FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA – UNIR
SOCIEDADE BRASILEIRA DE MATEMÁTICA – SBM
MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL – PROFMAT

ALGORITMOS PARA FATORAÇÃO E PRIMALIDADE COMO FERRAMENTA DIDÁTICA PARA O ENSINO DE MATEMÁTICA

JEAN PEIXOTO CAMPOS

PORTO VELHO
2014

JEAN PEIXOTO CAMPOS

ALGORITMOS PARA FATORAÇÃO E PRIMALIDADE COMO FERRAMENTA DIDÁTICA PARA O ENSINO DE MATEMÁTICA

Trabalho apresentado a Fundação Universidade Federal de Rondônia, como requisito parcial para obtenção do título de Mestre em Matemática sob orientação do professor Doutor Tomás Daniel Menéndez Rodríguez.

PORTO VELHO
2014

FICHA CATALOGRÁFICA

C1984a Campos, Jean Peixoto

Algoritmos para fatoração e primalidade como ferramenta didática para o ensino de matemática / Jean Peixoto Campos. - - Porto Velho: UNIR/ PROFMAT, 2014.

95 f.: il; 31cm.

Orientador: Prof. Dr. Tomás Daniel Menéndez Rodríguez

Dissertação (Mestrado) Universidade Federal de Rondônia (UNIR), Mestrado Profissional em Matemática em Rede Nacional (PROFMAT).

Bibliografia: p.90-94.

1. Teoria dos Números. 2. Programação. 3. Ensino-Aprendizagem Matemática - Dissertação I. Rodríguez, Tomás Daniel Menéndez II. Mestrado Profissional em Matemática em Rede Nacional - PROFMAT. III. Título.

CDD 510.7 - CDU 511

Bibliotecária Responsável: Miriã Santana Veiga CRB11/898

Jean Peixoto Campos

ALGORÍTMOS PARA FATORAÇÃO E PRIMALIDADE COMO FERRAMENTA DIDÁTICA PARA O ENSINO DE MATEMÁTICA

Trabalho de Conclusão de Curso defendido no Programa de Mestrado Profissional em Matemática em Rede Nacional – PROFMAT, do Departamento de Matemática da Fundação Universidade Federal de Rondônia – UNIR, como requisito parcial para obtenção do título de Mestre em Matemática, área de concentração Teoria dos Números, aprovado no dia 19 de setembro de 2014, pela Banca Examinadora constituída pelos docentes:

Prof. Dr. Tomás Daniel Menéndez Rodrigues
Orientador/Presidente
PROFMAT / UNIR

Prof. Dr. Adeilton Fernandes da Costa
PROFMAT / UNIR

Prof. Dr. Edcarlos Miranda de Souza
PROFMAT / UFAC

Dedico este trabalho àqueles que mais tenho orgulho: meus pais que sem pestanejar sempre me incentivaram a alçar voos mais altos, a minha esposa que me acompanhou e deu forças para o êxito e enfim a meu filho que me roubou tanto tempo de trabalho para viver alegrias incalculáveis.

Agradecimentos

Agradeço ao Pai Eterno que iluminou meus caminhos por toda a vida para que pudesse chegar a este momento.

Ao meu orientador prof. Dr. Tomás Daniel Menéndez Rodríguez que me compartilhou as dificuldades do trabalho e auxiliou na solução das mesmas sugerindo caminhos melhores e adequados ao objetivo.

Aos professores do PROFMAT: Dr. Adeilton Fernandes da Costa, Dr. Marinaldo Felipe da Silva, Flávio Batista Simão e Ronaldo Chaves Cavalcanti que contribuíram para uma formação sólida e promovendo meu crescimento acadêmico e profissional.

Aos professores da graduação: Ms. Fernando Luiz Cardoso, Dr. Carlos Mergulhão Júnior, Ms. Reginaldo Tudeia dos Santos, Ms. Marlos Gomes de Albuquerque e Dr. Aparecida Augusta da Silva, pelo auxílio nos primeiros passos dessa trajetória.

Aos grandes amigos que comigo caminharam e partilharam anseios, frustrações e glórias: Adalberto Carlos do Nascimento Silva, Alisson Gleike Moraes, Francenildo Cardoso de Oliveira, Francisco Sales Pereira, Gilson Marcos Caliani, Guilherme Alves de Sousa, José Inildo Alencar, Kleber Barbosa Sales, Luci Fabiane Belasquem Peter, Magno Carvalho Martins, Marizete Nink de Carvalho, Sézani Morais Gonçalves de Carvalho e Vicente Ferrer Trajano Bezerra que tornaram o caminho mais agradável.

E em especial a minha esposa e filho pelos momentos de abdicação em prol do sucesso deste trabalho e a força que me deram duram o percurso e sempre.

Toda educação científica que não se inicia com a Matemática é, naturalmente, imperfeita em sua base.

Augusto Comte

RESUMO

Este trabalho apresenta uma proposta de ensino de Matemática com o auxílio do desenvolvimento e implementação de algoritmos para fatoração e primalidade de inteiros positivos. Em outras palavras, utilizar recursos computacionais para discutir tópicos de teoria dos números que irão subsidiar o professor para discutir temas matemáticos como: conjuntos, variáveis e funções no sentido de desenvolver a capacidade de generalização e abstração junto a seus alunos. Para tanto, utiliza-se da teoria construcionista e o subsídio do Pensamento Matemático Avançado (PMA) para guiar a construção da sequência didática que discute temas de divisibilidade e números primos. As ferramentas computacionais utilizadas para a implementação foram as linguagens de programação *Scratch* e *Python*, ambas multiplataforma e gratuitas. O tema que constroi o enredo do trabalho é um objeto para motivação, pois, além de ser um grande recurso para a segurança da informação nos dias atuais, apresenta forte relação entre o mundo da escola e o contexto social em que o aluno está inserido. A construção da sequência didática é flexível para que o professor possa adequar a sua realidade educacional, podendo inclusive ser ampliada ou reduzida com facilidade. O trabalho proporciona, também, um roteiro para o professor que encontra exemplos que podem ilustrar suas aulas com aplicações da Matemática no contexto computacional sem desviar do objetivo fim que é o processo de ensino e aprendizagem curricular previamente definido, com qualidade e significado. A proposta construída possibilita essas facetas do processo e abre a possibilidade para o professor criar, inclusive, seus próprios recursos pedagógicos e aos alunos um aprendizado mais autônomo onde o professor promove a mediação da aprendizagem e não atua como principal fonte de informações do processo.

Palavras Chave: Teoria dos Números. Programação. Ensino-Aprendizagem Matemática.

ABSTRACT

This paper presents a proposal for teaching mathematics to aid the development and implementation of algorithms for factoring and primality of positive integers. In other words, use computational resources to discuss topics of number theory that will support the teacher to discuss mathematical themes such as sets, variables and functions in order to develop the ability of generalization and abstraction with their students. For this, it was used the Constructionist Theory and the subsidy of Advanced Mathematical Thinking (AMT) to guide the construction of instructional sequence that discusses topics of divisibility and primes. The computational tools used for implementation were the languages of Scratch and Python programming, both multiplatform and free. The theme that builds the plot of the work is an object for motivation because, besides being a great resource for information security today, shows strong relationship between the world of school and the social context in which the student is inserted. The construction of the instructional sequence is flexible so that the teacher can tailor their educational reality may even be enlarged or reduced with ease. The work also provides a roadmap for the teacher who finds examples that can illustrate their lessons with the applications of mathematics in the computational context without deviating from the objective end, which is the process of teaching and learning curriculum previously defined, with quality and meaning. The proposal provides built these facets of the process and opens the possibility for the teacher even create their own learning resources and students a more autonomous learning where the teacher promotes learning mediation and does not act as a primary source of information of the process.

Keywords: Number Theory. Programming. Mathematics Teaching and Learning.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A sociedade está cada vez mais imersa no mundo da tecnologia, e neste contexto o computador tem papel preponderante. Por outro lado, a escola, em sua maioria, não utiliza este recurso como uma ferramenta que favoreça a aprendizagem, seja por falta de capacitação, infraestrutura, ou outras razões. Diante disso, o aluno perde a oportunidade de ter o contato com o computador na escola, o que, lhe valeria posteriormente ingressar no meio social com saber adequado à atual demanda. Esta dicotomia pode ser reduzida ao se inserir o computador no âmbito escolar, como mais uma alternativa para o ensino.

Esse pressuposto está alinhado com o que indica Henriques (2010) ao relacionar o saber escolar com as mudanças e o rápido desenvolvimento da sociedade. Decorrente disso, o arcabouço de conhecimentos necessário é muito maior que em outros tempos, com ênfase para saber aplicar os saberes matemáticos ao mundo, desenvolver ideias matemáticas de forma autônoma e formular conjecturas.

Nesse sentido, o presente trabalho busca integrar ferramentas tecnológicas na rotina escolar da disciplina de Matemática, voltadas para alunos ingressantes do ensino médio, ou mesmo de ingressantes do ensino superior. Pois, utiliza-se do recurso de visualização para facilitar a aprendizagem, visualização essa proporcionada pelo recurso computacional que permite, inclusive, que o aluno manipule ideias e o professor pode suscitar temas matemáticos abstratos e complexos (KAWASAKI, 2008).

Com isso, reforçado pelo fato de que ambientes computacionais podem ser de grande valor para experimentações que ajudam, antes mesmo do desenvolvimento teórico, ao fazer com que alguns padrões e indícios de um dado fenômeno possam surgir (KAWASAKI, 2008). Uma vez que no entorno das ideias a serem discutidas está o desenvolvimento de um pensamento matemático voltado para a capacidade de encontrar padrões e abstrair alguns conceitos que são qualidades úteis, não só na Matemática, bem como para diversos contextos em que o aluno se encontre.

De fato, o objetivo do processo de ensino-aprendizagem não pode ser a transmissão de informações e conhecimentos, mas sim deixar pensamentos elementares para dar foco ao raciocínio, à resolução de problemas e a capacidade de investigação (HENRIQUES, 2010). Com vistas a fazer com que os alunos não criem o conceito de que Matemática é uma questão apenas escolar, pronta e terminada, mas sim um ambiente vivos e em constante remodelação.

Caso contrário, os alunos podem não ser capazes de construir o saber necessário diante de uma mudança de cenário. E como, notoriamente e de forma acelerada, ocorrem mudanças no cenário social que acaba por podar sujeitos que não possuem habilidades para se desenvolver, acabando por demandar ações externas do estado a fim de subsidiar tais indivíduos, que possivelmente ficam a margem em seu "próprio" ambiente. Assim, é imprescindível que alunos passem do simples ato de estudar, ao patamar de aprender com compreensão, e desta forma aplicar os conhecimentos adquiridos para apreender novos tópicos e resolver problemas que fogem ao escopo de conhecimento.

Em outro sentido, o ingresso do aluno no ensino médio é repleto de alterações do ponto de vista mental, no que concerne a Matemática, o aluno passa a ser cobrado sob temas mais abstratos, como: operações com conjuntos, compreensão e generalização de fórmulas e operações com variáveis e incógnitas. Além disso, também é notória a dificuldade que os professores enfrentam para que os conceitos desses temas sejam apreendidos por seus estudantes.

Nesse sentido, há ainda o problema motivacional em que o aluno não consegue compreender a necessidade de conhecer sobre tais objetos matemáticos ou mesmo em que medida serão necessários para seu futuro. Sendo assim, cabe ao professor apresentar o conteúdo de forma que seja possível ao aluno estabelecer relações com questões que fujam ao contexto escolar e por isso, a temática da criptografia e a utilização de computadores passa a ser uma interessante via de acesso a estas relações, extrapolando o ambiente escolar e permitindo ao aluno maior clareza e aplicabilidade destes temas, não só no contexto escolar e matemático, mas no âmbito social.

Ao pensar na capacidade de resolução de problemas e no público ao qual se pretende atingir, chegou-se a um tema matemático que praticamente não está contemplado na educação básica, mas que dada a atual conjuntura é de extrema importância para a sociedade: divisibilidade de números naturais, números primos e fatoração.

A estrutura da transmissão de informações mundiais ocorre por meio de redes de computadores, sendo possível que esta informação passe a ser conhecida por pessoas não autorizadas. Diante disso, diversos mecanismos de segurança são utilizados e estes sistemas estão construídos sobre um problema matemático secular, a determinação da primalidade de um natural ímpar grande, problema que recai sobre divisibilidade.

E com este tema, que contempla situações relativamente simples de se compreender, mas não necessariamente simples de se resolver, foi definido o objetivo de construir e implementar algoritmos para fatoração e primalidade como ferramenta didática que possibilite

o desenvolvimento das capacidades de generalização e abstração, juntamente com conceitos sobre variáveis, funções e alguns temas introdutórios sobre teoria dos números.

Uma vez que a programação, assim como a construção do saber matemático, é uma atividade que se baseia no método de tentativa e erro desde as primeiras fases de aprendizagem.

A alternativa de usar o computador se justifica, entre outras coisas, pelo fato de que, conforme Setzer (1998), a máquina necessita de instruções a partir de uma "linguagem formal", ou seja, que pode ser definida matematicamente, pois não permitem ambiguidades. Sinteticamente, o computador é uma máquina Matemática, por exigir raciocínio e linguagem matemáticos, similar ao processo de provas e demonstrações.

Assim o ato de programar se aproxima do trabalho do matemático em descrever sem ambiguidades o processo de resolução de um problema ou a prova de um teorema. Incorporar esta ferramenta ao contexto escolar avança o processo de desenvolvimento da lógica necessária à Matemática e às atuais demandas sociais, uma vez que alfabetização tecnológica é, também, um requisito essencial para a sociedade.

O tema traz consigo uma gama de conhecimentos que na exposição da sequência se mostram possíveis. O professor em sala de aula pode criar uma atmosfera "*hacker*"[1] para iniciar e motivar o estudo por meio da sequência didática que inicia com temas do ensino fundamental e avança a conceitos muitas vezes não vistos em um curso comum do ensino médio.

A priori, fez-se um levantamento bibliográfico em bases de dados buscando trabalhos na mesma linha de pesquisa que pudessem fornecer subsídios para a pesquisa. Encontrou-se uma série de trabalhos que utilizam o computador para diversas atividades no ensino de Matemática, como também diversos trabalhos versando sobre ensino de tópicos de teoria dos números, no entanto não foi identificado trabalho de mestrado ou doutorado que concilie ambas as vertentes, de tal forma que este trabalho buscará conciliar os dois temas para que em conjunto ofereçam resultado maior que a soma das partes, afinal o potencial de aprendizagem será maior que lógica/programação e teoria dos números, mas outros temas e competências pertinentes a aprendizagem matemática.

[1] Simular um ambiente em mensagens criptografas são interceptadas e tentam-se decodificá-las.

Posteriormente o trabalho foi guiado para encontrar uma linguagem de programação que fornecesse fácil aprendizado e bons resultados educacionais. Sobre este tema foram consideradas duas linguagens: *Scratch* e *Python*, pois ambas são gratuitas, multiplataforma e a primeira foi desenvolvida para uso, inclusive, com crianças em fase de alfabetização até programadores experientes e para uso em robótica enquanto a segunda por ter uma sintaxe mais próxima da linguagem matemática e que dispensa a declaração de variáveis e estas são de tipagem dinâmica[2] o que facilita o trabalho, principalmente de "programadores" iniciantes como é o caso da presente pesquisa.

O passo seguinte foi definir os aspectos teóricos de base do trabalho, estes dividem-se em: (i) Construcionismo; (ii) Pensamento Matemático Avançado e (iii) Matemática. Sendo estes os temas que comporão o capítulo sobre referenciais teóricos.

A seguir são dispostos os meios pelos quais a pesquisa foi desenvolvida abordando a revisão bibliográfica e as ferramentas computacionais utilizadas.

O capítulo seguinte trata da sequência didática, propriamente dita, que foi dividida em duas etapas, sendo a primeira com o intuito de discutir a parte elementar de divisibilidade e assim algoritmos que envolvem temas mais facilmente compreensíveis, enquanto a segunda utiliza temas mais complexos e suas implementações bem como as potencialidades para o processo de ensino e aprendizagem.

Essa divisão visa fazer com que o professor possa ter flexibilidade no uso da sequência, podendo suprimir a segunda parte, se seus objetivos de aprendizagem tiverem sido atingidos ou mesmo modificando-a para uma maior ênfase ao contexto de funções, seja para explorar a temática de substituição de valores para funções afins, quadráticas ou mesmo transcendentes, ou ainda, organizar algoritmos para a descoberta de raízes dessas funções, sejam por métodos exatos ou por aproximação.

No capítulo final, é exposto, de forma sucinta, a evolução do tema e suas principais contribuições para o fazer docente, bem como a exposição do resultado final do objetivo de pesquisa. E assim findando a estrutura do trabalho.

[2] São variáveis que mudam seu tipo de acordo com os valores que recebem.

2 REFERENCIAL TEÓRICO

Neste capítulo serão abordados os referenciais teóricos a serem utilizados para a elaboração da sequência didática.

Inicialmente, discutir-se-á sobre o construcionismo idealizado por Seymour Papert em conjunto com a ideia de Pensamento Matemático Avançado, abordado por David Tall e finalizando com conceitos matemáticos que serão suscitados no uso da sequência didática. As ferramentas matemáticas, ao serem apresentadas de forma flexível, poderão subsidiar, inclusive, ao professor na sua prática e fazer docente, fazendo com o que a ação docente não seja engessada.

2.1 CONSTRUCIONISMO

O construcionismo é uma teoria que surge com a inserção do uso do computador como uma ferramenta para o ensino. Em um contexto histórico, o uso de computadores na educação iniciou-se com atividades ditas instrucionistas, em que o computador era visto somente como uma ferramenta para pesquisa, ou seja, limitava o computador a uma máquina de ensinar e o aluno a um ser passivo que recebe instrução (MOTTA e SILVEIRA, 2010).

Segundo Santanchèe e Teixeira (1999, p. 1), o "[...] Instrucionismo fundamenta-se no princípio de que a ação de ensinar é fortemente relacionada com a transmissão de informação (instrução) ao aluno. A melhoria do ensino, sob esta ótica, consiste em aperfeiçoar as técnicas de transmissão da informação".

Assim, o computador começou a entrar neste contexto para auxiliar e incrementar o processo de comunicação. A instrução programada[3], proposta por Skiner, é uma das utilizações do computador com enfoque instrucionista, nela o computador assume o papel de "máquina de ensinar". Pode-se ainda citar outros mecanismos que utilizam o mesmo princípio: os sistemas tutores, programas de exercício e prática, navegação em hipertexto, etc.

[3] Que pode ser exemplificada com situação de um aluno estudando a partir de um livro, porém ao invés do livro quem disponibiliza a informação é o computador.

Sobre tal visão reducionista do uso do computador na educação, Papert pensou que "[...] o computador deve ser uma máquina de pensar com, auxiliando no desenvolvimento dos processos mentais incentivando o aluno a construir sua própria aprendizagem" (MOTTA e SILVEIRA, 2010, p. 120).

Seymour Papert desenvolveu a teoria construcionista sobre as bases do construtivismo piagetiano e segundo Altoé e Penati (2005, p. 1), o construtivismo é

> [...] uma corrente teórica empenhada em explicar como a inteligência humana se desenvolve, partindo do princípio de que o desenvolvimento da inteligência é determinado pelas ações mútuas entre o sujeito e o meio. Considera-se que a inteligência do homem não é inata, mas que o sujeito também não é passivo sob a influência do meio, isto é, ele responde aos estímulos externos agindo sobre eles para construir e organizar o seu próprio conhecimento, de forma cada vez mais elaborada.

De forma superficial e simplificada, o desenvolvimento do sujeito se dá na interação dele com o meio, o que provoca um processo de constante e gradual adaptação ao meio.

Então, o

> [...] Construcionismo se fundamenta numa perspectiva diversa [do instrucionismo]. O aprendizado é encarado como uma atitude ativa, onde o aluno constrói o próprio conhecimento. O uso dos computadores sob a ótica construcionista parte de uma direção inversa à do Instrucionismo. Nela o aluno, através de um software apropriado, aprende exercitando uma tarefa de 'ensinar' o computador (SANTANCHÈ e TEIXEIRA, 1999, p. 2).

Pois, segundo Altoé e Penati (2005), o termo construcionismo é definido como a construção do conhecimento pelo aluno por meio do computador, com o mínimo de ensino para o máximo de aprendizagem do aluno. Nesse cenário, o aluno constrói suas estruturas cognitivas a partir de suas ações com auxílio de suas construções de mundo.

O construcionismo além de uma teoria é uma estratégia para a educação. Nele “o conhecimento não é simplesmente transmitido pelo professor ao aluno, mas, é, efetivamente construído pela mente desse” (MISKULIN, 1999, p. 232). É uma forma de relacionar e utilizar o computador no ambiente em que a tríade: professor, aluno e saber se encontram

assim, “[...] o computador se torna um elemento de 'interação que propicia o desenvolvimento da autonomia do aluno [...]” (ALTOÉ e PENATI, 2005, p. 6), sem necessariamente direcionar a sua ação, mas fomentando por meio de explorações, experimentações, descobertas e reflexões a construção de conhecimento em diversas áreas do saber.

Uma das diferenças entre o construcionismo de Papert e o construtivismo piagetiano é que no primeiro não é dada ênfase às etapas de desenvolvimento da inteligência da criança, mas sim como o desenvolvimento cognitivo acontece, como as ideias piagetianas podem colaborar para uma aprendizagem mais efetiva. Desta forma, entra o computador como ferramenta educacional que possibilita a construção de situações de aprendizagens que acelerem as etapas do desenvolvimento.

No Construcionismo,

> O conhecimento não é fornecido ao aluno para que ele dê as respostas. É o aluno que coloca o conhecimento no computador e indica as operações que devem ser executadas para produzir as respostas desejadas. O programa fornece importantes pistas sobre o pensamento do aluno, uma vez que o seu pensamento está descrito explicitamente e a resposta do computador permite comparar o previsto com o obtido. O professor tem maiores chances de compreender o processo mental do aluno, ajudá-lo a interpretar as respostas, questioná-lo, colocar desafios que possam ajudá-lo na compreensão do problema e conduzi-lo a um novo patamar de desenvolvimento (ALMEIDA, 2000, p. 33-34).

Assim, o computador é o meio no qual o aluno resolve problemas utilizando suas estruturas cognitivas por meio de atividades propostas que considerem os princípios de liberdade e de autonomia do aprendiz. Com isso, o sujeito constrói seu conhecimento por meio de descobertas próprias, na interação com o objeto e elaborando e reelaborando as situações-problema propostas. "Nesse processo de reflexão para a superação do estado de desequilíbrio que pode ser ocasionado, ocorre a construção de novas estruturas cognitivas, sempre em um nível de conhecimento superior" (ALTOÉ e PENATI, 2005, p. 8).

Com isso, é preciso perceber que a concepção construcionista suscita mudanças no processo educacional, pois dá maior ênfase a aprendizagem do que ao ensino, na construção do conhecimento do que na instrução. E essa mudança não é algo simples, pois necessita que a prática pedagógica seja integrada à informática e disso decorre uma necessária e constante qualificação dos profissionais da educação para o desenvolvimento dessa prática.

Notoriamente há que ocorrer uma mudança de paradigma para que proporcione a formação de sujeitos mais críticos e autônomos para a construção do próprio conhecimento (POCRIFKA e SANTOS, 2009).

Em um ambiente construcionista, a interação com o conhecimento ocorre pela mediação do computador que molda um mundo para o estudante explorar os conceitos. Esse mundo é uma "simplificação" do real contexto que o saber estaria imerso, proporcionando foco no objeto de estudo planejado pelo docente. Esse ambiente é denominado micromundo que simboliza a realidade, ou seja, o objeto saber a ser apreendido. Assim, "[...] o termo micromundo consiste na ideia de mundos auto contidos, em que os alunos podem transferir seus hábitos de exploração da vida pessoal para o ambiente que propicia a construção do conhecimento científico" (BARROS e STIVAM, 2012, p. 188).

Nesse ambiente, o estudante tem a possibilidade de interagir e automaticamente verificar a veracidade ou não de suas conjecturas, isso ocorre, segundo Valente (2005), em um ciclo "*descrição-execução-reflexão-depuração*", nesse ciclo o aprendiz implementa (descreve) a solução, executa-a, verifica o resultado obtido e corrige as possíveis falhas apresentadas e torna a descrever a solução em um processo de fazer contínuo que o leva a obtenção do conhecimento planejado.

O outro tópico que guiará o desenvolvimento deste trabalho é o Pensamento Matemático Avançado, porém antes de discorrer sobre o mesmo far-se-á uma caracterização de como a variável pode ser compreendida e manipulada pelo aluno.

2.2 PENSAMENTO MATEMÁTICO AVANÇADO

A princípio pode se pensar nas qualidades abstratas inerentes à Matemática, com isso, alguns pesquisadores apontam que o pensamento algébrico é um tipo especial de pensamento, pois pode se manifestar em diferentes campos da Matemática, ou mesmo em outras áreas do conhecimento. A caracterização desse tipo de pensamento pode ser dada como: “[...] percepção de regularidades, percepção de aspectos invariantes em contraste com outros que variam, tentativas de expressar ou explicitar a estrutura de uma situação-problema e a

presença do processo de generalização" (FIORENTINI, MIORIM E MIGUEL *apud* GERETI ET AL., 2013, p. 1921).

A teoria do Pensamento Matemático Avançado (PMA) foi desenvolvida por um grupo de trabalho durante o *International Group for the Psychology of Mathematics Education* e desde então diversos pesquisadores como Tall, Resnick e Dreyfus, entre outros, tem se dedicado ao tema. E

> Segundo Dreyfus (2002), o PMA não é exclusivo de indivíduos que estão fazendo um curso do Ensino Superior, porém uma característica distintiva deste tipo de pensamento para o pensamento matemático elementar está no modo como os conceitos complexos são tratados. Para o autor os processos do PMA possibilitam fazer essa distinção. (GERETI et al., 2013, p. 1923)

Assim serão abordadas as características do PMA, de modo que se possa compreendê-lo e verificar sua função dentro do presente trabalho.

2.2.1 Características do Pensamento Matemático Avançado

O pensamento matemático está interligado a processos cognitivos que originam o conhecimento matemático. Estes processos variam entre os sujeitos, podendo ser rápidos "Eureca!" ou dependente de uma série de outros processos mentais decorrem de atividades de aprendizagem.

Os processos que originam o PMA podem também ser encontrados no Pensamento Matemático Elementar (PME) e isso decorre do fato não haver uma distinção acentuada entre muitos dos processos que são usados no PME e PMA.

Dreyfus (apud HENRIQUES, 2010, p. 16) faz uma tênue distinção entre PME e PMA, considerando que "É possível pensar em tópicos matemáticos avançados numa forma elementar e pode ter-se pensamento avançado sobre tópicos elementares".

Assim, segundo Henriques (2010, p. 16)

> É a complexidade dos processos usados no pensamento matemático e as mudanças cognitivas que se verificam no indivíduo que determinam o tipo de pensamento envolvido na aprendizagem de um dado conceito e que caracterizam a transição do PME para o PMA.

Desta forma, dentre os processos presentes no PMA a representação e a abstração são os mais poderosos para passar de um nível de detalhe para outro e assim gerir a complexidade na passagem do PME para o PMA.

A representação do conhecimento, sob a ótica do PMA necessita também de diversos processos são eles: (i) representação; (ii) mudança de representações e a tradução entre elas; e (iii) a modelação.

O desenvolvimento da Matemática se deu em boa parte pela capacidade de representar ideias através de símbolos e saber operar com seu auxílio, assim no que concerne ao PMA, além da representação simbólica que deve carregar em si, signos que tenham significados há também a representação mental que ocupa uma posição central no PMA, pois ao fazer referência a um objeto ou processo matemático cada indivíduo recorre a sua representação mental.

Nesse sentido, Machado e Bianchini (2013, p. 592) definem que o "**processo de representar** um conceito é aquele de gerar uma instância, um espécime, um exemplo, uma imagem dele. Ocorre em registros compartilhados como da escrita, do desenho, da fala, dos gestos e outros".

Henriques (2010, p. 20) aponta que

> Representar um conceito significa gerar um caso, um exemplo ou uma imagem dele. Mas esta descrição não especifica se o caso gerado é simbólico ou mental, nem indica o que 'gerar' significa em termos de processos pelos quais as representações mentais surgem e como são desenvolvidas. Desta forma, enquanto uma representação simbólica é externamente escrita ou verbalizada, usualmente com o objectivo de tornar a comunicação sobre o conceito mais fácil, a representação mental refere-se ao esquema interno ou imagens de referência que o indivíduo usa para interagir com o mundo exterior. A representação mental torna-se assim fundamental para permitir ao indivíduo comunicar o seu pensamento sobre um objecto ou processo. (HENRIQUES, 2010, p. 20)

É nesse sentido que o computador atuará como um meio que possibilite ao aluno externar suas representações mentais através do desenvolvimento de programas, além de possibilitar ao mesmo verificar se seus construtos estão respondendo adequadamente.

A visualização é outro fator que compõe o processo de representação auxiliando, inclusive, na criação das representações mentais. Pois, a visualização favorece a intuição e compreensão dos conceitos matemáticos ao gerar imagens que poderão ser associadas e/ou comparadas no processo de abstração e gerar representações mentais.

Desta forma, o indivíduo terá diversas representações as quais fará uso de acordo com a situação. Estas imagens mentais podem ser complementares e integradas em uma única para o conceito. O que favorece o desenvolvimento, uma vez que estas representações podem ser usadas simultaneamente e assim ganhar foco de acordo com a necessidade. Esta está associada ao processo de tradução, estas traduções podem ser compreendidas como a transformação de uma propriedade matemática ou problema para outro.

A modelação é outro subprocesso envolvido no processo de representação, ou seja, construir uma teoria ou estrutura matemática para algo não matemático (objeto, situação ou processo), mas que incorpore as características essenciais desse elemento em questão. "O modelo matemático ganha então o estatuto de uma representação da situação, embora esta, por si só, não seja suficiente para o indivíduo. Este necessita de formar também uma representação mental associada ao processo da modelação" (HENRIQUES, 2010, p.21).

Um dos problemas de alguns alunos é a capacidade de, ao ser exposto a diversos fatos correlacionados, identificar de que forma eles se relacionam e desenvolver uma generalização, ou seja, a habilidade de abstrair. A cada ciclo da escolaridade a capacidade de abstração se faz mais necessária na escola básica, ela se acentua a partir do final do ensino fundamental, onde o aluno deve operar explicitamente com valores desconhecidos (incógnitas) e no ensino médio é evidente a necessidade desde o primeiro ano.

Segundo Machado e Bianchini (2013, p. 592), "O **processo de abstrair** supõe os subprocessos de **generalizar** e **sintetizar**".

As autoras definem os processos de generalizar e sintetizar:

> O processo de generalização é aquele que permite ao sujeito tirar como consequência ou induzir do particular, identificar o que há de comum, expandir o domínio de validade. Enquanto o processo de sintetizar significa combinar ou compor partes de tal forma, que elas formem um todo isto é, um objeto matemático. É importante ressaltar que tais processos são indissociáveis. (MACHADO e BIANCHINI, 2013, p. 592)

Com isso, Machado e Bianchini (2013, p. 593), sintetizam os processos envolvidos no PMA por meio da figura 1.

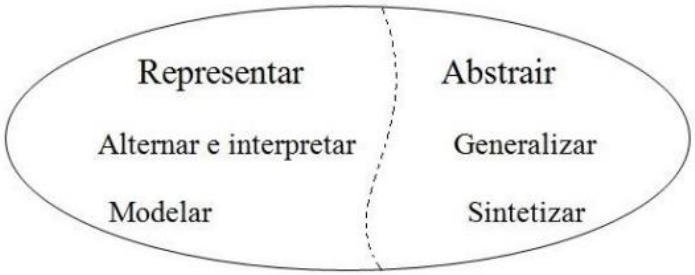

Figura 1: Principais processos do PMA

2.3 BASES MATEMÁTICAS DA SEQUÊNCIA DIDÁTICA

Este tópico aborda a teoria Matemática que será utilizada na sequência didática. Dado que o objetivo do trabalho é fomentar o desenvolvimento do conceito de variável e a habilidade de generalização, há uma grande possibilidade de temas matemáticos capazes de contribuir para esta questão. No entanto, optou-se por uma sequência que possa atingir um número maior de alunos, dada a heterogeneidade de níveis de desenvolvimento matemático com que os alunos chegam ao ensino médio.

Assim, as bases matemáticas concentram-se sobre a teoria dos números, especificamente: divisibilidade, números primos e alguns algoritmos sobre fatoração que são compreensíveis a todos os alunos, pelo fato do contato com a maioria dos itens (explícita ou implicitamente) durante toda sua vida escolar e que podem facilmente ser implementados no computador e por fazerem emergir uma grande quantidade de saberes matemáticos.

2.3.1 Divisibilidade

Os aspectos da divisibilidade são de fundamental importância para este trabalho, visto ser sobre eles que estarão alicerçados os demais elementos.

Definição 1: Se a e b são inteiros, diz-se que a divide b, denotando por $a|b$, se existir um inteiro c tal que $b = ac$. E se a não divide b escreve-se $a \nmid b$.

Desta definição pode-se extrair a seguinte proposição.

Proposição 2: Se a, b e c são inteiros, $a|b$ e $b|c$, então $a|c$.

Dem.: Como $a|b$ e $b|c$, existem k_1 e k_2 com $b = k_1 a$ e $c = k_2 b$ (I). Substituindo o valor de b em (I) tem-se $c = k_2 k_1 a$ o que implica $a|c$.

No teorema a seguir serão tratadas as propriedades da divisão.

Teorema 3:

i. $n|n$;
ii. Se $d|n$ então $ad|an$;
iii. Se $ad|an$ e $a \neq 0$ então $d|n$;
iv. $1|n$;
v. $n|0$;
vi. Se $d|n$ e $n \neq 0$ então $|d| \leq |n|$;
vii. Se $d|n$ e $n|d$ então $|d| = |n|$;
viii. Se $d|n$ e $d \neq 0$ então $\left(\frac{n}{d}\right)|n$.

Demonstrações:

i. Como $n = 1n$ segue da definição que $n|n$[4];
ii. Se $d|n$ então $n = dk$ para algum inteiro k. Logo $an = adk$ que pela definição implica que $ad|an$;
iii. Se $ad|an$ então $an = adk$ para algum inteiro k, como $a \neq 0$ implica que $n = dk$;

[4] É válido mesmo quando $n = 0$, pois existe c tal que $0 = c \cdot 0$.

iv. Se $1|n$ então $n = 1k$. Logo $k = n$ e portanto $1|n$;

v. Se $n|0$ então $0 = nk$. Logo $n = 0$ ou $k = 0$ o que conclui a demonstração;

vi. Se $d|n$ então $n = dk$ para algum inteiro k, como $n \neq 0$, isso implica que $|k| \geq 1$ e portanto $|d| \leq |n|$;

vii. Se $n|d$ e $d|n$ então $d = nk_1$(I) e $n = dk_2$ (II) para inteiros k_1 e k_2, multiplicando (I) e (II) membro a membro tem-se $dn = nk_1dk_2 = dnk_1k_2$ que implica que $k_1k_2 = 1$ como k_1 e k_2 são inteiros tem-se que $k_1 = k_2 = \pm 1$. E portanto $|d| = |n|$;

viii. Se $d|n$ então $n = k_1d$ que implica que $\frac{n}{d}$ é um inteiro. Como $\frac{n}{d}.d = n$, pela definição $\left(\frac{n}{d}\right)|n$.

A divisão inteira nem sempre é exata. Para isso usa-se um algoritmo que é aprendido nos anos iniciais de escolaridade que remete a uma imagem similar a figura 2.

```
342 |15
 42  22
 12
```

Figura 2: Divisão de dois inteiros.

De forma geral tem-se o dividendo a, o divisor b, o quociente q e o resto r, ou seja, para o algoritmo da divisão tem-se como entrada a e b e saída q e r de forma que $a = bq + r$ e $0 \leq r < b$.

Proposição 4: (Teorema de Eudoxius) Dados a e b inteiros com $b \neq 0$ então a é um múltiplo de b ou se encontra entre dois múltiplos consecutivos de b, isto é, correspondendo a cada par de inteiros a e $b \neq 0$ existe um inteiro q tal que:

i. Para $b > 0$,

$$qb \leq a < (q + 1)b$$

ii. E para $b < 0$,

$$qb \leq a < (q - 1)b$$

Teorema 5: (Teorema de divisão) Sejam a e b inteiros positivos. Existem números inteiros q e r tais que

$$a = bq + r \text{ e } 0 \leq r < b$$

Além disso, os valores de q e r satisfazendo as relações acima são únicos.

Dem.: (Existência) Pelo Teorema de Eudoxius, como $b > 0$, existe q satisfazendo a $qb \leq a < (q+1)b \Rightarrow 0 \leq a - qb < b$, assim definindo $r = a - qb$, pode-se garantir a existência de q e r.

Dem.: (Unicidade) Para isso, suponha que existam dois pares r_1, q_1 e r_2, q_2 tais que $a = q_1 b + r_1$, com $0 \leq r_1 < b$ e $a = q_2 b + r_2$, com $0 \leq r_2 < b$, isso implica que

$$q_1 b + r_1 = q_2 b + r_2 \Rightarrow b(q_1 - q_2) = r_1 - r_2$$

Logo $b|(r_1 - r_2)$, no entanto, $r_1 < b$ e $r_2 < b$, com isso, $|r_1 - r_2| < b$ e como $b|(r_1 - r_2)$ implica que $r_1 - r_2 = 0$ e portanto $r_1 = r_2$. Logo $q_1 b = q_2 b \Rightarrow q_1 = q_2$, pois $b > 0$.

Segundo Coutinho (2013, p. 20) o Algoritmo da Divisão é dado por:

> **Entrada:** inteiros positivos a e b;
> **Saída:** inteiros não-negativos q e r tais que $a = bq + r$ e $0 \leq r < b$;
> **Etapa 1:** Comece fazendo $Q = 0$ e $R = a$;
> **Etapa 2:** Se $R < b$ escreva o quociente é Q e o resto é R e pare; senão vá para a Etapa 3;
> **Etapa 3:** Se $R \geq b$ subtraia b de R, incremente Q de 1 e volte à Etapa 2.

Este é um dos primeiros algoritmos que serão implementados na sequência didática.

2.3.1.1 Algoritmo Euclidiano

O Algoritmo Euclidiano[5] é um meio de calcular o Máximo Divisor Comum entre dois números inteiros, ou seja, dados a e b inteiros (ambos não simultaneamente nulos), encontrar d, também denotado por (a, b) ou $mdc(a, b)$ que é o maior inteiro que divide a e b. Em particular se $d = 1$, diz-se que a e b são co-primos ou primos entre si.

[5] Apesar da História da Matemática apontar que o algoritmo fosse conhecido a mais tempo, o nome algoritmo euclidiano se deve ao fato de ser encontrado no livro VII de Os Elementos, escrito por Euclides aproximadamente 300 a. C., em suas proposições: I "Sendo expostos dois números desiguais, e sendo sempre subtraído de novo o menor do maior, caso o que restou nunca meça exatamente o antes dele mesmo, até que reste uma unidade, os números do princípio serão primos entre si" (EUCLIDES, p. 270); II "Sendo dados dois números não primos entre si, achar a maior medida comum deles" (EUCLIDES, p. 271).

O algoritmo euclidiano consiste em: dados dois inteiros a e b (suponha $a \geq b$), dividir a por b encontrando o resto r_1, se $r_1 \neq 0$, divide-se b por r_1 obtendo o resto r_2, se $r_2 \neq 0$, divide-se r_1 por r_2, encontrando-se r_3 e assim sucessivamente até que r_n seja igual a zero e $mdc(a,b) = r_{n-1}$. Ou seja,

$$\begin{array}{ccc} a = bq_1 + r_1 & \text{e} & 0 \leq r_1 < b \\ b = r_1 q_2 + r_2 & \text{e} & 0 \leq r_2 < r_1 \\ r_1 = r_2 q_3 + r_3 & \text{e} & 0 \leq r_3 < r_2 \\ r_2 = r_3 q_4 + r_4 & \text{e} & 0 \leq r_4 < r_3 \\ \vdots & & \vdots \end{array}$$

Com isso, os restos serão cada vez menores e o maior será ainda menor que b, ou seja, $b > r_1 > r_2 > r_3 > r_4 > \cdots \geq 0$, como todos os elementos dessa sequência pertencem aos inteiros e estão limitados superiormente por b e inferiormente por 0 esta sequência é finita o que garante que o algoritmo pare.

Ao usar o a linguagem de programação *Scratch* o professor terá a oportunidade de construir com os alunos esse algoritmo, visto que não é nativo da linguagem nenhum método para o cálculo do máximo divisor comum.

Para verificar a validade do algoritmo é necessário anteriormente a demonstração do seguinte lema:

Lema 6: Sejam a e b números inteiros positivos. Suponha que existam g e s tais que $a = bg + s$. Então $mdc(a,b) = mdc(b,s)$.

Dem.: Do fato de $a = bg + s$ pode-se concluir que todo divisor de b e s é um divisor de a (decorrente do fato de se a, b, c, m e n são inteiros, $c|a$ e $c|b$ então $c|(ma + nb)$[6]). Esta mesma relação, escrita na forma $s = a - bg$, diz que todo divisor de a e b é um divisor de s. Logo o conjuto dos divisores comuns de a e b é igual ao conjunto dos divisores comuns de b e r, o que nos garante o resultado $mdc(a,b) = mdc(b,s)$.

Para a demonstração do algoritmo euclidiano far-se-á uso do lema 6 aplicado a sequência:

[6] Esta proposição não exige muito trabalho para ser demonstrada.

$$\begin{array}{ccc} a = bq_1 + r_1 & \text{e} & 0 \leq r_1 < b \\ b = r_1q_2 + r_2 & \text{e} & 0 \leq r_2 < r_1 \\ r_1 = r_2q_3 + r_3 & \text{e} & 0 \leq r_3 < r_2 \\ r_2 = r_3q_4 + r_4 & \text{e} & 0 \leq r_4 < r_3 \\ \vdots & & \vdots \\ r_{n-2} = r_{n-1}q_n + r_n & \text{e} & 0 \leq r_n < r_{n-1} \\ r_{n-1} = r_nq_{n+1} + 0 & & \end{array}$$

A partir da última equação tem-se, pelo lema 6, que $mdc(r_{n-1}, r_n) = r_n$, a penúltima equação diz que $mdc(n_{n-2}, r_{n-1}) = mdc(r_{n-1}, r_n) = r_n$ e repetindo o processo obtém-se que

$$mdc(n_{n-2}, r_{n-1}) = mdc(r_{n-1}, r_n) = \cdots = mdc(b, r_1) = mdc(a, b) = r_n.$$

O teorema a seguir, conhecido como algoritmo euclidiano estendido (COUTINHO, 2013) é o último resultado que será exposto, no que concerne a divisibilidade.

Teorema 8: Seja d o máximo divisor comum de a e b, então existem inteiros n_0 e m_0 tais que $d = n_0a + m_0b$.

Dem. Seja B o conjunto de todas as combinações lineares $na + mb$ onde n e m são inteiros. Este conjunto contém números negativos, positivos e também o zero. Escolhendo n_0 e m_0 tais que $c = n_0a + m_0b$ seja o menor inteiro positivo pertencente ao conjunto B. Prova-se que $c|a$ e $c|b$. Devido a similaridade das demonstrações, será demonstrada a primeira. Por contradição, suponha que $c \nmid a$. Neste caso, pelo teorema 5, existem q e r tais que $a = qc + r$ com $0 < r < c$. Portanto $r = a - qc = a - q(n_0a + m_0b) = a(1 - qn_0) + b(-qm_0)$. Assim, $r \in B$, pois $(1 - qn_0)$ e $(-qm_0)$ são inteiros, o que é uma contradição, uma vez que $0 < r < c$ e c é o menor elemento positivo de B. Logo $c|a$ e, analogamente, prova-se que $c|b$.

Como d é um divisor comum de a e b, existem k_1 e k_2 tais que $a = k_1d$ e $b = k_2d$ e assim, $c = n_0a + m_0b = n_0k_1d + m_0k_2d = d(n_0k_1 + m_0k_2) \Rightarrow d|c$ e portanto $|d| \leq |c|$, como ambos são positivos tem-se $d \leq c$, como $d < c$ não pode ocorrer, pois d é o máximo divisor comum, só pode ocorrer que $c = d = n_0a + m_0b$.

Essa breve exposição sobre tópicos de divisibilidade é o suficiente para alicerçar os próximos temas a serem elencados. Assim, pode-se encerrar discussão sobre divisibilidade, no entanto, para uma discussão mais apurada sobre o tema e os demais tópicos recomenda-se a leitura de Santos (2007), Martinez et al. (2013), Coutinho (2013) e Ribenboim (2012).

2.3.2 Números Primos

Os números primos como aponta Daineze (2013), são abordados nos primeiros anos de escolaridade e não são mais retomados o que deixa uma lacuna de estudos sobre teoria dos números no saber dos alunos. Este problema é agravado atualmente dada a aplicabilidade desse item na sociedade do conhecimento, principalmente sob o aspecto da criptografia.

Assim, este trabalho faz suscitar a necessidade e aplicabilidade da Matemática ser vista como ciência capaz de estar diretamente ligada ao contexto social e científico atual, bem como meio para o desenvolvimento de outros conceitos inerentes a própria Matemática.

Em Os Elementos (EUCLIDES, 2009), no livro VII encontram-se as seguintes definições:

- "1. unidade é aquilo segundo o qual cada uma das coisas existentes é dita uma" p. 269
- "12. Um número primo é o medido por uma unidade só" p.269
- "13. Um número composto é o medido por algum número" p. 270

É possível ver que os conceitos sobre números estão imersos nos conceitos geométricos, ou seja, números "pertencem" à geometria. É neste contexto que Euclides expõe de forma sistematizada parte dos conceitos que serão utilizados neste trabalho.

Enquanto Euclides possibilitou os primeiros passos na Teoria dos Números, mesmo que com um olhar geométrico ele auxiliou na criação de uma grande área de estudos da Matemática. No entanto, os números primos, hoje, são apresentados com a seguinte definição:

Definição 9: Um número inteiro $n(n > 1)$ possuindo somente dois divisores n e 1 é chamado *primo*. Se $n > 1$ não é primo, é dito *composto*.

Teorema 10: Se $a|bc$ e $mdc(a, b) = 1$, então $a|c$.

Dem.: Como $mdc(a, b) = 1$, pelo teorema 8 existem m e n inteiros tais que $na + mb = 1$, multiplicando ambos os membros por c, implica em $nac + mbc = c$ é óbvio que $a|ac$, e por hipótese, $a|bc$ então $a|c$.

Proposição 11: Sejam a, b e p inteiros e p primo. Se $p|ab$, então $p|a$ ou $p|b$.

Dem.: Se $p \nmid a$, então $mdc(a,p) = 1$ que implica, pelo teorema 10, $p|b$.

Teorema 12[7]: (Teorema Fundamental da Aritmética) Dado um inteiro positivo $n \geq 2$ pode-se sempre escrevê-lo, de modo único, na forma

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k},$$

onde $1 < p_1 < p_2 < \cdots < p_k$ são números primos e $e_1, e_2, \cdots, e_k$ são inteiros positivos.

Este teorema proporciona o primeiro algoritmo o qual tem um vínculo direto com o presente trabalho. E é apresentado por Coutinho (2013, p. 38):

> **Algoritmo de fatoração.**
> **Entrada:** inteiro positivo n.
> **Saída:** inteiro f que é o menor fator primo de n ou uma mensagem indicando que n é primo.
> **Etapa 1**: Comece fazendo $F = 2$.
> **Etapa 2**: Se n/F é inteiro escreva 'F é $fator\ de\ n$' e pare; senão vá para a Etapa 3.
> **Etapa 3**: Incremente F de uma unidade e vá para a Etapa 4.
> **Etapa 4**: Se $F > \sqrt{n}$ escreva n é primo e pare; senão volte à Etapa 2.

O fato de a Etapa 4 do algoritmo da fatoração ter como parada $F > \sqrt{n}$ se deve ao Teorema 13.

Teorema 13: Se n não é primo, então n possui, necessariamente, um fator primo menor do que ou igual a $\sqrt{n}$.

Dem.: Se n é composto então $n = n_1 \cdot n_2$ onde $1 < n_1 < n$, $1 < n_2 < n$. Sem perda de generalidade suponha $n_1 \leq n_2$. Logo n_1 tem que ser menor ou igual a $\sqrt{n}$ pois, caso contrário, ter-se-ia $n = n_1 \cdot n_2 > \sqrt{n} \cdot \sqrt{n} = n$ o que é absurdo. Logo, pelo teorema 12, n_1 é primo ou possui algum fator primo p, este deve ser menor que ou igual a também a um fator de $\sqrt{n}$. Como p, sendo um fator primo de n_1 é também um fator de n, finda-se a demonstração.

[7] A demonstração deste teorema fica a cargo do leitor, dada a riqueza de informações que ele transmite, lembrando que o teorema requer que seja mostrado a existência e a unicidade da fatoração.

Com isso, apesar de o algoritmo ser simples e compreensível, sua eficiência não é adequada para números grandes, pois exige uma grande quantidade de ciclos (*loops*) para se chegar a definição de um fator, se este não for pequeno.

E esta é uma das questões motivadoras da evolução da Teoria dos Números, melhoramentos em algoritmos para fatoração de inteiros. Muito foi feito desde o Crivo de Eratóstenes, o qual,

> Limitando-se ao que o algoritmo se propõe a fazer, observamos que usa muita memória e executa laços muitas vezes, o que é ruim. Por outro lado, não é necessário executar nenhuma divisão (que são lentos), e o algoritmo é muito fácil de programar, o que é bom (COUTINHO, 2013, p. 65).

Assim, do ponto de vista educacional a que esse trabalho se destina é interessante considerar o Crivo como meio de ensino. Porém, uma questão poderia surtir efeito motivador nos alunos, é uma que relacione o foco da pesquisa e sua aplicabilidade. Uma possível é a exploração da criptografia RSA[8], ou seja, simular um ambiente de transmissão de informações que será alvo de ataques e as informações nessa rede deverão estar protegidas e seguras. Um mecanismo é a criptografia RSA que se vale de utilização de número primos grandes, com isso a atividade será desenvolver meios de encontrar primos grandes (rapidamente) com o intuito de proteção, enquanto o *hacker* busca encontrar meios de fatorar as chaves públicas e logo decodificar as mensagens.

Com isso, o estudo de algoritmos mais eficientes serão necessários para a atividade. Contudo não é intuito desenvolver meios complexos, dado que a sequência será aplicada a alunos do ensino médio, assim não será feito um estudo profundo sobre o tópico, mas sim uso de alguns algoritmos simples que mostram diferentes aplicações a certos grupos numéricos.

O tópico que será abordado é a Fatoração por Fermat, este método é útil quando os fatores dos números são próximos e se tem algum fator primo não muito menor que a raiz quadrada dele.

[8] O método de criptografia de chave pública mais conhecido e utilizado atualmente, desenvolvido em 1978 por Rivest, Shamir e Adleman (RSA) no MIT. Boas referências sobre isso podem ser encontradas em Coutinho (2013), Daineze (2013) ou Luz (2013).

Segundo Coutinho (2013, p. 40) este algoritmo pode ser expresso da seguinte forma:

> **Entrada**: inteiro positivo ímpar n.
> **Saída**: um fator de n ou uma mensagem indicando que n é primo.
> **Etapa 1**: Comece com $x = \sqrt{n}$; se $n = x^2$ então x é fator de n e podemos parar.
> **Etapa 2**: Caso contrário incremente x de uma unidade e calcule $y = \sqrt{x^2 - n}$.
> **Etapa 3**: Repita a Etapa 2 até encontrar um valor inteiro para y, ou até que x seja igual a $(n + 1)/2$: no primeiro caso n tem fatores $x + y$ e $x - y$, no segundo n é primo.

Para facilitar a abordagem deste processo, suponha n inteiro positivo e ímpar[9], a e b também inteiros positivos e $a < b$. Todo n pode ser escrito como $n = a.b$, em particular, quando n é primo, tem-se que $n = a.b = 1.n$, logo $a = 1$ e $b = n$. A fatoração por Fermat se estrutura sobre este fato, ou seja, em escrever n como produto de dois fatores, o método prevê que seja possível escrever $n = a.b = x^2 - y^2 = (x - y).(x + y)$, logo pode-se encontrar a e b tais que $a = x - y$ e $b = x + y$. Com isso, obtém-se que $x = \frac{a+b}{2}$ e $y = \frac{b-a}{2}$. De fato,

$$x^2 - y^2 = \left(\frac{a+b}{2}\right)^2 - \left(\frac{b-a}{2}\right)^2 \overset{\text{expandindo os produtos notáveis}}{=} a.b = n$$

Como n é ímpar, a e b também são, logo $(a + b)$ e $(a - b)$ são pares e portanto x e y são inteiros. Cabe agora fazer a consideração caso:

i. n seja primo;
ii. n seja composto

Caso n seja primo, a única possibilidade é que $a = 1$ e $b = n$ e com isso $x = \frac{1+n}{2}$ e este será o pior caso[10], logo o algoritmo para quando x alcança este valor.

Caso n seja composto, primeiramente é possível que $a = b$ nesse caso, no primeiro ciclo x será fator de n, pois o valor inicial de x é $\lfloor\sqrt{n}\rfloor$, caso contrário tem-se que $1 < a < b < n$. O que remete que o ponto de parada do algoritmo será antes de $x = \frac{n+1}{2}$, caso se prove que $\lfloor\sqrt{n}\rfloor \leq \frac{a+b}{2} < \frac{n+1}{2}$. À direita desta desigualdade, tem-se que

[9] Pois se n for par, 2 é um de seus fatores.
[10] É dito isso, pois será o caso que exigirá o maior número de ciclos do algoritmo.

$$a + b < n + 1 \overset{n=ab}{\Longrightarrow} a + b < ab + 1 \Rightarrow a + b - (b + 1) < ab + 1 - (b + 1) \Rightarrow$$

$$\Rightarrow a - 1 < ab - b \Rightarrow a - 1 < b(a - 1) \overset{a>1}{\Longrightarrow} 1 < b$$

Logo prova-se a parte direita da desigualdade, para provar a desigualdade à esquerda observe que $\lfloor \sqrt{n} \rfloor \leq \sqrt{n}$, assim basta provar que $\sqrt{n} \leq \frac{a+b}{2} \Rightarrow n \leq \left(\frac{a+b}{2}\right)^2$, mas como dito anteriormente,

$$x^2 - y^2 = \left(\frac{a+b}{2}\right)^2 - \left(\frac{b-a}{2}\right)^2 = n \Rightarrow \left(\frac{a+b}{2}\right)^2 - n = \left(\frac{b-a}{2}\right)^2 > 0$$

Portanto está provada a desigualdade à esquerda. E com isso pode-se garantir que o algoritmo sempre para antes de x chegar a $\frac{n+1}{2}$ quando n for composto. E para em $x = \frac{n+1}{2}$ quando n for primo.

Em se tratando de segurança da informação, este algoritmo poderia facilitar o descobrimento da decomposição de uma chave pública, caso esta seja composta por dois primos próximos um do outro. Segundo Coutinho (2013, p. 43)

> Este algoritmo nos diz que uma coisa muito importante sobre o RSA. Lembre-se que a segurança do RSA depende da dificuldade de fatorar a chave pública n, que é igual ao produto de dois primos. A primeira impressão é que basta escolher os primos grandes, para garantir que n é difícil de fatorar. Mas isto *não* é verdade. Por exemplo, se escolhermos primos grandes, mas próximos, então n é facilmente fatorável pelo algoritmo de Fermat.

Com isso, pode-se perceber um salto de qualidade do primeiro algoritmo de fatoração para este último. Este salto se traduz também em eficiência, pois a velocidade de fatoração para números grandes é maior que no anterior.

A seguir serão tratados sobre dois grupos notáveis de números: Números de Mersenne e Números de Fermat. E posteriormente serão tratados métodos de fatorar esses números ou testar sua primalidade, no entanto estes resultados não serão demonstrados, visto que exigem conhecimentos matemáticos além do objetivo deste trabalho.

Os números de Mersenne e Fermat conjugam uma propriedade comum, pois são da forma $2^n \pm 1$, talvez não por acaso, uma vez que Mersenne e Fermat eram contemporâneos[11].

Uma curiosidade histórica é que apesar de seus trabalhos serem fruto de intensa pesquisa matemática, nenhum dos dois era matemático. Mersenne era um monge francês, no entanto, envolveu-se com ciência e promoveu a divulgação científica em uma época em que a mesma não existia formalmente. Já para o advogado Fermat, a matemática era tida como lazer, ele contribuiu não só para a Teoria dos Números, como para o Cálculo Diferencial[12], Geometria Analítica e para a Teoria da Probabilidade.

Os números de Mersenne são abordados devido sua importância histórica e sua relação com os números perfeitos. Eles ganharam notoriedade quando Mersenne afirmou que sejam os números $M_p = 2^p - 1$, com p primo, em que $p \in \{2, 3, 5, 7, 13, 17, 19, 31, 67, 127, \text{e } 257\}$ eram todos primos, é evidente que dado o contexto histórico, a falta de uma justificativa e a impossibilidade de testar sua veracidade, a afirmação gerou grande repercussão, no entanto, existem alguns erros na afirmação de Mersenne, que foi corrigida posteriormente[13] ficando a lista formada por $p \in \{2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107 \text{ e } 127\}$.

O maior primo de Mersenne conhecido é $M_{57885161}$ que tem 17425170 algarismos descoberto em 2013, já o maior composto é $M_{183027 \times 2^{265440} - 1}$ descoberto em 2010.

Segundo Martinez *et al.* (2013), os dez maiores primos conhecidos são números de Mersenne, sendo o menor deles $M_{13466917}$, com 4053946 algarismos. Assim, esta é uma frutífera fonte de primos grandes e sabendo que a aplicação em segurança de chave pública é essencial, há instituições que de forma colaborativa buscam novos primos de Mersenne, contando com processamento de computadores pessoais espalhados pelo globo terrestre.

Dada a breve descrição sobre os números de Mersenne serão apresentados duas ferramentas para o estudo de sua primalidade ou descoberta de fatores de M_q. A primeira é o Método de Fermat (COUTINHO, 2013, p. 157) que possibilita encontrar fatores de M_q e o Teste de Lucas-Lehmer[14] (RIBENBOIM, 2012, p. 75 ou COUTINHO, 2013, p. 162) o qual avalia a primalidade de M_q, porém não encontra fatores do mesmo.

[11] Século XVII
[12] Com os infinitesimais, mesmo antes do nascimento de Newton.
[13] Por diversos matemáticos, até o início do século XX.
[14] Também denominado critério de Lucas-Lehmer.

O **Método de Fermat** diz que seja $p \neq 2$ um primo e q um fator M_p. Então $q = 1 + 2rp$ para algum inteiro positivo r. O Método de Fermat é eficiente para números de Mersenne não tão grandes. A demonstração desse método depende de alguns conceitos de: aritmética modular, teorema de Lagrange e ordem. Por este motivo será apresentado um exemplo para facilitar a compreensão do método.

Neste exemplo, buscar-se-á fatores de M_{11} que, como apontado, é composto. Inicialmente, como visto, se um inteiro positivo n é composto ele tem um fator primo $q \leq \sqrt{n}$, assim se M_p for composto, ele tem um fator primo menor ou igual a $\sqrt{M_p}$. Logo

$$q = 1 + 2rp \leq \sqrt{M_p}.$$

Como

$$\sqrt{M_p} = \sqrt{2^p - 1} < 2^{p/2}.$$

Então, por transitividade,

$$q = 1 + 2rp \leq \sqrt{M_p} = \sqrt{2^p - 1} < 2^{p/2} \Rightarrow 1 + 2rp < 2^{p/2} \Rightarrow r < \frac{2^{p/2} - 1}{2p}.$$

Assim, usando $p = 11$, tem-se que $r < \frac{2^{11/2}-1}{2.11}$, portanto $r < 2$, logo resta apenas $r = 1$, pois r é um inteiro positivo e $q = 1 + 2.1.11 = 23$ é um fator de M_{11}, o outro fator é 89.

Dado o exemplo é possível apresentar um algoritmo que se possa implementar em uma linguagem de programação.

Entrada: inteiro positivo ímpar primo p.

Saída: um fator de M_p ou uma mensagem indicando que M_p é primo.

Etapa 1: Comece com $r = 1$; se $q = (1 + 2r.p)|M_p$ então q é fator de M_p e pode-se parar.

Etapa 2: Caso contrário incremente r de uma unidade e verifique se $q = (1 + 2r.p)|M_p$.

Etapa 3: Repita a Etapa 2 até que $q = (1 + 2r.p)|M_p$, ou enquanto r for menor que $\frac{2^{p/2}-1}{2p}$: no primeiro caso q é fator de M_p, no segundo M_p é primo.

Como dito, esse processo é útil para valores relativamente pequenos de M_p[15], assim é necessário verificar a primalidade de forma mais eficiente para valores maiores. Nesse sentido, surge o **Teste de Lucas-Lehmer** que depende de uma sequência definida recursivamente por $S_{k+1} = S_k^2 - 2$, com $S_0 = 4$, agora pode surgir questionamentos sobre o cálculo dessa sequência recursivamente, visto que k será um primo relativamente grande e portanto o sistema necessitará de um alto número de cálculos de multiplicações e subtrações. Nesse sentido é que há uma distinção entre computação e matemática pois a segunda visa substituir métodos de "força bruta", da primeira, "[...] por maneiras indiretas de proceder, que produzam o resultado com um mínimo de cálculos. É claro que as duas são, na verdade, faces da mesma moeda" (COUTINHO, 2013, p. 91).

Assim, pode obter $S_n = \left(2 + \sqrt{3}\right)^{2^n} + \left(2 - \sqrt{3}\right)^{2^n}$, a prova é facilmente feita por indução em n. Agora é possível enunciar o **Teorema 14**[16]: (Teste de Lucas-Lehmer) Seja S_k a sequência definida por $S_0 = 4$, $S_{k+1} = S_k^2 - 2$ para todo inteiro não-negativo k. Seja $p > 2$ primo; $M_p = 2^p - 1$ é primo se, e somente se, S_{p-2} é múltiplo de M_p.

A partir do teorema 14 é possível construir um algoritmo simples, porém dependendo de p, demanda grande tempo para executar e que diga se dado número de Mersenne é primo ou composto, como segue:

Entrada: inteiro positivo ímpar primo p.

Saída: uma mensagem indicando se M_p é primo ou composto.

Etapa 1A: Comece com $S_0 = 4$ e calcule $S_n = S_{n-1}^2 - 2$, até $n = p - 2$.

Etapa 1B: Calcule $S_{p-2} = \left(2 + \sqrt{3}\right)^{2^{p-2}} + \left(2 - \sqrt{3}\right)^{2^{p-2}}$.

Etapa 2: Verifique se $M_p | S_{p-2}$, caso afirmativo M_q é primo, caso contrário M_p é composto.

Nesse algoritmo a escolha pela etapa 1A ou 1B dependerá do computador e do programa que será implementado o algoritmo, pois caso opte pela etapa 1A a partir de $p = 23$, ou seja, em S_{21} a sequência tem 1199461 algarismos e que demandará certo tempo. Porém, caso a escolha seja pela etapa 1B, o problema será estouro de memória, uma vez que o

[15] O quão pequeno dependerá do equipamento a disposição para os cálculos.
[16] A demonstração deste teorema faz uso da teoria de grupo que foge ao escopo do trabalho, porém em Martinez *et al.*(2013, p. 347), é possível encontrar uma demonstração deste fato.

programa estará atuando com números do tipo $float$[17] consumindo maior memória e portanto tendo menor capacidade de armazenamento pelo computador que limita em p ainda menores.

Os números de Fermat, por sua vez, são da forma $F_n = 2^{2^n} + 1$, como são duplamente exponenciais, crescem rapidamente. Por este motivo, caso esta sequência tivesse uma infinidade de números primos seriam extremamente grandes. Assim como sugeriu Fermat, no entanto, talvez por um erro de indução, Fermat afirmou que os números desta forma seriam todos primos para $n \geq 0$, o que é verdade para $n = 0, 1, 2, 3, 4$, porém somente 90 anos após a conjectura (em 1730), Euler provou que $F_5 = 2^{2^5} + 1 = 4294967297 = 641 \times 6700417$, logo composto. Apesar de Euler ter usado algo semelhante ao Método de Fermat[18] para decompor F_5.

Em 1880, Landry fatorou como:

$$F_6 = 18446744073709551617 = 274177 \times 67280421310721$$

E com isso, diversos outros números de Fermat foram decompostos por completo, são eles F_n com $n = 7, 8, 9, 10, 11$. Já os números de Fermat parcialmente decompostos são: F_n com $n = 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23$. Apesar de saberem que F_{20} e F_{24} serem compostos, ainda não se conhece nenhum de seus fatores.

Assim, podem surgir questionamentos a respeito da necessidade de estudar tal grupo de números, porém apesar de não se conhecer outros números de Fermat primos para $n \geq 5$, eles geram uma série de problemas ainda não resolvidos e como aponta Coutinho (2013).

> Desenvolver uma teoria geral, da qual problemas específicos são casos particulares é um princípio fundamental em matemática. Desta forma percebemos semelhanças entre resultados que pareciam nada ter em comum. A partir destas semelhanças somos muitas vezes levados a aplicar o mesmo tipo de técnica a problemas que antes pareciam inacessíveis. É por isso que estamos sempre buscando generalizar: para entender melhor e assim enxergar mais longe (p. 58-59).

Algumas questões relacionadas aos números de Fermat são:

[17] De ponto flutuante, equivalente aos reais.
[18] Usado para descobrir fatores dos números de Mersenne.

- Existe uma infinidade de números de Fermat primos? Esta questão ganhou importância devido que, em 1801, "Gauss mostrou que, para que um polígono regular de n lados possa ser construído com *régua e compasso*, é preciso que n seja o produto de uma potência de 2 por números de Fermat *primos* (*idem*, p.160);
- Existe uma infinidade de números de Fermat compostos?

Além disso, os números de Fermat são ótimos para se testar lógica e aritmética de supercomputadores, bem como para testar novos algoritmos de fatoração. E ainda, só o número de matemáticos excepcionais que se envolveram em estudos sobre esses números já seria motivador para discutir sobre os mesmos.

A seguir serão tratadas duas ferramentas matemáticas: uma para encontrar fatores dos números de Fermat (Método de Euler) e outra para testar a primalidade desses números (Teste de Pépin). Novamente recai-se sobre o fato que estas ferramentas demandam conhecimentos que fogem ao escopo do trabalho, assim serão enunciadas formalmente e apresentadas seus respectivos algoritmos para implementação dos mesmos.

O **Método de Euler** diz que se q é um fator primo de F_n $(n > 1)$ então existe um número inteiro positivo r tal que $q = 1 + r.2^{n+2}$, apesar de o método não ser eficiente para F_n com n grande[19], ele é simples, pois como r é um inteiro positivo e $q < \sqrt{2^{2^n} + 1}$, logo

$$1 + r.2^{n+2} < \sqrt{2^{2^n} + 1} \Rightarrow r < \frac{\sqrt{2^{2^n} + 1} - 1}{2^{n+2}}.$$

Desta forma, verifica-se facilmente a primalidade de F_2, F_3 e F_4, pois para $n = 2$ e $n = 3$, r deveria ser negativo e para $n = 4$, r deveria ser menor 3, como $r = 1$ ou $r = 2$, q não seria fator de F_4, deduz-se que F_4 é primo.

Agora será apresentado o algoritmo deste método:

Entrada[20]: inteiro positivo $n > 1$.
Saída: um fator de F_n ou uma mensagem indicando que F_n é primo.
Etapa 1: Comece com $r = 1$.
Etapa 2: Se $(1 + r.2^{n+2}) | F_n$ então r é fator de F_n.
Etapa 3: Caso contrário incremente r de uma unidade e repita a Etapa 2.

[19] A partir de $n = 8$, já há uma demora considerável em um computador pessoal.
[20] Por simplicidade, este algoritmo não testa se $(1 + r.2^{n+2})$ é primo.

Etapa 4: Repita a Etapa 3 até encontrar um valor de r tal que $(1 + r.2^{n+2})|F_n$, ou enquanto r seja menor que $\frac{\sqrt{2^{2^n}}-1}{2^{n+2}}$: no primeiro caso F_n tem fator $(1 + r.2^{n+2})$, no segundo F_n é primo.

Como dito, este método deixa de ser eficiente a partir de $n = 8$[21], assim, testar a primalidade de números com crescimento tão rápido exige métodos mais eficientes, porém exigem ferramentas matemáticas mais poderosas e necessariamente mais complexas, especificamente para os números de Fermat existe o denominado **Teste de Pépin** que pode ser enunciado como: seja $F_n = 2^{2^n} + 1$; F_n é primo se, e somente se,

$k^{(F_n-1)/2} \equiv -1 \pmod{F_n}$, ou seja, $F_n | k^{(F_n-1)/2} + 1$, onde k é necessariamente 3, 5 ou 10.

Apesar da praticidade do teste de Pépin, ele não revela nenhum fator de F_n.

Como pode ser visto no enunciado do teste ele usa recurso de aritmética modular, que torna sua compreensão de teste mais simples, porém para as pessoas que não são conhecedoras desta notação, pode-se enunciar como segue: seja $F_n = 2^{2^n} + 1$; F_n é primo se, e somente se, o resto da divisão de $k^{(F_n-1)/2}+1$ por F_n for igual a 0 (zero), onde k é necessariamente 3, 5 ou 10. A demonstração desse e de outros fatos podem ser encontradas em Martinez *et. al* (2013). E a partir dessas considerações, determinar a primalidade de F_n é facilmente verificável, de fato, porém não se pode esquecer que o crescimento dos números com essas características é exponencial "duplamente" que faz com que $k^{(F_n-1)/2}$ seja um número imensamente grande, assim, apesar de o método ser simples, executá-lo pode ser demorado.

Até o momento foram discutidos inúmeros fatos sobre números primos e assuntos correlatos e não foi dito a resposta a seguinte pergunta: o conjunto dos números primos é finito ou infinito? Euclides já sabia a reposta e apresentou um argumento simples, mas irrefutável para esta pergunta, no livro IX de seu compêndio, sendo a proposição 20. Ele afirmou que o conjunto dos números primos é infinito e este será o último teorema deste trabalho e será demonstrado como Euclides fez em Os Elementos (2009) e com argumentos similares, porém suscintamente, apesar de se conhecerem diversas demonstrações, com os mais diversos níveis de complexidade.

Teorema 15: Existe uma infinidade de números primos.

[21] Para os parâmetros de um computador pessoal que é o foco da discussão apresentada.

Dem.: Suponha que $\{p_1,\ p_2,\ p_3,\ \cdots, p_r\}$ seja o conjunto de todos os r primos existentes. Faça $P = p_1 \cdot p_2 \cdot p_3 \cdot \cdots \cdot p_r + 1$ e seja p um número primo tal que $p|P$. Esse número p não pode ser igual a qualquer um dos p_i $(i = 1, 2, \cdots, r)$ primos, pois então $p_i|(P - p_1 \cdot p_2 \cdot p_3 \cdot \cdots \cdot p_r) = 1$, o que é impossível. Assim p é um número primo que não pertence ao conjunto e, por consequência $\{p_1,\ p_2,\ p_3,\ \cdots, p_r\}$ não pode ser o conjunto de todos os primos. Portanto o conjunto dos primos é dado por $\{p_1,\ p_2,\ p_3,\ \cdots, p_n, \cdots\}$.

Teorema 15: Versão de **Euclides (2009)**: "Os números primos são mais numerosos do que toda quantidade que tenha sido proposta de números primos"(p. 342).

Dem. (EUCLIDES, 2009, p. 342)**:**

> Sejam os números primos que tenham sido propostos A, B, C; digo que os números primos são mais numerosos do que os A, B, C.
> Fique, pois, tomado o menor medido pelos A, B, C e seja o DE, e fique acrescida a unidade DF ao DE. Então, o EF ou é primo ou não. Primeiramente, seja primo; portanto, os números primos A, B, C, EF achados são mais numerosos do que os A, B, C.
> Mas, então, não seja primo o EF; portanto, é medido por algum número primo. Seja medido pelo primo G; digo que G não é o mesmo que algum dos A, B, C. Pois, se possível, seja. Mas os A, B, C medem o DE; portanto, o G também medirá o DE. E Também mede o EF; e o G, sendo um número, medirá a unidade DF restante; o que é absurdo. Portanto, o G não é o mesmo que algum dos A, B, C. E foi suposto primo. Portanto, os números primos achados, A, B, C, G são mais numerosos do que a quantidade que tenha sido proposta dos A, B, C; o que era preciso provar .

Com início a mais de dois milênios, a discussão sobre números primos e fatoração está longe de ser completamente explorada de forma satisfatória, tendo ainda diversos problemas por serem solucionados. Um dos problemas que geraram uma grande discussão foi verificar se há algum teste determinístico de primalidade que tenha custo em tempo polinomial. Esse problema foi solucionado em 2002 por um grupo de pesquisadores do *Indian Institute of Technology* (Manindra Agrawal, Neeraj Kayal e Nitin Saxena) ao determinar um algoritmo provando que o problema do teste de primalidade pertencia a classe $\mathcal{P}$.

> [...] o algoritmo decide se N é ou não primo em tempo que é da ordem de um polinômio com relação ao número de bits $k = \log_2 N$ da entrada. Em particular o algoritmo não executa nenhuma escolha aleatória como fazem todos os algoritmos eficientes conhecidos até então. Esta descoberta deixou a comunidade de cientistas da área surpresos pelo fato de que, não apenas esse algoritmo resolve um problema

> de longa data, ele também o faz de uma maneira brilhantemente simples (MARTINEZ et al., 2013, p. 331).

Este algoritmo é denominado AKS, devido a seus criadores, e apesar do apontamento anterior, não chega a ser tão simples que se alinhe ao foco deste trabalho. Desta forma, talvez seja interessante ao professor que se valer desta pesquisa, implementá-lo, há diversas fontes que apresentam como ou consultar a referência anterior que apresenta o algoritmo e seu requisito matemático.

3 PROCEDIMENTOS METODOLÓGICOS

Neste tópico, serão abordados os procedimentos pelos quais resultou esta pesquisa, bem como as ferramentas computacionais que serão utilizadas no desenvolvimento da sequência didática de acordo com a teoria de base.

3.1 LEVANTAMENTO BIBLIOGRÁFICO

O primeiro passo da pesquisa, dada a delimitação do tema, foi identificar se havia algum trabalho que fazia uso de temas relacionados à teoria dos números e computadores para ensino de conceitos matemáticos como: variável, função e demais elementos presentes no tópico anterior. Foram pesquisados o banco de teses e dissertações da CAPES, a Biblioteca Digital do PROFMAT e a Biblioteca Digital Brasileira de Teses e Dissertações (BDTD).

A pesquisa na BDTD resultou nos trabalhos de: Gatti (2009), Barbosa (2008), Okumura (2014) e Siqueira (2012).

Por sua vez, com a pesquisa no banco de teses CAPES foi possível identificar os trabalhos de: Olgin (2011), Martins (2012), Ramos (2011), Vecchia (2012) e Forigo (2012).

E por fim na busca realizada na Biblioteca Digital do PROFMAT, onde se encontram os trabalhos concluídos relacionados ao PROFMAT, tem-se identificado os resultados provenientes de: Cangussú (2013), Dias (2013), Sant'Anna (2013), Barbosa Júnior (2013), Oliveira (2013), Esquinca (2013), Carvalho (2013), Spenthof (2013), Santos (2013), Marques (2013), Luz (2013) e Daineze (2013). Sendo que, com exceção de Cangussú (2013), os demais fazem uma análise aplicada a educação básica de tópicos da Matemática relacionados a divisibilidade, números primos, criptografia, aritmética modular e congruências.

A seguir estão, de forma sucinta, resumos de todas as teses e dissertações encontradas e citadas anteriormente.

Gatti (2009) faz uma pesquisa focada no estudo da programação por meio de modelagem e nesse sentido o papel da aprendizagem matemática ficou em segundo plano, concentrando os estudos no desenvolvimento de práticas de programação e construção de algoritmos e suas implementações para alunos de um curso de ciência da computação.

Barbosa (2008) faz uma análise sobre atividades com alunos do 6º ano para estudos sobre o Teorema Fundamental da Aritmética e relações com outros temas, como: números primos, m.m.c. e m.d.c.

Okumura (2014) aborda a questão de números primos, fatoração, criptografia com algumas atividades voltadas para o ensino fundamental. Apesar deste trabalho ser proveniente do PROFMAT, o mesmo ainda não se encontrava na base de dados do programa até a última busca.

Siqueira (2012) discute a criação de algoritmos para a resolução de funções polinomiais do 2º grau utilizando-se de programação com o Visualg 2.0 Os resultados apresentados foram satisfatórios, segundo o autor, apontando para uma melhoria da aprendizagem de resolução do problema.

Olgin (2011) desenvolveu sua pesquisa com alunos do 3º ano do ensino médio no qual buscou desenvolver uma série de atividades para relacionar temas do ensino médio e criptografia, a qual aponta ser uma boa alternativa para desenvolver com o nível escolar estudado.

Martins (2012) pesquisou, por meio de oficinas, de que forma a programação usando o *Scratch* pode desenvolver o pensamento criativo em alunos do 6º ano do ensino fundamental. Nesta linha, a estrutura foi construída com etapas desde a ambientação dos alunos ao *Scratch* a momentos em que lhes eram atribuídas tarefas a serem cumpridas em duplas. Ainda segundo o autor, o resultado foi promissor podendo ser estendido a um grupo maior de estudantes dado o potencial do desenvolvimento dos alunos.

Ramos (2011) busca analisar em que medida a produção de jogos são recursos para o desenvolvimento das ideias do raciocínio lógico e lógica computacional com alunos pré-adolescentes à universitários, utilizando-se da linguagem *Action Script 2* nativo do *Flash*. Segundo Ramos (2011) a heterogeneidade do grupo criou dificuldades na evolução da pesquisa que pode ser melhorada, mas resultou em muitos pontos positivos, como interesse dos alunos pelo estudo da lógica.

Vecchia (2012) aborda um estudo com alunos de um curso de Licenciatura em Matemática, relacionando modelagem matemática e desenvolvimento de jogos com o *Scratch* e criação de imagens com o *Autodesk 3ds*, no qual foram abordadas temáticas do Cálculo Diferencial e Integral Equações Diferenciais. Segundo a autora, esses modelos são denominados matemáticos/tecnológicos, pois se diferenciam dos modelos matemáticos ao incorporarem recursos audiovisuais entre outros. E por fim a autora aponta que o método abre um grande potencial de aprendizagem e pesquisa.

Forigo (2012) aborda a resolução de problemas da geometria espacial com a construção e implementação de algoritmos em uma linguagem de programação denominada Pascalzim, no qual busca desenvolver habilidades com fórmulas junto a alunos do terceiro ano do ensino médio, os quais interagiram e aumentaram sua motivação e aprendizagem do conteúdo.

Cangussú (2013) aborda o uso da linguagem *Basic* para o estudo de sequências junto a alunos do ensino médio, com o intuito de extrapolar o que é previsto nos livros didáticos e melhorar a capacidade dos alunos de reconhecer padrões.

Como é possível notar, dentre os trabalhos citados nenhum relaciona tópicos de teoria dos números e programação, sendo assim este trabalho tem um diferencial em relação aos demais e pode muito bem ser combinado com alguns dos mesmos para ampliar o âmbito de ação ou o foco do processo de ensino e aprendizagem.

3.2 FERRAMENTAS COMPUTACIONAIS

Em um trabalho cujo fim seja utilizar programação de computadores para ensino de Matemática, escolher as ferramentas adequadas pode ser uma árdua tarefa, por este motivo, analisou-se primeiro a questão de abrangência, pois deviam ser ferramentas disponíveis para quaisquer plataformas (*Windows*, *Linux* ou *Mac*). Em segundo lugar deveria ser gratuita e por fim de fácil aprendizagem, pois a ferramenta não pode ser um empecilho para o desenvolvimento da sequência que está focada na Matemática.

Ao considerar esses critérios chegou-se a conclusão que a melhor opção seria o *Scratch*, pois é multiplataforma, gratuito e desenvolvido inclusive para crianças. No entanto, o

Scratch tem algumas limitações do ponto de vista de uma linguagem de programação, não sendo eficiente para algoritmos que demandam mais tempo de processamento. Assim, foi pensada uma alternativa e foi definida a linguagem *Python*, pois usa uma sintaxe similar a linguagem matemática e dispensa a declaração de variáveis, como: *C* ou *Java* por exemplo.

Um dos aspectos, dessas opções, é que para os usuários do sistema *Linux*, ou as escolas que receberam computadores do governo federal, tem essa linguagem instalada por padrão o que garante a possibilidade de uso da sequência por professores das redes públicas de ensino.

A seguir serão apresentadas algumas características dessas linguagens para facilitar seu uso pelos demais professores.

3.2.1 *Scratch*

O *Scratch* é uma linguagem de programação que segue a mesma filosofia do *Logo*, que é uma linguagem de programação procedural derivada do *LISP*[22]e foi desenvolvido por Papert em decorrência de sua pesquisa sobre o construcionismo, no final da década de 60 no MIT[23].

A princípio, não continha uma interface gráfica, devido ao desenvolvimento tecnológico da época, posteriormente foi implementada uma interface robótica em forma de tartaruga que respondia aos comandos do usuário, porém esse artefato tecnológico demandava uma estrutura que a tornava limitada a poucas instituições, além do mais, devido aos sistemas mecânicos a "tartaruga" apresentava alguns resultados com erro, o que poderia levar o aprendiz a formar conceitos errôneos em seu sistema cognitivo.

Com o desenvolvimento tecnológico e a redução de custos dos equipamentos a interface passou a ser o monitor, o que não fez com que perdesse a característica da tartaruga, pois na versão com interface gráfica os comandos eram dados a uma tartaruga digital.

---

[22] O nome vem de ListProcessing, pois a lista é a estrutura de dados fundamental desta linguagem, utilizada nas décadas de 70 e 80 pelas comunidades de inteligência artificial, um dos paradigmas considerados por Papert ao conceber o *Logo*. Nesta linguagem o próprio código fonte pode ser entendido como um dado a ser manipulado o que a torna tão flexível, do ponto de vista de aplicação.

[23] Massachusetts Institute of Tecnology (Instituto de Tecnologia de Massachusetts).

A linguagem *Logo* é composta de comandos de movimento, giro, variáveis, estruturas condicionais e diversas outras características de uma linguagem de programação. E entre outras pesquisas, a conduzida por Motta e Silveira (2010) aponta que com a utilização desse sistema foi alcançado um objetivo além do esperado que extrapolou o conhecimento matemático, pois desenvolveu nos alunos aspectos de autonomia e habilidade de refletir sobre as interações que executaram.

Pois,

> O contato do aluno com o SuperLogo contribuiu para o desenvolvimento de conceitos geométricos, por meio de uma matemática própria, presente no ambiente de aprendizagem interagindo com a geometria da tartaruga. Tal interação desenvolveu um modelo de pensamento reflexivo, no qual o conhecimento é obtido por meio de manipulações do próprio aluno, ou seja, em um processo de construção da aprendizagem (ibidem, p. 124).

Porém, o sistema apresenta alguns empecilhos, como:

i. o programa não permite a publicação de projetos em páginas *web*, com isso as escolas tem dificuldade de expor seu trabalho, pois apenas expõe seu trabalho em forma de procedimentos que precisarão ser executados em um computador com o *Logo* para que possa ser visualizado, as versões de *Logo* para *Linux* são bem diferentes daquelas desenvolvidas para *Windows*, apesar de em essência terem o mesmo objetivo, não é possível utilizar o procedimento feito em um, no outro sistema operacional, a não ser que se utilize um emulador de *Windows* no *Linux*.
ii. e segundo Motta e Silveira (2010, p. 123-124),

> Alguns professores, ao trabalharem com o SuperLogo, afirmam que os alunos sentem-se cansados em interagir com o software e solicitam outro tipo de atividade. Esse cansaço deve-se provavelmente ao fato de se proporem atividades em que os alunos somente aplicam os comandos do programa, não interagindo entre si e nem desenvolvendo suas habilidades intelectuais. Alguns teóricos criticam o fato do SuperLogo desenvolver, desde as séries iniciais, uma linguagem de programação. [...]. Hoje, há os que defendem a posição de que os conceitos e generalizações de uma programação não devem ser ensinados a todos. No entanto, a experiência em programação é útil ao desenvolvimento do raciocínio lógico, pois o ato de programar exige sucessivas antecipações e projeções sobre os resultados das várias partes de um objeto simbólico.

No Brasil, a versão gratuita mais conhecida e atual é o *SuperLogo* 3.0, que é a versão original, traduzida para o português pela equipe do NIED (Núcleo de Informática Aplicada à Educação) da Universidade de Campinas (Unicamp).

Como dito, neste trabalho será utilizado como ambiente com a perspectiva do *Logo* o *Scratch*, uma vez que segue a mesma perspectiva da linguagem original, porém não é necessário digitar os comandos, pois nesse programa os comandos são como blocos de montar[24] que são agrupados e executados em certa ordem para obter o resultado esperado.

O *Scratch* também foi desenvolvido no MIT, porém pela equipe do Media Lab coordenada por Mitchel Resnick, em 2006. O pressuposto principal do programa é que seja fácil para todos, tanto que o *slogan* é: Imagine, Programe, Compartilhe.

O nome deriva do trabalho de *Disc Jockeys* (*DJs*), de "arranhar" (*scratch*) os discos, pois segundo a filosofia do *Scratch* de montar blocos (programar) que levem a um resultado, é similar ao trabalho do DJ ao mixar sons para gerar um novo.

O programa é gratuito e está sob a licença GPLv2[25] que é uma das licenças de *software* livre, portanto seu código é aberto e passível de alterações de acordo com a necessidade do usuário. Podendo inclusive disponibilizar sua versão para a comunidade, tanto gratuitamente quanto não, no entanto, deve manter os preceitos da licença GPLv2, ou seja, disponibilizar também o código fonte do programa.

O *software* está disponível para instalação em sistemas operacionais livres, como proprietários: *Linux*, *Windows* e *Mac OS*, o que gera um ganho em relação ao *Logo*, por possibilitar a migração de um projeto entre sistemas operacionais distintos sem alteração na estrutura.

O *Scratch*, bem como o *Logo*, pode ser usado com alunos desde os primeiros anos escolares até o nível universitário. De acordo com o nível de escolaridade e o projeto do docente o sistema possibilita a criação de jogos interativos (com objetivos de explorar o conhecimento planejado), por poder adicionar diversos personagens (*sprites*) e portanto duas ou mais pessoas podem interagir simultaneamente em um mesmo projeto.

---

[24] Uma alusão aos sistemas compostos de blocos de plástico (Lego) que eram utilizados em robótica juntamente com o sistema *LOGO*.
[25] Significa *General Public License* (Licença Pública Geral), versão 2.

O programa dispõe suas ferramentas em espaços separados por categoria e em cada uma dessas categorias suas respectivas ferramentas, as categorias são:

- Movimento (que contém as ferramentas relacionadas a andar, girar, dirigir-se a determinado ponto da tela, etc.);
- Aparência (que controla mudança de personagem, tamanho, apresentação textual etc.);
- Som (como sugere, controla os possíveis efeitos sonoros do projeto);
- Caneta (onde gerencia a possibilidade de rastro, cor do mesmo, espessura, etc.);
- Controle (comporta ferramentas de *loop*, condicional, acionamento por tecla, etc.);
- Sensores (espaço abrange recursos de comparação ou toque do *sprite* em algum elemento ou cor, recepção de comandos do teclado, etc.);
- Operadores (nesse espaço é possível fazer operações e comparação matemáticas, função randômica, concatenação de caracteres, etc.); e
- Variáveis (neste bloco é possível criar e manipular variáveis e listas).

A tela de apresentação do programa é como mostrada na Figura 3:

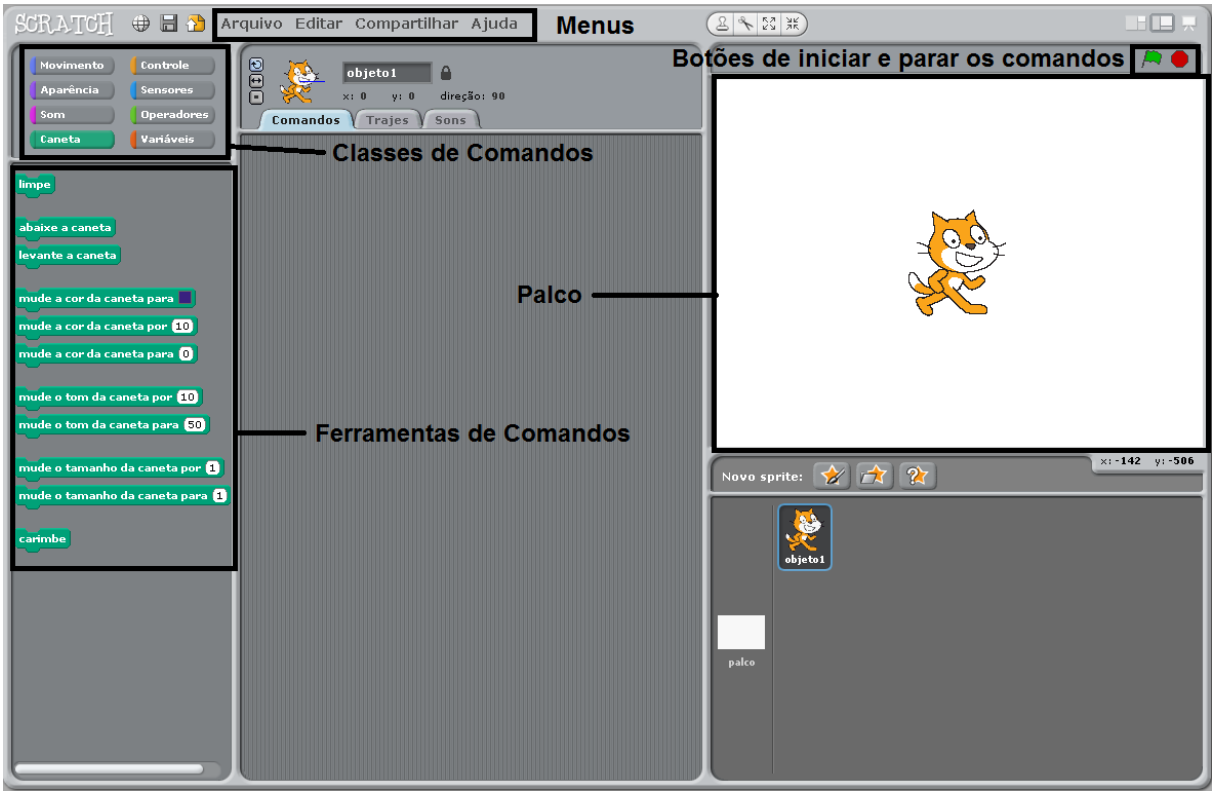

Figura 3: Apresentação do *Scratch*

O palco é onde são apresentados os resultados dos blocos de comandos executados no programa, ou seja, onde aparece o resultado do que foi programado e o aluno pode verificar e depurar seu projeto.

Com isso, o *Scratch* proporciona, além do desenvolvimento do raciocínio lógico, desenvolvimento da habilidade de dividir um problema em problemas menores, identificação e eliminação de erros, ainda proporciona o desenvolvimento de habilidades de concentração, perseverança, noções de programação, funcionamento de computadores e análise de *design* de interface (MARQUES, 2009). Ou seja, vai além do conhecimento, mas desenvolve habilidades que, talvez, não fossem possíveis de se fazer em um ambiente sem esta ferramenta.

3.2.2 *Python*

O *Python* é uma linguagem que está no auge no mundo do *software* livre e apresenta uma série de características que o faz muito atrativo para o ensino de programação, seu nome

foi inspirado em um grupo de humoristas britânicos chamado "*Monty Python*" e não devido ao réptil como se poderia imaginar. Segundo Duque (2011), todos deveriam conhecer a linguagem devidos suas qualidades e potenciais os quais proporcionam que o desenvolvimento de programas seja uma tarefa divertida. Por sua vez, Summerfield (2012) aponta que

> Python é, provavelmente, a linguagem de programação popular mais fácil e agradável de lidar. O código Python é simples para ler e escrever, e consegue ser conciso sem ser algo enigmático. Python é uma linguagem muito expressiva, o que significa que podemos normalmente escrever em muito menos linhas de código Python em comparação ao que seria necessário escrever em uma aplicação equivalente, digamos, em C++ ou Java (p. 1).

A linguagem é, também, "*de altíssimo nível [...] orientada a objeto, de tipagem dinâmica e forte, interpretada e interativa*" (BORGES, 2010, p. 13). Devido a suas características a linguagem é muito flexível, sendo utilizada em aplicações científicas, em jogos ou compondo aplicações que contam com outras linguagens de programação (como: *C* ou *Fortran*, geralmente utilizadas em aplicações científicas).

Uma das características que fizeram com que a linguagem *Python* fosse utilizada neste trabalho é o fato de ela possuir uma sintaxe clara (simples) e concisa, o que favorece a leitura do código fonte melhorando a produção do mesmo.

A linguagem foi desenvolvida em 1990 por Guido van Rossum[26], no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda e tinha como foco engenheiros, físicos e outras áreas científicas, o que a torna muito indicada para estes seguimentos até os dias atuais. De fato, Rossum, criou o *Python* depois de fazer parte de um projeto de desenvolvimento de uma linguagem de script denominada ABC, o que facilitou o desenvolvimento de sua linguagem por ter percebido as melhorias que poderia fazer a partir de sua experiência com a ABC.

O fato de funcionar em diversas plataformas (multiplataforma[27]) com raras ocasiões de adaptação o fez ganhar força para seu crescimento, aliado ao fato de ser multiparadigma, que se traduz no fato de ser possível desenvolver aplicações dinâmicas (tornando-se uma

[26] Também conhecido como BDFL (Benevolent Dictator for Life).
[27] Devido ao fato de ser uma linguagem interpretada.

calculadora avançada), modulares, funcionais ou orientadas a objetos, diferente de outras linguagens multiplataforma disponíveis.

Devido sua simplicidade e facilidade da leitura do código, é possível utilizá-lo como alternativa ao pseudocódigo[28], isso decorre pelo fato da estrutura do *Python* requerer que os blocos de comandos não sejam separados por sinais ou algum caractere, mas sim por sua indentação o que o torna mais legível, essa habilidade inclusive é desejada nas outras linguagens, porém como nelas não é obrigatório o aluno pode não desenvolvê-la

> Python utiliza tipagem dinâmica, o que significa que o tipo de uma variável é inserido pelo interpretador em tempo de execução (isto é conhecido como *Duck Typing*). No momento em que uma variável é criada através de atribuição, o interpretador define um tipo para a variável, com as operações que podem ser aplicadas (BORGES, 2010, p. 15).

Devido a isso não se declara variáveis no início do desenvolvimento que faz com que seja mais rápido, porém pode fazer com que o desenvolvedor (aluno) incorra em erros ao atribuir nomes diferentes para a mesma variável em partes distintas do código.

> A tipagem do Python é forte, ou seja, o interpretador verifica se as operações são válidas e não faz coerções automáticas entre tipos incompatíveis. Para realizar a operação entre tipos não compatíveis, é necessário converter explicitamente o tipo da variável ou variáveis antes da operação (BORGES, 2010, p. 16).

Nesse sentido, em nada é divergente da Matemática, pois a operação entre números de quaisquer conjuntos será executada, sempre apresentando a resposta no conjunto "maior" entre os envolvidos.

"O código Python pode ser escrito com qualquer editor de texto simples que possa carregar e salvar o texto usando uma das codificações de caracteres: ASCII ou Unicode UTF-8" (SUMMERFIELD, 2012, p. 7). No entanto existem editores especializados em código de programação com algumas funcionalidades adicionais, como: cor dos elementos do código (variáveis, números, comentários, etc.) ou exportam para outros formatos e fazem conversão

[28] É a forma de escrever um programa em língua materna, ao invés de uma linguagem de programação para depois traduzi-lo para a linguagem de desenvolvimento planejada.

de codificação de texto. Os arquivos de código tem como extensão *.py* ou *.pyw*, são exemplos de editores de código *Python*: *SciTE, Notepad++, Eclipse, Geany* ou mesmo o ambiente que acompanha o *Python* (IDLE), também denominado *shell*.

Como o objetivo deste trabalho é apresentar uma proposta que possa ser executada dando o mínimo de trabalho ao docente, será utilizado o próprio *shell* do *Python* como ambiente de desenvolvimento, o que evita com que o professor precise instalar outros *softwares*.

Este trabalho não tem o intuito de ser uma referência para a linguagem *Python* ou mesmo programação, por este motivo até o momento fez-se apenas uma explanação geral sobre o *Python* e serão apresentadas algumas características da sintaxe da linguagem e comandos que serão necessários para o desenvolvimento da sequência didática. O leitor interessado mais sobre o tema pode encontrar materiais de qualidade e gratuitamente na internet como Borges (2010) ou outras diversas referências na página http://www.python.org.br e também na documentação em português[29] em http://www.python.org.br/wiki/DocumentacaoPython ou em inglês https://www.python.org/doc/.

Entre outras coisas esta linguagem tem duas versões em desenvolvimento paralelo: A 2.x e a 3.x, sendo que ambas são funcionais e estão em desenvolvimento pela *Python Software Foundation*. A questão está centrada no fato de que o desenvolvimento caminha para somente a 3.x permanecer, no entanto uma série de bibliotecas só funcionam na versão 2.x, assim, ainda são mantidas as duas. Atualmente[30], as versões mais recentes disponíveis para *download* são: *Python* 2.7.6 e *Python* 3.4.0, é possível inclusive que o usuário instale ambas em seu computador sem prejuízos, porém o registro do sistema operacional deverá ser alterado para que uma ou outra tenha prioridade na execução dos arquivos de código-fonte.

Os próximos temas tratarão de estruturas da linguagem que serão utilizadas na sequência didática, devendo salientar que não faz parte do escopo deste projeto ser uma referência em *Python*, portanto serão utilizados somente elementos introdutórios para a discussão Matemática envolvida.

[29] Há um maior número de materiais em inglês.
[30] Em maio de 2014.

O *Python* oferece o conjunto padrão de operados de comparação binária: $<$ menor que, $<=$ menor que ou igual a, $==$ igual a, $!=$ não igual a (diferente de), $>=$ maior que ou igual a, $>$ maior que. Destas comparações binárias há o retorno de $True$ ou $False$ somente, conforme exemplos apresentados na Figura 4.

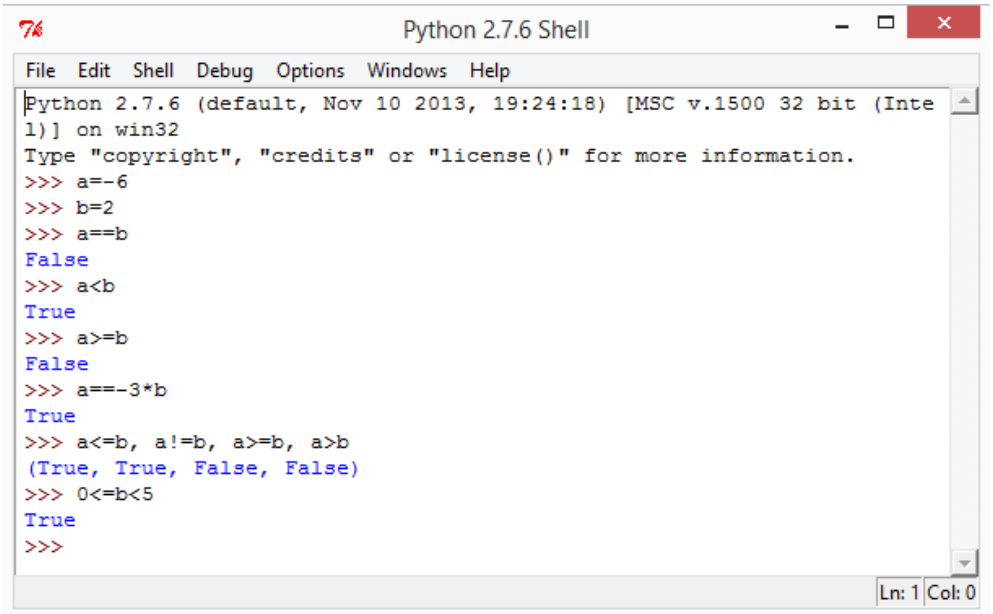

Figura 4: *Python* - operadores de comparação

A linguagem oferece um operador de adesão que será usado em casos de laços ou na manipulação de listas in ou sua negação $not\ in$. Este operador será usado quando forem apresentados os laços de controle de fluxo. Outros operadores que são usados nesses laços são os operadores lógicos que são: and, or ou not.

Os comandos para controle de fluxo são três: if, for e $while$, sendo os dois últimos também conhecidos como *loops*. Ao desenvolver um programa e executá-lo no computador a máquina o lê e o executa na mesma ordem em que os comandos foram inseridos, sendo assim, em alguns casos é necessário que a ordem não seja necessariamente esta e por isso é que existem os comandos para controle de fluxo que fazendo com que o comando seja executado em certas circunstâncias e em outras não.

A estrutura (sintaxe) do if é a seguinte:

$$
\begin{aligned}
&if\ expressão\_booleana\_1:\\
&\qquad opções\_a\_serem\_executadas\_1\\
&elif\ expressão\_booleana\_2:\\
&\qquad opções\_a\_serem\_executadas\_2\\
&\qquad\quad \vdots\\
&elif\ expressão\_booleana\_N:\\
&\qquad opções\_a\_serem\_executadas\_N
\end{aligned}
$$

$$else:$$
$$\quad opções\_a\_serem\_executadas$$

Nesta estrutura, somente os elementos das duas primeiras linhas são obrigatórios. A partir da versão 2.5 o *Python* também suporta a expressão:

$$variável = valor\_1\ if\ expressão\_booleana\ else\ valor\_2$$

Que seria equivalente a

$$if\ expressão\_booleana:$$
$$\quad variável = valor\_1$$
$$else:$$
$$\quad variável = valor\_2$$

"Diferentemente da maioria das outras linguagens de programação, o *Python* utiliza uma indentação para dar significado à suas estruturas de blocos" (SUMMERFIELD, 2012, p. 24). Além disso, é possível perceber que não existem parênteses ou chaves para determinar o início do bloco, mas sim ":". No exemplo a seguir (Figura 5) é fácil perceber essas características.

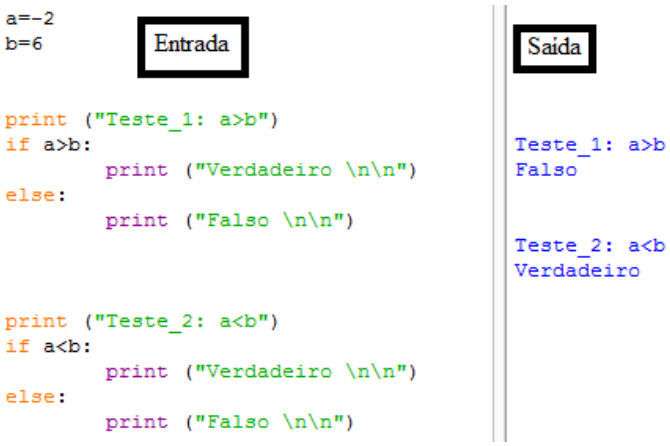

Figura 5: *Python* - Exemplo do uso do if.

O bloco *while* que pode ser traduzido como "enquanto". É utilizado para executar uma rotina zero ou mais vezes enquanto a condição booleana for verdadeira. Sua estrutura[31] é da forma:

$$while\ expressão\_booleana:$$
$$\quad opções\_a\_serem\_executadas$$

[31] Sua estrutura pode contar ainda com os comandos $continue$ e $break$, no entanto não serão abordados no trabalho.

Na Figura 6 serão impressos os números de 1 a 10 e é necessário verificar a que a condição booleana seja atendida em algum momento, pois caso contrário o programa entrará em um *loop* infinito e não encerrará sua execução até comando do usuário.

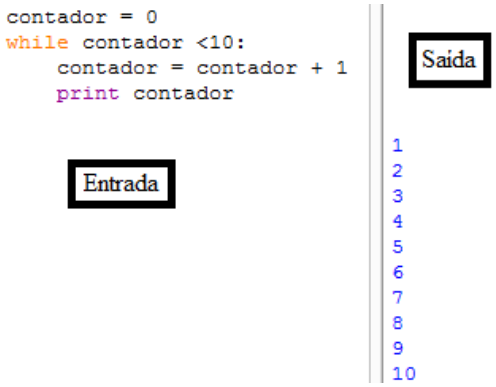

Figura 6: *Python* - exemplo do uso do *while*.

O segundo comando de *loop* é o *for*, o qual necessita da palavra-chave *in* para fornecer seu escopo de variação. A estrutura[32] do comando é dada por:

for variável in iterável[33]:
opções_a_serem_executadas

No exemplo de uso desta estrutura (Figura 7) o algoritmo decide se uma letra é vogal ou consoante.

[32] Sua estrutura, assim como no *while*, pode contar ainda com os comandos *continue* e *break*, no entanto não serão abordados no trabalho.
[33] Este elemento iterável pode ser uma lista, um texto ou uma sequência numérica proveniente do comando *range*.

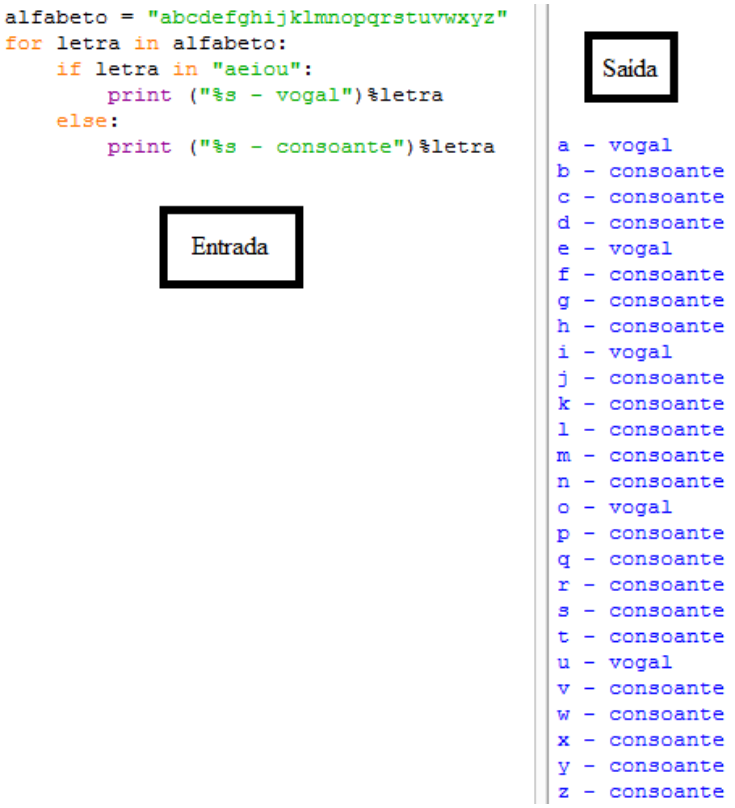

Figura 7: *Python* - exemplo do uso do for.

Como em qualquer outra linguagem, o *Python* possui os operadores aritméticos que se assemelham ao utilizado em Matemática (Tabela 1: *Python* - lista de operadores aritméticosTabela 1).

Operador	Função
+	Soma
−	Subtração
*	Multiplicação
/	Divisão
//	Divisão Inteira

Tabela 1: *Python* - lista de operadores aritméticos

Os três primeiros se comportam da mesma forma como em outras linguagens de programação, com exceção do fato de também poderem ser utilizados atributos incrementais, ou seja, ao invés de escrever $a = a + k$, ou $a = a - k$ ou $a = a * k$, onde k é uma constante, pode-se codificar respectivamente como: $a+= k$, ou $a-= k$ ou $a *= k$.

Independente de qual das forma usadas, isso vai de encontro ao senso matemático, nesse ponto cabe ao professor esclarecer essas diferenças a fim de não causar confusão entre as notações matemática e computacional.

A diferença em relação a divisão no *Python* é que ao usar $/$ obtém um número com sua parte decimal, enquanto a grande maioria das linguagens retorna um inteiro se a operação ocorrer entre dois inteiros e por esse motivo a existência de $//$ que retorna a parte inteira da divisão entre dois números, sejam eles inteiros ou não.

Além dos operadores expostos na Tabela 1, há ainda os apresentados na Tabela 2:

Operador / Função	**Objetivo**
$\boldsymbol{x\%y}$	Retorna o resto da divisão de x por y.
$\boldsymbol{x ** y}$	Retorna x^y.
$\boldsymbol{abs(x)}$	Retorna o valor absoluto de x ($\|x\|$).
$\boldsymbol{divmod(x,y)}$	Retorna os inteiros a e b que são, respectivamente, quociente e resto da divisão de x por y.
$\boldsymbol{pow(x,y)}$	Tem a mesma função que $x ** y$.

Tabela 2: *Python* - outros operadores e funções

O *Python* é extremamente flexível[34], no entanto parte de suas ferramentas não são nativas e são necessárias somente importar para fazer uso, ou em alguns casos fazer o *download* de do site oficial da linguagem ou de terceiros, esses itens são denominadas bibliotecas, para o presente trabalho serão utilizadas algumas dessas bibliotecas (ou módulos), especialmente *random* e *math*.

O módulo *random* fornece ferramentas para criar números aleatórios que serão utilizados para testar os algoritmos que serão criados e o módulo *math* apresenta algumas funções e constantes matemáticas. Para fazer uso dessas ferramentas é necessário incorporá-las ao projeto, assim o processo é feito ao inserir no programa "$import\ nome\_do\_módulo$", apesar de se poder inserir em qualquer parte, antes de usar algum item, é recomendado que se faça no início do projeto.

Praticamente todos os módulos em *Python* vem acompanhados da respectiva documentação, portanto é possível verificar quais são suas ferramentas e como utilizá-las.

A sequência didática versa sobre situações em subconjuntos dos números inteiros, no entanto, caso fosse utilizado o conjunto dos números reais, é de se levar em conta a questão

[34] Pois permite programação estruturada, orientada a objetos e orientada a funções.

do arredondamento, uma vez que as linguagens de programação (em sua grande maioria) trabalham com um subconjunto dos racionais, por este motivo a sequência foi pensada com algoritmos que não recaem sobre irracionais, a fim de evitar erros de arredondamento de ponto-flutuante.

Programadores, quando desenvolvem programas que manipularão dados em grandes quantidades, necessitam armazená-los para divulgar e/ou analisar, apesar desse não ser o intuito do trabalho, pode-se utilizar esse recurso como um meio do professor analisar os resultados obtidos pelos alunos e a averiguar seu desenvolvimento, ou mesmo para que os alunos façam análises de seus resultados e os comparem com o esperado.

Com isso, é possível utilizar diversos meios de armazenamento e entrada de dados, porém o mais indicado seria utilizar banco de dados, no entanto dada a possível complexidade, o mais adequado a situação é a leitura e escrita de arquivos de texto, assim o usuário pode consultar esses arquivos, inclusive, em equipamentos que não possuem nenhum *software* em específico instalado.

Para operar com leitura e escrita de arquivos é utilizado o comando $open()$, a seguir são apresentados exemplos de uso desse comando.

A fim de exemplificar foi criado um arquivo de texto simples com o conteúdo, conforme a Figura 8.

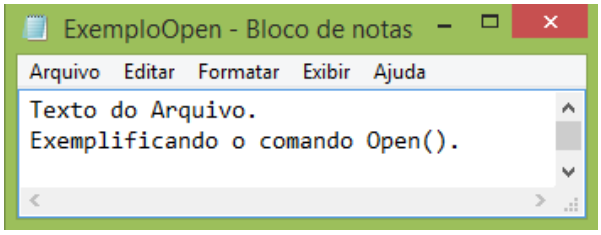

Figura 8: Exemplo de arquivo de texto a ser manipulado

Na Figura 9 é apresentado o exemplo de uso do *Python* para leitura do arquivo de texto: o comando $open()$, onde necessita do nome do arquivo (incluindo seu caminho n computador[35]) e o segundo argumento é a forma de acesso ao arquivo r, w, a, entre outras, que significam, respectivamente: leitura, escrita e adição. Assim no exemplo apresentado foi utilizado o r, uma vez que se desejava somente analisar o conteúdo do arquivo, o qual foi feito de três formas distintas, usando o comando:

[35] Neste caso não foi necessário, pois o arquivo de texto e o arquivo Python estavam na mesma pasta.

- $read()$, é apresentado o texto do arquivo na mesma estrutura como no arquivo de origem;
- $readline()$, é apresentado somente a primeira linha do arquivo, caso fosse necessário apresentar as demais linhas do arquivo, seria necessário repetir o comando, pois o mesmo apresenta apenas uma linha a cada execução e pula para a próxima;
- $readlines()$, é apresentado todo o conteúdo do arquivo, porém cada linha do mesmo é um item de uma lista (ou tupla), de forma que há uma ordem e se pode chamar cada uma delas individualmente e operar quando necessário.

O usuário fará uso de cada um desses modos de acordo com a necessidade de seu projeto, sendo assim não há como recomendar qual é a mais indicada.

```
# -*- coding: iso-8859-1 -*-

#Comando que faz a abertura do arquivo para leitura
arquivo = open("ExemploOpen.txt", "r")

print "01 - Tipo de leitura - read()\n"

#Imprime o conteúdo do arquivo.
print arquivo.read()

#Fecha o arquivo
arquivo.close()


#Comando que faz a abertura do arquivo para leitura
arquivo = open("ExemploOpen.txt", "r")

print "\n\n\n\n\n\n\n\n"
print "\n02 - Tipo de leitura - readline()\n"

#Imprime o conteúdo do arquivo.
print arquivo.readline()

#Fecha o arquivo
arquivo.close()


#Comando que faz a abertura do arquivo para leitura
arquivo = open("ExemploOpen.txt", "r")

print "\n\n\n\n\n\n\n\n"
print "\n03 - Tipo de leitura - readlines()\n"

#Imprime o conteúdo do arquivo.
print arquivo.readlines()

#Fecha o arquivo
arquivo.close()
```

```
01 - Tipo de leitura - read()

Texto do Arquivo.
Exemplificando o comando Open().


02 - Tipo de leitura - readline()

Texto do Arquivo.


03 - Tipo de leitura - readlines()

['Texto do Arquivo.\n', 'Exemplificando o comando Open().']
```

Figura 9: *Python* - abrindo arquivo para leitura.

No exemplo apresentado na Figura 10 são apresentados dois modos de abertura do mesmo arquivo anterior que possibilitam a edição do mesmo, para isso foram utilizados dois modos associados ao comando $open()$: w ou a. Como fica claro na figura, ao utilizar o a, o texto foi acrescido ao já existente, por sua vez ao reabrir o arquivo com o w, o texto existente no arquivo foi somente o escrito no exemplo, portanto todo o restante do texto foi apagado e esta é a diferença entre as opções w e a.

O comando $open()$ trata o arquivo sem a necessidade da abertura do mesmo, caso o arquivo indicado não exista o $open()$ o cria.

```
# -*- coding: iso-8859-1 -*-

print "\n\nUsando 'a'"
#Comando que faz a abertura do arquivo para escrita
arquivo = open("ExemploOpen.txt","a")

#Comando que adiciona texto ao final do arquivo
arquivo.write("Linha adicionada por write.\n")

#Fecha o arquivo
arquivo.close()

#Comando que faz a abertura do arquivo para escrita
arquivo = open("ExemploOpen.txt")

print arquivo.read()

#Fecha o arquivo
arquivo.close()

print "\n\n\n\n\n\n\n\n\n\n\n\nUsando 'w'"
#Comando que faz a abertura do arquivo para escrita
arquivo = open("ExemploOpen.txt","w")

#Comando que adiciona texto ao final do arquivo
arquivo.write("Linha adicionada por write.\n")

#Fecha o arquivo
arquivo.close()

#Comando que faz a abertura do arquivo para escrita
arquivo = open("ExemploOpen.txt")

print arquivo.read()

#Fecha o arquivo
arquivo.close()
```

```
Usando 'a'
Texto do Arquivo.
Exemplificando o comando Open().
Linha adicionada por write.
Usando 'w'
Linha adicionada por write.
```

Figura 10: *Python* - arquivo para escrita

Com isso, é possível alterar e armazenar dados e divulgá-los pelos alunos e/ou docentes no espaço educacional.

Para poder encerrar a discussão sobre o *Python*, será apresentado o conceito de funções nesta linguagem e que será usado na sequência didática, assim o professor pode correlacionar os conceitos matemáticos e computacionais e verificar suas similaridades e esclarecer para os alunos o conceito que muitos deixam a escola sem de fato o compreenderem.

Segundo Summerfield (2012, p. 161), "Funções [em programação] são recursos pelos quais podemos empacotar e parametrizar funcionalidades", a estrutura de funções no *Python* é da seguinte forma:

$$\text{def}\, nome\_função(parâmetros):$$
$$comandos$$

E esta estrutura será útil em alguns algoritmos nos quais serão usados uma mesma rotina diversas vezes em vez de reescrevê-lo todas as vezes que se fizerem necessários para utilizá-los.

Com isso, serão apresentados dois exemplos de funções no *Python*. No primeiro (Figura 11), o algoritmo recebe dois números e apresenta a média aritmética resultante enquanto no segundo (Figura 12) a entrada é um natural n e a saída será $n!$.

```
# -*- coding: iso-8859-1 -*-

#Definição da função
def media(a, b):
    media = (a+b)/2
    return media

print media(4, 6)
```

```
>>>
5
```

Figura 11: *Python* - exemplo de função média aritmética

```
# -*- coding: iso-8859-1 -*-

#Definição da função
def fatorial(n):
    if 0<=n<=1:
        return 1
    return n*fatorial(n-1)

print fatorial(10)
```

```
>>>
3628800
```

Figura 12: *Python* - exemplo da função fatorial

Nos dois exemplos é possível notar a presença dos comandos $print$ e $return$, o primeiro se deve a peculiaridade inerente as funções, elas só serão executadas se chamadas, o segundo é devido ao fato de se desejar utilizar o resultado que provém da função e portanto o comando $return$ apresenta esse resultado ao mecanismo que fez a chamada da função.

Analisando as funções é fácil perceber a simplicidade da primeira em relação a segunda, a função que calcula a média simplesmente soma os dois elementos e os divide retornando o valor desejado, por sua vez a função fatorial é enquadrada como uma função recursiva, pois usa a si mesmo no cálculo desejado, no entanto esse é um meio para facilitar e reduzir linhas de código, porém seria possível construir uma função que retorne o fatorial de um número sem utilizar recursividade.

Assim, a introdução ao tema de programação em *Python* está encerrado, no que concerne ao uso na sequência didática. Agora é o momento de iniciar seu uso em tarefas e algoritmos que auxiliem os alunos a desenvolver seu potencial cognitivo e o PMA.

4 SEQUÊNCIA DIDÁTICA

A construção da sequência didática consistirá de duas partes: uma introdutória com a parte inicial sobre divisibilidade e introdução aos números primos e outra que trata dos algoritmos e temas mais complexos.

4.1 PRÉ-REQUISITOS

A sequência está alicerçada sobre as ideias da lógica clássica, o que do ponto de vista educacional é essencial para dedução de padrões ou mesmo compreensão dos diversos temas matemáticos discutidos na vida escolar.

Dessa forma, uma introdução à lógica se faz necessária, uma fonte de abordagem desse tema é Morais Filho (2012), pois caso o professor deseje pode abordar outras conexões com o ensino da matemática além da lógica.

Outro ponto que é necessário discutir antes de iniciar a sequência é a abordagem das linguagens de programação, essencialmente os conceitos de programação são os mesmos, assim como primeiro contato, a discussão pode se restringir ao *Scratch*, devido sua facilidade de uso e construção. Porém no trabalho serão apresentadas as atividades desenvolvidas nas duas linguagens, assim o professor pode comparar e compreender a dinâmica de ambas.

O estudo de programação, assim como de Matemática, é preferível que seja feito com prática, desta forma, o professor pode abordar o *software* e suas telas e funções utilizando alguns aspectos da geometria ou mesmo com algoritmos simples do estudo de números e operações elementares, como: soma de sequências numéricas, construção de tabuada e etc.

Como a parte de geometria não será o foco desse trabalho, serão apresentados alguns exemplos que levam os alunos a compreenderem como construir algoritmos e resolver problemas usando o *Scratch*. Uma questão inicial é: de que forma poderia se fazer um algoritmo que retorne a tabuada de um número n até seu décimo múltiplo? As Figuras 13 e 14

representam uma possibilidade de resposta à questão no *Scratch* enquanto a Figura 15 apresenta no *Python*.

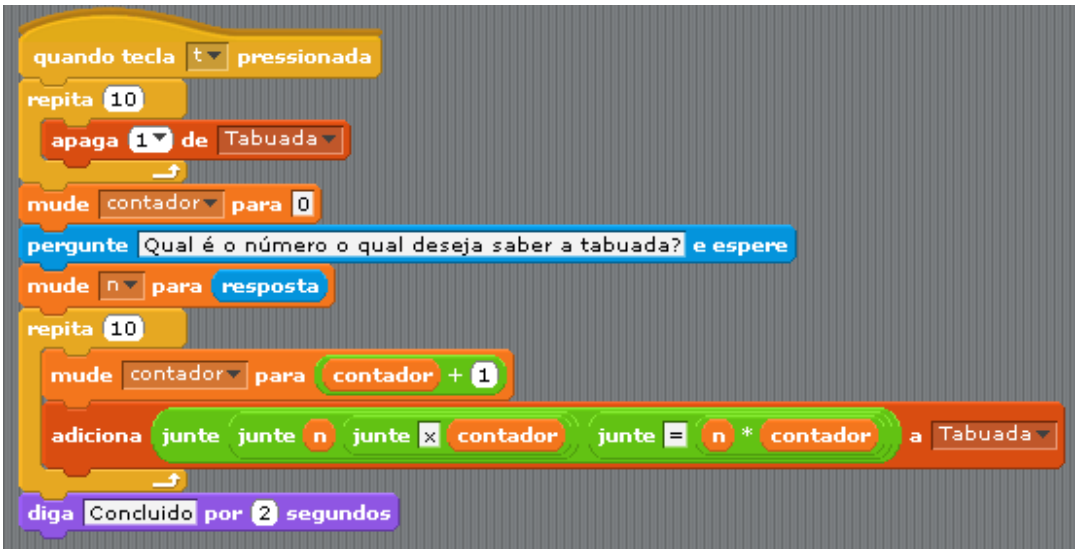

Figura 13: *Scratch* - algoritmo da tabuada

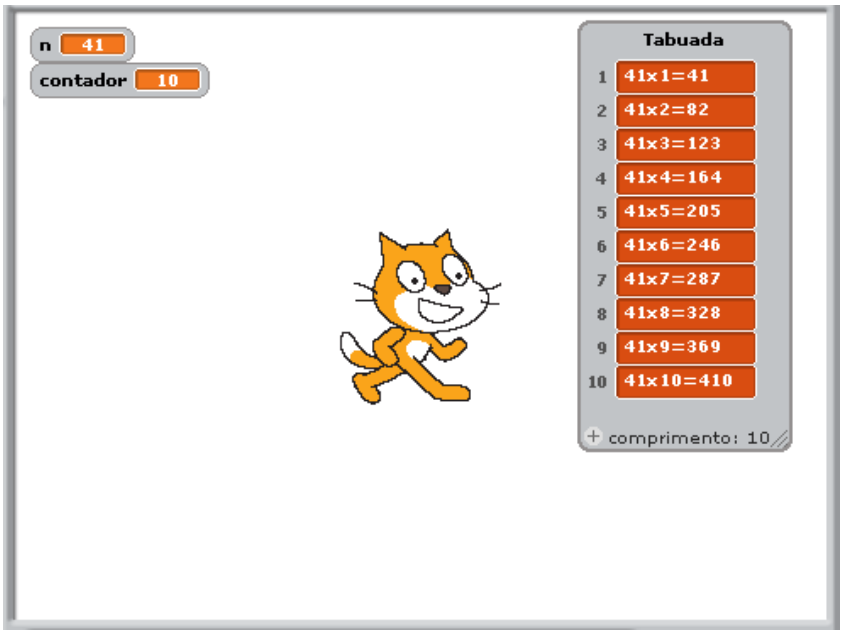

Figura 14: *Scratch* - saída do algoritmo da tabuada com entrada $n = 41$

```
# -*- coding: iso-8859-1 -*-
n = int(input("Qual é o número que deseja saber a tabuada?\n"))
print "\n"
for contador in range(1,11):
        print ("%dx%d=%d")%(n,contador,n*contador)
```

```
>>>
Qual é o número que deseja saber a tabuada?
41


41x1=41
41x2=82
41x3=123
41x4=164
41x5=205
41x6=246
41x7=287
41x8=328
41x9=369
41x10=410
```

Figura 15: *Python* - algoritmo da tabuada e sua saída com entrada $n = 41$

Apesar da simplicidade do conceito matemático da tabuada, o intuito é compreender alguns aspectos da programação que leva a busca de padrões.

Nesse algoritmo (implementado na Figura 13), utilizou-se como método de início a tecla *t* (*Scratch*), a criação de uma lista para receber os valores e duas variáveis uma que recebe a entrada e outra que conta os ciclos do programa. Neste caso, a parte mais complexa é a organização do texto de saída. Não foi feito nenhum tratamento de erros, pois o usuário é o próprio aluno, portanto ele conhece o intervalo de valores possíveis de entrada.

Outra questão de cunho inicial é: qual é a soma dos naturais[36] pares até um dado n?

Como esta sequência é pensada inicialmente para alunos ingressantes do ensino médio, eles podem não conhecer os meios para a soma de uma progressão aritmética, assim, os algoritmos implementados não levarão isso em consideração (conforme Figuras 16 e 18 e suas respectivas saídas, Figuras 17 e 19).

[36] Os números naturais serão considerados como os pertencentes ao conjunto $\{1, 2, 3, \cdots, n, \cdots\}$.

Figura 16: *Scratch* - soma dos naturais pares até um dado n.

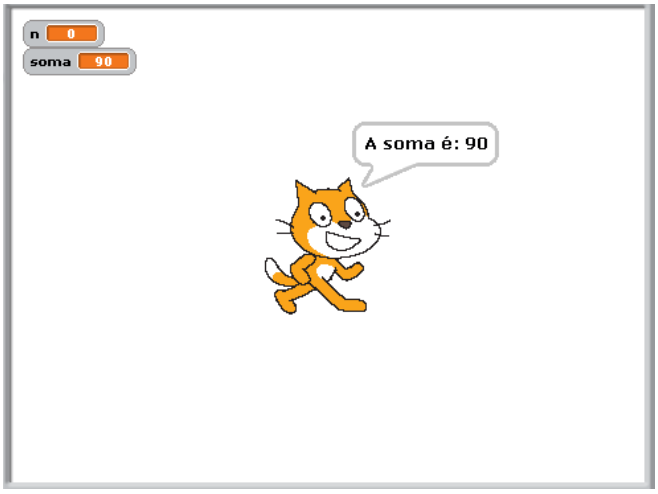

Figura 17: *Scratch* - resultado da soma dos naturais pares até 19.

```
# -*- coding: iso-8859-1 -*-
n=int(input("Até qual número deseja somar os pares?\n"))
if n%2!=0:
    n=n-1
soma=0
while n>0:
    soma=n+soma
    n=n-2
print ("A soma é: %d") %soma
```

Figura 18: *Python* - soma dos naturais pares até um dado n.

```
Até qual número deseja somar os pares?
19
A soma é: 90
```

Figura 19: *Python* – resultado da soma dos naturais pares 19.

A parte interessante da programação é que cada aluno pode fazer suas conjecturas e testá-las imediatamente. Isso resulta em algoritmos distintos o que fortalece o aspecto matemático que possibilita diversas resoluções para uma mesma situação problema, com maior ou menor complexidade ou eficiência.

No algoritmo apresentado (Figura 16), utilizou-se como método de início a tecla de espaço, mudança da variável que retorna a soma para zero, pois o *Scratch* mantem o último resultado, depois é que se inicia a resolução do problema propriamente dito. Novamente não foi feito nenhum tratamento de erros.

Uma simples variação no problema para ensino é questionar sobre a soma de naturais ímpares, nesse sentido o aluno já precisará identificar quais características comuns na soma de pares e ímpares, o intervalo de entrada (em $\mathbb{N}$) bem como ter possibilidade de explorar a ideia de função crescente em qualquer uma das situações problema. Pois, para a soma dos pares, tem-se que seja $f: \mathbb{N} \to \mathbb{N}$, uma função tal que

$$f(n) = \begin{cases} \left(\frac{n}{2} + 1\right)\left(\frac{n}{2}\right) & se\ n\ par \\ \left(\frac{n+1}{2}\right)\left(\frac{n-1}{2}\right) & se\ n\ ímpar \end{cases}.$$

Esta função cumpre o mesmo papel que o algoritmo e com isso, o professor já pode explorar de forma direta ou indireta o conceito de função crescente.

Com as ideias de lógica e tabelas-verdade, juntamente com esses algoritmos iniciais, a sequência didática que tem por fundamento conceitos matemáticos abordados no referencial teórico pode ser iniciada.

4.2 SEQUÊNCIA DIDÁTICA: PARTE I

Dada a introdução à lógica e a discussão sobre o *Scratch* do item anterior é possível ao professor iniciar a sequência para discutir o tema central: uma introdução aos testes de primalidade e fatoração.

Nesta primeira etapa serão discutidos temas com melhor construção algorítmica e assimilação pelo aluno, como o algoritmo da divisão, cálculo do máximo divisor comum de dois números, um primeiro método de fatoração de inteiros e o Crivo de Eratóstenes.

Ao solicitar a construção do algoritmo da divisão, conforme exposto anteriormente, obtém-se um resultado similar ao apresentado nas Figuras 20 e 22, referente respectivamente ao *Scratch* ou *Python*:

Figura 20: *Scratch* - algoritmo da divisão.

Figura 21: *Scratch* - resultado da divisão de 127 por 7.

```
# -*- coding: utf-8 -*-
dividendo = input ("Qual é o dividendo?")
divisor = input("Qual é o divisor?")
quociente = 0
resto = dividendo
while not(resto < divisor):# ou while resto >= divisor
    resto = resto-divisor
    quociente +=1
print ("O quociente é %d. O resto é %d.") %(quociente, resto)
```

Figura 22: *Python* - algoritmo divisão.

```
Qual é o dividendo?
127

Qual é o divisor?
7
O quociente é 18. O resto é 1.
```

Figura 23: *Python* - resultado da divisão de 127 por 7.

Com esse algoritmo (implementado nas Figuras 20 e 22) o professor pode retomar os conhecimentos do princípio do ensino fundamental e pode discutir as ideias que o envolvem em relação ao PMA. Abordando a existência e unicidade do respectivo quociente e resto da divisão. Um fato a considerar é que o *Scratch* e o *Python* possuem o comando que retorna o quociente inteiro da divisão e, portanto o aluno pode usufruir disso e não desenvolver o algoritmo como previsto.

O algoritmo para calcular o mdc de dois números (apresentado a seguir e implementado conforme as Figuras 24 e 26) pode gerar um conflito interessante no quadro mental dos alunos, pois os alunos aprendem a calcular o mdc a partir dos fatores primos dos números envolvidos e o algoritmo que será construído é dado pelo Algoritmo Euclidiano, com isso cabe ao professor apresentar essa divergência e possíveis questionamentos dos alunos, lembrando que as dúvidas promovem o conhecimento, isso pode ser frutífero para a aprendizagem matemática e para a definição de padrões de diversas iterações do processo até definir o mdc.

Entrada: inteiros positivos a e b (supondo $a > b$).

Saída: um inteiro positivo $d = mdc(a, b)$.

Etapa 1: Encontre r tal que $a = bq + r$ e $0 \leq r < b$.

Etapa 2: Se $r = 0$, escreva o mdc é b, senão, faça $a = b$ e $b = r$ e volte a Etapa 1.

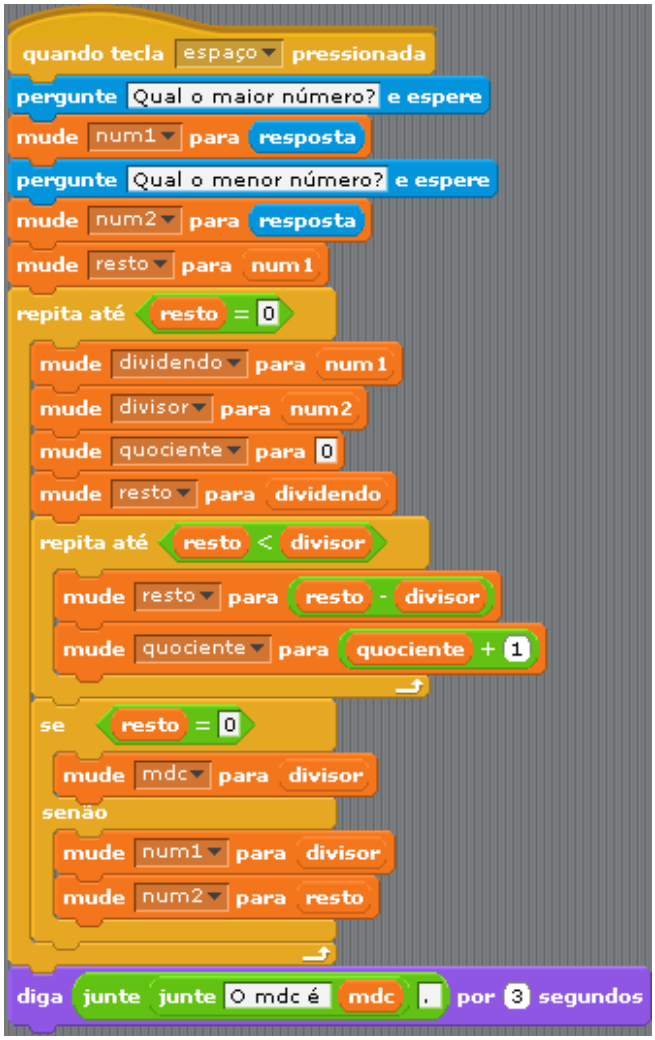

Figura 24: *Scratch* - cálculo do mdc.

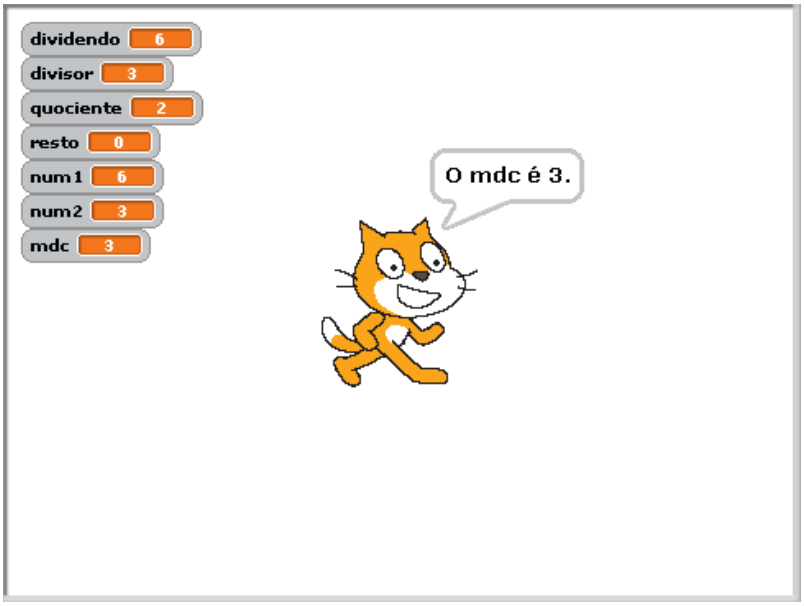

Figura 25: *Scratch* - resultado do cálculo do mdc entre 123 e 84.

```python
# -*- coding: utf-8 -*-
def divisao_inteira(dividendo, divisor):
    quociente = 0
    resto = dividendo
    while not(resto < divisor):
        resto = resto-divisor
        quociente +=1
    return resto

num1 = input("qual o maior números?\n")
num2 = input("Qual o menor número?\n")
resto = 1
while resto!=0:
    resto = divisao_inteira(num1, num2)
    if resto == 0:
        mdc = num2
    else:
        num1 = num2
        num2 = resto
print ("O mdc é %d.") % mdc
```

Figura 26: *Python* - cálculo do mdc.

```
Qual o maior número?
123

Qual o menor número?
84
O mdc é 3.
```

Figura 27: *Python* - resultado do cálculo do mdc entre 123 e 84.

A partir das Figuras 22 e 26 é possível notar uma diferença nas implementações dos algoritmos, isso se deve ao fato de no *Scratch* não possuir a possibilidade de construir funções explicitamente, diferentemente que no *Python* o que facilita a divisão do problema e sua resolução em partes, que de certa modo, o torna mais simples, uma vez que no presente caso foi aproveitado o algoritmo da divisão para o desenvolvimento do cálculo do mdc entre dois inteiros. Esse fato reforça o porquê de usar somente o *Python* na segunda parte da sequência didática e não haver implementações em *Scratch*.

É possível ainda que o professor desenvolva também um algoritmo que envolva o método de cálculo do *mdc* pelos fatores primos, no entanto, para isso é necessário primeiro construir um algoritmo que o faça, sendo assim, o professor deve fazer uma revisão teórica sobre números primos, apontar os dados históricos e curiosidades como motivadoras e o Teorema Fundamental da Aritmética que embasa esse algoritmo.

O algoritmo da fatoração apresentado aqui é o proposto por Coutinho[37] (2013, p. 38) e, como pode se ver na construção (Figuras 28 e 30), é bem simples, porém a carga conceitual traz um resultado importante o fato de se n é natural então n tem um fator menor que ou igual

[37] Apresentado também nos tópicos sobre a base Matemática da sequência-didática.

a $\sqrt{n}$ ou n é primo. O que diminui a busca de divisores de n e melhora a eficiência do algoritmo, mas antes de expor esse resultado o professor deve, como em todos os casos, requisitar que os alunos desenvolvam seus próprios algoritmos, assim podem comparar entre si esses elementos e verificar a eficiência e devidas melhorias. Ou seja, o professor deve fomentar o estudo pelo ciclo descrição – execução – reflexão – depuração – descrição, em um contínuo pensar e refletir sobre seus resultados e ações.

Com isso, obtém-se o algoritmo de fatoração implementado no *Scratch* conforme a Figura 28 e no *Python* de acordo com a Figura 30 e seus exemplos de saída nas Figuras 29 e 31:

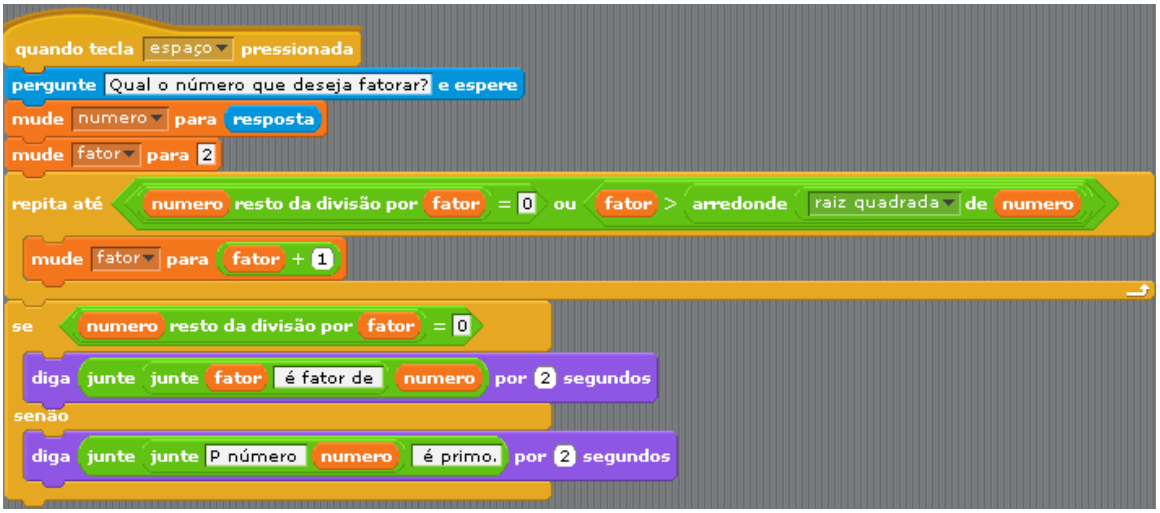

Figura 28: *Scratch* - algoritmo de fatoração

Figura 29: *Scratch* - primeiro fator primo de 1037.

```python
# -*- coding: utf-8 -*-
def fator(n):
    fator=2
    while (n%fator > 0) and (fator < int(n**0.5)):
        fator += 1
    if (n%fator==0):
        return "%d é fator de %d." %(fator, n)
    else:
        return "O número %d é primo." %n
```

```
17 é fator de 1037.
```

Figura 30: *Python* - algoritmo de fatoração e resultado do primeiro fator primo de 1037

Como é possível notar, só retorna o primeiro (e menor) fator primo encontrado, sendo assim, é necessário armazenar o resultado do algoritmo e repeti-lo com a razão resultante até encontrar todos os fatores do número que resulta em algo como apontado nas Figuras 30 e 32:

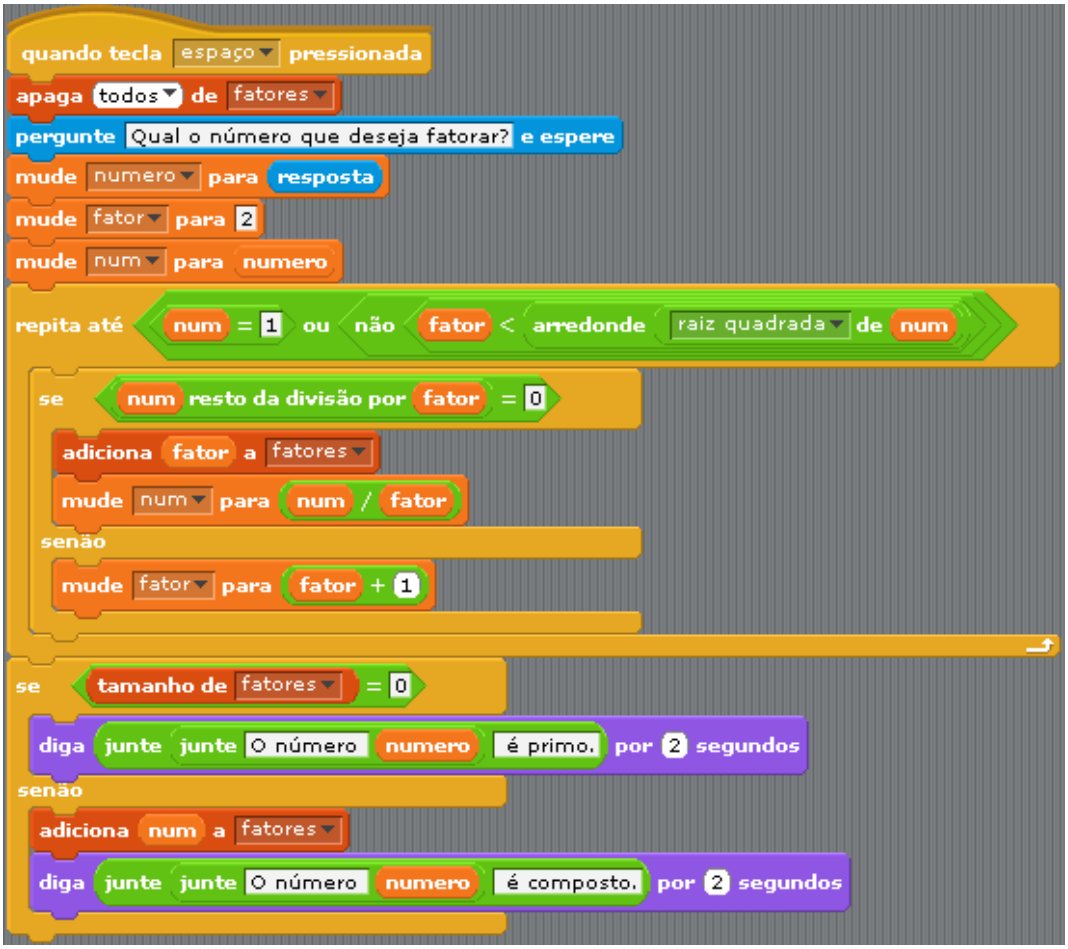

Figura 31: *Scratch* - algoritmo de fatoração

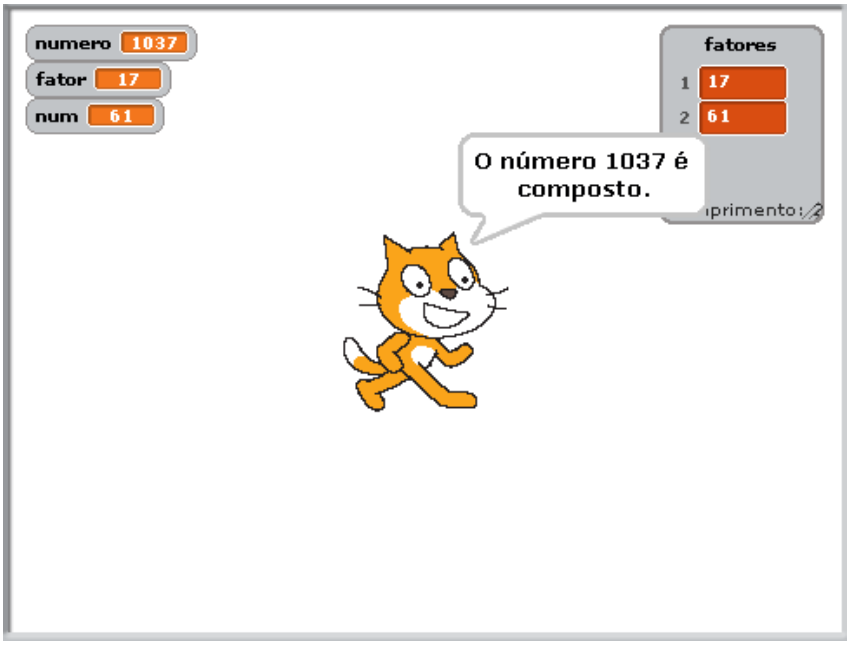

Figura 32: *Scratch* – resultado da fatoração de 1037.

```python
# -*- coding: utf-8 -*-
def fatora_completa(numero):
    fatores = []
    fator = 2
    num = numero
    while (num!=1) and (fator <= int(num**0.5)):
        if (num%fator) == 0:
            fatores.append(fator)
            num = num/fator
        else:
            fator += 1
    if len(fatores) == 0:
        return "O número %d é primo" %numero
    else:
        fatores.append(num)
        return "O número %d é composto e seus fatores são %s" %(numero, str(fatores))
```

Figura 33: *Python* - algoritmo de fatoração completa

```
O número 1037 é composto e seus fatores são [17, 61]
```

Figura 34: *Python* - resultado da fatoração de 1037.

Outro fator que faz parte do algoritmo da fatoração é o fato de ser necessário encontrar a raiz quadrada de um natural, portanto há casos em que isso é um número irracional, porém o que interessa para os algoritmos aqui dispostos é somente a parte inteira da raiz, sendo assim, devido às questões expostas pelas peculiaridades computacionais, basta um meio de se encontrar a parte inteira da raiz quadrada de um número, Coutinho (2013) apresenta um método para tal problema, porém este método apresenta divisões consecutivas que, do ponto de vista do computador, são operações que requerem mais esforço de máquina que somas e subtrações além de não ser possível se o número for muito grande.

Desta forma, é interessante um algoritmo que dependa apenas dessas operações e o aluno tem condições de testá-lo sem muito esforço com lápis e papel. Esse algoritmo tem como referência as Equações de Pell que sua forma mais simples é $x^2 - Ay^2 = 1$ em que $A \in \mathbb{N}$ e $\sqrt{A} \notin \mathbb{Q}$ tal que as soluções $x, y \in \mathbb{Z}$.

Seja n natural, o qual se deseja encontrar a parte inteira de $\sqrt{n}$ o algoritmo consiste em subtrações sucessivas dadas da seguinte forma:

- Tomando $n_1 = n$, então

$$n_1 - 1 = n_2$$
$$n_2 - 3 = n_3$$
$$\vdots$$
$$n_k - (2k - 1) = n_{k+1}$$

Esse processo enquanto $n_{k+1} > 0$, e quando o algoritmo parar, ou seja, k será a parte inteira de $\sqrt{n}$.

A prova disto vem do fato que $1, 3, 5, \cdots, 2k - 1$ são os termos de uma progressão aritmética e portanto sua soma é $\frac{(1+2k-1)k}{2} = k^2$, como $k \geq 1$ é um quadrado perfeito, $n_{k+1} \leq 0 \Longrightarrow n \leq k$, pois $n_{k+1} = n - k^2$. Então os algoritmos implementados podem ser construídos como as Figuras 35 e 37.

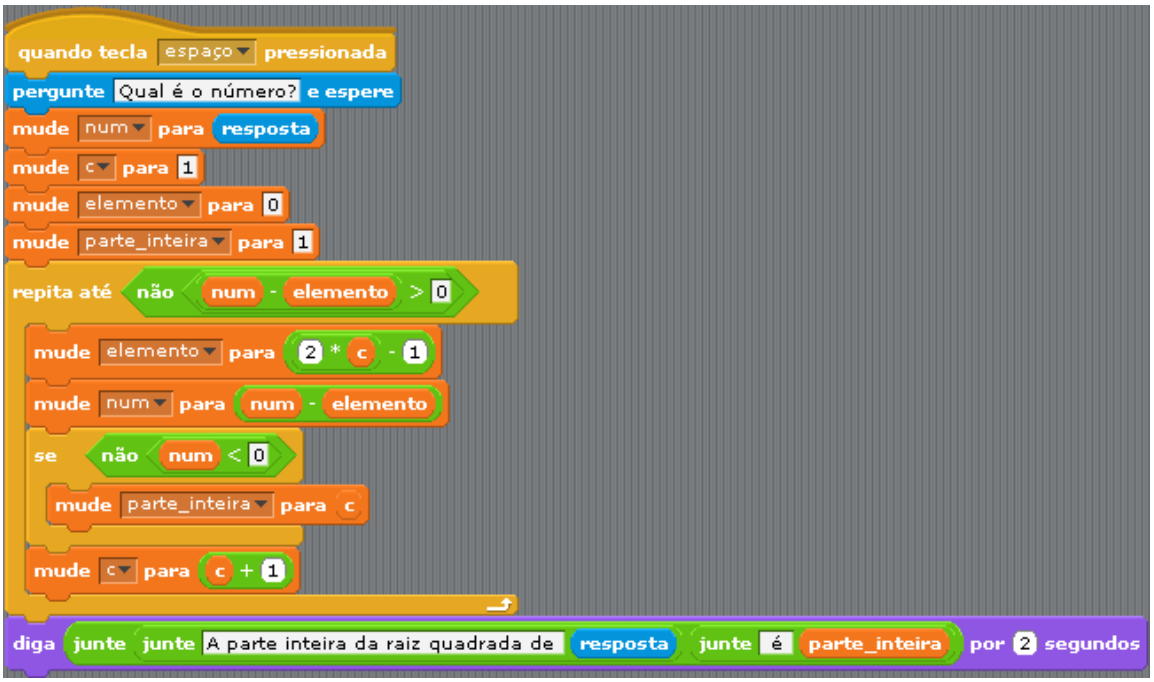

Figura 35: *Scratch* - parte inteira da raiz quadrada de um inteiro positivo

Figura 36: *Scratch* - resultado da parte inteira da raiz quadrada de 4294967297.

```
# -*- coding: utf-8 -*-
def parte_inteira_raiz(num):
    c = 1
    elemento = 0
    parte_inteira = 1
    while (num - elemento) > 0:
        elemento = (2*c)-1
        num = num - elemento
        if num>=0:
            parte_inteira = c
        c += 1
    return parte_inteira

print parte_inteira_raiz(51)
```

Figura 37: *Python* - parte inteira da raiz quadrada de um inteiro positivo

```
A parte inteira da raiz quadrada de 4294967297 é 65536.
```

Figura 38: *Python* - resultado da parte inteira da raiz quadrada de 4294967297.

A beleza desse fato é que conceitos simples resultam em aplicações instigantes e esse é o cunho que pauta este trabalho, tornar o trabalho educacional em um meio de aprendizagem que não seja tedioso e sim criativo e inovador.

E por fim, é de se esperar que os alunos já tenham adquirido maturidade para desenvolver um algoritmo que requeira maior complexidade, dado que demanda maior número de etapas.

Como dito, o Crivo de Eratóstenes não é um mecanismo eficiente para definir se um número é primo ou não, no entanto dado seu contexto histórico, ele pode ser explorado,

porém é possível fazer algumas alterações em sua estrutura com o intuito de torná-lo melhor computacionalmente (mais eficiente), pois em sua forma original certos números serão "riscados" diversas vezes, por serem múltiplos de mais de um número, sendo assim Coutinho (2013, p. 64-65) sugere o seguinte algoritmo:

> **Crivo de Eratóstenes (Melhorado)**
> **Entrada:** inteiro positivo ímpar n.
> **Saída:** lista de primos ímpares$\leq n$.
> **Etapa 1:** comece criando um vetor **v** de $(n-1)/2$ posições, cada uma das quais deve estar preenchida com o valor 1; e fazendo $P = 3$.
> **Etapa 2:** Se $P^2 > n$ escreva os números $2j + 1$ para os quais a j-ésima entrada do vetor **v** é 1 e pare; senão vá para a Etapa 3.
> **Etapa 3:** Se a posição $(P-1)/2$ do vetor **v** está preenchida com 0 incremente P de 2 e volte à Etapa 2; senão vá para a Etapa 4.
> **Etapa 4:** Atribua o valor P^2 a uma nova variável T: substitua por zero o valor da posição $(T-1)/2$ do vetor **v** e incremente T de $2P$; repita essas duas intruções até que $T > n$; quando isto acontecer incremente P de 2 e volte à Etapa 2.

Que ao se implementar resulta em nas Figuras 39 e 41.

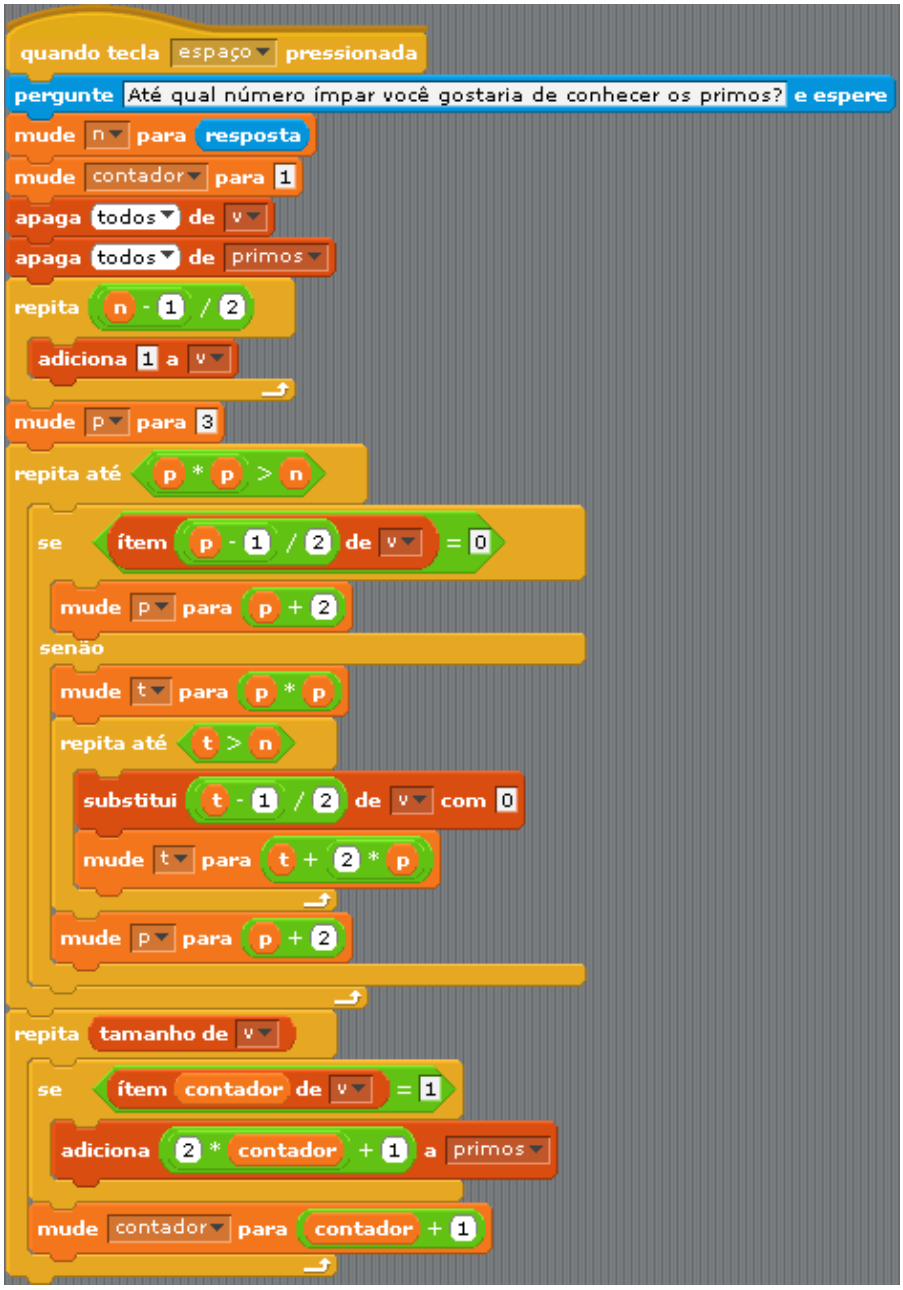

Figura 39: *Scratch* - Crivo de Eratóstenes melhorado

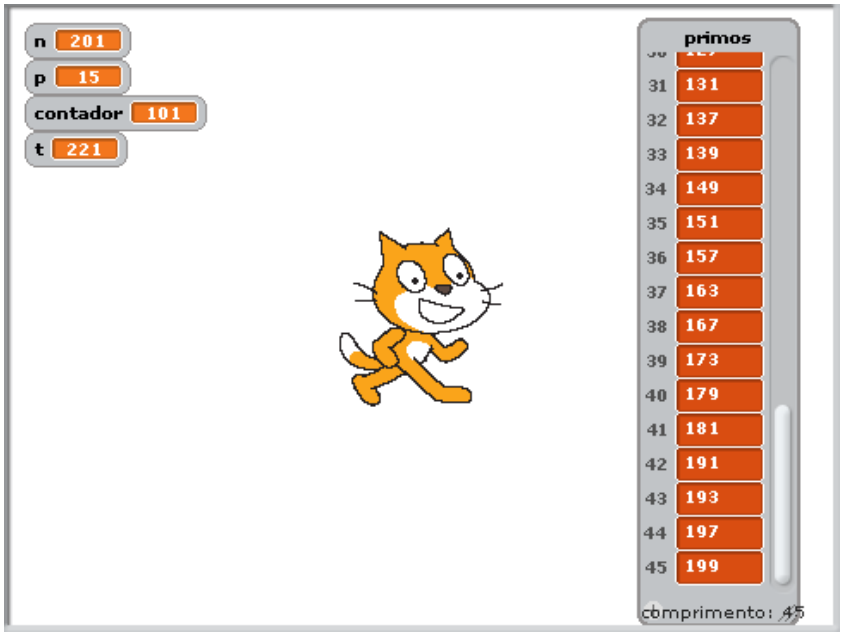

Figura 40: *Scratch* - resultado com o valor dos primos até 201 pelo Crivo de Eratóstenes.

```
# -*- coding: utf-8 -*-
def crivo_eratostenes(n):
    contador = 1
    v = []
    primos = []
    for a in range(1, (n+1)/2):
        v.append(1)
    p = 3
    while (p**2) <= n:
        if v[(p-3)/2] == 0:
            p = p + 2
        else:
            t = p**2
            while t <= n:
                v[(t-3)/2] = 0
                t = t +(2*p)
            p = p + 2
    for a in range(0, len(v)):
        if v[contador -1] == 1:
            primos.append(2*contador +1)
        contador += 1
    return primos
print crivo_eratostenes(input("Até qual número ímpar "+
                              "você gostaria de conhecer os primos?\n"))
```

Figura 41: *Python* - Crivo Eratóstenes melhorado

```
Até qual número ímpar você gostaria de conhecer os primos?
201
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89,
97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181,
191, 193, 197, 199]
```

Figura 42: *Python* - resultado com o valor dos primos até 201 pelo Crivo de Eratóstenes.

Explorar esses algoritmos e temas matemáticos já resulta em uma grande discussão sobre os conceitos de variáveis, domínio, função, imagem, sequências entre outras que podem

surgir em aula e o papel do professor será mediar e conduzir esta aprendizagem. O computador em si não será capaz de ensinar ao aluno, será apenas uma ferramenta a mais no arcabouço dos professores para melhorar o desenvolvimento dos seus alunos, sendo portanto necessário mesclar o uso do computador com discussões provenientes de seus usos, as relações que foram definidas a partir das construções. O refletir sobre a ação é que será o desencadeador do PMA e irá criar conjecturas que o aluno pode testar e comprovar ou refutar, passando ao professor as provas formais que se fizerem necessárias e socializando as descobertas de modo rigoroso quando couber.

Nessa introdução às ferramentas computacionais e temas matemáticos são utilizados temas quase que de senso comum para incentivar os alunos a explorarem ainda mais o seu saber. Na segunda fase que vem a seguir serão explorados alguns temas que não costumam compor o currículo das escolas de educação básicas brasileiras, mas que podem ser fontes de boas discussões.

4.3 SEQUÊNCIA DIDÁTICA: PARTE II

Com a discussão ocorrida a partir do item anterior é possível ao docente abordar, matematicamente e computacionalmente, temas mais complexos como: Fatoração por Fermat, Método de Fermat, Teste de Lucas-Lehmer, Método de Euler e um sistema *pseudo-RSA* (ou seja, um RSA simplificado), sendo este último apenas de cunho ilustrativo e motivador, pois é interessante o professor usá-lo como meio de motivação para os alunos com uma situação problema como: “Dados alguns critérios, quem consegue números primos para proteger uma mensagem secreta?”.

Com a questão ou algo similar, apresentar o mecanismo de segurança (RSA), histórico, embasamento matemático para sua validação e questionar como descobrir se um número é primo ou composto e/ou ainda solicitar para criarem suas mensagens “secretas” e suas respectivas chaves públicas[38] para que o professor examine o tempo gasto e a mensagem secreta dos alunos a partir de primos computados por eles, sendo assim o professor precisará ter seu próprio algoritmo para fatoração e que tenha um bom desempenho.

[38] Ponto central da abordagem do RSA.

Para que as atividades não se estendam por longos períodos de tempo, pode ser imposto aos alunos um tempo para que descubram dois primos que comporão a chave pública e este primo será melhor ou pior dependendo do algoritmo que os alunos irão utilizar e aí cabe apresentar os algoritmos dos testes de primalidade, pois poderá ser mostrado que cada teste tem sua eficiência e sua base matemática e a sociedade científica busca mecanismos cada vez melhores de encontrar primos que podem ser utilizados na segurança RSA entre outros métodos.

Assim, será apresentado o mecanismo para fatoração de um dado natural $n > 2$ e ímpar, caso não seja, basta dividir por 2 até que a razão seja um número ímpar, este algoritmo denominado Fatoração por Fermat é muito eficiente quando n tem seus fatores próximos e evidentemente mais eficiente que o algoritmo de fatoração apresentado na primeira parte para definir se um número é par ou ímpar. Isso compõe o estudo desta sequência, a evolução da Matemática e seus refinamentos em busca de métodos melhores para resolução de um dado problema.

Com a estrutura do algoritmo apresentada anteriormente, uma das possibilidades de implementar o algoritmo é proposto na Figura 43, cuja saída é a Figura 44.

```python
# -*- coding: utf-8 -*-

def parte_inteira_raiz(num):
    c = 1
    elemento = 0
    parte_inteira = 1
    while (num - elemento) > 0:
        elemento = (2*c)-1
        num = num - elemento
        if num>=0:
            parte_inteira = c
        c += 1
    return parte_inteira

def fatoracao_fermat(n):
    x = parte_inteira_raiz(n) # int(n**0.5)
    y = 0
    if n == (x**2):
        resultado = "%d é fator de %d." %(x, n)
    else:
        while (x < ((n+1)/2)) and ((y**2) != ((x**2)-n)):
            x += 1
            y = parte_inteira_raiz((x**2)-n) # int(((x**2)-n)**0.5)
    if (x-y) == 1:
        resultado = "O número %d é primo." %n
    else:
        resultado = "%d e %d são fatores de de %d." %(x-y, x+y, n)
    return resultado

print fatoracao_fermat(input("Qual inteiro positivo ímpar deseja testar?"))
```

Figura 43: *Python* - Fatoração por Fermat

```
Qual inteiro positivo ímpar deseja testar?
4294967297
641 e 6700417 são fatores de de 4294967297.
```

Figura 44: *Python* - resultado da fatoração por Fermat de 4294967297.

Os comentários presentes no código se devem ao fato de que se o número de entrada não for muito grande, ou seja, não acarretar em erro devido ao tamanho de sua conversão em *float*, pode ser usado o comando nativo para encontrar a parte inteira da raiz quadrada de um número que resulta em melhor eficiência do algoritmo.

Como pode ser visto no algoritmo implementado (Figura 44), ele requer a chamada do algoritmo que encontra a parte inteira da raiz quadrada de um número (Figura 37), caso o número seja grande, nesse sentido, o professor pode explorar a ideia de função composta, em matemática a partir do conceito similar da informática, proporcionando ao aluno uma visão real de aplicabilidade.

Com a construção desses algoritmos, pode-se pensar inclusive em testes de eficiência, em que o usuário entra com uma lista de números e mede o tempo necessário para determinar se é primo ou composto com base nos três mecanismos apresentados: algoritmo de fatoração, Crivo de Eratóstenes e Fatoração por Fermat, além de outros possíveis.

A partir desse ponto, os algoritmos serão implementados somente no *Python*, pois serão utilizados números de crescimento muito rápido, serão os números de Mersenne e Fermat, para tanto serão discutidos ou apresentados os contextos matemáticos e suas respectivas implementações do: Método de Fermat, Teste de Lucas-Lehmer, Método de Euler e Teste de Pépin, onde os dois primeiros se relacionam com os números de Mersenne e os demais com os números de Fermat.

No que concerne ao Método de Fermat, pode ser útil na dinâmica da aula por encontrar fatores dos números de Mersenne ou primos de Mersenne, os quais crescem rapidamente, e assim pode ser um meio em que o aluno encontra os primos para codificar sua mensagem.

O algoritmo pode inicialmente definir qual é o valor limite que o incremento poderá chegar evitando cálculos repetitivos e desnecessários pelo computador, porém se M_p for grande as operações podem ser demoradas, dada a quantidade de operações a serem feitas para alcançar o fim do algoritmo.

A estrutura a ser implementada, com base no algoritmo apresentado na Figura 43, é apresentada na Figura 45.

```
# -*- coding: utf-8 -*-
def parte_inteira_raiz(num):
    c = 1
    elemento = 0
    parte_inteira = 1
    while (num - elemento) > 0:
        elemento = (2*c)-1
        num = num - elemento
        if num>=0:
            parte_inteira = c
        c += 1
    return parte_inteira

def divisao_inteira(dividendo, divisor):
    quociente = 0
    resto = dividendo
    while not(resto < divisor): # ou while resto >= divisor
        resto = resto-divisor
        quociente +=1
    return [quociente, resto]

def metodo_fermat(primo):
    r=1
    mersenne = (2**primo)-1
    limite = int((parte_inteira_raiz(2**primo)-1)/(2.0*primo))
    # limite = int((int(2**(primo/2.0))-1)/(2.0*primo))
    resto = 1
    while (resto != 0) and (r <= limite):
        q = 1+(r*primo)
        resto = divisao_inteira(mersenne, q)[1] #resto = mersenne % q
        if resto == 0:
            return "\n%d é fator de M(%d)=%d" %(q, primo, mersenne)
        r += 1
    return "\nO número M(%d)=%d é primo." %(primo, mersenne)

print metodo_fermat(input("Qual o índice primo do número de Mersenne deseja "+
                          "testar a primalidade?\n"))
```

Figura 45: *Python* - Método de Fermat

```
Qual o índice primo do número de Mersenne que deseja testar a primalidade?
29

233 é fator de M(29)=536870911
```

Figura 46: *Python* - Resultado da primalidade de M(29) pelo Método de Fermat.

É possível notar que novamente será chamada a função que retorna a parte inteira da raiz quadrada de um número, agora em novo contexto, como o ciclo de aprendizagem do aluno é crescente pode-se supor que o aluno compreenderá esse recurso e o conceito de uma melhor forma e com maior segurança.

E uma “evolução” do processo pode ser entendido como o Teste de Lucas-Lehmer que determina se um número de Mersenne é primo ou não, mas não apresenta nenhum de seus fatores caso seja composto.

Esse teste abre espaço para o professor evoluir conceitualmente a partir da sequência definida recursivamente podendo inclusive implementar essa proposta e discutir a ideia de recursividade, um conceito amplamente utilizado em diversas áreas da Matemática. O algoritmo resulta na Figura 47, a seguir.

```
# -*- coding: utf-8 -*-
def divisao_inteira(dividendo, divisor):
    quociente = 0
    resto = dividendo
    while not(resto < divisor): # ou while resto >= divisor
        resto = resto-divisor
        quociente +=1
        if quociente%1000000 == 0:
             print resto
    return [quociente, resto]

def lucas_lehmer_recursivo(p):
    s = 4
    mersenne = (2**p)-1
    for n in range(1, p-1):
        s= (s**2)-2
    if s%mersenne==0:# ou if divisao_inteira(s, mersenne)[1] == 0:
         return "\nO número M(%d)=%d é primo." %(p, mersenne)
    else:
        return "\nO número M(%d)=%d é composto." %(p, mersenne)

print lucas_lehmer_recursivo(input(u'Qual o índice primo que deseja testar '+
                                   'a primalidade de M(p)?'"\n"))
```

Figura 47: *Python* - teste de Lucas-Lehmer

```
Qual o índice primo que deseja testar a primalidade de M(p)?
29

O número M(29)=536870911 é composto.
```

```
Qual o índice primo que deseja testar a primalidade de M(p)?
31

O número M(31)=2147483647 é primo.
```

Figura 48: *Python* - resultados do teste de Lucas-Lehmer para os números M(29) e M(31).

Neste ponto, é possível combinar algoritmos desenvolvidos anteriormente, como o caso do Crivo de Eratóstenes que retorna uma lista com os primos até um dado n de forma a testar todos os M_p (com $p \neq 2$) contidos nesta lista, lembrando que para $p > 23$ o algoritmo requer um tempo considerável para determinar a primalidade ou não de M_p. Assim, o código anterior tem como final a Figura 49.

```
    else:
        return "\nO número M(%d)=%d é composto." %(p, mersenne)

def crivo_eratostenes(n):
    contador = 1
    v = []
    primos = []
    for a in range(1, (n+1)/2):
        v.append(1)
    p = 3
    while (p**2) <= n:
        if v[(p-3)/2] == 0:
            p = p + 2
        else:
            t = p**2
            while t <= n:
                v[(t-3)/2] = 0
                t = t +(2*p)
            p = p + 2
    for a in range(0, len(v)):
        if v[contador -1] == 1:
            primos.append(2*contador +1)
        contador += 1
    return primos

for a in crivo_eratostenes(25):
    if a != 2:
        print lucas_lehmer_recursivo(a)
```

Figura 49: *Python* - teste de Lucas-Lehmer com Crivo de Eratóstenes

Agora serão tratados algoritmos que abarcarão os números de Fermat que crescem mais rapidamente que os números de Mersenne, apesar de até o momento não se ter encontrado nenhum número de Fermat primo maior que F_4. Os maiores fatores primos de F_n, com $n > 4$, são primos que crescem muito e portanto nas condições da situação-problema da aula podem ser interessantes para a situação didática e o professor pode utilizar esses fatores como meios de criar chaves públicas.

Com isso, ao estudar e implementar o Método de Euler poderão ser encontrados fatores de F_n, enquanto que com o Teste de Pépin só poderá definir se F_n é primo ou composto. O professor pode fazer com que os alunos reflitam sobre as utilidades de cada algoritmo e decidir sobre qual a aplicabilidade buscarão para implementar suas chaves.

Em se tratando do Método de Euler, dada sua similaridade com o Método de Fermat, pode-se iniciar com a definição do limitante de r a priori[39] e posteriormente implementar, resultando no algoritmo implementado na Figura 50, a qual tem duas saídas apresentadas na Figura 51.

[39] Seja como função ou procedimento.

```
# -*- coding: utf-8 -*-
def metodo_euler(n):
    limite_r=((2**(2**(n-1)))-1)/(2**(n+2))
    fermat = (2**(2**n))+1
    r=1
    f = 1+(r*(2**(n+2)))
    while (fermat % f) != 0 and (r < limite_r):
        r += 1
        f = 1+(r*(2**(n+2)))
    if not(r < limite_r):
        return "\nO número F(%d)=%d é primo." %(n, fermat)
    else:
        return "\n%d é fator de F(%d)=%d." %(f, n, fermat)

print metodo_euler(input("Qual índice do número de Fermat deseja "+
                   "testar a primalidade?\n"))
```

Figura 50: *Python* - Método de Euler

```
Qual índice do número de Fermat deseja testar a primalidade?
4

O número F(4)=65537 é primo.
```

```
Qual índice do número de Fermat deseja testar a primalidade?
6

274177 é fator de F(6)=18446744073709551617.
```

Figura 51: *Python* - resultados da primalidade de F(4) e F(6) pelo Método de Euler.

E o último item da sequência é a discussão em torno do Teste de Pépin, que como dito, é alicerçado sobre o fato de que se $k^{(F_n-1)/2}+1|F_n$ então F_n é primo e $k \in \{3,5,10\}$. Como se sabe os números de Fermat crescem muito rapidamente e portanto $k^{(F_n-1)/2}$ crescerá ainda mais, o que torna esse teste difícil de aplicar para números de Fermat com n grande. Esse fator abre espaço para o professor discutir inclusive as limitações da tecnologia atual e fomentar uma discussão sobre o papel da matemática como mecanismo para resolver problemas com condições especiais, ou seja, o quão importante é o estudo das ciências matemáticas.

Para implementar o Teste de Pépin é relativamente simples, dado que terá um pequeno número de etapas, e resulta em algo similar a Figura 52.

```
# -*- coding: utf-8 -*-
def teste_pepin(n):
    fermat = (2**(2**n))+1
    expoente = (2**(2**n))/2
    if ((3**expoente)+1) % fermat == 0:
        return "\nO número F(%d)=%d é primo." %(n, fermat)
    else:
        return "\nO número F(%d)=%d é composto." %(n, fermat)

print teste_pepin(input("Qual o índice do número de Fermat gostaria de "+
                        "testar a primalidade?\n"))

#Ou uma lista de naturais a serem testados como:
for x in range(1,6):
   print teste_pepin(x)
```

Figura 52: *Python* - Teste de Pépin

```
Qual o índice do número de Fermat gostaria de testar a primalidade?
4

O número F(4)=65537 é primo.
```

Figura 53: *Python* - resultado da primalidade de F(4) pelo Teste de Pépin.

Com isso, é possível verificar que apesar de o teste ser útil é demorado para fornecer seu resultado, dada a dimensão dos números de entrada, mas ainda é útil o uso dos números de Fermat para, inclusive, testar os demais algoritmos implementados pelos alunos, evidentemente para números de Fermat que não sejam tão grandes[40].

Assim, fica concluída as discussões sobre as temáticas: divisibilidade, números primos e fatoração, ou seja, uma introdução a teoria dos números ainda no ensino médio, mas que suscitam diversos temas do currículo "comum" do ensino médio: variável, conjuntos domínio e imagem, função e relação, lógica e demonstrações, sendo estes temas intrínsecos a Matemática como um todo e habilidades aplicáveis a outras áreas do saber.

[40] Neste caso, a partir de $n \geq 5$ o tempo de resposta passa a ser horas.

5 CONSIDERAÇÕES FINAIS

Ao refletir sobre o uso da tecnologia pela sociedade, é notório seu aumento em todos os setores. Principalmente para automatização de processos, por facilitar o conhecimento de informação em tempos muito inferiores antes da automatização. Para um ente gerencial, seja do setor público ou privado é essencial para tornar as tomadas de decisão coerentes com a realidade e de forma a minimizar possíveis problemas, tornando seus projetos mais coerentes com o planejamento.

Assim, é cada vez mais visível a utilização de tecnologias pelas diversas áreas do conhecimento, seja por engenheiros, físicos, químicos, matemáticos entre outras profissões devido a sua capacidade de trazer cálculos ou ambientes de simulação, difíceis de se obter sem o uso de ferramenta computacional.

Neste cenário, é de se conjecturar que a escola como entidade que deve formar um cidadão apto a se inserir em seu meio, bem como ascender a educação superior deve ter conhecimentos que lhe permitam, não só consumir tecnologia, mas fazer de forma crítica, ou seja, usar com o intuito de melhorar sua qualidade de vida e da sociedade em que vive.

A partir disso, é possível apontar que a dissertação atende a seu objetivo, ou seja, apresenta uma sequência didática que pode desenvolver os conceitos de variável e função como meios de relacionar grandezas, utilizando-se programação de algoritmos para discutir temas da teoria dos números, com o uso das linguagens Python e Scratch. Bem como, possibilita o desenvolvimento da capacidade de generalizar e abstrair, sob a perspectiva do PMA, dado que o professor poderá partir de casos particulares, sempre em busca de uma regra que valha para um conjunto maior.

Com o saber proveniente do desenvolvimento da lógica e seus conectivos e sentenças o aluno, inclusive passe a compreender melhor a Matemática como um sistema ordenado e regido por definições e teoremas que seguem os mesmos princípios. Ou seja, o aluno poderá usufruir desse conhecimento para facilitar a compreensão de entes matemáticos, sejam: axiomas, teoremas, proposições, corolários, lemas ou definições

É necessário pensar em uma escola voltada para o aprendizado de competências, pois a atual realidade exige cada vez mais um ser pensante capaz de adaptar-se a cenários distintos de trabalho. De aprender de forma autônoma e com grande capacidade de resolver problemas. Sendo assim, o tema aqui abordado pode suscitar essas habilidades e desenvolver uma forma de ensino na qual o professor deixa de ser o centro do processo para auxiliar e conduzi-lo.

O professor passa a imergir os alunos em uma Matemática com sentido que pode ser percebida e discutida dentre os temas da sociedade que faz uso de equipamentos digitais, para segurança financeira e pessoal. Ou seja, promovendo a integração da Matemática a outras áreas do conhecimento e abrindo a possibilidade para a interdisciplinaridade.

A sequência propicia desde uma introdução ao tema de teoria dos números podendo ser utilizada para aprofundar o tema em si e outros relacionados como: o conceito de variável, função, domínio, imagem dentre outros elementos da álgebra.

A parte de lógica necessária para a tradução dos conceitos em algoritmos implementados, expostos no trabalho, fará com que o processo de desenvolvimento lógico necessário para diversas situações cotidianas e para as aulas de Matemática possam ser encaradas com outro ponto de vista, facilitando a compreensão de teoremas, definições, axiomas e mesmo enunciados de questões pertencentes ao conjunto de temas explorados na educação básica.

A relevância deste trabalho não é apenas voltada para o aluno, mas também para o professor, ao passar a ter mais uma alternativa para encarar as dificuldades inerentes ao ensino de Matemática. Em que as implementações desenvolvidas pelo presente autor e apresentadas durante todo o trabalho podem ser um guia para o professor ter suas primeiras experiências com o ensino de matemática com o auxílio de tecnologia computacional.

Com isso, o docente pode abrir espaço para outras formas e usos da ferramenta que pode ser elemento motivador para o ensino de uma ciência tão necessária e ao mesmo tempo, temida pelos discentes.

É sabido que o uso do computador não resolverá todos os problemas, uma vez que Giraldo e Carvalho (2008) apontam a existência de estudos que mostram que a inserção da tecnologia não contribui para o desenvolvimento da aprendizagem matemática, porém eles relacionam esse fato com a abordagem que foi feita com tecnologia. Que não pode ser vista

somente como uma calculadora, mas como um elemento que leve o aluno a refletir sobre suas ações e saberes com os quais está lidando.

Desta forma, encerra-se este trabalho que se estima ser uma fonte agradável para professores que desejam inovar em suas práticas e refletir sobre as mesmas. E, ainda, ser fonte de pesquisa para outros que também buscam fazer o ensino de matemática um prazer para a comunidade discente. Apesar do encerramento deste trabalho, perguntas provenientes das ações apresentadas surgem, abrindo espaço para pesquisas que analisam uma série de alternativas, como:

- é possível desenvolver situações que envolvam outros temas matemáticos que atendam ao mesmo fim?
- quais implicações para a aprendizagem dos demais temas do currículo tem a sistemática apresentada?

Questões como estas, entre outras, trarão elucidações cada vez melhores para a realidade da educação auxiliada por ferramentas computacionais, principalmente se for pensado na aprendizagem individualizada, em que cada aluno aprende no seu próprio ritmo.

REFERÊNCIAS

ALMEIDA, Maria.Elizabeth B. de (org.). **ProInfo: informática e formação de professores**. Secretaria de Educação a Distância. Prolnfo - Brasília: Ministério da Educação, SEED. 2000 (Série de estudos / Educação a distância). Disponível em: <http://www.dominiopublico.gov.br/download/texto/me002401.pdf>. Acesso em: 28 nov 2013.

ALMEIDA, Maria Elizabeth B. de. Tecnologias na educação: dos caminhos trilhados aos atuais desafios. In. **Bolema: Boletim de Educação Matemática**, São Paulo, n.29. 2008. Disponível em: <http://www.periodicos.rc.biblioteca.unesp.br/index.php/bolema/article/download/1723/1497>. Acesso em: 29 fev. 2014.

ALTOÉ, Anair; PENATI, Marisa Morales. O Construtivismo e o Construcionismo Fundamentando a Ação docente. In: ALTOÉ, Anair; COSTA, Maria LuizaFurlan; TERUYA, Teresa Kazuko. **Educação e Novas Tecnologias**. Maringá: Eduem, 2005, p. 55-67. Disponível em: <http://www.dtp.uem.br/gepiae/pde/constru.pdf>. Acessado em: 20 jul. 2013.

BARBOSA, Gabriela dos Santos. **O teorema fundamental da aritmética:** problemas com alunos do sexto ano do ensino fundamental. São Paulo – SP, 2008, 308 f. Tese (Doutorado em Educação Matemática) Programa de Pós-graduação em Educação Matemática, Pontifícia Universidade Católica de São Paulo.

BARBOSA JUNIOR, José Hélio. **Congruências modulares**: construindo um conceito e as suas aplicações no ensino médio. São Cristóvão – SE, 2013, 51 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Programa de Pós-graduação em Matemática, Universidade Federal de Sergipe.

BARROS, Ana Paula M. R. STIVAM, Elen Priscila. O software Geogebra na concepção de micromundo. In. **Revista do Instituto Geogebra Internacional de São Paulo (IGISP)**, São Paulo, n. 1, v. 1, 2012 . Disponível em:<http://revistas.pucsp.br/index.php/IGISP/article/download/8388/6915>. Acesso em: 08 ago. 2013.

BORGES, Luiz Eduardo. **Python para desenvolvedores**. 2. ed. Rio de Janeiro: Edição do autor, 2010.

CANGUSSÚ, Everton Soares. **O ensino de sequências de recorrências na educação básica com o auxílio de linguagem de programação**. São Luís – MA, 2013, 70 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Departamento de Matemática, Universidade Federal de Rondônia.

CARVALHO, Glauber Cristo Alves de. **Números primos:** pequenos tópicos. Goiânia – GO, 2013, 41 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Programa de Pós-graduação Matemática em Rede Nacional, Instituto de Matemática e Estatística, Universidade Federal de Goiás.

COUTINHO, S. C. **Números inteiros e criptografia RSA**. 2. ed. Rio de Janeiro: IMPA, 2013.

DAINEZE, Kelly Cristina S. A. de L. **Números primos e criptografia: da relação com a educação ao sistema RSA**. Seropédica-RJ, 2013, 42 p. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Instituto de Ciências Exatas, Universidade Federal Rural do Rio de Janeiro.

DIAS, Cristina Helena B. Batista. **Números primos e divisibilidade**: estudo de propriedades. Rio Claro – SP, 2013, 49 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Programa de Pós-graduação em Matemática em Rede Nacional, Universidade Estadual Paulista.
.
DUQUE, Raúl Gonzales. **Python para todos**. Espanha: Edição do autor, 2011.

ESQUINCA, Josiane Colombo Pedrini. **Aritmética**: códigos de barras e outras aplicações de congruências. Campo Grande – MS, 2013, 72 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) Programa de Pós-graduação Matemática em Rede Nacional, Universidade Federal de Mato Grosso do Sul.

EUCLIDES. **Os elementos**. Tradução por Irineu Bicudo. São Paulo: UNESP, 2009.

FONSECA, Daila S. S. de M.; FRANCHI, Regina Helena de O. L. O uso do pensamento matemático avançado e de tecnologias na aprendizagem de séries infinitas. **Anais XV EBRAPEM**, Campina Grande – PB, 201. Disponível em < http://www.editorarealize.com.br/revistas/ebrapem/trabalhos/e95fd0c97716c41454166916270980c6.pdf >. Acesso em: 15 mai. 2014.

FORIGO, Franciele Meinerz. **O ambiente de programação Pascalzim como ferramenta de auxílio para ensino de Matemática**. Santo Ângelo – RS, 2012, 174 f. Dissertação (Mestrado em Ensino Científico e Tecnológico) Programa de Pós-graduação em Ensino Científico e Tecnológico, Universidade Regional Integrada do Alto Uruguai e das Missões.

GATTI, Daniel Couto. **Ensino de programação**: a modelagem como estratégia para ampliar a compreensão dos alunos. São Paulo – SP, 2009, 144 f. Tese (Doutorado em Educação Matemática) Programa de Pós-graduação em Educação Matemática, Pontifícia Universidade Católica de São Paulo.

GERETI, Laís Cristina Viel. et al. Pensamento matemático avançado e pensamento algébrico evidenciados em tarefas de sistemas de equações lineares. In. VII Congresso Iberoamericano de Educação matemática (CIBEM), 7., 2013, Montevidéu (Uruguai). **Anais do VII CIBEM**. Montevidéu, 2013. p. 1920-1928.

GIRALDO, Victor; CARVALHO, Luiz Mariano. Uma breve revisão bibliográfica sobre o uso de tecnologia computacional no ensino de matemática avançada. In. CARVALHO, Luiz Mariano *et. al.* **História e tecnologia no ensino de matemática, vol. 2**. Rio de Janeiro – RJ: Editora Ciência Moderna Ltda, 2008, p. 153-206.

GIORDAN, Marcelo. O computador na educação em ciências: breve revisão crítica acerca de algumas formas de utilização. **Ciência & Educação**, Bauru, v. 11, n. 2, p. 279-304, 2005.

HENRIQUES, Ana Cláudia C. B. **O pensamento matemático avançado e a aprendizagem da análise numérica num contexto de actividades de investigação**. 2010. Tese (Doutorado em Educação - Didáctica da Matemática). Instituto de Educação, Universidade de Lisboa, Lisboa (Portugal).

KAWASAKI, Teresinha Fumi. **Tecnologias na sala de aula de matemática**: resistência e mudanças na formação continuada de professores. Belo Horizonte – MG, 2008, 212 f. Tese (Doutorado em Educação) Programa de Pós-graduação, Conhecimento e Inclusão Social. Faculdade de Educação, Universidade Federal de Minas Gerais.

LUZ, Welington Batista. **Introdução à Matemática do Criptossistema RSA**. São Cristóvão-SE, 2013, 50 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Universidade Federal de Sergipe.

MACHADO, Silvia D. A.; BIANCHINI, Bárbara L. Aportes dos processos do pensamento matemático avançado para a reflexão do professor sobre sua "forma" de pensar matemática. **Educação Matemática Pesquisa**, São Paulo, v. 15, n. 3, p. 590-605, 2013.

MARQUES, Maria Teresa P. M. **Recuperar o engenho a partir da necessidade, com recurso às tecnologias educativas: Contributo do ambiente gráfico de programação Scratch em contexto formal de aprendizagem**. Lisboa, 2009. 198 p. Dissertação (Mestre em Ciências da Educação). Faculdade de Psicologia e de Ciências da Educação, Universidade de Lisboa. Disponível em: <http://eduscratch.dgidc.min-edu.pt/index.php?option=com_docman&task=doc_download&gid=43&Itemid=17>. Acesso em: 15 ago. 2013.

MARQUES, Thiago Valentim. **Criptografia**: abordagem histórica, protocolo Diffie-Hellman e aplicações em sala de aula. João Pessoa – PB, 2013, 75 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Programa de Pós-graduação em Matemática em Rede Nacional, Universidade Federal da Paraíba.

MARTINEZ, Fabio Brochero; et al. **Teoria dos números:** um passeio com primos e outros números familiares pelo mundo inteiro. 2. ed. Rio de Janeiro: IMPA, 2013.

MARTINS, Amilton Rodrigo de Quadros. **Usando o Scratch para potencializar o pensamento criativo em crianças do ensino fundamental**. Passo Fundo – RS, 2012, 113 f. Dissertação (Mestrado em Educação) Programa de Pós-graduação em Educação, Universidade de Passo Fundo.

MISKULIN, Rosana G. S. **Concepções teórico-metodológicas sobre a introdução e a utilização de computadores no processo ensino/aprendizagem da geometria**. Campinas, 1999. 577 p. Tese (Doutorado em Educação). Faculdade de Educação, Universidade Estadual de Campinas.

MORAIS FILHO, Daniel Cordeiro. **Convite à Matemática**. Rio de Janeiro: IMPA, 2012.

MOTTA, Marcelo S; SILVEIRA, Ismar F. Contribuições do Superlogo ao ensino de geometria. In. **Informática na Educação: teoria e prática**, Porto Alegre, n. 1, v. 13, 2010 . Disponível em: <http://seer.ufrgs.br/InfEducTeoriaPratica/article/view/9142/12036>. Acesso em: 18 dez. 2013.

OKUMURA, Mirella Kiyo. **Números primos e criptografia RSA**. São Carlos – SP, 2014, 41 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) Programa de Pós-graduação Matemática em Rede Nacional, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

OLGIN, Clarissa de Assis. **Currículo no ensino médio:** uma experiência com o tema criptografia. Canoas – RS, 2011, 136 f. Dissertação (Mestrado em Ensino de Ciências e Matemática) Programa de Pós-graduação em Ensino de Ciências e Matemática, Universidade Luterana do Brasil.

OLIVEIRA, Maykon Costa de. **Aritmética**: criptografia e outras aplicações de congruências. Campos Grande – MS, 2013, 73 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) Programa de Pós-graduação Matemática em Rede Nacional, Universidade Federal de Mato Grosso do Sul.

POCRIFKA, Dagmar H. SANTOS, Taís W. Linguagem logo e a construção do conhecimento. In. **IX Congresso nacional de educação (EDUCERE) e III Encontro sul brasileiro de Psicopedagogia**. Curitiba, out. 2009 (26 a 29). Disponível em: <http://www.pucpr.br/eventos/educere/educere2009/anais/pdf/2980_1303.pdf>. Acesso em: 02 ago. 2013.

RAMOS, Reinaldo Augusto de Oliveira. **O uso de mídias interativas na compreensão de conceitos da lógica computacional**. São Paulo – SP, 2011, 90 f. Dissertação (Mestrado em Tecnologias da Inteligência e Design Digital) Programa de Pós-graduação em Tecnologias da Inteligência e Design Digital, Pontifícia Universidade Católica de São Paulo.

RIBENBOIM, Paulo. **Números primos:** velhos mistérios e novos recordes. Rio de Janeiro: IMPA, 2012.

SANTANCHÈ, André; TEIXEIRA, Cesar Augusto C. Integrando instrucionismo e construcionismo em aplicações educacionais através do casa mágica. In. **WIE/SBC**, Rio de Janeiro, jul. 1999. Disponível em: <http://www.lis.ic.unicamp.br/~santanch/publications/WIE99-CasaMagica.pdf >. Acesso em: 12 dez. 2013.

SANT'ANNA , Iury Kersnowsky de. **A aritmética modular como ferramenta para as séries finais do ensino fundamental**. Rio de Janeiro – RJ, 2013, 36 f. Dissertação (Mestrado em Educação Matemática) Instituto de Matemática Pura e Aplicada.

SANTOS, José Plínio de Oliveira. **Introdução à teoria dos números**. Rio de Janeiro: IMPA, 2007.

SANTOS, José Luiz dos. **A arte de cifrar, criptografar, esconder e salvaguardar como fontes motivadoras para atividades de matemática básica**. Salvador – BA, 2013, 81 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) Programa de Pós-

graduação Matemática em Rede Nacional, Instituto de Matemática, Universidade Federal da Bahia.

SETZER, Waldemar W. **Computadores na educação**: por quê, quando e como. 1998. Disponível em: <http://www.ime.usp.br/~vwsetzer/PqQdCo.html>. Acesso em: 04 mar. 2014

SIQUEIRA, Fábio Rodrigues de. **A programação no ensino médio como recurso de aprendizagem dos zeros da função polinomial do 2º grau**. São Paulo – SP, 2012, 125 f. Dissertação (Mestrado em Educação Matemática) Programa de Pós-graduação em Educação Matemática, Pontifícia Universidade Católica de São Paulo.

SPENTHOF, Roberto Luiz. **Primos**: da aleatoriedade ao padrão. Maringá – PR, 2013, 33 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) Programa de Pós-graduação Matemática em Rede Nacional, Centro de Ciências Exatas, Universidade Estadual de Maringá.

SUMMERFIELD, Mark. **Programação em Python 3:** uma introdução à linguagem Python. Tradução por Fabiane Fiorin. Rio de Janeiro: Alta Books, 2012.

VALENTE, José Armando. **A espiral da espiral de aprendizagem: o processo de compreensão do papel das tecnologias de informação e comunicação na educação**. Campinas, 2005. 232 p. Tese (Livre Docência). Instituto de Artes, Universidade Estadual de Campinas. Disponível em:<http://www.bibliotecadigital.unicamp.br/document/?down=000857072&idsf=>. Acesso em: 10 jan. 2014.

VECCHIA, Rodrigo Dala. **A modelagem Matemática e a realidade do mundo cibernético**. Rio claro – SP, 2012, 275 f. Tese (Doutorado em Educação Matemática) Programa de Pós-graduação em Educação Matemática. Universidade Estadual Paulista.