UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

MICHAEL DOUGLAS GONÇALVES NÓBREGA

UM PROGRAMA DE ESTUDOS BASEADO EM PROBLEMAS DE PROGRAMAÇÃO PARA ALGORITMOS MATEMÁTICOS

QUIXADÁ

2022

MICHAEL DOUGLAS GONÇALVES NÓBREGA

UM PROGRAMA DE ESTUDOS BASEADO EM PROBLEMAS DE PROGRAMAÇÃO PARA ALGORITMOS MATEMÁTICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Wladimir Araújo Tavares

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

N675p Nóbrega, Michael Douglas Gonçalves.
Um programa de estudos baseado em problemas de programação para algoritmos matemáticos / Michael Douglas Gonçalves Nóbrega. – 2022.
63 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Computação, Quixadá, 2022.
Orientação: Prof. Dr. Wladimir Araújo Tavares.

1. Aprendizagem por descoberta. 2. Aprendizagem baseada em problemas. 3. Algoritmos. 4. Matemática-computação. I. Título.

CDD 621.39

MICHAEL DOUGLAS GONÇALVES NÓBREGA

UM PROGRAMA DE ESTUDOS BASEADO EM PROBLEMAS DE PROGRAMAÇÃO PARA ALGORITMOS MATEMÁTICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Aprovada em: ____/____/_____

BANCA EXAMINADORA

Prof. Dr. Wladimir Araújo Tavares (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Henrique Macêdo de Araújo
Universidade Federal do Ceará (UFC)

Prof. Dr. Fábio Carlos Sousa Dias
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Gostaria de agradecer primeiramente para minha mãe, Ana Cleide, que sempre esteve presente na minha vida e me incentivou em todos os momentos. Por sempre fazer tudo por mim sem medir sacrifícios, por ser uma mãe mais do que incrível e me ensinar a nunca desistir dos meus sonhos.

Também queria agradecer a todos de minha família, com ênfase na minha avó Anacleta, que sempre me apoiou, cuidou de mim como se fosse seu filho e por sempre querer me fazer feliz.

Ao Prof. Dr. Wladimir Araújo Tavares por me orientar em meu trabalho de conclusão do curso, apresentar a maratona de programação e por ser meu coach nessa competição que teve um impacto imenso na minha vida.

Aos meus amigos do grupo Resistência, Alan Nascimento, David Tavares, Paulo Miranda, Gregório Neto, Jorge Lucas, Natan Nobre e Ruan Felipe por todo o apoio durante o curso e por fazerem esses anos divertidos e leves.

Aos integrantes da minha equipe da Maratona de Programação, Claro Henrique e Paulo Miranda, por todo o tempo estudado, toda a dedicação nessa competição e por me ajudarem a alcançar o nosso sonho de ir a mundial desta competição.

A minha namorada, Erika Lopes, por fazer esse ultimo ano da graduação mais tranquilo e feliz.

Ao meu melhor amigo, João Victor Da Silva Simão, por estar sempre presente enquanto eu precisava e por sempre acreditar no meu potencial.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Hoje, posso dizer que uma das minhas melhores decisões de minha vida foi escolher o campus Quixadá da Universidade Federal do Ceará (UFC) e ter participado da maratona de programação. Obrigado a todos que fizeram parte dessa parte da minha história.

“Se você não gosta do seu destino, não o aceite. Em vez disso, tenha a coragem de mudá-lo do jeito que você quer que ele seja.”

(Naruto Uzumaki)

RESUMO

Uma dificuldade recorrente na vida dos estudantes que entram em cursos que envolvem tecnologia da informação é a aprendizagem de algoritmos. O método tradicional foca pouco no desenvolvimento do pensamento algorítmico. O foco deste TCC é reverter essa dificuldade através de um programa de estudo para algoritmos matemáticos baseado em problemas. A técnica de aprendizagem baseada em problemas tem como princípio o estímulo da criatividade e da curiosidade do estudante através da disponibilização de problemas relacionados com o assunto apresentado. Os problemas são divididos em 3 níveis, sendo eles: introdução, reforço e síntese. Os níveis de reforço e síntese requerem um conhecimento maior sobre o assunto ou alguma percepção de uma característica importante, estimulando assim buscas em outras fontes e o pensamento lógico e criativo do estudante. Todos os problemas escolhidos são provenientes de competições de programação, as quais requerem um conjunto de habilidades do aluno, como criatividade, conhecimento teórico e a capacidade de escrever algoritmos eficientes em memória e tempo.

Palavras-chave: Aprendizagem por descoberta. Aprendizagem baseada em problemas. Algoritmos. Matemática-computação.

ABSTRACT

A recurring difficulty in the lives of students who enter courses involving information technology is learning algorithms. The traditional method does not focus on developing algorithmic thinking. However, the focus of this TCC is to reverse this difficulty through a program of study for mathematical algorithms based on problem-based learning. The problem-based learning technique has as its principle the stimulation of the student's creativity and curiosity through the provision of problems related to the presented subject. The problems are divided into 3 levels: introduction, reinforcement and synthesis. Reinforcement and synthesis levels require a greater knowledge of the subject or some perception of an important characteristic, thus stimulating searches in other sources and also the student's logical and creative thinking. All problems chosen come from programming competitions, which require a set of skills, such as creativity, theoretical knowledge and the ability to write memory and time efficient algorithms.

Keywords: *Discovery Learning. Problem-based Learning. Algorithm. Mathematics-computing.*

LISTA DE ABREVIATURAS E SIGLAS

SBC	Sociedade Brasileira de Computação
OBI	Olimpíada Brasileira de Informática
IOI	Olimpíada Internacional de Informática
ABP	Aprendizagem Baseada em Problemas
APD	Aprendizagem Por Descoberta
ICPC	*International Collegiate Programming Contest*

SUMÁRIO

1 INTRODUÇÃO

> A programação pode ser um processo doloroso para os programadores iniciantes. Eles devem possuir habilidades de memorização, compreensão, resolução de problemas, abstração, capacidade de pensamento lógico, conhecimentos declarativos e procedimentais e entre outros. (COSTA; PITEIRA, 2013, tradução nossa)

No ensino superior, geralmente, o estudo da programação é realizada através do método tradicional. Neste método, o professor escolhe uma linguagem de programação em específico e todos os alunos devem aprender a utilizar esta linguagem e, em simultâneo, desenvolver o pensamento algorítmico. O pensamento algorítmico pode ser entendido como uma forma de analisar um problema tendo como objetivo encontrar uma solução utilizando o computador.

Para Piteira e Haddad (2011), o processo de aquisição de habilidades relacionadas a programação é de grande importância para os cursos de informática. Entretanto, apesar de estas habilidades serem necessárias para solucionar problemas e estar relacionada com o desempenho da vida profissional, o nível de motivação dos alunos para aprender programação é baixo e existe uma alta taxa de desistência. Por si só, problemas de programação já tem uma alta complexidade para serem resolvidos, na maioria das vezes essa dificuldade está correlacionada com a utilização de conhecimentos prévios, como estrutura de dados, matemática, técnicas de resolução de problemas e entre outros. De acordo com Martin (2000), a resolução de problemas de programação que contém algum passo matemático é um processo que envolve as faculdades contemplativas humanas e é abstrato e complexo, por isso, quando pensamos na resolução desses problemas é perceptível um aumento na dificuldade.

De acordo com Bosse e Gerosa (2015), o índice de reprovação da disciplina introdutória a programação na USP nos anos entre 2010 e 2014 foi de 30 por cento, entretanto em alguns casos chegou a 50 por cento da turma. A autora complementa que esses dados se mantiveram constantes ao longo dos anos e que 25 por cento dos aprovados cursaram a disciplina de programação pelo menos 2 vezes.

Combefis *et al.* (2017) defendem o desenvolvimento de estratégias de ensino que possibilitem a dominação da arte de projetar algoritmos eficientes pelos futuros cientistas e engenheiros. Hidayah *et al.* (2019) argumentam que o ensino de algoritmos requer uma abordagem diferente da abordagem convencional. Eles complementam que a habilidade de projetar e implementar algoritmos deve ser desenvolvida pela experimentação, tentativa e erro.

Apesar da estreita relação, existe uma diferença entre o ensino da programação

tradicional e o ensino do pensamento algorítmico. Christodoulou *et al.* (2018) alertam que os algoritmos são independentes da linguagem utilizada e cada algoritmo pode ser expresso em diferentes linguagens de programação.

Recentemente, o termo *algoritmic programming* está sendo utilizado para indicar quando o objetivo da programação é resolver problemas de natureza algorítmica, ou seja, problemas que podem ser resolvidos através da utilização de algoritmos. Tais problemas são bem definidos com uma saída desejada para cada entrada dada. As soluções dos problemas exigem o conhecimento de técnicas de solução de problema, estrutura de dados e algoritmos. Cada problema possui restrições com relação ao gasto de memória e o tempo de execução.

Esses problemas com restrições ao gasto de memória e o tempo de execução são amplamente utilizados em competições de programação, como a Maratona de Programação organizada pela Sociedade Brasileira de Computação (SBC) e a Olimpíada Brasileira de Informática (OBI) que serve como classificatória para Olimpíada Internacional de Informática (IOI). Essas competições não exigem que o candidato saiba uma linguagem específica dado que basta que o seu código resolva os problemas nas restrições estabelecidas previamente, valorizando assim o pensamento algorítmico.

Vale destacar que os problemas apresentados nessas competições são contextualizados exigindo que o competidor leia com atenção o problema, divida o problema complexo em partes isoladas, construa uma solução para cada parte, analise os elementos que são relevantes, modele uma solução para o problema como um todo e automatize essa solução através de um algoritmo. Observe que esses passos fazem parte do pensamento algorítmico. Esses problemas podem ser utilizados em conjunto com a metodologia ativa chamada Aprendizagem Baseada em Problemas (ABP).

A técnica de ensino conhecida como ABP, visa aprimorar a proatividade, estabelecer conhecimento aprofundado sobre determinado assunto, estimular a criatividade e o desenvolvimento de habilidades cognitivas do estudante. Essa técnica é muito útil para o pensamento algorítmico devido à utilização de problemas que estimulam a habilidade do aluno de identificar o assunto e o algoritmo que consegue resolver ele.

Atualmente existem plataformas que servem de repositórios de problemas, elas são conhecidas como juízes *online*, pois, qualquer usuário cadastrado pode submeter o código para analise, onde será submetido a vários casos de teste compostos de entradas e saídas. O código do usuário será considerado aceito se respeitar as restrições de tempo, memória e se todos os casos

de teste passarem. Alguns exemplos de juízes *online* seria o *CodeForces, CodeChef, TopCoder* e *HackerRank*.

Apesar de ser uma excelente forma de se treinar a parte do pensamento algorítmico existe uma dificuldade no aprendizado apenas utilizando essas plataformas, devido à falta de estruturação entre os assuntos e as dificuldades dos problemas. Desta forma, a falta de um roteiro para se seguir pode dificultar a evolução do aluno. Portanto, será desenvolvido um roteiro utilizando essas plataformas como fonte de questões.

Barrows (1996) listou algumas das principais características da utilização do ABP, sendo elas:

1. A aprendizagem é direcionada ao aluno;
2. Aplicada em grupos com poucos alunos;
3. Os professores são considerados mentores;
4. Estimular a aprendizagem é o objetivo principal dos problemas selecionados;
5. Através do roteiro de questões sugeridas ocorre o desenvolvimento das habilidades necessárias e conhecimentos para a resolução do problema.

Essas características são muito interessantes dado que podemos até utilizar as competições como estímulo adicional sabendo que algumas destas competições seguem o formato de equipes com 3 pessoas.

Neste trabalho é apresentado um programa de estudos para algoritmos matemáticos voltado aos alunos do ensino médio e do ensino superior, seguindo os princípios definidos pela metodologia ativa ABP. O programa de estudos é direcionado à aprendizagem de alguns tópicos da Aritmética Modular e utiliza um conhecimento básico relacionado à combinatória.

A aritmética modular atua no domínio dos números naturais. Para isso foi elaborado um conjunto de operações as quais garantem essa propriedade. A ideia base é realizar uma transformação para o domínio modular, onde o número a ser trabalhado será substituído pelo resultado do resto de sua divisão por um valor fixo qualquer maior que zero. A aritmética modular é amplamente utilizada em *hashing*, criptografia, teoria da computação, equações de congruência linear, cifra de Ceasar e entre outros.

Em cada tópico, apresentaremos uma explicação teórica com as devidas demonstrações. Em seguida, é proposto uma relação de problemas sugeridos dividido em 3 níveis sendo eles introdução, reforço e síntese que compõem o programa de estudos. Podemos ver um exemplo de programa de estudos no Quadro 1.

Quadro 1 – Exemplo de problemas sugeridos compondo um programa de estudos.

Nome	Introdução	Reforço	Síntese
Combinatória	CF — *JOE is on TV!*	CF — *Bus Number*	CF — *Gerald and Giant Chess*

Fonte: Quadro adaptada de Nikházy (2020b)

Os problemas introdutórios são exercícios que normalmente se dão pela aplicação direta do assunto ensinado, utilizados apenas para o estudante confirmar o entendimento e praticar.

Os problemas de reforço são exercícios que exigem um raciocínio lógico para identificar como resolver o problema com aquela técnica e/ou mostrar novas situações que aquele assunto pode ser aplicado.

Já os problemas de síntese são exercícios que exigem um alto domínio do assunto ou necessitam de algumas técnicas aplicadas em conjunto.

Todos os problemas escolhidos podem ser enviados e testados através das plataformas de juízes *online*.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver um plano de estudos para algoritmos matemáticos utilizando a ABP, uma das sub-técnicas da Aprendizagem Por Descoberta (APD), e analisar sua aplicação com alunos do primeiro ano do ensino superior.

1.1.2 Objetivos Específicos

Como objetivos específicos temos:

1. Definição da fundamentação teórica necessária para aplicação dos planos de estudos.
2. Definição do plano de estudos baseado na ABP com as devidas divisões entre níveis e resolução dos problemas com implementação e análise de complexidade de tempo e memória.
3. Análise do nível de comprometimento e satisfação da aplicação do método nos alunos.

1.2 Organização

No Capítulo 2 são abordados os fundamentos teóricos para facilitar o entendimento do trabalho. Em seguida, no Capítulo 3 são apresentados os trabalhos relacionados a este TCC. Logo após, no Capítulo 4 são apresentados os tópicos de aritmética e, em seguida, são apresentados o programa de estudos e a resolução dos problemas propostos. Por fim, no Capítulo 7 temos a conclusão deste trabalho.

1.3 Resultados esperados

Os resultados esperados para esta monografia é fornecer um programa de estudos para algoritmos matemáticos que dê maior autonomia para o estudante e estimule o estudante buscar novos conhecimentos. Além disso, devido aos problemas serem retirados de competições de programação e ter um incentivo à participação, espera-se que os grupos de estudantes tenham interesse nessas competições e comecem a praticar e participar.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta capítulo, os conceitos necessários para o desenvolvimento e compreensão do trabalho serão apresentados.

2.1 Aprendizagem Por Descoberta

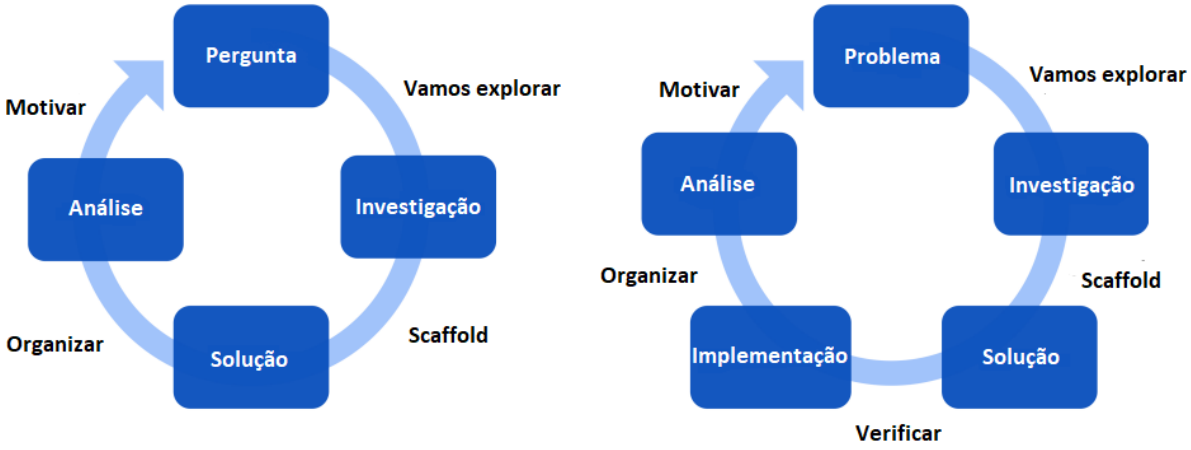
> Ensinar aos alunos a noção de capacidade de descoberta, pensamento crítico, questionamento e resolução de problemas é um dos princípios fundamentais do ensino de ciências e tecnologia. (BALIM, 2009, tradução nossa)

A Aprendizagem por descoberta é uma abordagem construtiva cujo foco é estimular o desenvolvimento intelectual do aluno, criando assim uma aprendizagem duradoura sobre o assunto em questão. Para alcançar esses objetivos essa metodologia ativa tem como principio estimular a curiosidade e a busca por mais informações sobre o assunto, desta forma transformando um assunto que antes era considerado chato em algo instigante.

Existem dois estilos de abordagens no processo de aprendizado, a primeira abordagem é seguir as instruções necessárias para chegar em determinado objetivo, como se fosse uma receita, já na segunda abordagem é proposto através da pesquisa e do estudo a formulação dos procedimentos a serem realizados para chegar ao objetivo. Svinicki (1998) argumenta que o nível de processamento das informações é mais aprofundado na segunda abordagem e afirma que esse processamento garante ao estudante uma melhor retenção do assunto, dado que para cada problema a ser analisado e solucionado gera um conjunto de procedimentos únicos para o aluno.

Observando a Figura 1, podemos concluir que o método de aprendizagem para matemática e para programação são bastante similares apenas se diferenciando na parte da implementação do algoritmo que solucione o problema. O primeiro passo a ser feito é analisar o problema, identificar todas as dicas e informações fornecidas de forma direta ou indireta, olhar as contantes, o tempo máximo para a solução e entre outras informações relacionadas as variáveis para assim ter uma noção da complexidade esperada do algoritmo, fazendo assim uma investigação minuciosa. O segundo passo é o *scaffold*, que na aprendizagem esse termo inglês se refere ao estágio inicial ao ser submetido a uma experiência pela primeira vez. Nessa etapa será criado um método definindo os procedimentos a serem seguidos para chegar no objetivo final. O terceiro passo seria verificar o método através de casos de teste que podem quebrar a lógica desenvolvida para assim partir para a implementação do método em um algoritmo e por enfim

Figura 1 – Exemplo de processo da aprendizagem por descoberta na matemática (esquerda) e na programação (direita).



Fonte: Nikházy (2020a)

organizar todas as ideias para fazer uma análise e identificar o que foi aprendido nesse problema servindo como motivação para continuar no aprendizado.

Nikházy (2020a) argumenta que a parte de implementação do código é bastante importante, devido ser um meio de verificar sua solução. Enquanto no processo da matemática um professor teria que analisar sua solução e por fim definir se está correta ou não, no caso da programação isso pode ser feito automaticamente com a ajuda do computador. Além disso, durante a etapa de implementação o estudante pode acabar por identificar problemas na lógica desenvolvida.

Unir o estudo de programação com o estudo de matemática fornece mais benefícios ainda para a aprendizagem autodirigida e a aprendizagem por descoberta, pois gera mais autonomia ao estudante. Além disso, a etapa de implementação auxilia na construção do pensamento algorítmico e com isso o estudante consegue trabalhar duas áreas simultaneamente.

2.2 Aprendizagem Baseada em Problemas

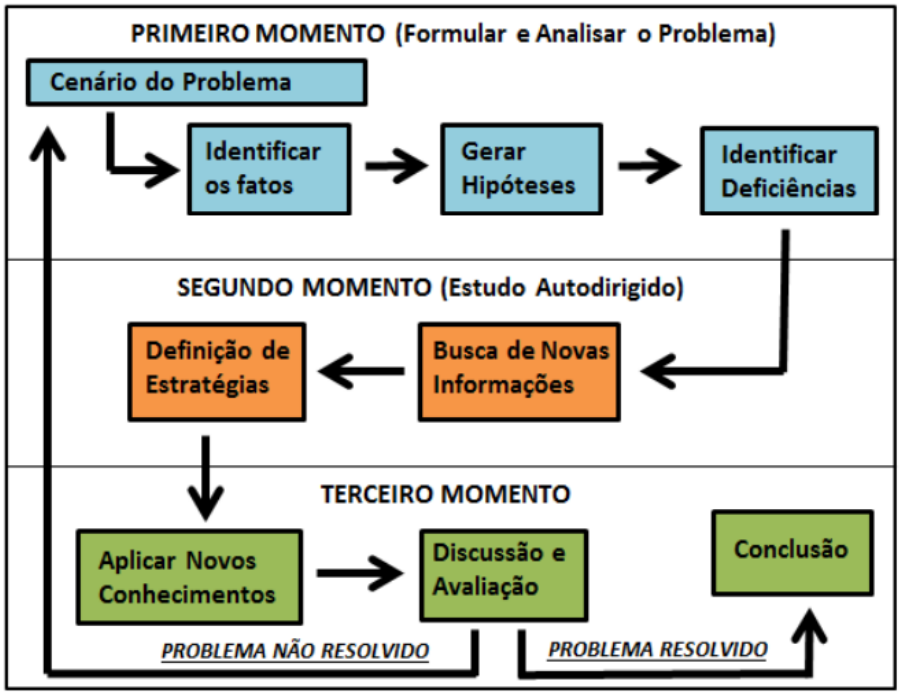
A aprendizagem baseada em problemas por ser uma sub-técnica da aprendizagem por descoberta contém boa parte de suas características. A forma de estimular a construção do raciocínio nessa técnica é por questões que possuem relações com o nosso dia a dia. Essa metodologia estimula o desenvolvimento de habilidades de *Problem Solving* que está altamente relacionado com a programação tornando assim uma excelente metodologia de ensino.

Wood (2003) listou um conjunto de características que auxiliam a formar um cenário

favorável para a aprendizagem baseada em problemas, sendo elas:

- Os problemas propostos devem estar adequados ao nível de conhecimento dos estudantes;
- Cenário moldado para estimular a curiosidade dos alunos na busca de informações diversas;
- Os exercícios devem despertar interesse intrínseco para a aprendizagem do aluno;
- As questões precisam ser contextualizadas com o intuito de facilitar a integração do conhecimento.

Figura 2 – O ciclo de aprendizagem na Aprendizagem Baseada em Problemas.



Fonte: Pierini e Lopes (2017)

Esta monografia utilizará desses passos a passos na criação do plano de estudos após as devidas adaptações para o uso na área de algoritmos matemáticos. Desta forma tira-se maior proveito da metodologia ativa.

2.3 Pensamento Algorítmico

Para compreendermos o pensamento algorítmico o primeiro passo é definir o que é um algoritmo.

> Na computação um algoritmo é uma sequência precisa que pode ser utilizada por um computador para a solução de um problema. O algoritmo é composto por um conjunto finito de passos, cada um requerendo uma ou mais instruções. Cada uma destas instruções, também chamadas de operações, deve ser perfeitamente definida e clara. (LOPES; MELO, 2002)

Portanto, o pensamento algorítmico é toda a parte de raciocínio lógico para se desenvolver um algoritmo independente da linguagem escolhida. Ou seja, é a capacidade de solucionar um problema específico descrevendo um conjunto finito de instruções a serem seguidas.

Na concepção de Futschek (2006), o pensamento algorítmico é um conjunto de habilidades que envolvem entendimento e construção de algoritmos. O mesmo listou uma série de habilidades que ele considera importantes sendo elas:

1. Extrair informações a partir da análise do problema;
2. Identificar com precisão as demandas necessárias para chegar a solução e listá-las;
3. Construir um algoritmo para solucionar as demandas identificadas;
4. Detectar casos especiais que o algoritmo pode falhar;
5. Aprimorar a eficiência de um algoritmo.

Com isso podemos notar a importância do pensamento algorítmico para a aprendizagem. Entretanto, existe uma correlação muito grande entre pensamento algorítmico e a criatividade, dado que todas as habilidades citadas previamente envolvem diretamente o pensamento criativo do estudante. "Como ensinar alguém a ser criativo?" é uma das principais perguntas que temos que nós fazer ao ensinar o pensamento algorítmico e uma das possíveis respostas seria resolver muitos problemas para estimular a criatividade direcionada e isso possibilita uma conexão com a metodologia ativa ABP.

De acordo com Luo e Zheng (2021), uma abundante quantidade de treinamento de algoritmos clássicos e de treinamento do pensamento computacional são extremamente necessários para os estudantes que querem ser programadores de alto nível. Ele complementa citando os benefícios das competições de programação para o treinamento desses estudantes, como:

1. Habilidade elevada em pensamento lógico e computacional, devido a problemas complexos serem frequentemente compostos de assuntos diversificados e técnicas de programação;
2. Conhecimento abundante em algoritmos, os estudantes que planejam se classificar para as próximas fases devem compreender uma vasta gama de algoritmos e utilizá-los da melhor forma para solucionar uma questão;
3. Espirito de equipe, devido às competições ocorrerem em equipes os participantes experienciam a coletividade que no futuro ajudará na vida profissional;
4. Naturalidade em escrever códigos, o autor comenta que para uma equipe se classificar

nas regionais da *International Collegiate Programming Contest* (ICPC), os membros que compõem a equipe precisam escrever de 50 a 100 mil códigos de soluções.

Portanto, a utilização dos problemas que normalmente aparecem nessas competições são proveitosos para o pensamento algorítmico já que para solucionar um problema é necessário ter diversos referenciais e imaginar os procedimentos de unificação de vários conhecimentos distintos.

3 TRABALHOS RELACIONADOS

Neste capítulo serão introduzido os trabalhos relacionados a este TCC.

3.1 *A Problem-based Curriculum for Algorithmic Programming*

Em Nikházy (2020b), a autora introduz os conceitos básicos de aprendizagem por descoberta e aprendizagem baseada em problemas para desenvolver um plano de estudos direcionado a jovens talentos e alunos do ensino médio com o objetivo de auxiliar na aprendizagem de programação do nível básico ao nível mais avançado. Ela também cita alguns acampamentos de matemática que existem na Hungria, especificamente o *The Joy of Thinking foundation* liderado pelo Lajos Pósa e comenta sobre o *Pósa Method* que é uma variação do *Problem-based Learning*. No fim, é apresentado um roteiro de problemas baseado nos princípios do *Problem-based learning* para otimizar o desenvolvimento da programação algorítmica.

Este trabalho também utiliza-se do conceito de *Discovery Learning* e *Problem-based Learning* para produzir um roteiro de problemas. A principal diferença é que este trabalho é focado em algoritmos que estão interligados com tópicos matemáticos, também será elaborada uma explicação demonstrativa dos assuntos selecionados antes das questões serem apresentadas. Além disso, todos os problemas propostos serão resolvidos didaticamente para que seja possível esclarecer as dúvidas e a compreensão da solução.

3.2 *Planets: A System for Autonomous Learning of Algorithmic Programming*

Em Nikházy (2020a), a autora propôe a construção de um *site* chamado *Planets* que tem como princípio o ensino da programação através de uma gamificação de um roteiro de estudos. No *site* os planetas representaram tópicos a serem estudados e cada problema sugerido representa uma tarefa ou desafio apresentado pelos moradores daquele planeta. Após você concluir todos os desafios os outros planetas serão liberados conforme a ordem estipulada pelo autor. O roteiro de estudos e suas respectivas dependências foram criados baseado em alguns conceitos como *Discovery Learning* e *Pósa Method*. Essa categoria de roteiro de problemas a serem solucionados tem como intuito estimular o desenvolvimento do conhecimento através de uma participação mais ativa do estudante e com o auxílio do site o aluno consegue aprender mesmo sem ter um professor fazendo o seu acompanhamento. O público-alvo é voltado aos jovens talentos, a escolha do público foi feita observando os índices de sucesso das metodologias

ativas em cada grupo.

Esta monografia utiliza as mesmas metodologias ativas com exceção da *Pósa Method*, tendo também o intuito de servir como ferramenta para o estudo de programação sem o acompanhamento de um professor. Entretanto, a principal diferença é que este trabalho tem como princípio o foco do ensino de algoritmos matemáticos, fornecendo o conhecimento necessário para o desenvolvimento das soluções das questões e disponibilizando um plano de estudos de aprendizagem linear sem utilizar o conceito de gamificação como estimulo.

3.3 Aplicações de *hashing*

Em Bento (2012), o trabalho introduz a história e as teorias do *hashing*, em seguida é exposta uma explicação sobre suas funções de dispersão, sendo elas: método da divisão, meio do quadrado, método da dobra, transformação de base e análise dos dígitos. Também é apresentado os assuntos de *hashing* perfeito, dinâmico e universal.

Após exposto todos os tópicos, a autora explica possíveis aplicações do *hashing* em problemas aritméticos, algébricos e em teoria de conjuntos. A autora também fala sobre o uso de *hashing* em buscas exaustivas como: *backtracking* e em algoritmos de ordenação (*bucket sort*). Entretanto, a autora não utilizou a metodologia ativa ABP que utilizaremos neste trabalho.

No trabalho citado as aplicações em *hashing* são apresentadas de maneira direta, ou seja, a autora apresenta o problema e descreve como aplicar *hashing* em simultâneo. Após isso, a autora não disponibilizou um programa de estudos com problemas a serem resolvidos e também não fez uso de nenhuma metodologia ativa no ensino.

As aplicações envolvendo os assuntos expostos nesta monografia tem como intuito ser apresentado de maneira indireta. Os problemas sugeridos necessitam de tópicos matemáticos diversificados, entretanto o próprio estudante tem que identificar e utilizar o conhecimento para resolver a questão, estimulando a habilidade de *Problem Solver* através da utilização da ABP na criação do plano de estudos.

3.4 Quadro Comparativo

No Quadro 2, podemos ver de maneira resumida as relações entre os trabalhos citados e a dissertação. A coluna de fundamentação teórica significa se o assunto abordado foi explicado e demonstrado devidamente.

Quadro 2 – Análise comparativa deste trabalho com os trabalhos relacionados

Trabalhos	Fundamentação Teórica	APD	ABP	Resolução de atividades sugeridas ou exemplos	Análise de complexidade
(NIKHÁZY, 2020b)	Não	Sim	Sim	Sim	Sim
(NIKHÁZY, 2020a)	Não	Sim	Sim	Não	Não
(BENTO, 2012)	Sim	Não	Não	Sim	Sim
Este trabalho	Sim	Sim	Sim	Sim	Sim

Fonte: O Autor.

4 REFERENCIAL TEÓRICO DO PROGRAMA DE ESTUDOS

O nosso programa de estudos tem o foco em aritmética modular e utilizará conceitos da combinatória. Neste capítulo, apresentaremos o referencial teórico do nosso programa de estudo para algoritmo matemáticos.

4.1 Aritmética Modular

Ao observamos um relógio de ponteiros convencional notamos que existem representações diferentes para cada um dos dois ponteiros. Um ponteiro representará as horas assumindo valores entre 1 e 12, já o outro ponteiro representará os minutos assumindo valores entre 0 e 59.

Imagine que você anotou a posição do ponteiro das horas no momento em que o relógio estava marcando 10 horas. Daqui a 12 horas, este ponteiro estará na mesma posição. Após 12 horas, a mesma configuração do relógio se repete. O mesmo acontece com o ponteiro dos minutos, porém precisamos esperar 60 minutos para a mesma posição ser repetida.

Em 1798, com 21 anos, Carl Friedrich Gauss escreveu seu livro sobre teoria dos números, chamado *Disquisitiones Arithmeticae* o qual foi publicado no ano de 1801. Neste livro foi apresentado pela primeira vez formalmente o conceito de aritmética modular.

A aritmética modular consiste em um conjunto de operações nos números inteiros as quais trazem o conceito de congruência modular. A congruência modular define que dois inteiros **a** e **b** são considerados congruentes no módulo **m**, se e somente se, o resto da divisão de **a** e **b** pelo módulo **m** forem iguais. Observe que o resto da divisão ser igual é equivalente a **m** dividir **(a - b)**.

A representação da congruência módulo **m** pode ser visualizada na equação 4.1.

$$a \equiv b \pmod{m} \tag{4.1}$$

Para a congruência modular ser considerada uma relação de equivalência, ela tem que respeitar a propriedade de ser reflexiva, simétrica e transitiva.

A propriedade reflexiva pode ser observada na equação 4.2.

$$\forall a \in \mathbb{Z} : a \equiv a \pmod{m} \tag{4.2}$$

Prova:

$$m|0 \therefore m|(a-a) \rightarrow a \equiv a \pmod{m} \tag{4.3}$$

A propriedade simétrica pode ser observada na equação 4.4.

$$\forall a,b \in \mathbb{Z} : a \equiv b \pmod{m} \rightarrow b \equiv a \pmod{m} \tag{4.4}$$

Prova:

$$m|(a-b) \rightarrow m|-(a-b) \therefore m|(b-a) \tag{4.5}$$

A propriedade transitiva pode ser observada na equação 4.6.

$$\forall a,b,c \in \mathbb{Z} : a \equiv b \pmod{m} \wedge b \equiv c \pmod{m} \rightarrow a \equiv c \pmod{m} \tag{4.6}$$

Prova:

$$m|(a-b) \wedge m|(b-c) \rightarrow m|(a-b)+(b-c) \therefore m|(a-c) \tag{4.7}$$

4.1.1 Grupo Finito

Um grupo $(S,\oplus)$ é um conjunto S e uma operação binária $\oplus$ definido sobre S para a qual são válidas as seguintes propriedades:

1. Fechamento: Para todo $a,b \in S, a \oplus b \in S$.
2. Identidade: Existe um elemento $e \in S$ tal que $a \oplus e = e \oplus a = a$ para todo $a \in S$.
3. Associatividade: Para todo $a,b,c \in S$, temos que $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
4. Inverso: Para cada $a \in S$, existe um elemento único $b \in S$ tal que $a \oplus b = b \oplus a = e$.

4.1.2 Grupo aditivo módulo n

Dado um inteiro n, os inteiros podem ser divididos em n classes de equivalências de acordo com seus restos, módulo n. A classe de equivalência que contém um inteiro a é:

$$[a]_n = \{a + kn \mid k \in \mathbb{Z}\} \tag{4.8}$$

O conjunto de todas as classes de equivalência Z_n é

$$Z_n = \{[a]_n | 0 \leq a \leq n-1\} \tag{4.9}$$

A adição módulo n denotada por $+_n$ pode ser definida da seguinte maneira:

$$[a]_n +_n [b]_n = [a+b]_n \tag{4.10}$$

$(Z_n, +_n)$ é o grupo aditivo módulo n. O quadro seguinte mostra as operações para o grupo $(Z_6, +_6)$:

$+_6$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

A operação de subtração pode ser definida da seguinte maneira:

$$[a]_n -_n [b]_n = [a-b]_n \tag{4.11}$$

O inverso aditivo de um inteiro a módulo n é o elemento $[-a]_n$ ou $[n-a]_n$.

A operação de subtração modular pode ser definida usando o inverso aditivo da seguinte maneira:

$$[a]_n -_n [b]_n = [a]_n + [-b]_n \tag{4.12}$$

4.1.3 Grupo Multiplicativo módulo n

Z_n^* são os elementos de Z_n que formam um grupo com a operação $*_n$. O conjunto Z_n^* é formado pelos seguintes elementos:

$$Z_n^* = \{[a]_n \in Z_n \mid mdc(a,n) = 1\} \tag{4.13}$$

Um exemplo de Z_n^* é

$$Z_8^* = \{[1],[3],[5],[7]\} \tag{4.14}$$

A operação de multiplicação módulo n pode ser definida da seguinte maneira:

$$[a]_n *_n [b]_n = [a*b]_n \tag{4.15}$$

O quadro seguinte mostra as operações para o grupo $(Z_8^*, *_6)$:

$*_8$	1	3	5	7
1	1	3	5	7
3	3	1	7	5
5	5	7	1	3
7	7	5	3	1

4.1.4 Inverso modular

Dado um inteiro **a** e um inteiro positivo **m**, o inverso multiplicativo modular para **a** módulo **m** é um inteiro **b** que pode ser visto através da equação 4.16.

$$a * b \equiv 1 \pmod{m} \tag{4.16}$$

Portanto, podemos definir **b** como sendo $[a]_m^{-1}$.

Uma forma de se calcular o valor de **b** seria passar por todos os valores entre 1 e **m** - 1 e testar se a equação 4.16 é verdadeira. Entretanto, dependendo do valor do **m** escolhido encontrar a variável **b** pode ser muito lenta.

Dessa maneira, podemos definir a **divisão módulo n** utilizando o inverso multiplicativo da seguinte maneira:

$$[a]_n /_n [b]_n = [a]_n *_n [b]_n^{-1} \tag{4.17}$$

4.1.4.1 Pequeno teorema de Fermat

O pequeno teorema de Fermat diz que dado **m** um número primo qualquer e **a** um inteiro qualquer, se o maior divisor em comum de **m** e **a** for igual a 1, então a equação de congruência 4.18 é verdadeira.

$$a^{m-1} \equiv 1 \pmod{m}. \tag{4.18}$$

Sabendo disso, vamos multiplicar os dois lados da equação por a^{-1}, desta forma mantendo a equivalência e obtendo a equação 4.19.

$$a^{m-1} * a^{-1} \equiv 1 * a^{-1} \pmod{m}. \tag{4.19}$$

Aplicando a propriedade da multiplicação na mesma base obtemos a nossa equação final 4.20.

$$a^{m-2} \equiv a^{-1} \pmod{m}. \tag{4.20}$$

Portanto, para descobrirmos nosso inverso multiplicativo modular quando o maior divisor em comum de **m** e **a** for igual a 1 basta calcularmos o resultado de a^{m-2} módulo **m**.

4.1.4.2 Teorema de Euler

Como foi observado na seção anterior, o pequeno teorema de Fermat consegue resolver o problema de encontrar o inverso multiplicativo de um número, entretanto de maneira parcial devido a exigir a necessidade de que **m** seja um número primo. E se por acaso o **m** não for um número primo?

O teorema de Euler afirma que se o maior divisor em comum de **m** e **a** for igual a 1, então a equação de congruência 4.21 é verdadeira.

$$a^{\phi(m)} \equiv 1 \pmod{m}. \tag{4.21}$$

Observe que podemos fazer os mesmos passos realizados na seção do teorema de Euler até obter a equação 4.22.

$$a^{\phi(m)-1} \equiv a^{-1} \pmod{m}. \tag{4.22}$$

Portanto, para descobrirmos nosso inverso multiplicativo modular quando o maior divisor em comum de **m** e **a** for igual a 1 basta calcularmos o resultado de $a^{\phi(m)-1}$ módulo **m**. Observe que quando **m** é primo, o $\phi(m)$ é igual a **m** - 1, resultando na equação do pequeno teorema de Fermat.

4.1.4.3 Solução utilizando o lema de Bezout

O Lema de Bezout estabelece que dado dois inteiros a e b existem inteiros m, n tal que

$$am + bn = mdc(a,b)$$

Os inteiros m e n podem ser encontrado utilizando o algoritmo de Euclides estendido.

Quando a e b são primos entre si, o máximo divisor comum de a e b é 1. Utilizando o algoritmo de Euclides estendido podemos encontrar m e n tal que

$$am + bn = 1$$

Tomando a equação acima mod b, temos

$$am + bn \equiv 1 \pmod{b}$$

Simplificando a equação acima,

$$am \equiv 1 \pmod{b}$$

Logo, o inteiro m representa o inverso multiplicativo de a $\pmod{b}$. De igual modo, o inteiro n representa o inverso multiplicativo de b $\pmod{a}$.

Os algoritmos de exponenciação modular, exponenciação modular rápida e Euclides estendido serão apresentados durante a resolução dos problemas no próximo capítulo.

4.1.5 Equação de congruência linear

Na matemática convencional, uma equação é considerada linear se seguir o seguinte formato:

$$a * x = b. \tag{4.23}$$

Nesta equação, nosso objetivo é encontrar o valor de **x**, conhecendo os valores reais das variáveis **a** e **b**.

Entretanto, uma equação de congruência linear é um pouco diferente devido à utilização do módulo. Podemos definir uma equação de congruência linear, como a equação 4.24 onde os valores de **a**, **b** e **m** são inteiros conhecidos e o nosso objetivo é encontrar o valor **x**.

$$a * x \equiv b \pmod{m}. \tag{4.24}$$

Podemos descobrir os valores de x de duas formas distintas, a primeira seria utilizando o inverso multiplicativo modular e a segunda utilizando o algoritmo de Euclides estendido após reescrever a equação.

A primeira solução pretende encontrar o inverso multiplicativo modular para a variável **a** de tal forma que ao multiplicarmos os dois lados da equação pelo inverso vamos obter a seguinte equação:

$$x \equiv b * a^{-1} \pmod{m}. \tag{4.25}$$

Portanto, se o maior divisor comum entre **a** e **m** for igual a 1, podemos encontrar o inverso multiplicativo modular de **a** e multiplicarmos por **b** para obtermos o valor de **x**.

Para a solução utilizando Euclides estendido, precisamos reescrever a equação 4.24 para o formato de uma equação linear diofantino, como podemos ver a seguir:

$$a * x + k * m = b. \tag{4.26}$$

Após isso, podemos utilizar o algoritmo de Euclides estendido para solucionar essa equação diofantino como visto anteriormente.

4.1.6 Teorema chinês do resto

> O surgimento do Teorema Chinês dos Restos ocorreu no livro do matemático chinês Sun Zi Suanjing chamado "Manual de aritmética do Sol"durante os primeiros séculos entre 280 d.C a 483 d.C. (GLÓRIA *et al.*, 2019)

A história mais conhecida sobre a origem do teorema chinês do resto acontece no período das guerras lideradas pelos grandes generais chineses. Nessa história, os generais chineses precisavam contar suas tropas antes de algum confronto para assim determinar a melhor estratégia possível a ser seguida e ao final do confronto para identificar quantas tropas foram perdidas.

Para fazer essa contagem, os generais chineses ordenavam que as suas tropas formassem filas de determinados tamanhos e após isso, o general anotava a quantidade de guerreiros que sobraram e não conseguiram pertencer a nenhuma fila. Esse método se repetia algumas vezes com tamanhos diferentes até o general conseguir determinar o número exato da tropa.

Por exemplo, suponha que um determinado general chinês fez a contagem de suas tropas e observou ter exatamente 3200 tropas, após um determinado tempo ele montou a melhor estratégia para uma batalha e quando acabou ele queria contabilizar a quantidade de tropas perdidas.

Para isso, ele usou a estratégia mencionada anteriormente e começou ordenar os soldados para montarem filas de determinados tamanhos. A primeira ordem foi que todos os soldados se alinhassem em filas de tamanho cinco e então o general anotou que apenas um soldado não pertenceu a nenhuma fila, após isso ele solicitou que a fila tivesse tamanho sete e anotou dois, por fim ao solicitar o tamanho onze e treze ele anotou que em ambas não restou nenhum soldado. Depois de um curto tempo pensando, o general determinou que agora lhe restou 2431 tropas utilizando o teorema chinês do resto.

O teorema chinês do resto afirma que dado inteiros $p_1, p_2, ..., p_k$ tais que todos os números sejam primos entre si e valores inteiros $a_1, a_2, ..., a_k$ formam um sistema de congruência. Existe uma solução para este sistema de congruência única no modulo $p = p_1 * p_2 * ... * p_k$.

Podemos observar o sistema de congruências lineares mencionado a seguir:

$$\begin{cases} a & \equiv a_1 \pmod{p_1} \\ a & \equiv a_2 \pmod{p_2} \\ & \dots \\ a & \equiv a_k \pmod{p_k} \end{cases}$$

4.1.6.1 Solução para sistema de congruências lineares com duas equações.

Primeiramente, vamos encontrar a solução para um conjunto com apenas duas equações e após isso, vamos expandir para encontrar a solução de um conjunto de tamanho qualquer.

Observe o sistema de congruências lineares a seguir:

$$\begin{cases} a & \equiv a_1 \pmod{p_1} \\ a & \equiv a_2 \pmod{p_2} \end{cases}$$

A primeira coisa a ser feita é substituir a representação dessas equações como uma equação convencional, a nova representação pode ser observada no sistema de equações a seguir:

$$\begin{cases} a = a_1 + k_1 * p_1 \\ a = a_2 + k_2 * p_2 \end{cases}$$

O próximo passo, é substituir a primeira equação na segunda equação do sistema de equações acima, o qual pode ser visto na equação a seguir:

$$a_1 + k_1 * p_1 = a_2 + k_2 * p_2 \tag{4.27}$$

Observe que após agruparmos o a_1 e o a_2 do lado esquerdo e agruparmos o $k_1 * p_1$ e $k_2 * p_2$ do lado direito, vamos obter a equação diofantino 4.28 devido a conhecermos os valores de p_1, p_2, a_1 e a_2.

$$a_1 - a_2 = (-k_1) * p_1 + k_2 * p_2 \tag{4.28}$$

Agora, vamos definir que **d** é igual ao maior divisor comum de p_1 e p_2. Como sabemos, o lado direito da equação 4.28 é divisível por **d** e para existir solução o lado esquerdo também precisa ser divisível.

Graças ao algoritmo de Euclides estendido, nós conseguimos descobrir os coeficientes da equação 4.29.

$$d = x * p_1 + y * p_2 \tag{4.29}$$

Agora precisamos manipular a equação 4.29 para ficar igual à equação 4.28. Para isso, multiplicaremos os dois lados da equação 4.29 por $\frac{a_1 - a_2}{d}$ obtendo a equação 4.30.

$$d * \frac{a_1 - a_2}{d} = (x * p_1 + y * p_2) * \frac{a_1 - a_2}{d}. \tag{4.30}$$

Aplicando a distributiva na equação 4.30, obtemos a equação 4.31.

$$a_1 - a_2 = p_1 * (x * \frac{a_1 - a_2}{d}) + p_2 * (y * \frac{a_1 - a_2}{d}) \tag{4.31}$$

Comparando as equações 4.28 e 4.31, conseguimos obter os valores das variáveis k_1 e k_2 as quais podem ser constatadas no seguinte sistema de equações:

$$\begin{cases} k_1 = -(x * \dfrac{a_1 - a_2}{d}) \\ k_2 = (y * \dfrac{a_1 - a_2}{d}) \end{cases}$$

Podemos agora, substituir k_1 na primeira equação na equação a = a_1 + k_1 * p_1, obtendo a equação:

$$a = a_1 + -(x * \frac{a_1 - a_2}{d}) * p_1 \tag{4.32}$$

A equação acima gera um valor a que respeita às duas equações de congruências.

Exemplo 4.1 *Considere o seguinte sistema de equação:*

$$\begin{cases} a \equiv 2 \pmod 5 \\ a \equiv 3 \pmod 7 \end{cases}$$

O máximo divisor de 5 e 7 é 1. Pelo Lema de Bezout, existe x e y tal que 5x + 7y = 1. Tome x = 3 e y = -2. Utilizando, o resultado acima,

$$a = 2 + -(3 * \frac{-1}{1}) * 5 = 2 + 3 * 5 = 17,$$

a = 17 é uma solução particular do sistema de equação acima.

Observe que como estamos lidando com aritmética modular, não existe apenas uma solução para essa equação. Suponha que temos duas soluções validas chamadas x_1 e x_2, onde o seguinte sistema de congruências é respeitado:

$$\begin{cases} x_1 \equiv a_1 \pmod{p_1} \\ x_2 \equiv a_1 \pmod{p_1} \\ x_1 \equiv a_2 \pmod{p_2} \\ x_2 \equiv a_2 \pmod{p_2} \end{cases}$$

Por transitividade, sabemos que o seguinte sistema de congruências também é respeitado:

$$\begin{cases} x_1 \equiv x_2 \pmod{p_1} \\ x_1 \equiv x_2 \pmod{p_2} \end{cases}$$

Podemos reescrever as equações acima da seguinte forma:

$$\begin{cases} x_1 = x_2 + k_1 * p_1 \\ x_1 = x_2 + k_2 * p_2 \end{cases}$$

Agora podemos manipular para obtermos o seguinte sistema de equações:

$$\begin{cases} (x_1 - x_2) = k_1 * p_1 \\ (x_1 - x_2) = k_2 * p_2 \end{cases}$$

Esse sistema nos informa que $(x_1 - x_2)$ é divisível por p_1 e p_2.

Como sabemos o MMC é o menor múltiplo comum, observe que também podemos afirmar pela definição que o MMC(p_1, p_2) é divisível por p_1 e p_2 . Sabe-se que $(x_1 - x_2)$ e MMC(p_1, p_2) são múltiplos de p_1 e p_2. Sendo assim podemos concluir que $(x_1 - x_2)$ também é divisível pelo menor múltiplo comum entre p_1 e p_2.

Assim, podemos escrever a seguinte equação:

$$(x_1 - x_2) = k_3 * MMC(p_1, p_2) \tag{4.33}$$

Podemos então reescrever a equação para o formato de uma congruência modular:

$$x_1 = x_2 + k_3 * MMC(p_1, p_2) \tag{4.34}$$

Portanto, unindo às duas equações de congruência linear em uma só módulo MMC(p_1, p_2), desta forma provando que existem infinitas soluções, entretanto existe apenas uma solução para este módulo.

Exemplo 4.2 *Considere o seguinte sistema de equação:*

$$\begin{cases} a \equiv 2 \pmod 5 \\ a \equiv 3 \pmod 7 \end{cases}$$

Como sabemos que a = 17 é uma solução particular do sistema de equação acima. Podemos encontrar a solução geral, usando a equação acima. O mínimo múltiplo comum de 5 e 7 é 35. Tome $x_2 = 17$ *e renomeando* x_1 *para a*

$$a = 17 + 35 * k$$

Logo, a solução do sistema de equação acima é:

$$a \equiv 17 \pmod{35}$$

4.1.6.2 Solução para sistema de congruências lineares com t equações.

Como vimos na seção anterior, existe apenas uma solução válida para o MMC dos módulos informados, ou seja, podemos unir duas equações de congruência linear em uma só de tal forma que o novo modulo seja igual ao MMC e o resultado irá respeitar ambas equações.

Para isso, será realizado a união entre duas equações e após isso a resultante será utilizada para fazer a junção com a próxima equação. O sistema de equações a seguir representa a junção de duas equações em uma, onde a variável x representa o número que satisfaz às duas primeiras equações:

$$\begin{cases} a \equiv a_1 \pmod{p_1} \\ a \equiv a_2 \pmod{p_2} \\ a \equiv x \pmod{MMC(p_1, p_2)} \end{cases}$$

Observe que podemos utilizar a mesma solução da seção anterior para fazer a junção das duas equações e enfim descobrir o valor de x modulo MMC(p_1, p_2). Após isso, basta utilizar essa nova equação de congruência como a primeira equação e continuar unindo par a par.

5 PROGRAMA DE ESTUDOS

Neste capítulo, apresentaremos nosso programa de estudo para algoritmo matemáticos. O programa de estudo segue o padrão sugerido no artigo (NIKHÁZY, 2020b).

As soluções de todos os problemas apresentados em C++ pode ser vista na plataforma github no seguinte link, https://github.com/dougnobrega/Solucoes-Tcc.

Quadro 3 – Exemplo de problemas sugeridos.

Introdução	(RIBEIRO, 2014) Canetas e Borrachas SPOJBR - Miojo CSES - Exponentiation KATTIS - Chinese Remainder Theorem (non-relatively prime moduli)
Reforço	CSES - Exponentiation II SPOJ - Good Predictions
Síntese	CODEFORCES - B. Remainders Game CODECHEF - Moves NEPSACADEMY - Fantastic Beasts

5.1 Problemas de Introdução

Nesta seção, discutiremos quatro problemas:

- Canetas e Borrachas adaptado do trabalho sobre equações diofantinas do mestrado profissional em matemática apresentado em (RIBEIRO, 2014)
- Miojo retirado do juiz online SPOJ-BR (https://br.spoj.com/problems/main/)
- Exponentiation retirado do juiz online Code Submission Evaluation System (CSES) (https://cses.fi/)
- Chinese Remainder Theorem (non-relatively prime moduli) retirado do juiz online KATTIS (https://open.kattis.com/)

5.1.1 Canetas e Borrachas

Problema: Anita comprou um número ímpar de canetas e algumas borrachas, gastando R$ 37,40. Sabendo-se que os preços unitários das canetas e das borrachas são, respectivamente, R$ 1,70 e R$ 0,90, determine quantas canetas e quantas borrachas ela comprou. Considere que essa solução é única. (RIBEIRO, 2014)

Solução: Sejam x o número de canetas e y o número de borrachas. Obtemos a

seguinte equação:

$$1,7x+0,9y=37,4 \tag{5.1}$$

Multiplicando a equação por 10, temos:

$$17x+9y=374 \tag{5.2}$$

Primeiramente, precisamos checar se a equação possui solução. A equação $ax+by=c$ possui soluções inteiras quando $mdc(a,b)|c$. No problema acima, mdc(9,17)|374.

Teorema 1 *Sejam $a,b,c \in \mathbb{Z}$ tal que $ax+by=c$ e $mdc(a,b)|c$ então $ax+by=c$ possui soluções inteiras.*

Suponha por absurdo que sejam $a,b,c \in \mathbb{Z}$ tal que $ax+by=c$ e $mdc(a,b)|c$ e $ax+by=c$ não possui soluções inteiras. Pelo lema de Bezout, para $a,b \in \mathbb{Z}$ existem inteiros m e n tal que

$$am+bn=mdc(a,b) \tag{5.3}$$

(podem ser obtidos pelo algoritmo de Euclides estendido). Como $mdc(a,b)|c$, existe um inteiro k tal que $c=k*mdc(a,b)$. Logo, multiplicando a equação 5.3 por k, encontramos uma solução inteiro para $ax+by=c$, o que é uma contradição.

$$a(km)+b(kn)=k*mdc(a,b)=c \tag{5.4}$$

Em seguida, vamos encontrar uma solução particular para a seguinte equação:

$$17x \equiv 374 \pmod 9 \tag{5.5}$$

Seja x_0 uma solução particular inteira positiva para a primeira equação. Observe que a partir do valor x_0, podemos encontrar o valor de y_0. Além disso, se (x_0,y_0) é uma solução da Equação 5.2 então (x_0+b,y_0-a) é uma solução da Equação 5.2.

Teorema 2 *Sejam $a,b,c \in \mathbb{Z}$. Se (x_0,y_0) é uma solução para $ax+by=c$ se somente se (x_0+b,y_0-a) é uma solução para $ax+by=c$.*

As seguintes equivalências demonstram o teorema:

$$\begin{aligned} ax_0 + by_0 = c \quad &\Leftrightarrow \quad a(x_0 + b - b) + b(y_0 - a + a) = c \\ &\Leftrightarrow \quad a(x_0 - b) - ab + b(y_0 - a) + ab = c \\ &\Leftrightarrow \quad a(x_0 - b) + b(y_0 - a) = c \end{aligned}$$

O algoritmo que encontra o maior divisor comum entre A e B pode ser visto a seguir:

Algorithm 1 MDC

Data: A e $B \in \mathbb{Z}$
Result: Maior divisor comum entre A e B

```
1  início
2      while b! = 0 do
3          r ← a mod b
4          a ← b
5          b ← r
6      return a
```

Fonte: elaborado pelo autor.

Para encontra a solução particular, podemos implementar um algoritmo como a seguir:

Algorithm 2 SOLUÇÃO PARTICULAR INTEIRA PARA $A * X == C(mod)B$

Data: A, B e $C \in \mathbb{Z}$
Result: X tal que $A * X == C \pmod{B}$

```
7  início
8      d ← MDC(A,B)
9      if c mod d! = 0 then
10         return (-1,-1)
11     else
12         x ← 0
13         while x ≤ b & a*x mod b = c mod b do
14             x ← x+1
15         y ← (c−a*x)/b
16         repeat
17             if x mod 2 == 1 then
18                 return (x, y)
19             x ← x+b
20             y ← y−a
21         until x > 0 & y > 0;
22     return (-1,-1)
```

Fonte: elaborado pelo autor.

Outra maneira de resolver este problema é encontrando uma solução particular da Equação 5.2 utilizando o algoritmo de Euclides estendido. Dado $a, b\mathbb{Z}$, o algoritmo de Euclides estendido encontra inteiros m, n tal que $am + bn = mdc(a,b)$. Podemos exemplificar

o algoritmo de Euclides utilizando um quadro com quatro colunas rotuladas como : r, q, m e n, respectivamente resto, quociente e os coeficientes m e n. Nas duas primeiras linhas, temos $(a, *, 1, 0)$ e $(b, *, 0, 1)$. Em cada linha, manteremos o seguinte invariante:

$$r = am + bn \tag{5.6}$$

As linhas seguintes podem ser calculadas da seguinte maneira:

$$\begin{aligned} q_{j+1} &= r_{j-1}/r_j \\ r_{j+1} &= r_{j-1} - q_{j+1} r_j \\ m_{j+1} &= m_{j-1} - q_{j+1} m_j \\ n_{j+1} &= n_{j-1} - q_{j+1} n_j \end{aligned}$$

Utilizando o algoritmo de Euclides estendido com $a = 17$ e $b = 9$.

r	q	m	n	Invariante
17	*	1	0	17 = 17(1) + 9(0)
9	*	0	1	9 = 17(0) + 9(1)
8	1	1	-1	8 = 17(1) + 9(-1)
1	1	-1	2	1 = 17(-1) + 9(2)

Uma solução particular da Equação 5.2 é (-374, 748). Uma solução inteira da Equação 5.2 tem o seguinte formato (-374 + 9t , 748 - 17t). Como estamos procurando uma solução inteira positiva, temos que

$$-374 + 9t \geq 0$$

$$748 - 17t \geq 0$$

Por essas duas equações temos que:

$$t \geq 42$$

$$t \leq 44$$

Então vamos procurar neste intervalo, as soluções que satisfazem as restrições impostas pelo problema.

O algoritmo do Euclides estendido pode ser visto a seguir:

Algorithm 3 EUCLIDES ESTENDIDO

Data: $a, b \in \mathbb{Z}$
Result: x, y tal que $ax + by = mdc(a,b)$

```
início
    x ← 1
    y ← 0
    x1 ← 0
    y1 ← 1
    a1 ← a
    b1 ← b
    while b1 != 0 do
        q ← a1/b1
        (x, x1) ← (x1, x − q ∗ x1)
        (y, y1) ← (y1, y − q ∗ y1)
        (a1, b1) ← (b1, a1 − q ∗ b1)
    return (a1, x , y)
```

Fonte: elaborado pelo autor.

O algoritmo que encontra as soluções que satisfazem as restrições impostas pelo problema pode ser visto a seguir:

Algorithm 4 SOLUÇÃO INTEIRA (MC + BT, NC - AT)

Data: $a, b \in \mathbb{Z}$
Result: X tal que $A * X == C \pmod{B}$

```
início
    (d, m, n) ← EuclidesEstendido(a, b)
    if c mod d != 0 then
        return (-1, -1)
    else
        lu ← ceil((−m ∗ c)/b)
        ub ← floor((n ∗ c)/a)
        t ← lu
        while t ≤ ub do
            x ← m ∗ c + b ∗ t
            y ←= n ∗ c − a ∗ t
            if x mod 2 == 1 then
                return (x, y)
            t ← t + 1
        return (-1,-1)
```

Fonte: elaborado pelo autor.

5.1.2 Problema Miojo

João é um fanático por miojos; ele os adora, e, como era de se esperar, ele levou vários pacotes quando foi acampar com seus colegas. Como João só gosta de miojos feitos com o tempo exato, ele se desesperou ao perceber que havia esquecido seu relógio em casa.

Por sorte, ele conseguiu, no caminho, comprar duas ampulhetas de durações diferentes. Por exemplo, se o miojo precisa de 3 minutos para ficar pronto, e João tiver uma ampulheta de 5 minutos e outra de 7, uma possível forma de cozinhar o miojo é:

1. João começa virando às duas ampulhetas em simultâneo.
2. Quando a areia da ampulheta de 5 minutos se esgotar, João torna a virá-la.
3. João começa a preparar o miojo quando a areia da ampulheta de 7 minutos acabar.
4. João tira o miojo do fogo quando a ampulheta de 5 minutos acabar novamente.

Dessa forma, o miojo ficará 3 minutos no fogo (do minuto 7 ao minuto 10). Assim, apesar do miojo levar apenas três minutos para ser cozido, ele precisa de 10 minutos para ficar pronto.

Faça um programa que, dado o tempo de preparo do miojo, e os tempos das duas ampulhetas (ambos maiores que o tempo do miojo), determina o tempo mínimo necessário para o miojo ficar pronto. Você pode supor que é sempre possível cozinhar o miojo no tempo correto.

Dado os inteiros a, b, t queremos encontrar o tempo mínimo necessário para cozinhar um miojo no tempo t utilizando às duas ampulhetas de tempo a e b. Note que queremos encontrar uma solução para

$$ax + by = t \tag{5.7}$$

Podemos encontrar simulando o processo de viradas das ampulhetas com o algoritmo a seguir:

Algorithm 5 MIOJO SOLUÇÃO INICIAL

Data: $a, b, t \in \mathbb{Z}$
Result: Tempo mínimo de preparo

```
início
    while 1 do
        if ta > tb then
            if ta − tb == t then
                return ta
            tb ← tb + b
        else
            if tb − ta == t then
                return tb
            ta ← ta + a
```

Fonte: elaborado pelo autor.

Queremos encontrar a primeira solução inteira que satisfaça:

$$(ax \equiv t \pmod{b}) \vee (bx \equiv t \pmod{a})$$

O algoritmo a seguir encontra a primeira solução inteira que satisfaça a equação a cima:

Algorithm 6 MIOJO

Data: $a, b, t \in \mathbb{Z}$
Result: Tempo mínimo de preparo

```
início
    k ← 1
    while 1 do
        ta ← a ∗ k
        tb ← b ∗ k
        if ta mod b == t mod b then
            return ta
        if tb mod a == t mod a then
            return tb
```

Fonte: elaborado pelo autor.

Utilizando o algoritmo de Euclides, podemos encontrar uma solução particular. Em seguida, estudar a solução geral:

r	q	m	n	Invariante
7	*	1	0	7 = 7(1) + 5(0)
5	*	0	1	5 = 7(0) + 5(1)
2	1	1	-1	2 = 7(1) + 5(-1)
1	2	-2	3	1 = 7(-2) + 5(3)

A solução particular para 7x + 5y = 3 é (-6, 9). A solução geral para a equação é (-6 + 5k, 9 - 7k). Queremos encontrar o valor mínimo positivo para $7(-6+5k)$ ou $5(9-7k)$. Para os valores serem positivos, temos as seguintes restrições:

$k \geq 2$ para $-42+35k$

$k \leq 1$ para $45-35k$

k	x	y	7(-6 + 5k)	5(9-7k)
2	4	-5	28	-25
3	9	-12	63	-60
1	-1	2	-7	10
0	-6	9	42	45

Podemos observar que o tempo mínimo foi obtido para k=2 para a expressão $7(-6+5k)$ e $k=1$ para a expressão $5(9-7k)$.

Nossa solução final, pode ser implementada como a seguir:

Algorithm 7 MIOJO VERSÃO FINAL

Data: $a,b,t \in \mathbb{Z}$
Result: Tempo mínimo de preparo
início
  $(d,m,n) \leftarrow EuclidesEstendido(a,b)$
  $u \leftarrow ceil((double)(-t*m)/b)$
  $v \leftarrow floor((double)(t*n)/a)$
  $x1 \leftarrow a*abs(t*m+b*u)$
  $y1 \leftarrow b*abs(t*n-a*u)$
  $x2 \leftarrow a*abs(t*m+b*v)$
  $y2 \leftarrow b*abs(t*n-a*v)$
  return $min(max(x1,y1),max(x2,y2))$

Fonte: elaborado pelo autor.

5.1.3 Problema Exponentiation

O problema Exponentiation pode ser encontrado no seguinte link https://cses.fi/problemset/task/1095.

Os assuntos abordados neste problema são:

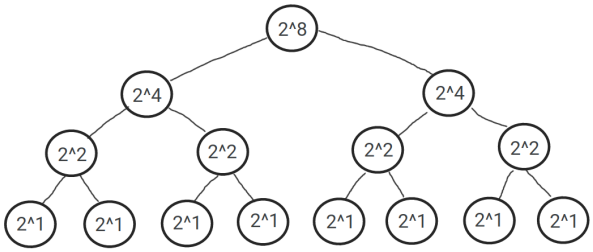
- Matemática
- Aritmética modular
- Exponenciação rápida.

Neste problema, sua tarefa é calcular eficientemente a equação a^b módulo $10^9 + 7$. As variáveis a, b podem assumir valores entre 1 e 10^9.

A primeira forma de se calcular esta equação seria através de uma repetição sendo executado b vezes, na qual uma variável é utilizada para guardar a resposta da exponenciação. Entretanto, esse algoritmo teria um custo bastante elevado devido ao b poder assumir o valor de 10^9. Portanto, precisamos encontrar uma solução melhor que essa.

Para isso, vamos observar de uma forma diferente como construir o número correspondente a 2^8. Pelas propriedades da exponenciação sabemos que 2^8 é igual a $2^4 * 2^4$, e podemos continuar fazendo isso diversas vezes até chegarmos em 2^1. Observe a imagem a seguir do grafo que representa essas operações.

Figura 3: Figura de um grafo representando outras formas de visualizar o 2^8.



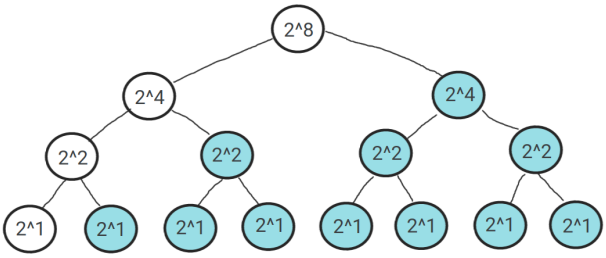
Fonte: O Autor.

A primeira observação em relação ao grafo a ser feita é que vários destes vértices se repetem. O vértice referente a 2^1 se repete 8 vezes, o vértice referente a 2^2 se repete 4 vezes e o vértice referente a 2^4 se repete 2 vezes. Portanto, sabemos que se calcularmos o valor de 2^4 do lado esquerdo e multiplicarmos por ele mesmo obteremos o valor de 2^8, com isso já conseguimos reduzir um total de 7 vértices que estariam sendo utilizados pelo 2^4 do lado direito. E podemos continuar fazendo isso para todos os vértices que já conhecemos, como 2^2 e 2^1. Observe a imagem a seguir do grafo referente ao 2^8 onde os vértices pintados não serão calculados.

Sabendo disso, podemos implementar nosso algoritmo de exponenciação rápida. Para isso, vamos implementar um algoritmo recursivo.

O caso base será quando o expoente for igual a 0 o qual retornaremos 1, devido a qualquer número elevado a 0 resultar em 1.Após isso, precisaremos verificar se o expoente é par ou impar, se o expoente for par podemos calcular a resposta para metade do expoente e depois elevar-la ao quadrado. Caso o expoente seja impar, podemos subtrair 1 no expoente e realizar a operação do par e após isso multiplicar o resultado pela base.

Figura 4: Figura de um grafo representando outras formas de visualizar o 2^8 com vértices pintados significando que não será calculado.



Fonte: O Autor.

Devido a sempre dividirmos o expoente por 2, a quantidade de passos será log2(b). Sendo mais eficiente que o algoritmo com loop.

Por fim, a solução do problema pode ser implementada dessa maneira:

Algorithm 8 EXPMOD

Data: $a, b, m \in \mathbb{Z}$
Result: $a^b \mod m$

```
79 início
80     if b == 0 then
81         return 1
82     x ← expMod(a, b/2, m)
83     if b mod 2 == 0 then
84         return x*x (mod m)
85     else
86         return a*x*x (mod m)
```

Fonte: elaborado pelo autor.

5.1.4 Problema Chinese Remainder Theorem (non-relatively prime moduli)

O problema Chinese Remainder Theorem (non-relatively prime moduli) pode ser encontrado no seguinte link https://open.kattis.com/problems/generalchineseremainder.

Neste problema, sua tarefa é dado os valores inteiros das variáveis a, n, b e m, encontrar dois inteiros x e K, onde K = MMC(n, m) e 0 <= x < K que satisfaça o seguinte sistema de equações de congruência linear:

$$\begin{cases} x \equiv a \pmod{n} \\ x \equiv b \pmod{m} \\ x \equiv x \pmod{MMC(n,m)} \end{cases}$$

Observe que este problema é o mesmo que foi solucionado na subseção 4.1.6.1.

Portanto, basta que a solução oferecida seja implementada.

O algoritmo que executa a união de duas equações de congruência linear pode ser vista a seguir:

Algorithm 9 UNIRDUASEQUAÇÕES

Data: $a1, a2, b1, b2 \in \mathbb{Z}$
Result: Equação de congruência linear resultante da união de duas equações.

```
início
    (d, x, y) ← EuclidesEstendido(a2, b2)
    if b1 − a1 mod d! = 0 then
        return (−1, −1)
    k ← MMC(a2, b2)
    k1 ← (x ∗ ((b1 − a1)/d mod K
    ans ← a1 + (k1 ∗ a2) mod K
    if ans < 0 then
        ans ← ans + k
    return (ans, k)
```

Fonte: elaborado pelo autor.

5.2 Problemas de Reforço

Nesta seção, discutiremos dois problemas:

- O problema Exponentiation II retirado do juiz online CSES (https://cses.fi/problemset/).
- Good Predictions retirado do juiz online Sphere Online Judge(SPOJ) (https://www.spoj.com/)

5.2.1 Problema Exponentiation II

O problema Exponentiation II pode ser encontrado no seguinte link https://cses.fi/problemset/task/1712.

Os assuntos abordados neste problema são:

- Propriedades da aritmética modular
- Pequeno Teorema de Fermat
- Exponenciação rápida modular.

Neste problema, sua tarefa é calcular eficientemente a equação a^{b^c} modulo 10^9+7.

A primeira coisa que é necessária saber para resolver este problema é o teorema de Fermat, o qual afirma que se o modulo **m** escolhido for um número primo, então a seguinte equação é verdadeira,

$$a^{m-1} \equiv 1 \pmod{m}$$

Logo, precisamos reescrever o expoente a^{b^c} em grupos de potências a^{m-1} utilizando o algoritmo da divisão, obtemos que

$$b^c = k * (m-1) + y$$

Logo,

$$(a^{m-1})^k a^y$$

Note que

$$a^{b^c} \equiv (a^{m-1})^k a^y \equiv a^y \pmod{m}$$

Para entender a expressão acima, precisamos utilizar as seguintes propriedades da aritmética modular:

(1) $a \equiv b \pmod{m} \Leftrightarrow a \pmod{m} = b \pmod{m}$

(2) $ab \pmod{m} = (a \pmod{m} * b \pmod{m}) \pmod{m}$

Agora, vamos aplicar essas regras:

$$\begin{aligned} a^{b^c} \pmod{m} &\Leftrightarrow (a^{m-1})^k a^y \pmod{m} \\ &\Leftrightarrow ((a^{m-1})^k \pmod{m})(a^y \pmod{m}) \pmod{m} \\ &\Leftrightarrow (1 \pmod{m})^k (a^y \pmod{m}) \pmod{m} \\ &\Leftrightarrow (a^y \pmod{m}) \pmod{m} \end{aligned}$$

A solução do problema pode ser implementado dessa maneira:

Algorithm 10 SOLVE

Data: $a, b, c \in \mathbb{Z}$
Result: $a^{b^c} \mod 10^9 + 7$

```
início
    m ← 10^9 + 7
    y ← expMod(b, c, m − 1)
    return expMod(a, y, m)
```

Fonte: elaborado pelo autor.

5.2.2 Problema GOODB - Good Predictions

O problema GOODB - Good Predictions pode ser encontrado no seguinte link https://www.spoj.com/problems/GOODB/.

Os assuntos abordados neste problema são:

- Combinatória;
- Aritmética modular;
- Exponenciação rápida.

Neste problema, você precisa calcular a quantidade de configurações distintas módulo $10^9 + 7$ da resposta de N submissões de um time contendo exatamente W respostas erradas(WA), T tempo limite excedido(TLE) e R erros de execução(RE). Onde W + T + R = N.

Para facilitar a compreensão, observe um exemplo de entrada e sua respectiva saída.

Imagine que temos 3 submissões que precisa ser distribuída entre 2 respostas erradas, 1 tempo limite excedido e 0 erros de execuções.

A quantidade de maneiras distintas que essas submissões podem ser feitas para essa entrada é 3, sendo elas:

- WA, WA e TLE;
- WA, TLE e WA;
- TLE, WA e WA.

Vamos resolver agora para um caso geral, dado os valores inteiros de N, W, T e R.

No primeiro momento, temos N posições para W respostas erradas a serem inseridos. Portanto, temos a combinação N escolhe W.

Após escolher as posições das W respostas erradas, vamos ter um total de N-W posições para inserir T tempo limite excedido que contém N-W escolhe T possibilidades.

Por fim, vamos ter um total de N-W-T posições para inserir R erros de execução as quais podem ser reorganizadas em contém N-W-T escolhe R formas.

Para unir as 3 combinações bastas multiplicá-las e estaremos gerando todas as formas possíveis que as submissões podem ser feitas, assim a resposta final pode ser vista a baixo:

$$C\binom{N}{W} * C\binom{N-W}{T} * C\binom{N-W-T}{R} \tag{5.8}$$

A combinação N escolhe K pode ser calculada através do fatorial como pode ser vista a seguir:

$$C\binom{N}{K} = \frac{n!}{k! * (n-k)!} \tag{5.9}$$

Entretanto, como sabemos, devido às operações modulares a parte da divisão não pode ser feita de maneira convencional. Para isso, precisamos calcular o fatorial no módulo desejado e após isso na parte da divisão utilizaremos o conceito de inverso multiplicativo modular.

O algoritmo que calcula a combinação N escolhe R pode ser visto a seguir:

Algorithm 11 NCR - Combinação N escolhe R

Data: $n, r, m \in \mathbb{Z}$
Result: Combinação N escolhe R

```
101 início
102     if r > n then
103         return 0
104     dividendo ← fatorial[n]
105     divisor ← (fatorial[r] * fatorial[n − r]) mod m
106     inversoModular ← expMod(divisor, m − 2, m)
107     return (dividendo * inversoModular) mod m
```

Fonte: elaborado pelo autor.

A solução do problema pode ser feita como o algoritmo a seguir:

Algorithm 12 SOLVE

Data: $n, t, w, r \in \mathbb{Z}$
Result: Quantidade de formas distintas

```
108 início
109     m ← 10^9 + 7
110     ans ← nCr(n, t) mod m
111     ans ← ans * nCr(n − t, w) mod m
112     ans ← ans * nCr(n − t − w, r) mod m
113     return ans
```

Fonte: elaborado pelo autor.

5.3 Problemas de Síntese

Nesta seção, discutiremos três problemas:

- O problema Remainder Game retirado do juiz online CodeForces (https://codeforces.com/problemset/)
- Moves retirado do juiz online Code Chef (https://www.codechef.com/)
- Fantastic Beasts retirado do juiz online Neps Academy (https://neps.academy)

5.3.1 Problema DIV1B - Remainders Game

O problema Remainders Game pode ser encontrado no seguinte link https://codeforces.com/problemset/problem/687/B.

Neste problema, Pari e Arya estão jogando um jogo chamado Remainder. Neste jogo, Pari escolhe dois inteiros positivos x e k e desafia Arya a descobrir qual o valor de x módulo k. Para isso, existem n números anciões chamados c_1, c_2, ..., cn os quais Pari conta pra Arya o valor de x módulo c_i, caso Arya queira. Sua missão é descobrir se existe uma estratégia em que Arya sempre consegue responder corretamente independente do x escolhido.

Primeiro, vamos assumir que a resposta é não. Então, para isso acontecer, precisamos que pelo menos duas soluções existam x_1 e x_2 de tal forma que o resto desses números modulo c_i são iguais, mas o resto pelo módulo k sejam diferentes.

Como foi observado na seção 4.1.6.1, se uma solução x_1 e x_2 possuírem o mesmo resto para cada módulo c_i, podemos concluir que $\text{MMC}(c_1, c_2, ..., c_n) | (x_1 - x_2)$.

Também podemos concluir que $(x_1 - x_2)$ não é um múltiplo de k, já que sabemos que $x_1 \not\equiv x_2$ módulo k. Devido a isso podemos concluir que $k \not| \text{MMC}(c_1, c_2, ..., c_n)$.

Agora, assumindo que $k \not| \text{MMC}(c_1, c_2, ..., c_n)$, vamos provar que existem valores inteiros para x_1 e x_2 que possuem o mesmo resto no módulo c e diferentes, módulo k.

Um exemplo seria, x_1 = 2*MMC$(c_1, c_2, ..., c_n)$ e x_2 = MMC$(c_1, c_2, ..., c_n)$.

Com isso, conseguimos provar que para conseguirmos encontrar o valor de x módulo k, basta que MMC$(c_1, c_2, ..., c_n)$ e x_2 seja divisível por k.

A solução do problema pode ser implementado dessa maneira:

Algorithm 13 SOLVE

Data: $n, k, c[n] \in \mathbb{Z}$
Result: Se a Arya consegue ganhar o jogo!
Entrada: n, k, c[n]

```
início
    atual ← 1
    i ← 0
    while i < n do
        atual ← MDC(k, MMC(atual, c))
        i ← i + 1
    if atual == k then
        return "Yes"
    else
        return "No"
```

Fonte: elaborado pelo autor.

5.3.2 Problema MOVES

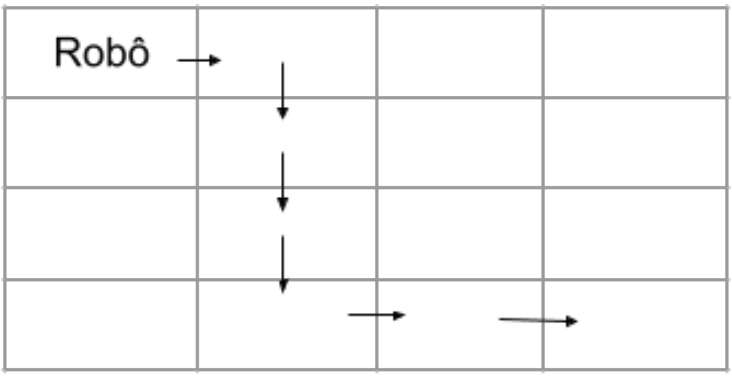
O problema MOVES pode ser encontrado no seguinte link https://www.codechef.com/problems/MOVES/. Os assuntos abordados neste problema são:

- Combinatória;
- Aritmética modular;
- Exponenciação rápida.
- Problema de combinatória estrelas e barras.

Neste problema, é fornecido uma matriz de tamanho N por N e um robô localizado na célula (1, 1). A tarefa do robô é chegar na célula (n, n) fazendo movimentos apenas para direita(R) e para baixo(D). Já sua tarefa é calcular a quantidade de caminhos distintos que este robô faz concluindo sua tarefa executando exatamente K viradas módulo $10^9 + 7$.

É considerado uma virada quando o robô muda de direção, ou seja, ele está indo para direita e depois vai para baixo ou vice e versa. Podemos observar um caminho válido com 2 viradas em uma matriz de tamanho 4 por 4 na imagem 5.

Figura 5: Exemplo de um caminho valido com 2 viradas em uma matriz de tamanho 4 x 4



Fonte: O Autor.

Todos os movimentos que o robô pode fazer em uma matriz de tamanho 4 por 4 dando exatamente 2 viradas são:

- RRDDDR;
- RDDDRR;
- DRRRDD;
- DDRRRD.

Observe que conseguimos representar os caminhos que começam com R como um conjunto de 1 ou mais R's concatenados com um conjunto de 1 ou mais D's e por fim concatenado com um conjunto de 1 ou mais R's de tal forma que a quantidade de R's e D's distribuídos seja igual a N-1.

Podemos representar todos os conjuntos de caminhos distintos começando com R e dando exatamente duas viradas da seguinte forma:

(R...)(D...)(R...).

Observe que essa afirmação também é valida para o caso onde começamos com o movimento indo para baixo, ou seja, a quantidade de caminhos distintos começando em cada direção é a mesma. Portanto, podemos calcular a resposta para uma direção específica e multiplicar por 2 para obtermos a resposta completa.

Sabendo disso, podemos supor sem perder a generalidade que o primeiro passo foi um movimento a direita e queremos saber agora quantos grupos de cada direção será necessário para cumprir a restrição de ter exatamente K viradas em uma matriz N por N.

Para isso vamos analisar os grupos para os quatro primeiros valores que K pode assumir, onde o primeiro movimento seria R:

- K = 1 -> (R...)(D...)
- K = 2 -> (R...)(D...)(R...)
- K = 3 -> (R...)(D...)(R...)(D...)
- K = 4 -> (R...)(D...)(R...)(D...)(R...)

Observe que a quantidade de conjuntos que representam uma sequência de R's é de tamanho $\lceil \frac{K+1}{2} \rceil$ e a quantidade de conjuntos que representam uma sequência de D's é de tamanho $\lfloor \frac{K+1}{2} \rfloor$.

Com isso, sabemos que precisamos dividir N-1 R's em $\lceil \frac{K+1}{2} \rceil$ grupos e N-1 D's em $\lfloor \frac{K+1}{2} \rfloor$ grupos de tal forma que cada grupo não fique vazio. Esse problema pode ser resolvido com combinatória, para isso vamos utilizar de um problema conhecido como estrelas e barras.

Figura 6: Figura com 9 estrelas antes da divisão em grupos.

Fonte: O Autor.

Para isso, suponha que temos um total de 9 estrelas e temos que dividir essas estrelas

em três grupos de tal forma que nenhum grupo fique vazio após esta divisão.

Cada grupo é considerado diferente do outro, entretanto as estrelas são consideradas iguais, portanto, apenas a quantidade de estrelas que compõe um grupo é considerada.

A primeira coisa a se fazer é garantir que cada grupo tenha pelo menos uma estrela, para isso será removido 3 estrelas para ser colocada uma em cada grupo, desta forma, teríamos apenas 6 estrelas para serem distribuídas nestes três novos grupos.

Uma forma de abstrair a divisão das estrelas nos grupos seria utilizar duas barra para dividir as estrelas, de tal modo que as estrelas que estiverem até a primeira barra pertencerá ao primeiro grupo, as estrelas que estiverem entre a primeira e a segunda barra pertencerá ao segundo grupo e por fim, as estrelas que estiverem após a segunda barra pertencerá ao terceiro grupo.

Um exemplo de uma possível divisão com barras pode ser observado na imagem 7.

Figura 7: Figura com 6 estrelas divididas por duas barras, representando a distribuição das estrelas para os três grupos.

Fonte: O Autor.

Observe que a quantidade de barras necessárias sempre será igual a 3 - 1.

O próximo passo é determinar a quantidade de distribuições que podem acontecer e isso pode ser respondido com combinatória.

A solução para o problema com 6 estrelas e 3 grupos seria a combinação $C\binom{8}{2}$, devido a existirem 8 possíveis posições onde essas barras podem permanecer e às duas barras a serem colocadas.

Portanto, dado um número inteiro qualquer X representando as estrelas e Y representando a quantidade de grupos a resposta seria $C\binom{X-Y+(Y-1)}{Y-1}$, ou seja, $C\binom{X-1}{Y-1}$. Onde o primeiro - Y seria da remoção das estrelas para distribuir entre os Y grupos e o +(Y - 1) seria a quantidade de barras necessárias.

Agora que já se sabe calcular as possíveis distribuições, vamos voltar ao problema original e concluir sua resposta. Sabemos que a quantidade de possíveis distribuições dos N-1 R's em $\lceil \frac{K+1}{2} \rceil$ grupos e dos N-1 D's em $\lfloor \frac{K+1}{2} \rfloor$ grupos, de tal forma que nenhum grupo fique vazio são respectivamente iguais a $C\binom{N-2}{\lceil \frac{K+1}{2} \rceil - 1}$ e $C\binom{N-2}{\lfloor \frac{K+1}{2} \rfloor - 1}$.

Observe que ao multiplicarmos a quantidade de possíveis distribuições dos N-1 R's e dos N-1 D's, vamos obter todos os caminhos distintos iniciando pela direita. Entretanto, o nosso problema quer a resposta de todos os caminhos distintos podendo iniciar em ambas as direções, para isso, basta multiplicarmos por 2 o resultado da multiplicação anterior, desta forma gerando todos os possíveis caminhos distintos saindo da célula (1,1) e chegando na célula (n, n). Para obter a resposta no módulo desejado, basta executar todas as operações utilizando a aritmética modular.

A solução do problema pode ser implementado dessa maneira:

Algorithm 14 SOLVE

Data: $n, k \in \mathbb{Z}$
Result: Quantidade de caminhos distintos
início
  $m \leftarrow 10^9 + 7$
  $rightAnswer \leftarrow (nCr(n-2, ceil((k+1)/2.0) - 1) * nCr(n-2, floor((k+1)/2.0) - 1)) \mod m$
  return $(2 * rightAnswer) \mod m$

Fonte: elaborado pelo autor.

5.3.3 Problema Fantastic Beasts

O problema Fantastic Beasts pode ser encontrado no seguinte link https://neps.academy/exercise/1217.

Este problema apareceu na Fase Nacional da Maratona de Programação Brasileira, competição que seleciona as melhores equipes do Brasil para a final mundial do International Collegiate Programming Contest (ICPC). Durante a competição, apenas 3 equipes brasileiras resolveram esse problema.

Neste problema, o zoólogo de animais mágicos Newt Scamande em uma visita Nlogônia acabou por deixar escapar b feras fantásticas de seu objeto mágico e agora ele está atrás delas para capturar novamente. Na cidade de Nlogônia, os habitantes amam zoológicos e por causa disso a cidade contém muito zoológicos espalhados pela cidade, para ser mais específico a cidade tem z zoológicos. Devido a acontecimentos passados, o Newt colocou um rastreador em cada uma de suas feras mágicas e ele consegue acompanhar a localização de suas feras em tempo real. Após observar por um tempo, o Newt observou um certo padrão nas suas feras.

A primeira coisa observada é que as feras apenas se movimentam entre os zoológicos e a cada segundo se passa uma fera escolhe mudar de zoológico ou se manter no zoológico

atual. Após um certo tempo, Newt notou que o próximo zoológico que uma fera escolhe ir depende de qual zoológico a fera está atualmente e, portanto anotou para cada uma de suas feras os movimentos que cada uma faz.

Newt percebeu que existe a possibilidade de ele pegar todas as feras em simultâneo, portanto, solicitou para você que faça um programa que responda em qual zoológico todas as feras vão se encontrar e em que tempo isso ocorrerá, caso exista.

Como entrada do problema, você receberá na primeira linha dois inteiros b e z representando respectivamente a quantidade de feras e a quantidade de zoológicos. Nas próximas b linhas, você receberá Z+1 inteiros P_0, P_1,...P_Z, onde P_0 representa onde a fera atual está e as outras posições P_i representa que se está fera estiver no zoo i o próximo zoológico a ser visitado sera o com valor P_i.

Para facilitar a compreensão do problema, observe a seguinte entrada com sua respectiva saída:

Entrada:

2 6

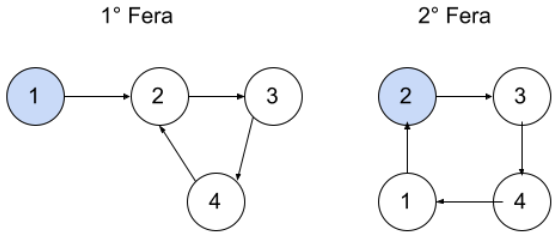
1 2 3 4 2

2 2 3 4 1

Saída:

2 4 .

Figura 8: Figura representando o movimento de cada fera em formato de grafo.



Fonte: O Autor.

A primeira observação que precisamos fazer é que o grafo gerado pelo movimento das bestas possui ciclos e após um determinado tempo todas as bestas estarão em algum ciclo formado por zoológicos.

A segunda observação é que a resposta está antes ou depois de todas as bestas

entrarem em algum ciclo de zoológicos. Observe que se andarmos com todas as bestas 2*z vezes vamos ter a certeza de que todas as bestas já vão estar em um ciclo e se guardarmos o momento que cada besta acessou aquele zoológico pela última vez vamos saber se aquele zoológico faz ou não parte de um ciclo, para isso basta checar se a última vez que ela foi acessada foi em um tempo menor ou igual z.

Devido ao ciclo ter tamanho máximo z, se andarmos 2*z vamos ter a garantia que vamos passar por todos os elementos do ciclo no mínimo 2 vezes. Portanto, se o tempo da última vez que aquela besta acessou aquele zoológico for menor ou igual z significa que ele só foi acessado uma vez e que não faz parte do ciclo.

Como o valor máximo de z é 100, então podemos simular o movimento das bestas até 2*z e se em algum momento todas as bestas estiverem na mesma posição, então a resposta já foi encontrada.

Caso a solução ainda não tenha sido encontrada, vamos precisar calcular de uma maneira mais inteligente utilizando as informações dos ciclos.

Depois que todas as bestas estão nos seus respectivos ciclos, podemos utilizar o teorema chinês do resto para descobrir em que momento todas as bestas se encontraram em um zoológico específico.

Na figura 8, observe que a 1° fera entra no ciclo de tamanho 3 no tempo 1 e após esse momento, toda vez que o resto da divisão do tempo pelo tamanho do ciclo for igual a 1, a 1° fera estará no zoológico 2. Já a segunda fera tem um ciclo de tamanho 4 e entra nesse ciclo no tempo 0, podemos fazer a mesma analise que a 1° fera e visualizar que toda vez que o resto da divisão do tempo pelo tamanho do ciclo for igual a 0, então a 2° fera estará no zoológico 2.

O seguinte sistema de congruências representa o tempo em que cada fera aparece no zoológico 2:

$$\begin{cases} t \equiv 1 \pmod 3 \\ t \equiv 0 \pmod 4. \end{cases}$$

Executando o teorema chinês do resto para duas equações, vamos obter a seguinte equação:

$$t \equiv 4 \pmod{12}. \tag{5.10}$$

Portanto, considerando apenas essas duas bestas, nós sabemos que elas se encontraram no zoológico 2 sempre que o resto da divisão do tempo por 12 for igual a 4. Caso houvesse uma terceira besta, seria necessário unir essa equação com a equação da outra besta.

Então, para cada zoológico, precisamos checar se esse zoológico faz parte de todos os ciclos gerados pelas feras, se alguma fera não consegue alcançar esse zoológico, então a resposta é descartada. Após isso, será formado um conjunto de equações utilizando o tempo que cada fera acessou aquele zoológico. O formato da equação pode ser visto a seguir:

$$t \equiv UltimoAcesso[fera][zoo] \pmod{TamanhoCiclo[fera]} \tag{5.11}$$

Por fim, basta utilizar o teorema chinês do resto para múltiplas equações e utilizar a menor resposta gerada.

O algoritmo que implementa o teorema chinês do resto para todas as equações pode ser visto a seguir:

Algorithm 15 CRT - TEOREMA CHINES DO RESTO

Data: $a[n], b[n] \in \mathbb{Z}$
Result: Equação de congruência linear resultante da união de todas equações.

```
início
    (ans, mod) ← (a[0], b[0])
    i ← 0
    while i < n do
        (ans, mod) ← UnirDuasEquações(ans, a[i], mod, b[i])
        if ans == −1 then
            return (−1, −1)
        i ← i + 1
    return (ans, mod)
```

Fonte: elaborado pelo autor.

Devido a complexidade da questão e o tamanho do seu código a solução final pode ser vista no GitHub https://github.com/dougnobrega/Solucoes-Tcc.

6 MINICURSO DE ALGORITMOS MATEMÁTICOS

Foi ofertado um minicurso de algoritmos matemáticos com carga horária estipulada de 8 horas. Este minicurso foi planejado de acordo com este trabalho, focando no desenvolvimento do aluno através do programa de estudos e fornecendo a explicação necessária para conseguir resolver os problemas.

Este minicurso foi proposto para ser desenvolvido no decorrer de 4 aulas com cada aula tendo uma duração de 2 horas. Deste tempo, 1 hora e 30 minutos foi reservado para explicação e os outros 30 minutos restantes para as dúvidas existentes e a apresentação das questões a serem resolvidas envolvendo aquele tópico.

O planejamento das aulas pode ser visto a seguir:

- Introdução à aritmética modular, exponenciação rápida e inverso multiplicativo modular;
- Resolução do inverso multiplicativo modular com Euclides estendido e curiosidades relacionadas.
- Introdução à equação de congruência linear;
- Resolução dos problemas sugeridos.

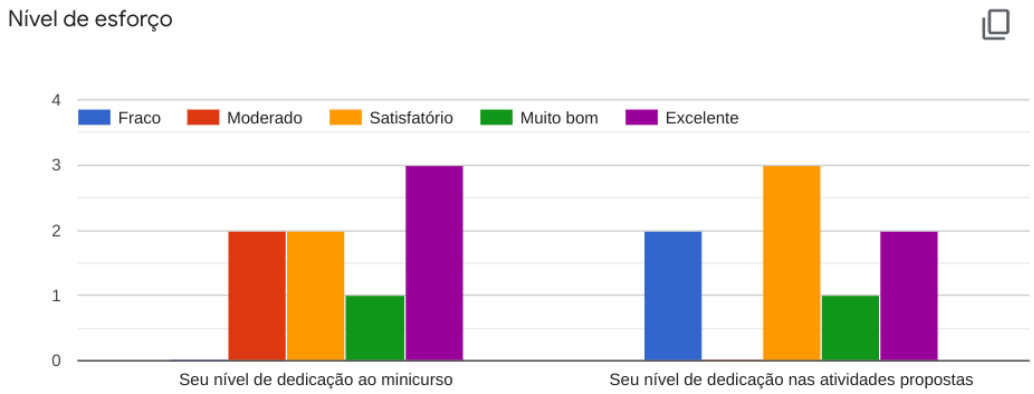
O minicurso de algoritmos matemáticos contou com vinte e uma inscrições e uma boa participação nas aulas oferecidas. Na primeira aula, tivemos um total de 14 participantes, e nas aulas seguintes em média 9 participantes.

Após a finalização do minicurso, foi repassado um formulário com o intuito de recolher o *feedback* dos alunos. Entretanto, devido à quantidade de participantes, não pôde ser utilizado com teor estatístico.

Este minicurso ocorreu de forma *online*, entretanto o recomendado seria ser executado de forma presencial para que houvesse a formação de equipes entre os participantes, desta forma, tendo maior proveito na comunicação, retenção de conteúdo e aproveitando de todas as características estipuladas que uma aprendizagem baseada em problemas deve possuir.

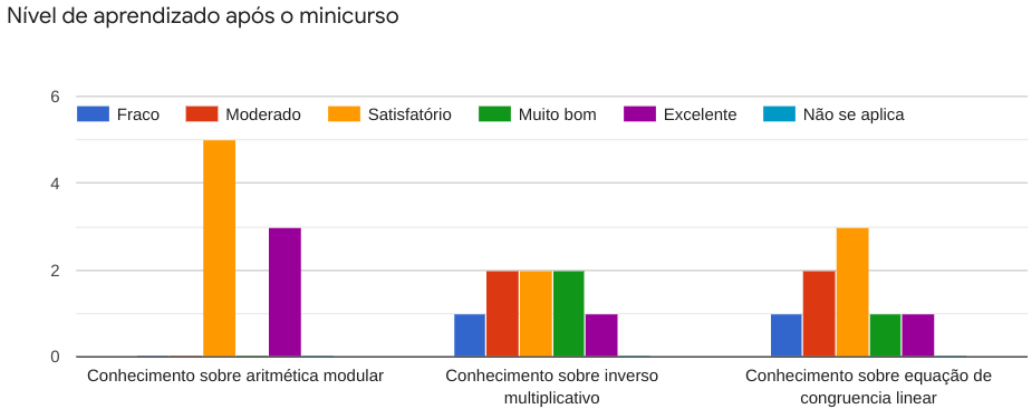
Na figura 10, podemos concluir que os conhecimentos adquiridos pelo minicurso foram satisfatórios. Principalmente, o assunto de aritmética modular que foi trabalhado durante todas as aulas. Observando a figura 9 podemos notar que apesar do nível de dedicação às atividades propostas terem sido abaixo do nível de dedicação ao minicurso, o objetivo final do minicurso foi alcançado.

Figura 9: Figura representando a estatística gerada pelo formulário em relação ao nível de esforço do estudante.



Fonte: O Autor.

Figura 10: Figura representando a estatística gerada pelo formulário em relação ao nível de aprendizado do estudante.



Fonte: O Autor.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este programa de estudos foi desenvolvido utilizando as características das metodologias ativas, a aprendizagem por descoberta e aprendizagem baseada em problemas, com o intuito de fornecer um material didático sistematizado que pode ser seguido de maneira autônoma ou com o auxílio de um professor.

Os problemas selecionados almejam desenvolver o conhecimento do aluno sobre os tópicos matemáticos de forma sequencial, propondo problemas contextualizados os quais estimulam a aprendizagem e motivam os estudantes. Formando então, um roteiro de estudos a ser seguido o qual desenvolve a habilidade de pensamento computacional e o conhecimento do aluno.

A utilização deste programa de estudos é recomendada para alunos de ensino superior por despertar a curiosidade, estimular a capacidade de ser autodidata, o desenvolvimento das habilidades comunicativas do aluno, coletividade no trabalho, resolução de problemas, pensamento computacional e conhecimento teórico.

Como trabalhos futuros, pretende-se desenvolver outros programas de estudos relacionados a outras temáticas que também podem ser aplicadas na programação, desenvolvendo em conjunto o tópico escolhido e o pensamento algorítmico. Alguns exemplos de tópicos que se relacionam com programação seria geometria, grafos, programação dinâmica, strings e entre outros. Além disso, pretende-se adicionar mais problemas ao programa de estudos ofertado.

REFERÊNCIAS

BALIM, A. G. The effects of discovery learning on students' success and inquiry learning skills. **Eurasian Journal of Educational Research (EJER)**, n. 35, 2009.

BARROWS, H. S. Problem-based learning in medicine and beyond: A brief overview. **New directions for teaching and learning**, Wiley Online Library, v. 1996, n. 68, p. 3–12, 1996.

BENTO, L. M. d. S. **Aplicações de Hashing**. Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012.

BOSSE, Y.; GEROSA, M. A. Reprovações e trancamentos nas disciplinas de introdução à programação da universidade de são paulo: Um estudo preliminar. In: SBC. **Anais do XXIII Workshop sobre Educação em Computação**. [*S. l.*], 2015. p. 426–435.

CHRISTODOULOU, M.; SZCZYGIEŁ, E.; KŁAPA, Ł.; KOLARZ, W. **Algorithmic and Programming**. [*S. l.*]: PTEA Wszechnica Sp. Zoo, 2018.

COMBEFIS, S.; BARRY, S. A.; CRAPPE, M.; DAVID, M.; MOFFARTS, G. de; HACHEZ, H.; KESSELS, J. Learning and teaching algorithm design and optimisation using contests tasks. **Olympiads in Informatics**, v. 11, p. 19–28, 2017.

FUTSCHEK, G. Algorithmic thinking: the key for understanding computer science. In: SPRINGER. **International conference on informatics in secondary schools-evolution and perspectives**. [*S. l.*], 2006. p. 159–168.

GLÓRIA, W. d. S. *et al.* **Teorema chinês dos restos**: ensino e aplicações. Dissertação (Mestrado em Matemática) – Universidade Federal do Amazonas, Manaus, 2019.

HIDAYAH, I.; ADJI, T. B.; SETIAWAN, N. A. Development and evaluation of adaptive metacognitive scaffolding for algorithm-learning system. **IET Software**, IET, v. 13, n. 4, p. 305–312, 2019.

LOPES, D. de C.; MELO, E. de C. **Desenvolvimento de algoritmos**, 2002.

LUO, Y.; ZHENG, H. An innovative teaching mode based on programming contest. In: SUGUMARAN, V.; XU, Z.; ZHOU, H. (Ed.). **Application of Intelligent Systems in Multi-modal Information Analytics**. Cham: Springer International Publishing, 2021. p. 469–477. ISBN 978-3-030-51556-0.

MARTIN, W. G. **Principles and standards for school mathematics**. [*S. l.*]: National Council of Teachers of, 2000. v. 1.

NIKHÁZY, L. Planets: A system for autonomous learning of algorithmic programming. In: **ISSEP (CEURWS Volume)**. [*S. l.*: *s. n.*], 2020. p. 79–90.

NIKHÁZY, L. A problem-based curriculum for algorithmic programming. **Central-European Journal of New Technologies in Research, Education and Practice**, p. 76–96, 2020.

PIERINI, M. F.; LOPES, R. M. **A Formação Interdisciplinar dos Professores de Ciências da Natureza Para a Integração Curricular Através da Aprendizagem Baseada em Problemas**. 2017. Disponível em: https://www.researchgate.net/publication/321918645_A_Formacao_Interdisciplinar_dos_Professores_de_Ciencias_da_Natureza_Para_a_Integracao_Curricular_Atraves_da_Aprendizagem_Baseada_em_Problemas. Acesso em: 05 ago. 2021.

PITEIRA, M.; COSTA, C. Learning computer programming: study of difficulties in learning programming. In: **Proceedings of the 2013 International Conference on Information Systems and Design of Communication**. [*S. l.*: *s. n.*], 2013. p. 75–80.

PITEIRA, M.; HADDAD, S. R. Innovate in your program computer class: an approach based on a serious game. In: **Proceedings of the 2011 Workshop on Open Source and Design of Communication**. [*S. l.*: *s. n.*], 2011. p. 49–54.

RIBEIRO, R. **Equações Diofantinas**: uma abordagem para o ensino médio. Dissertação (Mestrado Profissional em Matemática) – Universidade de Brasília, Brasília, 2014.

SVINICKI, M. D. A theoretical foundation for discovery learning. **Advances in physiology education**, v. 275, n. 6, p. S4, 1998.

WOOD, D. F. Problem based learning. **Bmj, British Medical Journal Publishing Group**, v. 326, n. 7384, p. 328–330, 2003.