



UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

JOÃO PEDRO DE SOUZA OLIVO TARDIVO

**RELATÓRIO TÉCNICO SOBRE O SIMULADOR DE
MÁQUINA DE TURING**



APUCARANA – PR
2023

JOÃO PEDRO DE SOUZA OLIVO TARDIVO

**RELATÓRIO TÉCNICO SOBRE O SIMULADOR DE MÁQUINA DE
TURING**

Trabalho apresentado à disciplina de
Linguagens Formais Autômatos e
Computabilidade, do curso de Bacharelado
em Ciência da Computação.

Professor: Guilherme Nakahata

**APUCARANA – PR
2023**

SUMÁRIO

INTRODUÇÃO	4
CAPÍTULO 1 - MOTIVAÇÃO E RECURSOS	6
1.1 LINGUAGENS DE PROGRAMAÇÃO E BIBLIOTECAS	6
1.2 FERRAMENTAS DE BUILD	7
1.3 ESTRUTURA DA INTERFACE GRÁFICA	9
1.4 ESTRUTURAS DE DADOS PARA A LÓGICA DA APLICAÇÃO	9
1.5 ESTRUTURA LÓGICA DA SIMULAÇÃO	10
CAPÍTULO 2 - RESULTADOS	13
CONCLUSÃO	17
REFERÊNCIAS	18

INTRODUÇÃO

No domínio da ciência da computação, a exploração da teoria computacional permanece como uma pedra angular. Dentro desta disciplina, as Máquinas de Turing, concebidas pelo visionário matemático britânico Alan Turing, assumiram um papel eminente. A sua importância reside não apenas no seu estatuto fundamental, mas também na sua utilidade como poderosas construções teóricas que iluminam os limites e capacidades da computação. É neste contexto intelectual que se desenvolve este trabalho.

Essas construções teóricas, com sua fita infinita, transições de estado e capacidade computacional, têm sido há muito tempo a vanguarda da compreensão dos limites do poder computacional. A sua importância reside na sua capacidade de modelar abstractamente o funcionamento de computadores e algoritmos. Como máquinas versáteis, mas elegantemente simples, as Máquinas de Turing têm sido fundamentais na decifração das linguagens formais subjacentes à computação e têm desempenhado um papel indispensável na elucidação de problemas que podem ser resolvidos pela computação.

Na esfera das Linguagens Formais e da Teoria dos Autômatos, as Máquinas de Turing servem como pedra angular. Eles fornecem uma estrutura rigorosa para explorar as construções teóricas que definem os limites da computação, iluminando assim a complexidade inerente e a capacidade de decisão dos problemas. Este formalismo, por sua vez, equipa os cientistas da computação com as ferramentas para analisar o poder expressivo das linguagens de programação, investigar as complexidades das expressões regulares e sondar a profundidade das gramáticas livres de contexto.

No panorama mais amplo da ciência da computação, as Máquinas de Turing não estão apenas confinadas aos domínios teóricos, mas também têm significado prático. Sua influência se estende às áreas de projeto de algoritmos, inteligência artificial e construção de compiladores, onde a compreensão dos limites e possibilidades computacionais é fundamental.

Desta forma, este trabalho busca transformar a teoria abstrata em software tangível. O objetivo é criar um simulador de Máquina de Turing que

não apenas encapsule os princípios básicos das Máquinas de Turing, mas também preencha a lacuna entre a teoria e a aplicação, proporcionando aos alunos uma compreensão prática desses conceitos fundamentais.

CAPÍTULO 1

MOTIVAÇÃO E RECURSOS UTILIZADOS DURANTE A IMPLEMENTAÇÃO

Na jornada de desenvolvimento de software, a implementação bem-sucedida de um projeto depende da seleção e utilização criteriosa de recursos. Estes recursos abrangem uma ampla gama de elementos, desde linguagens de programação e bibliotecas até estruturas de dados, ferramentas e metodologias. Neste capítulo, será aprofundado na intrincada rede de recursos que desempenharam um papel fundamental na definição do resultado deste projeto.

Ao dissecar esses componentes e explorar a lógica por trás de sua seleção e utilização, será esclarecido como a aplicação foi elaborada e quais escolhas foram feitas em cada etapa.

1.1 LINGUAGENS DE PROGRAMAÇÃO E BIBLIOTECAS

Inicialmente, a lógica central da aplicação foi desenvolvida em Java, principalmente por ser a linguagem de programação enfatizada nas aulas universitárias do autor. Java é conhecido por sua portabilidade e robustez, tornando-o uma escolha popular para instituições educacionais e desenvolvimento de software. No entanto, à medida que o projeto avançava, tornou-se evidente que a biblioteca gráfica JavaFX, comumente usada com Java para criar interfaces de usuário, apresentava alguns desafios. A experiência do autor com JavaFX não foi ideal e ficou evidente que era necessário explorar opções alternativas para melhorar a interface do usuário e a experiência geral de desenvolvimento.

Na busca por uma alternativa, Python surgiu como uma escolha atraente. Python é conhecido por sua simplicidade e legibilidade, o que o torna uma linguagem excelente para o rápido desenvolvimento de aplicativos. Embora possa não ter o mesmo desempenho que o Java em termos de velocidade de execução bruta, ele se destaca por fornecer um ambiente de desenvolvimento mais rápido, permitindo que os desenvolvedores iterem rapidamente em seus projetos.

A verdadeira virada de jogo, entretanto, foi a descoberta do PyQt6, uma ligação Python para o framework Qt. As capacidades do Qt na criação de interfaces gráficas multiplataforma e sua extensa biblioteca de widgets e ferramentas abriram novas possibilidades para o projeto. A integração do PyQt6 ao projeto possibilitou a utilização do poder da estrutura Qt e, ao mesmo tempo, a sintaxe concisa e expressiva do Python. Essa combinação não apenas melhorou a qualidade da interface gráfica, mas também acelerou significativamente o processo de desenvolvimento. O resultado foi um aplicativo mais sofisticado e fácil de usar, que se beneficia do melhor dos dois mundos: a versatilidade do Qt e a agilidade do Python.

A título de esclarecimento, Qt é um *framework* popular e robusto para a criação de interfaces gráficas, que possui uma comunidade de usuários grande e ativa, isso significa que os desenvolvedores têm acesso a uma grande variedade de recursos, incluindo fóruns, listas de discussão e documentação online. Este apoio comunitário pode ser inestimável quando se deparam com desafios durante o desenvolvimento.

Além disso, Qt é conhecido por seu desempenho, tornando-o adequado para aplicações que exigem capacidade de resposta em tempo real e gerenciamento eficiente de recursos. Isso é especialmente crucial para aplicativos como jogos, simulações e software de desktop de alto desempenho.

A arquitetura modular do Qt permite que os desenvolvedores selecionem apenas os componentes necessários, minimizando o tamanho e a complexidade da aplicação. Além disso, o Qt pode ser facilmente estendido com código personalizado, garantindo flexibilidade e adaptabilidade para requisitos específicos do projeto.

1.2 FERRAMENTAS DE BUILD

Outrossim, o projeto utiliza o pacote PyInstaller, o que permite aos desenvolvedores empacotar aplicativos Python em executáveis independentes, facilitando a distribuição de software aos usuários finais. Isso elimina a necessidade dos usuários instalarem o Python e as dependências separadamente, simplificando o processo de instalação.

O pacote agrupa automaticamente o interpretador Python e as dependências necessárias no executável, eliminando a necessidade de gerenciamento manual de dependências. Isso reduz problemas de compatibilidade e torna a implantação mais confiável.

Ademais, é uma ferramenta de código aberto, o que significa que está disponível gratuitamente e pode ser usada para projetos pessoais e comerciais sem custos de licenciamento. Isto o torna uma escolha atraente para projetos com restrições orçamentárias.

Por outro lado, aplicativos Java empacotados como arquivos .jar podem encontrar problemas de execução, muitas vezes exigindo que os usuários naveguem na linha de comando ou criem arquivos .bat, o que pode ser confuso para usuários não técnicos.

Ora, aplicativos Java geralmente exigem que os usuários tenham a versão correta do Java Runtime Environment (JRE) instalada, levando a incompatibilidades de versão e erros de tempo de execução.

Por fim, para atingir o objetivo de multiplataforma, tendo em vista que o ambiente de desenvolvimento do autor é o Windows 10 Enterprise LTSC, foi utilizada uma combinação de WSL (Windows Subsystem for Linux), Oracle VM VirtualBox e Ubuntu.

O WSL permite que os desenvolvedores executem uma distribuição Linux junto com o Windows, fornecendo uma maneira perfeita de desenvolver e testar aplicativos compatíveis com Linux em uma máquina Windows. Essa compatibilidade entre plataformas garante que os desenvolvedores possam trabalhar em seu ambiente Windows preferido enquanto direcionam executáveis Linux.

Todavia, como trata-se de uma aplicação com interface gráfica, a Oracle VM VirtualBox também foi utilizada, uma plataforma de virtualização que permite a criação de máquinas virtuais (VMs) isoladas. Ao configurar uma VM Ubuntu no VirtualBox, os desenvolvedores podem criar um ambiente de teste Linux dedicado, garantindo que o executável gerado pelo PyInstaller funcione corretamente em um sistema Linux genuíno.

O Ubuntu é uma das distribuições Linux mais populares, com ampla compatibilidade e uma grande base de usuários. Desenvolver e testar no

Ubuntu garante compatibilidade com uma parte significativa da comunidade de usuários Linux.

Depois de testar e empacotar o aplicativo Python no Ubuntu, o executável compatível com Linux resultante pode ser distribuído aos usuários Linux com confiança, sabendo que foi testado em um ambiente Linux autêntico.

1.3 ESTRUTURA DA INTERFACE GRÁFICA

Neste projeto, uma abordagem de layout empilhado é adotada para gerenciar diferentes telas ou layouts de forma eficiente. A aplicação inicia com uma tela de boas-vindas e, à medida que o usuário interage com a GUI, novos layouts são carregados, adicionados a uma pilha e exibidos com base em suas escolhas.

Essa abordagem oferece vários benefícios como a modularidade, onde cada tela ou layout é desenvolvido de forma independente, facilitando o foco em um aspecto do aplicativo por vez.

Ademais, tem-se uma melhor eficiência de memória, pois a medida que o usuário avança, novos layouts são adicionados à pilha. No entanto, se o usuário optar por voltar, o layout anterior será removido da pilha e efetivamente “destruído”. Isso ajuda a economizar memória e recursos, garantindo que o aplicativo permaneça responsivo e eficiente mesmo durante fluxos de trabalho complexos.

Finalmente, temos uma separação relativa de funções, onde cada layout pode encapsular sua própria lógica e funcionalidade, promovendo uma divisão mais clara de interesses na base de código do aplicativo.

1.4 ESTRUTURAS DE DADOS PARA A LÓGICA DA APLICAÇÃO

Neste projeto de simulação da Máquina de Turing, diversas estruturas de dados desempenham um papel fundamental na modelagem e execução do comportamento da máquina. Uma estrutura de dados chave é uma matriz 2D que serve como tabela de transição. Esta matriz possui os estados da Máquina de Turing como linhas e cada símbolo do alfabeto como colunas. Cada índice

nesta matriz armazena uma instância de uma classe de transição personalizada.

A classe de transição encapsula informações vitais sobre a transição, incluindo o estado atual, o próximo estado, a letra atual na fita, a letra de substituição, a direção na qual o ponteiro da fita se moverá (esquerda ou direita) e se o estado é final. Essa matriz 2D bem organizada e a classe de transição personalizada formam, juntas, a espinha dorsal da funcionalidade da Máquina de Turing, facilitando o gerenciamento e a execução eficiente de transições complexas.

Garantir que os símbolos do alfabeto usados na Máquina de Turing sejam únicos e bem definidos é crucial para a correção e previsibilidade da simulação. Para conseguir isso, uma estrutura de dados HashSet ou Set em Python é empregada ao receber as entradas de símbolos do alfabeto. Um HashSet é uma coleção que garante a exclusividade de seus elementos. Neste contexto, um HashSet é utilizado para armazenar e validar os símbolos do alfabeto fornecidos pelo usuário ou definidos nas regras de transição. Quando um novo símbolo é encontrado, ele é adicionado ao HashSet. Caso o símbolo já exista no HashSet, o alfabeto inserido não será aceito, evitando a introdução acidental de símbolos duplicados.

Ao usar um HashSet para gerenciar símbolos alfabéticos, o projeto garante que a Máquina de Turing opere em um conjunto válido e distinto de símbolos, evitando ambiguidades e erros nas definições de regras de transição. Essa estrutura de dados simplifica a tarefa de verificação da exclusividade do símbolo, aumentando a confiabilidade da simulação e facilitando aos usuários a definição e o trabalho com regras de transição personalizadas.

1.5 ESTRUTURA LÓGICA DA SIMULAÇÃO

O núcleo da simulação da Máquina de Turing reside na sua lógica orquestrada por um loop while que se adapta dinamicamente à operação da máquina. A utilização deste loop se justifica na impossibilidade de determinação de quantas iterações são necessárias para executar a Máquina

de Turing, tendo em vista que o número de passos pode variar com base nas regras de entrada e transição.

O loop while começa no estado inicial, pronto para modificar a fita e executar transições. Dentro do loop while, um loop for aninhado entra em ação. Este loop interno é responsável por examinar todas as regras de transição associadas ao estado atual.

A Máquina de Turing percorre a tabela de transição, movendo-se ao longo de suas colunas para verificar cada regra. Para cada regra, verifica se o símbolo da fita atual corresponde ao símbolo especificado na regra. Se houver uma correspondência, a máquina atualiza a fita de acordo com a regra, desloca o cabeçote da fita para a esquerda ou para a direita, faz a transição para o próximo estado e continua o processo. Essas iterações continuam enquanto transições válidas são encontradas.

No entanto, se a Máquina de Turing examinar exaustivamente todas as regras de transição disponíveis para o estado atual e não conseguir identificar uma transição válida, ela é considerada presa. A incapacidade de progredir indica que a palavra de entrada não pode ser processada posteriormente usando as regras de transição definidas.

Nesse caso, a máquina avalia se está no estado final. Se estiver e a cabeça da fita estiver posicionada no primeiro índice da fita, a palavra de entrada é considerada aceita. Por outro lado, se a máquina estiver presa em um estado final, mas a cabeça da fita não estiver no primeiro índice, ela sinaliza uma rejeição, identificando a causa exata da rejeição. Essas mensagens de rejeição servem como ferramentas de diagnóstico para entender o que especificamente deu errado na execução da máquina.

Para fornecer ao usuário final uma compreensão abrangente da execução da Máquina de Turing, um mecanismo de log é implementado na simulação. Este mecanismo de log consiste em um array que armazena strings customizadas para cada transição executada pela máquina.

Durante a execução da Máquina de Turing, à medida que ela processa entradas e realiza transições, uma entrada de log é gerada para cada etapa. Essas entradas de log são dinamicamente elaboradas para incluir informações essenciais sobre a transição, como o estado atual, o símbolo de entrada, o símbolo de substituição, a direção do movimento do cabeçote da fita e o

próximo estado. Essas entradas servem como um log detalhado de cada ação realizada pela Máquina de Turing durante sua execução.

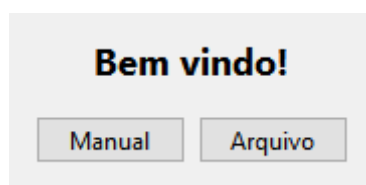
A matriz de log atua como um relato cronológico da jornada da máquina, permitindo ao usuário final rastrear toda a execução passo a passo. Esse recurso é inestimável para fins educacionais e de depuração e para obter insights sobre como a Máquina de Turing processa entradas e toma decisões com base nas regras de transição. Os usuários podem revisar o log para identificar qualquer comportamento inesperado ou para confirmar a exatidão da simulação.

Em essência, o mecanismo de registro aumenta a transparência e a interpretabilidade da simulação da Máquina de Turing, tornando-a uma ferramenta valiosa para fins de aprendizagem e solução de problemas. Ele permite que os usuários se aprofundem nas complexidades da computação, promovendo uma compreensão mais profunda do funcionamento interno da máquina.

CAPÍTULO 2

RESULTADOS

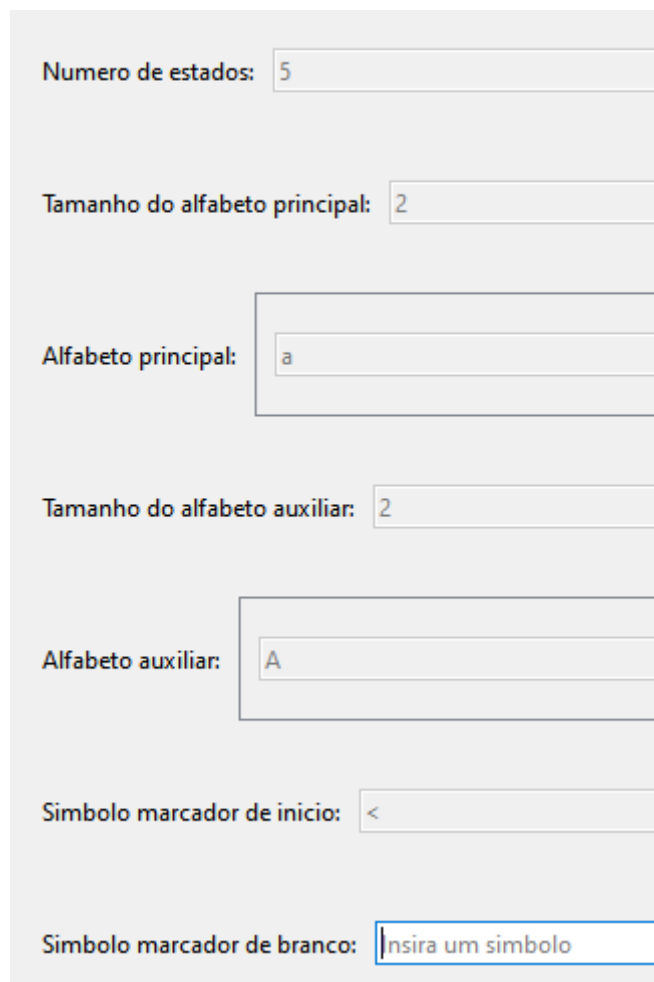
A aplicação de simulação Máquina de Turing oferece uma experiência de usuário intuitiva e interativa por meio de uma interface gráfica amigável. Os usuários são guiados por um processo estruturado, começando com uma tela de boas-vindas e depois tendo a opção de inserir os parâmetros da máquina manualmente ou por meio de arquivo.



Bem vindo!

Manual Arquivo

No modo de entrada “manual”, os usuários são conduzidos através de um processo passo a passo para definir os parâmetros essenciais da Máquina de Turing:



Numero de estados: 5

Tamanho do alfabeto principal: 2

Alfabeto principal: a

Tamanho do alfabeto auxiliar: 2

Alfabeto auxiliar: A

Simbolo marcador de inicio: <

Simbolo marcador de branco: Insira um simbolo

Desta forma, o usuário estabelece o número de estados, bem como o conjunto de símbolos utilizados pela Máquina de Turing, o que terá um grande impacto na tabela de transição que terá N linhas e M colunas, onde N é o número de estados e M é a quantidade de símbolos.

Aqui, são aplicadas medidas de verificação rigorosas para garantir que os usuários forneçam informações precisas e válidas para configurar a Máquina de Turing. Estas etapas de verificação visam evitar erros e inconsistências na definição dos parâmetros da máquina.

Resumidamente, a aplicação verifica se cada entrada de quantidades é um número inteiro, bem como garante que cada símbolo inserido seja único. Isso inclui verificações do comprimento do símbolo e do tipo de caractere para manter a consistência.

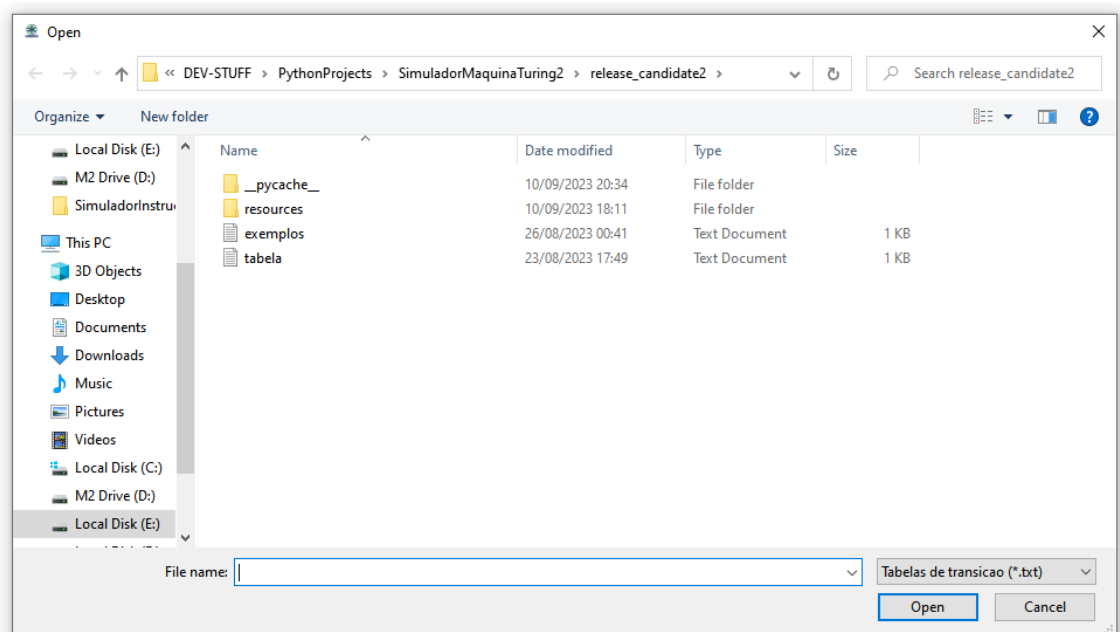
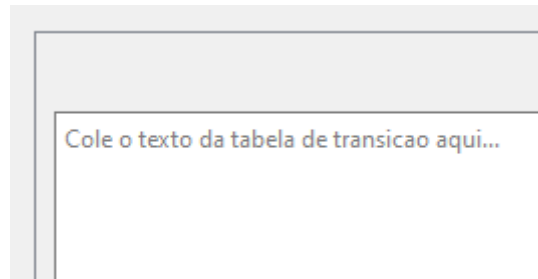
Após a conclusão bem-sucedida dessas entradas, os usuários avançam para uma tela de tabela de transição dinâmica, onde podem definir os detalhes de cada transição, incluindo substituições de símbolos, direção de movimento e transições de estado. Eles também podem designar determinados estados como iniciais ou finais.

Estados	a	b
S[0]	0,0	0,1
S[1]	1,0	1,1
S[2]	2,0	2,1
S[3]	3,0	3,1
S[4]	4,0	4,1

	Estado Inicial?	Estado Final?
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

No modo de entrada “arquivo”, o usuário tem a opção de colar um texto pré-formatado em uma caixa de texto ou abrir um navegador de arquivos para

abrir .txts contendo parâmetros da máquina e informações da tabela de transição. Este método de entrada simplificado agiliza o processo de configuração para usuários que preferem definir sua máquina fora da aplicação.



Independentemente do modo de entrada escolhido, os usuários podem inserir palavras para serem testadas pela Máquina de Turing. O aplicativo simula meticulosamente cada execução de teste, fornecendo aos usuários resultados detalhados que mostram como ocorreu cada transição. Esse feedback abrangente permite que os usuários entendam completamente o comportamento da máquina e o resultado de seus testes.

Resultado da fita

<AABB>>>

Palavra aceita!

Transicoes realizadas

Trocou 'a' com 'a'. Agora movendo-se 'R' para o estado '1'
Trocou 'a' com 'a'. Agora movendo-se 'R' para o estado '1'
Trocou 'b' com 'b'. Agora movendo-se 'L' para o estado '2'
Trocou 'a' com 'a'. Agora movendo-se 'L' para o estado '2'
Trocou 'A' com 'A'. Agora movendo-se 'L' para o estado '2'
Trocou '<' com '<'. Agora movendo-se 'R' para o estado '0'
Trocou 'A' com 'A'. Agora movendo-se 'R' para o estado '0'
Trocou 'a' com 'a'. Agora movendo-se 'R' para o estado '1'
Trocou 'B' com 'B'. Agora movendo-se 'R' para o estado '1'
Trocou 'b' com 'b'. Agora movendo-se 'L' para o estado '2'
Trocou 'B' com 'B'. Agora movendo-se 'L' para o estado '2'
Trocou 'A' com 'A'. Agora movendo-se 'L' para o estado '2'

A interface do usuário do aplicativo foi projetada para ser fácil de usar, melhorando a experiência geral do usuário. No geral, o aplicativo permite que os usuários explorem e experimentem simulações da Máquina de Turing, oferecendo controle manual e uma abordagem conveniente baseada em arquivo, ao mesmo tempo que fornece insights detalhados sobre a execução de cada caso de teste.

CONCLUSÃO

Isto posto, o esforço para desenvolver uma simulação de Máquina de Turing usando Python e PyQt6 provou ser uma exploração interessante no reino da teoria computacional e nos aspectos práticos do desenvolvimento de algoritmos, estruturas de dados e aplicações GUI.

Ao longo deste projeto, uma melhor compreensão das Máquinas de Turing foi alcançada. Essas construções teóricas, que servem como pilares fundamentais da ciência da computação contemporânea, foram examinadas de forma mais detalhada. Conceitos fundamentais como estados, regras de transição e manipulação de fitas foram analisados, contribuindo para o aprendizado da teoria computacional.

O processo de implementação da simulação da Máquina de Turing usando Python e PyQt6 facilitou um aprimoramento das habilidades de programação bem como na resolução de problemas que abrangem aspectos como a validação de entradas, a garantia da correção das regras de transição e o fornecimento de feedback aos usuários.

A importância da documentação do projeto foi sublinhada. O fornecimento de instruções claras e precisas é fundamental, pois permite aos usuários compreender e utilizar o simulador da Máquina de Turing de maneira eficaz.

Em suma, esta tarefa proporcionou uma oportunidade distinta para aprofundar as dimensões teóricas e práticas das Máquinas de Turing e do desenvolvimento de software, expandindo, em última análise, horizontes no campo da ciência da computação.

REFERÊNCIAS

Java. Disponível em: <https://www.java.com>. Acesso em: 23 jul. 2023.

OpenJFX. Disponível em: <https://openjfx.io/>. Acesso em: 23 jul. 2023.

Python. Disponível em: <https://www.python.org/>. Acesso em: 23 jul. 2023.

Qt. Disponível em: <https://www.qt.io/>. Acesso em: 23 jul. 2023.

PyQt6 - PyPI. Disponível em: <https://pypi.org/project/PyQt6/>. Acesso em: 23 jul. 2023.

PyQt - Riverbank Computing. Disponível em: <https://www.riverbankcomputing.com/software/pyqt/>. Acesso em: 23 jul. 2023.

PyInstaller - PyPI. Disponível em: <https://pypi.org/project/pyinstaller/>. Acesso em: 23 jul. 2023.

PyInstaller Documentation. Disponível em: <https://pyinstaller.org/en/stable/>. Acesso em: 23 jul. 2023.

Windows Subsystem for Linux Installation Guide. Disponível em: <https://learn.microsoft.com/en-us/windows/wsl/install>. Acesso em: 23 jul. 2023.

Ubuntu. Disponível em: <https://ubuntu.com/download>. Acesso em: 23 jul. 2023.

VirtualBox. Disponível em: <https://www.virtualbox.org/>. Acesso em: 23 jul. 2023.