MAPA DO TESOURO

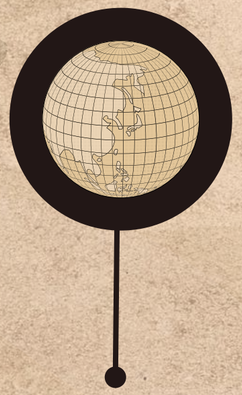
QUAL A IDEIA PARA O PROJETO?

Gerar um trajeto para que um aventureiro ou aventureira possa encontrar o tesouro com o melhor tempo possível, ou seja, aquele que percorrer menos pela ilha ganha!

MAPA DA ILHA

OBJETIVOS

Achar o tesouro!



Não perder tempo em caminhos alternativos!

Memorizar os caminhos utilizados!

Ter o melhor tempo

PROCESSO

01

Você inicia seu caminho pela praia.

02

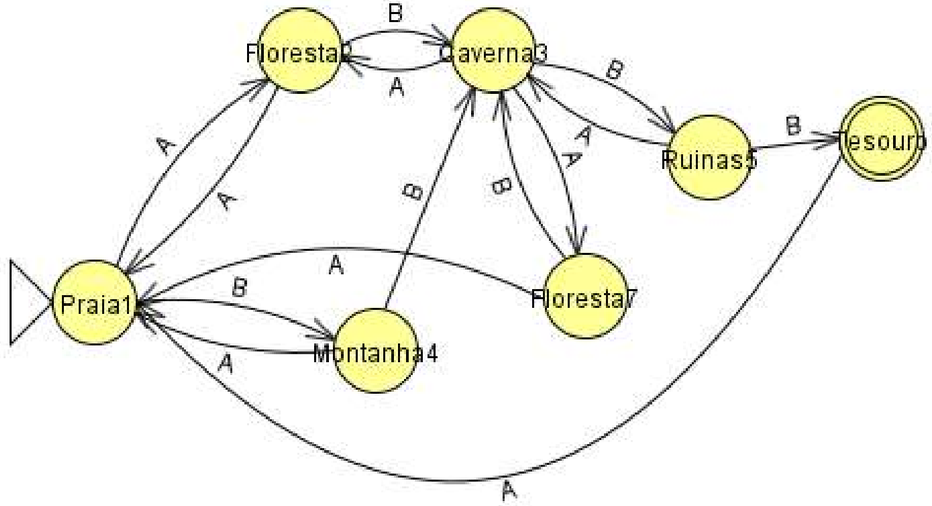
Decide por onde vai passar.

03

A glória eterna, encontra o tesouro perdido.

JFLAP

O JFLAP, ou "Java Formal Language and Automata Package", é um software gratuito e multiplataforma que permite criar e simular diversos tipos de autômatos, como autômatos finitos determinísticos (AFDs), autômatos finitos não determinísticos (AFNs), autômatos com pilha (APs) e máquinas de Turing (MTs).



FUNCIONAMENTO

Definição da Classe “MapaTesouro”

- Esta classe define o mapa do tesouro com seus estados possíveis, transições entre os estados e dicas associadas a cada estado e transição.
- Também define o estado inicial e final do mapa

Método “transitar(self, estado_atual, pista)”

- Este método recebe o estado atual e uma pista como entrada e retorna o próximo estado com base na transição especificada pela pista.
- Ele verifica se o estado atual e a pista fornecida existem nas transições definidas. Se existirem, retorna o próximo estado. Caso contrário, retorna None.

FUNCIONAMENTO

Método obter_dica(self, estado_atual, pista)

- Este método retorna a dica associada ao estado atual e à pista fornecida.
- Ele verifica se o estado atual e a pista fornecida existem nas dicas definidas. Se existirem, retorna a dica correspondente. Caso contrário, retorna None.

Função testar_autômato()

- Esta função cria uma instância da classe MapaTesouro e inicializa o estado atual como o estado inicial do mapa.
- Em um loop while, solicita ao jogador uma pista (A ou B) e avança para o próximo estado com base na pista fornecida, usando os métodos transitar() e obter_dica() da instância do mapa.
- O loop continua até que o estado atual seja o estado final (o tesouro).
- Durante cada iteração do loop, imprime a dica correspondente, se houver, e o próximo estado.
- Ao encontrar o tesouro, imprime o caminho percorrido.

CÓDIGO

```python
class MapaTesouro:
    def __init__(self):
        # Define os estados possíveis do mapa do tesouro
        self.estados = {'Praia', 'Floresta', 'Caverna', 'Montanha', 'Ruínas', 'Tesouro'}
        # Define as transições possíveis entre os estados, representadas por pistas "A" e "B"
        self.transicoes = {
            'Praia': {'A': 'Floresta', 'B': 'Montanha'},
            'Floresta': {'A': 'Praia', 'B': 'Caverna'},
            'Caverna': {'A': 'Floresta', 'B': 'Ruínas'},
            'Montanha': {'A': 'Praia', 'B': 'Caverna'},
            'Ruínas': {'A': 'Caverna', 'B': 'Tesouro'}
        }
        # Define as dicas para cada estado e pista
        self.dicas = {
            'Praia': {'A': 'As árvores guardam o caminho.', 'B': 'As alturas revelam segredos.'},
            'Floresta': {'A': 'Retornar à origem pode revelar novos caminhos.', 'B': 'A escuridão esconde a verdade.'},
            'Caverna': {'A': 'A luz natural te guiará de volta.', 'B': 'Antigas construções sussurram histórias de tesouros.'},
            'Montanha': {'A': 'Onde a areia encontra o mar, um novo começo te espera.', 'B': 'Profundezas ocultas guardam segredos.'},
            'Ruínas': {'A': 'Os ecos do passado podem te confundir.', 'B': 'O final da jornada está próximo.'}
        }
        # Define o estado inicial e o estado final do mapa do tesouro
        self.estado_inicial = 'Praia'
        self.estado_final = 'Tesouro'

    def transitar(self, estado_atual, pista):
        # Verifica se o estado e a pista fornecidos levam a um próximo estado válido
        if estado_atual in self.transicoes and pista in self.transicoes[estado_atual]:
            return self.transicoes[estado_atual][pista]
        else:
            return None
```

```python

    def obter_dica(self, estado_atual, pista):
        # Retorna a dica correspondente ao estado e pista fornecidos
        if estado_atual in self.dicas and pista in self.dicas[estado_atual]:
            return self.dicas[estado_atual][pista]
        else:
            return None


def testar_autômato():
    mapa = MapaTesouro()
    estado_atual = mapa.estado_inicial
    caminho = [estado_atual]  # Inicializa a lista para armazenar o caminho percorrido
    print(f"Começando na {estado_atual}")
    # Loop enquanto o estado atual não for o estado final (Tesouro)
    while estado_atual != mapa.estado_final:
        # Solicita ao jogador uma pista (A ou B)
        pista = input("Insira a pista encontrada (A ou B): ").upper()
        # Obtém o próximo estado com base na pista fornecida
        proximo_estado = mapa.transitar(estado_atual, pista)
        if proximo_estado:
            # Se a transição for válida, imprime a dica, o próximo estado e adiciona ao caminho
            dica = mapa.obter_dica(estado_atual, pista)
            if dica:
                print(f"Dica: {dica}")
            print(f"Seguindo a pista '{pista}' para a {proximo_estado}")
            estado_atual = proximo_estado
            caminho.append(estado_atual)
```

```python
            estado_atual = proximo_estado
            caminho.append(estado_atual)
        else:
            # Se a pista fornecida for inválida, solicita ao jogador que tente novamente
            print("Pista inválida. Tente novamente.")
    # Imprime o caminho percorrido ao encontrar o tesouro
    print("Parabéns! Você encontrou o tesouro!")
    print("Caminho percorrido:")
    print(" -> ".join(caminho))


testar_autômato()
```

ACESSO

Google Colab

AFD_tesouro_CC_D1.ipynb

OBRIGADO!

Alunos

Nicollas Monteiro - 28287274
Gustavo Henrique - 27636666
Ian Reis - 28158083

Curso e Matéria

Ciência da Computação
Linguagens Formais e Autômatos