Universidade Federal do Amazonas

Engenharia de Software

Algoritmos e Estruturas de Dados II

Joseph Ferreira Paz - 22051566

Rebeca Freitas Pedrosa - 22153224

Codibentinho e o Labirinto

MANAUS / AM

2023

1. Descrição do problema escolhido.

Codibentinho é o mais novo calouro do curso de Engenharia de Software na UFAM. Infelizmente, Codibentinho está atrasado para sua primeira aula do curso, ao ver Codibentinho aflito, você (Veterano) lembra da sua época de calouro e decide ajudar Codibentinho a chegar a sua sala o mais rápido possível. Codibentinho pode se mover livremente pelas dependências da faculdade até encontrar a sala. Sua tarefa é acompanhar Codibentinho até a sala em um labirinto de M linhas e N colunas.

2. Explicação do procedimento de busca implementado.

Utilizamos o algoritmo A* para realizar a busca do caminho mais eficiente a partir do ponto de partida até o fim do labirinto. O processo se inicia com a inicialização de uma fila de prioridades, denominada "heap" no código, responsável por armazenar os estados a serem explorados e classificá-los com base em uma estimativa de custo total. Essa estimativa é calculada pela função "heuristic," que utiliza a distância de Manhattan para determinar o custo.

$$d(p1\,,p2\,)\;=\;|\,p1\,p2\,x1\;-\;x2|\;+\;|y1\;-\;y2|$$

A busca procede selecionando o estado com o menor custo na fila de prioridade e expandindo seus estados vizinhos válidos, evitando aqueles que já foram visitados. O controle das visitas é realizado por meio de um conjunto chamado "visited."

O caminho percorrido é construído passo a passo, com as coordenadas armazenadas em um dicionário chamado "path." Esse dicionário é essencial para a construção da solução do labirinto na interface.

Quando a fila de prioridade se esgota sem encontrar a saída, retornamos "None" para indicar o término da busca.

3. Elaboração de interface.

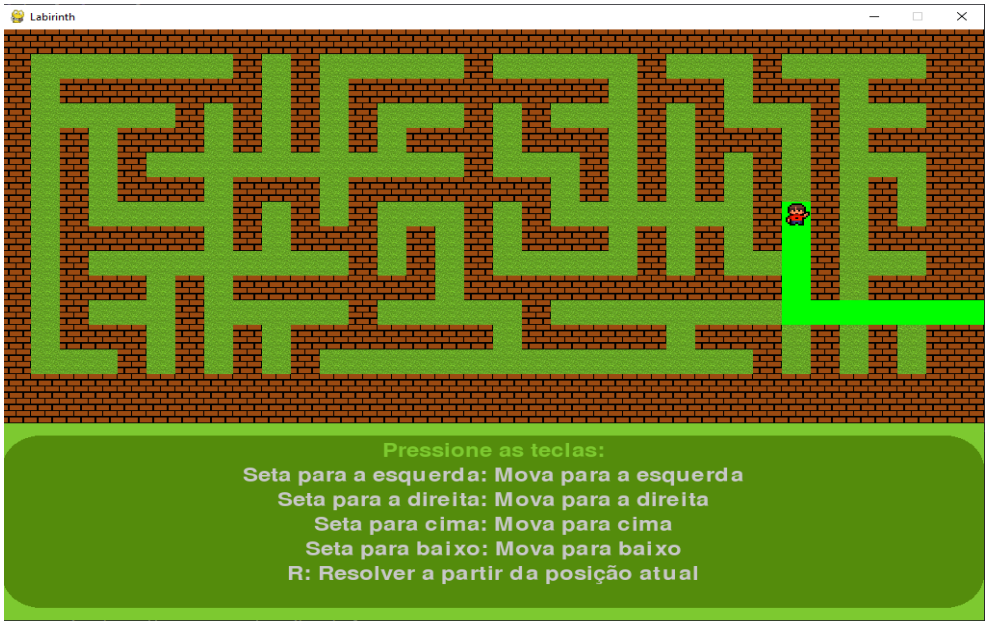
Optamos por utilizar a biblioteca Pygame para desenvolver a interface do nosso projeto devido à sua capacidade de lidar com eventos de forma prática. Essa facilidade foi útil, pois nos permitiu incorporar diretamente no código condições baseadas nos eventos capturados após o usuário pressionar as teclas, no nosso

caso, as teclas de direção (cima, baixo, esquerda e direita) e o R. Isso facilitou o controle dos movimentos do Codibentinho durante a exploração do labirinto.

Além disso, o Pygame simplificou a tarefa de adicionar elementos visuais lúdicos, como imagens, aos blocos do labirinto, o que deixou o visual muito mais agradável com poucas linhas de código.

Consideramos também a quantidade de informação presente na internet sobre a construção de labirintos utilizando essa biblioteca.

Atualmente, essa é a nossa interface. O bloco amarelo simboliza a saída, o bonequinho é o Codibentinho. Quando a tecla R é pressionada, o caminho gerado pelo algoritmo de busca é desenhado em verde neon.



4. As instruções de instalação e uso do aplicativo desenvolvido.

Para executar o projeto, siga os passos abaixo:

1. Clone o repositório para a sua máquina local. (https://github.com/reb27/codibentinho)
2. Certifique-se de ter uma versão do Python **superior** à 3, pois estamos utilizando alguns recursos das versões mais atuais.
3. Tenha o pip instalado, pois será usado para a instalação da biblioteca **Pygame**.
4. Execute o comando pip install pygame para finalizar o passo 3.
5. Por fim, execute o comando python main.py para visualizar a interface do aplicativo.
6. As instruções estão no Menu presente na interface. As teclas de seta são responsáveis pela movimentação no labirinto, enquanto a R acha o caminho até a saída.

1. Referências Bibliográficas

[1] https://thepythoncode.com/article/build-a-maze-game-in-python
[2]https://pythonspot.com/maze-in-pygame/
[3]https://iq.opengenus.org/maze-in-python/