UFAM - Universidade Federal do Amazonas
IComp - Instituto de Computação
Engenharia de Software
AED 2 - Algoritmo e Estrutura de Dados 2
Prof. Rafael Giusti

Discentes:
Johnatas Philipe Rodrigues da Silva
Pedro Henrique Souza dos Santos

Trabalho Prático 2 - Relatório de Atividade

Problema 3: Dogo

Descrição do problema:

O problema envolve a tarefa de encontrar o menor caminho entre um ponto e outro. Especificamente, Dogo é um cãozinho que precisa atravessar um salão até o seu pote de ração, porém, o salão contém obstáculos que Dogo precisa evitar. Dogo sempre começa no canto superior esquerdo do salão, ou seja, na posição (0,0), e seu pote de ração está sempre no canto inferior direito, ou seja, na posição (M-1, N-1), sendo M e N linha e coluna respectivamente. Os obstáculos geram zonas proibidas em que Dogo deve evitar, por exemplo, se há um obstáculo na posição (1,4), todas as células ao redor dessa posição são zonas proibidas. Há duas possibilidades para a solução desse problema, ou existe um caminho até o pote de ração, ou não há nenhum caminho possível

Procedimento de busca:

O procedimento de busca utilizado é o algoritmo A* (A estrela). Esse é um algoritmo de busca em grafos ponderados, que combina técnicas de Busca em Largura (BFS) e Busca em Profundidade (DFS), utilizado para encontrar o caminho mais curto entre um ponto inicial e um ponto final.

O algoritmo é composto pelos seguintes procedimentos:

- Cria um conjunto de nós abertos e nós fechados.
- Define estimativas de custo para os nós, como o custo real e estimativa heurística.
- A exploração inicia a partir do primeiro nó.
- Do conjunto de nós abertos, seleciona iterativamente o nó com o menor custo total.
- É feita uma avaliação dos vizinhos do nó analisado para encontrar o próximo nó promissor.
- É feita uma atualização dos custos caso um caminho mais curto seja encontrado.

- Quando o nó destino é encontrado, a reconstrução do caminho é feita a partir dos predecessores.
- Como último passo, o caminho encontrado é retornado.

Interface

No início do projeto, cogitou-se utilizar a biblioteca *pygame* para construir a interface (Figura 1). Obtivemos sucesso em representar o tabuleiro, o objeto móvel (Dogo) e sinalizar os obstáculos, mas tivemos dificuldades de manipular a interface corretamente.

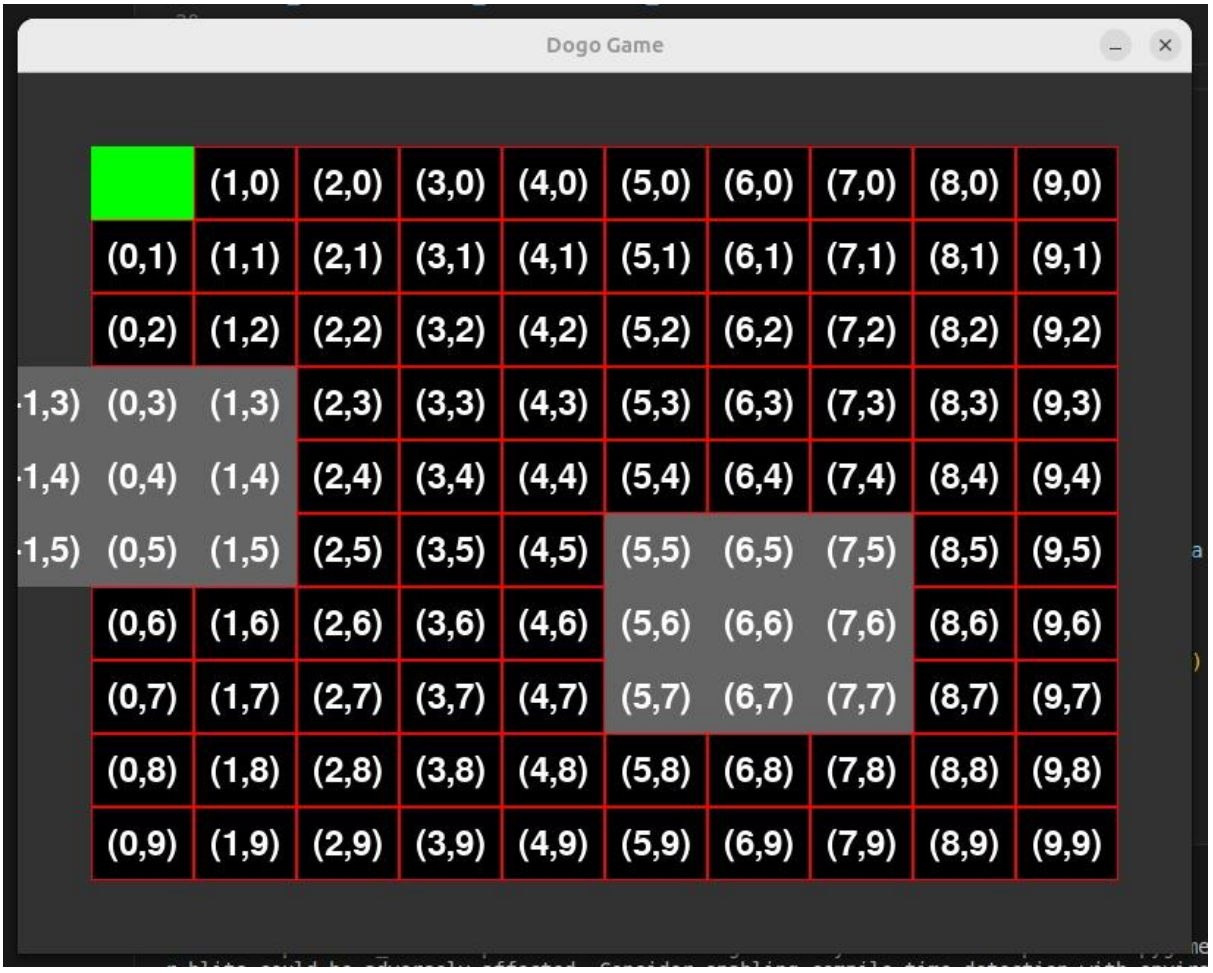

Figura 1. Representação do salão, dos obstáculos e da zona proibida com a biblioteca *pygame*. Note que a zona proibida se extendia para além dos limites do tabuleiro.

Devido aos problemas de construção da inteface gráfica, optamos por mostrar o tabuleiro no próprio terminal, realizando uma impressão simples das transições de estados em que o salão assumia de acordo com a interação do usuário (Figura 2 - 4).

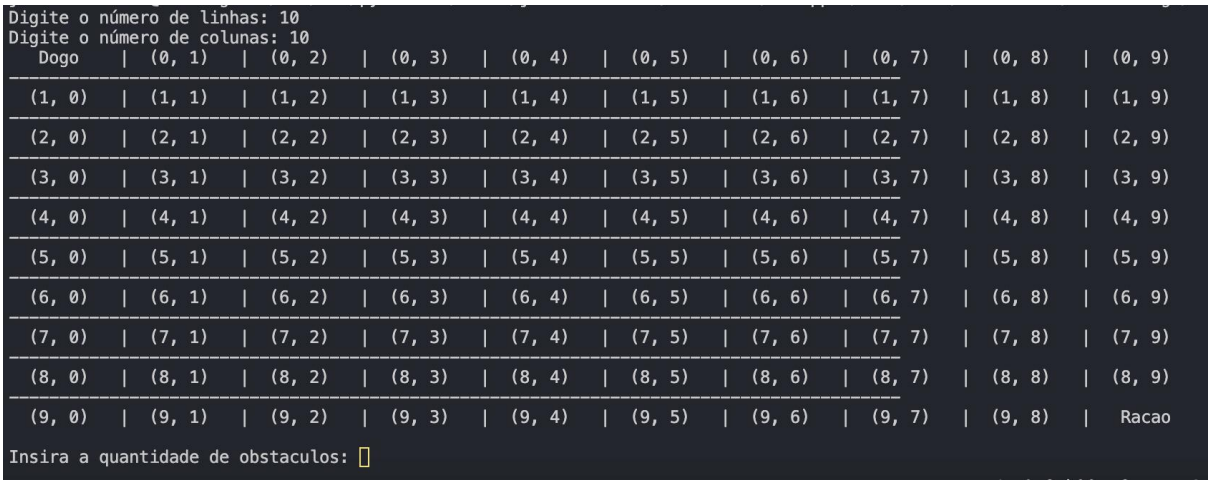

Figura 2. Representação do salão, estado inicial, no terminal padrão.

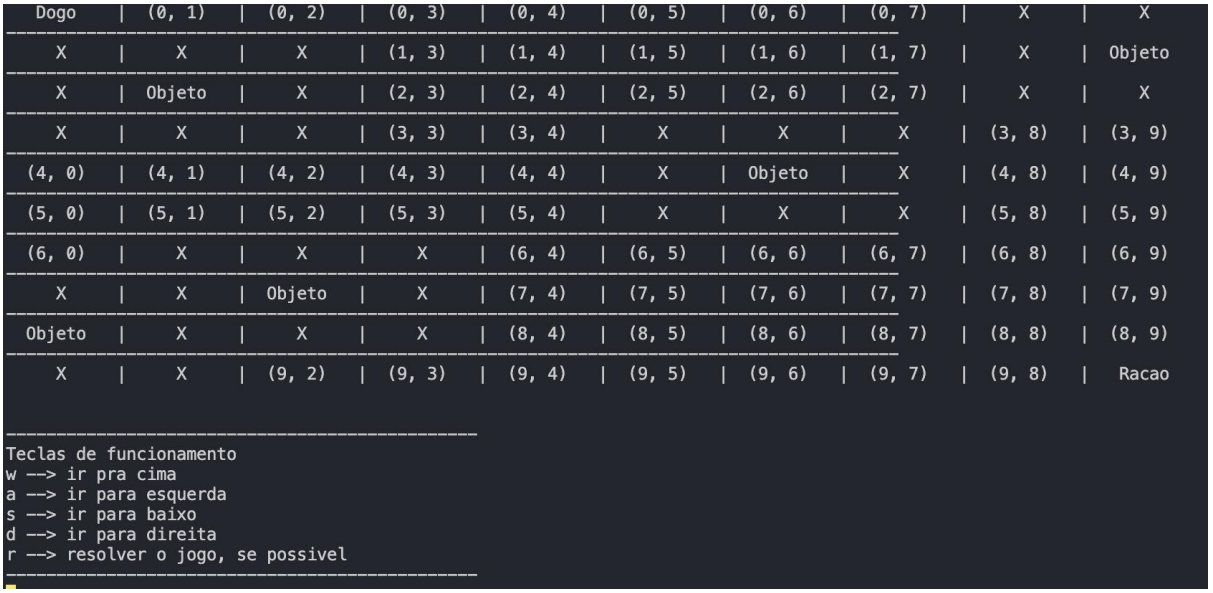

Figura 3. Representação do salão com os obstáculos inseridos e o menu de interação.

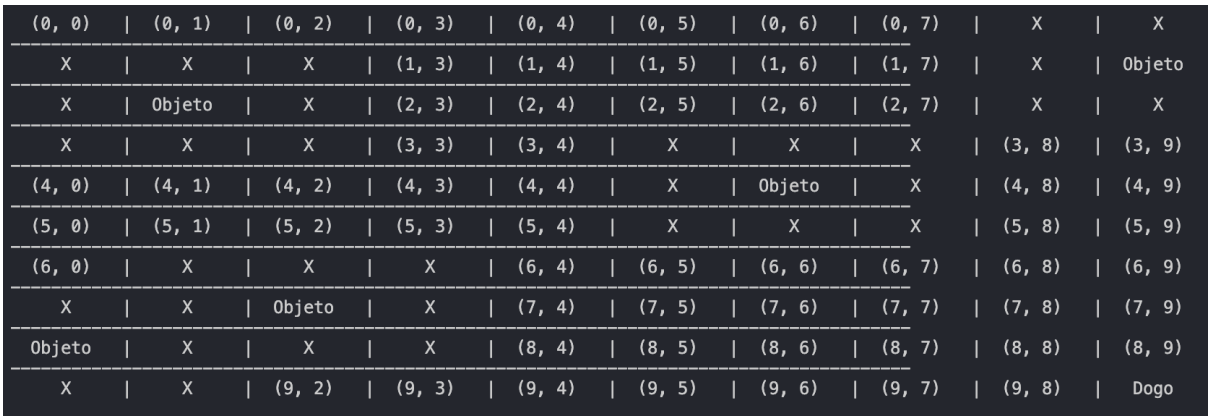

Figura 4. Representação do salão no estado final. Note que foi possível alcançar o destino, Dogo se encontra no canto inferior direito.

Usabilidade e Conclusão

Devido a interação ocorrer diretamente no terminal, a execução ocorre a partir de comandos em linha ou por meio do uso de uma IDE. Nesse caso, utilizamos o VS Code. O código foi escrito em Python, dessa forma não geramos um arquivo executável. Uma versão mais elaborada de interface e conversão do arquivo de código em um arquivo executável é o próximo objetivo, nesse primeiro momento, queríamos demonstrar a contrução da solução com a busca informada.

Esse projeto demonstrou uma das aplicabilidades de fila de prioridade, operações de busca em grafos valorados e uso de heaps para resolver um problema de busca por rota otimizada. Além da aplicação em resolução de jogos de quebra-cabeça, essas técnicas são usadas constantemente em mais diversos problemas do quotidiano, desde problemas de transporte e logísticas quanto em problemas mais complexos como o processamento de linguagem natural, muito utilizada por aplicações que usam IA.