MINISTÉRIO DA EDUCAÇÃO

Universidade Federal do Amazonas

Instituto de Computação

Trabalho De AED II

Alexandre Gabriel Gadelha de Lima, Martinho Prata e Samuel Monteiro Gomes

Manaus – AM

2024

Descrição do problema

Dogo é um cachorrinho que quer atravessar um salão retangular, onde há objetos assustadores (aspiradores, vassouras, etc.) que ele deve evitar. O objetivo é encontrar o caminho mais curto até seu pote de ração, localizado no canto inferior direito do salão. Dogo não pode se aproximar de menos de dois azulejos de qualquer objeto assustador.

Resumo do Problema:

- **Estado Inicial:** Dogo está no canto superior esquerdo do salão (0, 0).
- **Estado Final:** O pote de ração está no canto inferior direito do salão (M-1, N-1).
- **Objetos Assustadores:** Representados por 'x'. Dogo não pode pisar nem se aproximar de dois azulejos desses objetos.
- **Objetivo:** Encontrar o caminho mais curto de Dogo até o pote de ração, respeitando as restrições de distância dos objetos assustadores.

Representação do Estado:

- **Matriz de Acessibilidade:** Uma matriz de dimensões M × N onde True indica posições acessíveis e False indica posições com objetos ou muito próximas a eles.

Requisitos do Programa:

1. **Interface Gráfica:** Mostrar um salão de M x N, com Dogo no canto superior esquerdo e a comida no canto inferior direito.
2. **Interatividade:** Permitir adicionar objetos assustadores e mostrar suas coordenadas.
3. **Redimensionamento do Salão:** Ajustar o tamanho do salão conforme necessário.
4. **Movimentação Manual:** Permitir mover Dogo manualmente sem passar por paredes ou objetos.
5. **Busca Automática:** Implementar a estratégia de busca com fila de prioridades para encontrar o caminho mais curto. Exibir o caminho ou informar se não há caminho possível.
6. **Funções Extras:** Repor Dogo à origem, remover objetos do salão, e limpar o estado do salão.

Estrutura Geral

1. Classe Estado:
    - Representa uma posição no salão com coordenadas (x, y).
    - Implementa métodos para comparar estados, calcular o hash, e mover-se para outro estado.
2. Função reconstruir_caminho:
    - Reconstrói o caminho do estado objetivo de volta ao estado inicial, a partir de um dicionário que rastreia o caminho percorrido.
3. Classe FilaPrioridade:
    - Implementa uma fila de prioridade usando um heap binário.
    - Adiciona e remove itens com base em suas prioridades.
4. Função a_star:
    - Executa o algoritmo A* para encontrar o caminho mais curto entre um estado inicial e um estado objetivo.
    - Usa uma fila de prioridade para explorar os estados, mantém um registro dos custos e do caminho percorrido.

Como Funciona

1. Inicialização:
    - Cria o estado inicial e a fila de prioridade.
    - Adiciona o estado inicial à fila.
2. Exploração:
    - Enquanto houver estados na fila, remove o estado com a menor prioridade.
    - Se o estado removido é o objetivo, reconstrua e retorne o caminho.
3. Expansão:
    - Para cada estado, explora os estados vizinhos.
    - Calcula o custo de mover para cada vizinho e atualiza a fila de prioridade.
4. Finalização:
    - Se encontrar o estado objetivo, retorna o caminho.
    - Se a fila de prioridade estiver vazia e o objetivo não for encontrado, retorna None.

Essencialmente, o algoritmo A* utiliza uma fila de prioridade para explorar estados com base no custo acumulado e na estimativa de custo até o objetivo, garantindo que o caminho mais curto seja encontrado.

Nina's Adventure

Nina's Adventure é um jogo interativo que utiliza o algoritmo de busca heurística A*
para encontrar o caminho mais curto entre um ponto inicial (dogo) e um objetivo (osso),
evitando obstáculos em um grid. O jogo possui uma interface gráfica criada com a biblioteca Pygame.

Compilação do programa :

bash

```
pip install pygame

python main.py
```

Características :

- Algoritmo A*: Implementação do algoritmo de busca A* para encontrar o caminho mais curto.
- Interface Gráfica: Desenvolvida com Pygame, com suporte para interação via teclado e mouse.
- Modos de Jogo:
  - Manual: Controle o dogo com as setas do teclado.
  - Automático: O algoritmo A* encontra o caminho até o osso.
- Personalização: Escolha manualmente as posições dos obstáculos ou gere-os automaticamente.
- Feedback Visual: Mensagens de status e feedback durante o jogo (ex: "Game Over", "Good Job!").

Utilização :

1. Configuração Inicial:
   - Ao iniciar, uma janela perguntará se você deseja definir as dimensões do grid.
   - Pressione SIM ou NÃO.

2. Definição de Dimensões(se aplicável):
   - Insira o número de linhas e colunas desejadas e pressione Enter
3. Escolha de Obstáculos:
   - Clique em "Escolher obstáculos" para definir manualmente ou "Gerar obstáculos" para que o programa os gere aleatoriamente.
4. Definição de Obstáculos (se aplicável):
   - Siga o tutorial interativo para posicionar os obstáculos. Clique nas células para colocar ou remover obstáculos. Pressione Enter para salvar.
5. Seleção do Modo de Jogo:
   - Clique em "Modo Manual" ou "Modo Automático".
6. Modo Manual:
   - Use as setas do teclado para guiar o dogo até o osso, evitando obstáculos.