Documentação Pathfinder

Autor: Tomás Lacerda Muniz – 2021088116

Introdução

Essa documentação é referente ao trabalho de Inteligência Artificial, cujo desafio consistiu em implementar cinco algoritmos de busca – A*, UCS, IDS, BFS e Greedy. Esses algoritmos são utilizados para encontrar o melhor caminho entre dois pontos em um mapa, considerando diferentes tipos de terrenos e custos associados.

Estrutura de Implementação

O código está organizado em múltiplos arquivos para melhor modularização:

- **`pathfinder.py`** (arquivo principal): Contém a função `main`, onde os algoritmos são chamados e integrados.
- Arquivos separados para cada algoritmo: Cada um carrega o nome do algoritmo correspondente.

Representação do Mapa

O mapa é representado por uma matriz bidimensional (`map_matrix`), onde cada célula corresponde a um tipo de terreno com custos definidos pelo dicionário `terrain_costs`:

- `.` → Grama (custo 1.0)
- `;` → Grama alta (custo 1.5)
- `+` → Água (custo 2.5)
- `x` → Fogo (custo 6.0)
- `@` → Parede (intransponível)

Os algoritmos operam entre uma posição inicial (`start`) e final (`end`), ambas definidas como tuplas `(x, y)`.

Fluxo do Programa

1. **Carregamento do Mapa:** A função `load_map` carrega a matriz representando o mapa.
2. **Chamada do Algoritmo:** O algoritmo correspondente é chamado com os parâmetros fornecidos.

Principais Diferenças entre os Algoritmos

Todos os algoritmos seguem um padrão de implementação:

- **Construtor:**

```
class Algoritmo:
    def __init__(self, map_matrix, start, end, terrain_costs):
        # Inicialização
```

- **Validação de Movimento:**

```
    def is_valid(self, x, y):
        # Verifica se a posição é válida
```

- **Busca:** Implementação específica do algoritmo.

Tabela Comparativa dos Algoritmos

Algoritmo	Estratégia	Heurística	Custo Considerado	Melhor para
BFS	Busca em largura	Nenhuma	Não considera custo	Explorar caminhos curtos em nós uniformes
UCS	Busca de custo uniforme	Nenhuma	Custo acumulado do terreno	Encontrar o caminho mais barato (custo real)
IDS	Busca em profundidade iterativa	Nenhuma	Não considera custo	Reduzir consumo de memória em árvores profundas
Greedy	Busca gulosa	Distância Manhattan	Somente a heurística	Caminhos rápidos, mas não necessariamente ótimos
A*	Busca A*	Distância Manhattan	Custo acumulado + heurística	Encontrar o caminho mais barato considerando heurística

---

Análise dos Algoritmos

Qualidade dos Caminhos

- **A*** e **UCS:** Sempre encontram o caminho ótimo.
- **IDS** e **BFS:** Sempre encontram uma solução, mas não necessariamente a mais eficiente.
- **Greedy:** Pode falhar em encontrar uma solução, pois prioriza caminhos rápidos baseando-se exclusivamente na heurística.

Tempo de Execução

- **A***, **UCS** e **Greedy** são mais rápidos que **IDS** e **BFS**.
- **A*** é geralmente mais eficiente que **UCS** devido ao uso de heurística.

- Nos testes feitos por mim, localmente, os algorítimos de **UCS**, **Greedy** e **A*** executam nos mapas propostos instantâneamente, enquanto o **BFS** e o **IDS** gastam mais do que um minuto para executar.

Heurística de Manhattan

É admissível, pois nunca superestima o custo real para atingir o objetivo. Isso garante que os algoritmos **A*** e **Greedy**, que utilizam essa heurística, sejam eficazes no contexto.

Conclusão

Para um problema semelhante ao apresentado a nós no enunciado, o A* se apresenta superior aos demais por sua velocidade, e garantia de chegar em um resultado ótimo.