Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

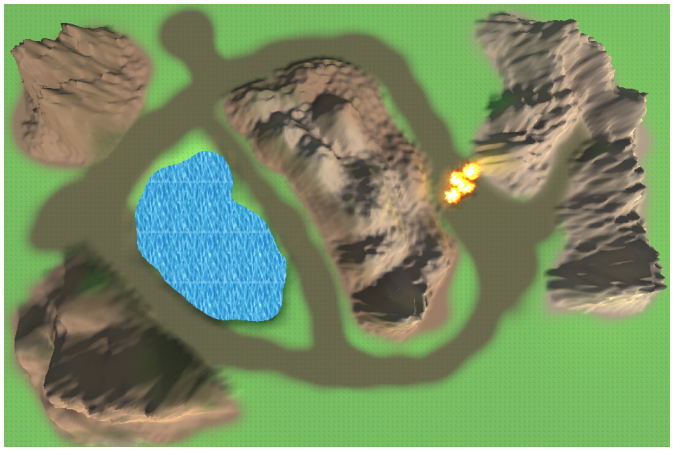
Introdução à Inteligência Artificial
Professor: Luiz Chaimowicz
Monitor: Marco Tulio de Pinho

Trabalho Prático I: Algoritmo de Busca para Menor Caminho

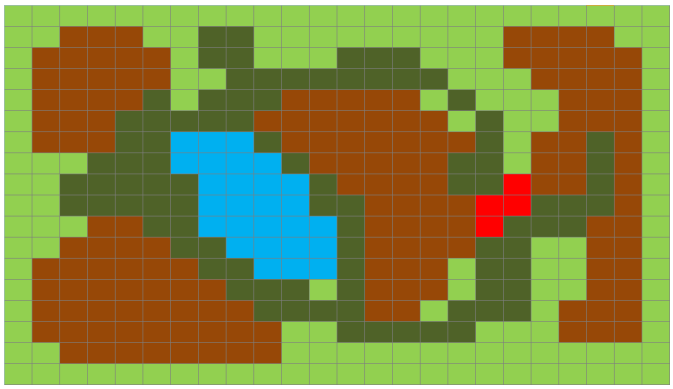
Valor: 15 pontos
Data de entrega: a ser definida.

1 Definição do Problema

Este trabalho tem como objetivo praticar os conceitos e algoritmos de busca em espaço de estados. Além disso, proporcionará uma forma de comparar diferentes tipos de métodos. Para tal, será proposto a resolução do Path-Finding. Este é o problema de encontrar o menor caminho entre dois pontos. Sua aplicação é essencial para diversas áreas. Em jogos, os métodos são usados para guiar a movimentação de NPCs. O mundo do jogo (podendo ser 2D ou 3D) é discretizado em áreas geométricas. A figura 1 mostra um exemplo desse processo. Com esse processo, o tamanho do espaço de estados é reduzido. As possíveis ações do agente são caminhar entre essas áreas executando um movimento entre quatro direções (cima, baixo, direita e esquerda). Quando uma solução é encontrada, o agente apenas executa as ações até alcançar o alvo. Uma questão é que, muitas das vezes, alguns caminhos são preferíveis a outros. Um exemplo é entre escolher o caminho pegando fogo mas que é direto ao objetivo ou decidir contorná-lo e gastando mais passos. Uma forma de resolver esse problema é atribuir para cada tipo de terreno um custo diferente. Dessa forma, você deverá desenvolver cinco métodos de busca: **Busca em Largura (BFS)**, **Aprofundamento Iterativo (IDS)**, **Busca de Custo Uniforme (UCS)**, **Gulosa** e **A***. O seu programa receberá uma descrição do mapa discretizado e deverá retornar um caminho de um ponto inicial ao final (considerando os diferentes custos do terreno). Além disso, o seu programa deve retornar o custo total do caminho encontrado.



(a) Mundo 3D do jogo



(b) O mapa discretizado

Figura 1: Figura exemplificando o processo de discretização de um mapa. Na figura 1a, é uma representação do mundo 3D com diferentes tipos de terreno. A figura 1b mostra uma versão discretizada, com cores para marcar os diferentes tipos de terrenos.

2 Mapa

O mapa que seu programa receberá será a versão discretizada de um mundo. Essa é modelada por uma matriz bidimensional. Cada posição dessa representa uma área do mapa original do jogo e terá um símbolo que identifica o tipo de terreno. Os símbolos e seus respectivos custos estão descritos na tabela 1 a seguir.

Terreno	Símbolo	Custo
Grama	.	1.0
Grama Alta	;	1.5
Água	+	2.5
Fogo	x	6.0
Parede	@	∞

Tabela 1: Símbolos e custos associados a cada terreno

Note que os custos se referem ao custo real do caminho, não à heurística que será utilizada pelos métodos **Guloso** e **A***. O simbolo @ é um caractere especial que significa uma parede ou terreno intransponível. O agente não pode, em momento nenhum, estar em uma posição com esse tipo de terreno.

3 Ações

O agente pode movimentar em quatro direções (cima, baixo, direita e esquerda). Os movimentos para fora dos limites do mapa ou para terrenos intransponíveis são inválidos.

4 Métodos

Para o trabalho, você deve implementar cinco métodos de busca. Três são sem informação: **Busca em Largura (BFS)**, **Aprofundamento Iterativo (IDS)** e **Busca de Custo Uniforme (UCS)**. Os outros são o busca **Gulosa** e **A***. Para ambos, é necessário a definição de uma função heurística. Você deve utilizar uma função válida para o problema.

5 Entrada do programa

O seu programa receberá como argumento uma string que representa o nome do arquivo texto onde o mapa e suas informações serão descritas. Além disso, uma outra string que identifica qual método que deve ser utilizado. Por fim, 4 inteiros que representam as coordenadas (x_i, y_i) do ponto inicial e (x_f, y_f) do objetivo. Um exemplo de um programa compilado em C/C++ nomeado como *pathfinder*, o seguinte comando deve ser usado para rodá-lo.

./pathfinder [caminho_para_arquivo_mapa] [identificador_metodo] x_i y_i x_f y_f

Em que [identificador_metodo] pode ser BFS, IDS, UCS, $Greedy$ ou $Astar$ representando os métodos: **Busca em Largura**, **Aprofundamento Iterativo**, **Busca de Custo Uniforme**, **Guloso** e **A***, respectivamente. [caminho_para_arquivo_mapa] será a string com o caminho para o arquivo texto.

O arquivo texto apresentará todas as informações referentes ao mapa. A primeira linha do arquivo possui dois inteiros W e H, representando o número de colunas e linhas da matriz respectivamente. Em seguida, as próximas H linhas terão W caracteres representando o tipo de terreno naquela posição. Um exemplo de entrada é descrito a seguir.

```
5 3
@ + + + @
@ . x . @
@ . ; . @
```

6 Saída do Programa

Seu programa deve computar um caminho entre os pontos dados na entrada. O custo do caminho deve considerar os custos de cada terreno como dado na tabela 1. Ao final da busca, deve-se imprimir na tela duas coisas. Primeiramente é o custo real total do caminho. Em seguida, a sequencia de pontos que se deve percorrer para realizar o caminho. Um exemplo de saída considerando o mapa exemplo descrito na seção anterior com ponto inicial (1, 1) e com ponto final (3, 1).

4.5 (1, 1) (1, 2) (2, 2) (3, 2) (3, 1)

7 Entrega

Você deverá entregar um arquivo **ZIP** no Moodle da disciplina. O arquivo **ZIP** deve conter o código-fonte da sua implementação **(C, C++ ou python)**. Além disso, deve contem uma documentação em formato **PDF**. Essa deve descrever de forma sucinta as decisões de implementação e análises. Mais especificamente, deve-se incluir:

- Apresentação das estruturas usadas e da modelagem dos componentes da busca (estado, função sucessora, etc);
- Breve descrição das principais diferenças entre os algoritmos.
- Especificação das heurísticas utilizadas. Elas são admissíveis? Por quê?
- Análise quantitativa comparando os algoritmos com relação ao número de estados expandidos e tempo de execução à medida que o ponto final se distancia do ponto inicial. **Apresente tabelas e/ou gráficos comparativos**.
- Discussão dos resultados obtidos.

Por fim, deve-se ter um **README** descrevendo como rodar (e compilar caso a linguagem necessite) o seu programa. Assim como uma lista de bibliotecas utilizadas.

8 Considerações Finais

- Sinta-se livre para discutir o trabalho com seus colegas. Entretanto, o compatilhamento de códigos ou texto é plágio e será devidamente punido. Seu trabalho deve ser de sua autoria.
- Use o fórum de discussão *Dúvidas/Discussão* do Moodle em caso de dúvidas. Caso prefira, pode enviar um email para o monitor da disciplina (marcotuliopin@ufmg.br).
- Comece o trabalho o mais cedo possível!
- **Bom trabalho!**