Tarefa 08
Implementação do A*

Autores: **Lucas e Luis**

Projeto de busca de caminhos utilizando o algoritmo A* com heurística Manhattan.

ÍNDICE

INTRODUÇÃO

- **Título**: "Introdução", alinhado à esquerda e em negrito.
- **Subseções**:
    - **Objetivo do Projeto**: Texto introdutório explicando o propósito do algoritmo A*.

      Exemplo: "Este projeto implementa o algoritmo A* para busca de caminhos eficientes em mapas representados por matrizes."
    - **Descrição Geral**: Texto detalhado sobre o problema resolvido e como o A* funciona.
    - **Contexto de Uso**: Tópicos com possíveis aplicações, como:
        - Navegação em jogos.
        - Robótica.
        - Sistemas de GP

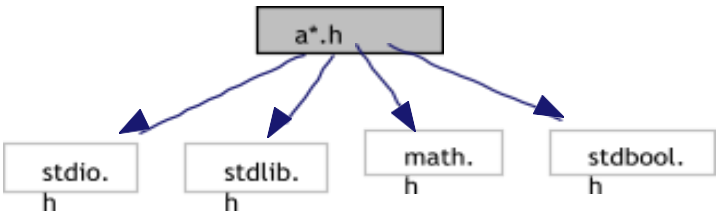
Estruturas de Dados

1.1 Estrutura `TNoCaminho`

A estrutura `TNoCaminho` representa um nó no caminho do algoritmo A*. Ela contém os seguintes atributos:

- `x` e `y`: Coordenadas do nó.
- `custo_g`: Custo acumulado do início até este nó.
- `custo_h`: Heurística estimada até o objetivo.
- `custo_f`: Soma de `custo_g` e `custo_h`.
- `pai`: Ponteiro para o nó anterior no caminho.

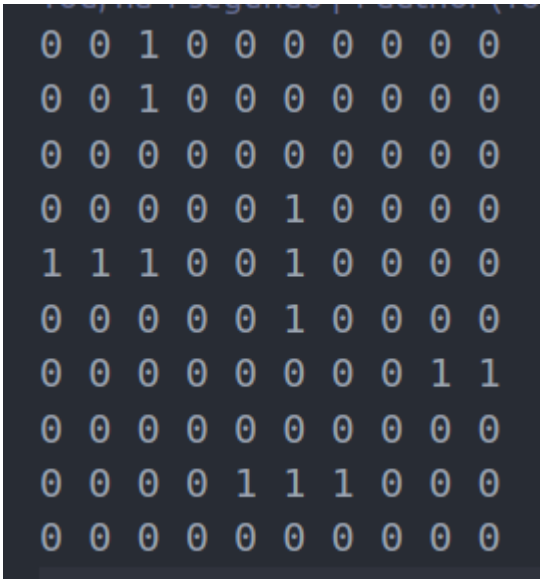

```c
typedef struct NoCaminho {
    int x;         /**< Coordenada x do nó. */
    int y;         /**< Coordenada y do nó. */
    int custo_g;   /**< Custo acumulado do início até este nó. */
    int custo_h;   /**< Heurística estimada até o objetivo. */
    int custo_f;   /**< Custo total (g + h). */
    struct NoCaminho* pai; /**< Ponteiro para o nó pai (para reconstrução do caminho). */
} TNoCaminho;
```



1.2 Representação do Mapa

O mapa é representado por uma matriz de inteiros onde cada célula pode ter os seguintes valores:

- 0: Célula transitável.
- 1: Obstáculo.
- 3: Ponto inicial.
- 4: Ponto objetivo.



Funções e Métodos

Função: `heuristica`

- **Descrição: Calcula a distância Manhattan entre dois pontos.**

Assinatura da Função:
int heuristica(int x1, int y1, int x2, int y2);

- **Exemplo de Execução:**
  - **Entrada: `(x1 = 2, y1 = 3), (x2 = 5, y2 = 1)`**
  - **Saída: `5`**
  - **Fórmula: `|2 - 5| + |3 - 1| = 3 + 2 = 5`**
- **Figura Ilustrativa: Inclua uma tabela com as coordenadas de entrada e o cálculo.**

Função: `ehValido`

- **Descrição: Verifica se uma célula do mapa é válida para movimentação.**

Assinatura da Função:
bool ehValido(int x, int y, int mapa[LINHAS][COLUNAS], bool lista_fechada[LINHAS][COLUNAS]);

-
- **Condições de Validade:**
  - **`x` e `y` devem estar dentro dos limites do mapa.**
  - **A célula não pode ser um obstáculo (`mapa[x][y] != 1`).**
  - **A célula não pode ter sido explorada anteriormente (`!lista_fechada[x][y]`).**
- **Exemplo de Execução:**
  - **Entrada: `x = 3, y = 2, mapa[3][2] = 0, lista_fechada[3][2] = false`**
  - **Saída: `true`**
- **Figura Ilustrativa: Inclua um mapa com uma célula destacada que atenda às condições.**

Função: `estaListaAberta`

- **Descrição: Verifica se um nó já está na lista aberta.**

Assinatura da Função:
bool estaListaAberta(TNoCaminho* lista[], int contagem, int x, int y);

-
- **Funcionamento:**
  - **Itera sobre os nós da lista aberta.**
  - **Retorna `true` se encontrar um nó com as coordenadas `(x, y)`.**
- **Exemplo de Execução:**
  - **Entrada: Lista aberta contendo nós nas posições `(1, 1)`, `(3, 2)`, `(5, 6)`. Coordenada a verificar: `(3, 2)`.**
  - **Saída: `true`**
- **Figura Ilustrativa: Inclua uma representação da lista aberta e destaque o nó verificado.**

Função: `criarNo`

- **Descrição: Cria um novo nó da estrutura `TNoCaminho` para o algoritmo A*.**

Assinatura da Função:
TNoCaminho* criarNo(int x, int y, int custo_g, int custo_h, TNoCaminho* pai);

-
- **Exemplo de Execução:**
  - **Entrada:**
    - **`x = 3, y = 2`**
    - **`custo_g = 10, custo_h = 5`**
    - **`pai = NULL`**
  - **Saída:**
    - **Nó criado com `custo_f = 15`.**
- **Figura Ilustrativa: Inclua um diagrama do nó com seus atributos e conexão ao nó pai (se houver).**

Função: `reconstruirCaminho`

- **Descrição: Reconstrói o caminho encontrado pelo algoritmo A* no mapa.**

Assinatura da Função:
void reconstruirCaminho(TNoCaminho* atual, int mapa[LINHAS][COLUNAS]);

-
- **Funcionamento:**
  - **Marca o caminho no mapa, traçando uma linha de células (2) até o ponto inicial.**
- **Exemplo de Execução:**
  - **Entrada: Nó final em `(5, 6)` com pai `(4, 6)` e mapa original.**
  - **Saída: Mapa com o caminho traçado.**
- **Figura Ilustrativa: Mostre o mapa antes e depois do traçado.**

Função: `liberarListaAberta`

- **Descrição: Libera a memória alocada para a lista aberta.**

Assinatura da Função:
void liberarListaAberta(TNoCaminho* lista[], int contagem);

-
- **Funcionamento:**
  - **Itera sobre a lista aberta e libera cada nó.**
- **Exemplo de Execução:**
  - **Entrada: Lista aberta com três nós.**
  - **Saída: Todos os nós são liberados da memória.**
- **Figura Ilustrativa: Mostre uma representação visual da lista antes e depois da liberação.**

Função: `buscaAestrela`

- **Descrição: Implementa o algoritmo A*.**

Assinatura da Função:
void buscaAestrela(int mapa[LINHAS][COLUNAS], int inicio_x, int inicio_y, int objetivo_x, int objetivo_y);

-
- **Fluxo Geral:**
  - **Inicializa o nó inicial.**
  - **Expande nós vizinhos com menor custo.**
  - **Reconstrói o caminho quando o objetivo é alcançado.**
- **Exemplo de Execução:**
  - **Entrada:**
    - **`mapa`: Matriz com obstáculos e células transitáveis.**
    - **Ponto inicial: `(0, 0)`.**
    - **Ponto objetivo: `(5, 6)`.**
  - **Saída:**
    - **Caminho encontrado e traçado no mapa.**
- **Figura Ilustrativa: Inclua um fluxograma do funcionamento do algoritmo e o mapa antes/depois.**

Função: `imprimirMapa`

- **Descrição: Imprime o mapa no console.**

Assinatura da Função:
void imprimirMapa(int mapa[LINHAS][COLUNAS]);

-
- **Exemplo de Execução:**
  - **Entrada: Mapa com células marcadas.**
  - **Saída: Impressão no console do mapa formatado.**
- **Figura Ilustrativa: Mostre como o mapa aparece no console.**

Função: `corrigirMapa`

- **Descrição: Marca os pontos inicial e final no mapa.**

Assinatura da Função:
void corrigirMapa(int mapa[LINHAS][COLUNAS], int inicio_x, int inicio_y, int objetivo_x, int objetivo_y);

-
- **Exemplo de Execução:**
  - **Entrada:**
    - **Ponto inicial: `(0, 0)`.**
    - **Ponto objetivo: `(5, 6)`.**
  - **Saída: Mapa atualizado com marcadores `3` e `4`.**
- **Figura Ilustrativa: Mostre o mapa antes e depois da marcação.**

Função: `lerMapa`

- **Descrição: Lê o mapa de um arquivo de texto.**

Assinatura da Função:
void lerMapa(const char* arquivo, int mapa[LINHAS][COLUNAS]);

-
- **Funcionamento:**
  - **Carrega valores de um arquivo para a matriz `mapa`.**
- **Exemplo de Execução:**
  - **Entrada: Arquivo `mapa.txt`.**
  - **Saída: Matriz preenchida.**
- **Figura Ilustrativa: Mostre um exemplo do arquivo de entrada e a matriz gerada.**

Fluxo do Algoritmo

Descrição Geral
O algoritmo A* é uma técnica de busca informada que utiliza duas listas (aberta e fechada) e uma heurística para encontrar o caminho mais curto entre dois pontos em um grafo ou mapa. Este fluxo detalha os passos realizados pelo algoritmo para atingir seu objetivo.

Fluxograma

1. Inicializar: Configura o mapa e adiciona o nó inicial à lista aberta.
2. Selecionar Nó: Encontra o nó com o menor custo total (`custo_f`) na lista aberta.
3. Verificar Objetivo: Se o nó selecionado for o objetivo, reconstrua o caminho.
4. Expandir Vizinhos: Adicione vizinhos válidos à lista aberta, calculando seus custos.
5. Atualizar Listas: Mova o nó atual para a lista fechada.
6. Repetir: Volte ao passo 2 até encontrar o objetivo ou esgotar a lista aberta.

Explicação Passo a Passo

1. Inicializar: O nó inicial é adicionado à lista aberta, com custo `g = 0` e heurística calculada.
2. Selecionar Nó: Percorra a lista aberta para encontrar o nó com menor custo total.
3. Expandir Vizinhos: Para cada vizinho:
    - Verifique se está dentro dos limites do mapa.
    - Certifique-se de que não seja um obstáculo ou já explorado.
    - Calcule o custo acumulado (`g`) e a heurística (`h`).
4. Atualizar Listas: Mova o nó processado para a lista fechada e continue.

Exemplos de Saída

Mapa Inicial

0 1 0 0 0

0 1 0 1 0

0 0 0 1 0

1 1 0 0 0

0 0 0 1 4

- `0`: Célula transitável.
- `1`: Obstáculo.
- `4`: Objetivo.

Execução

1. Nó inicial `(0, 0)` adicionado à lista aberta.
2. Expansão do nó `(0, 0)` para `(1, 0)` e outros vizinhos válidos.
3. Lista aberta reduzida até atingir o objetivo `(4, 4)`.

Mapa Final

3 1 0 0 0

2 1 0 1 0

2 2 2 1 0

1 1 2 0 0

0 0 2 1 4

- 3: Ponto inicial.
- 2: Caminho traçado.
- 4: Objetivo.

Referências e Apêndices

Referências

- Documentação oficial do C.
- Tutoriais ou artigos sobre o algoritmo A*.

Apêndice

- Definições de constantes (`#define`).
- Estruturas de dados (`TNoCaminho`).
- Outras funções utilitárias

Exemplo:

```
#define LINHAS 10

#define COLUNAS 10

typedef struct NoCaminho {

   int x, y;

   int custo_g, custo_h, custo_f;

   struct NoCaminho* pai;

} TNoCaminho;
```
