Projeto 2 - Algoritmos de Busca
(07/07/2025)

Disciplina de Inteligência Artificial
Curso de Ciência da Computação – UNIOESTE
Grupos de 2 alunos

Na mitologia grega, Minotauro era uma criatura metade homem e metade touro que vivia no labirinto da ilha de Creta (Figura 1). Como forma de tributo, Creta exigia que cidadãos de Atenas fossem enviados periodicamente como sacrifício para o monstro. Revoltado com a situação, Teseu, príncipe de Atenas, se voluntaria para integrar um dos grupos de sacrifício e então eliminar a criatura.

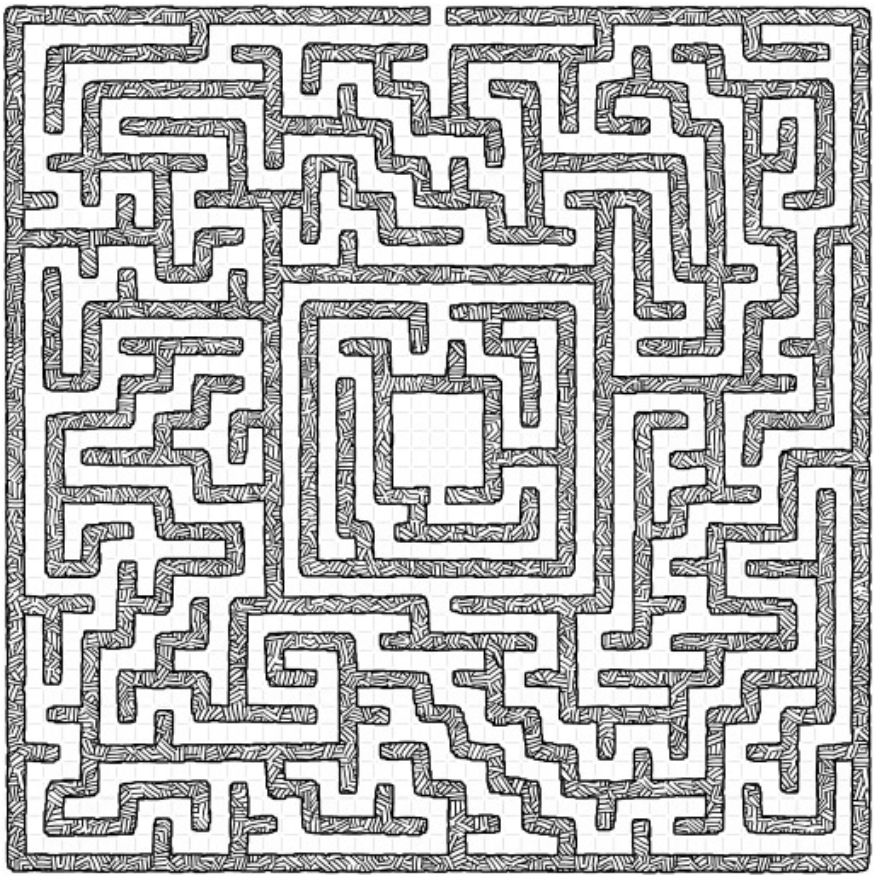

Figura 1 – Labirinto do Minotauro.
Fonte: https://dysonlogos.blog/2020/04/13/minotaur-maze/

Ao chegar em Creta, Teseu recebe a ajuda de Ariadne, que lhe entrega um fio para que ele possa marcar o caminho de ida e retornar do centro do labirinto após derrotar o Minotauro.

Considerando que o comprimento do fio concedido por Ariadne é limitado, Teseu precisa chegar até o Minotauro passando pelo caminho mais curto. Assim, para definir o caminho a ser percorrido no labirinto, considere e implemente pelo menos duas abordagens, entre as vistas em sala de aula, para propor:

- Melhor solução;
- Pior solução.

A implementação deve:

- Permitir a leitura de um arquivo texto, detalhado no final deste documento, com a especificação do ponto inicial e do ponto final;
- Ser desenvolvida usando uma das seguintes linguagens: Java (versão 1.8+), Python (versão 3+), C++ ou C;
- Permitir a escolha de qual algoritmo será executado dentre as opções disponíveis;
- Demonstrar o processo de execução do algoritmo de modo iterativo, exibindo, no mínimo, o estado atual da estrutura de controle, por exemplo, a lista que está sendo utilizada e uma medida de desempenho escolhida, seguindo o exemplo detalhado no final deste documento;
- Apresentar um resumo da execução do algoritmo ao final da execução, indicando a medida de desempenho, o custo e o caminho para ir do ponto inicial até o final;
- Não é necessário implementar interface gráfica.

Outras definições e escolhas, adequabilidade da abordagem/algoritmo e peso de arestas, devem ser realizadas e justificadas levando em consideração o problema descrito.

Preparar um relatório técnico de quatro a dez páginas (incluindo referências e capa), apresentando, ao menos, os algoritmos escolhidos e as medidas para a avaliação de desempenho definidas pelo grupo, bem como as justificativas para as escolhas realizadas.

Bônus: Implemente um algoritmo (dentre os visto em sala de aula) que permite a entrada do comprimento do fio e não explore caminhos cuja distância é maior que esse comprimento. O exemplo de saída a ser seguida para esse algoritmo está no final do documento.

Enviar para a Tarefa Teams Projeto 2 até 31/08/2024 às 23:59hs:

- Todos os arquivos necessários para a execução da implementação;
- Arquivo README com o passo a passo detalhado de como executar o projeto. Caso seja necessário a instalação de alguma biblioteca adicional para execução do código, incluir os passos para instalação;
- Relatório técnico conforme especificação.

Observações:

- A ordem das apresentações finais das implementações será divulgada na Equipe Teams da disciplina e ocorrerão a partir de 02/09/2024;
- Se for julgado necessário, poderá ser solicitada uma nova apresentação;
- A entrega com atraso acarretará a perda 25% da nota do projeto no 1º dia e mais 25% a cada dia adicional;
- É de responsabilidade do discente a certificação do envio/carregamento correto do projeto ao Tarefas na plataforma Teams.

Formato do arquivo de entrada (o arquivo de teste não conterá comentários):

```
ponto_inicial(a0).  % Ponto inicial da busca
ponto_final(f0).    % Ponto final da busca
orientado(s).       % Indica grafo orientado
%orientado(n).      % Indica grafo não orientado
                    % Existe uma ligação entre a0 e b0
pode_ir(a0,b0,95).  % O custo para chegar em b0 partindo de a0 é 95
pode_ir(a0,c0,44).  % Caso não orientado, o custo para a0 partindo de c0 é 44
pode_ir(a0,d0,98).  % Comentários não serão incluídos no arquivo de teste
pode_ir(a0,e0,49).
pode_ir(b0,c0,60).
pode_ir(b0,e0,31).
pode_ir(b0,f0,44).
pode_ir(d0,c0,32).
pode_ir(d0,e0,28).
pode_ir(d0,f0,34).
h(a0,f0,58).        % A distância em linha reta de a0 até f0 é 58
h(b0,f0,24).
h(c0,f0,34).
h(d0,f0,37).
h(e0,f0,5).
h(f0,f0,0).
```

Formato obrigatório de saída a ser seguido:

A cada iteração exibir:

```
Iteração n:                          % Distância atual, heurística e soma para cada
Lista: (nó_x: dist + h = soma)       % vértice da estrutura de controle
Medida de desempenho: valor_x
```

Exemplo de execução:

```
Início da execução
Iteração 1:
Lista: (e0: 12 + 5 = 17) (c0: 20 + 25 = 45) (b0: 15 + 36 = 51)
Medida de desempenho: 1.2

Iteração 2:
Lista: (c0: 16 + 25 = 41) (b0: 14 + 36 = 50) (d0: 27 + 26 = 53)
Medida de desempenho: 2.7
⋮
Fim da execução
Distância: 52
Caminho: a0 – e0 – c0 – f0
Medida de desempenho: 7.6
```

As informações apresentadas acima não representam a execução real de um algoritmo, apenas demostram a sintaxe esperada. Alterar “Medida de desempenho” para o nome da medida escolhida.

Formato obrigatório de saída para algoritmo bônus:

A cada iteração exibir:

```
Qual o comprimento do fio?           % Pergunta realizada por meio do console
Um valor_inteiro                     % Entrada de um inteiro por meio do console
Iteração n:
Fila: (no_x: dist + h = soma)
Medida de desempenho: 0.0
Comprimento restante: Um valor_inteiro
```

Exemplo de execução:

```
Início da execução
Qual o comprimento do fio?
21
Iteração 1:
Fila: (e0: 12 + 5 = 17) (c0: 20 + 25 = 45) (b0: 15 + 36 = 51)
Medida de desempenho: 1.2
Fio restante: 21

Iteração 2:
Fila: (c0: 16 + 25 = 41) (b0: 14 + 36 = 50) (d0: 27 + 26 = 53)
Medida de desempenho: 2.7
Fio restante 0 – Caminho descartado
⋮
Fim da execução
Distância: 18
Caminho: a0 – e0 – c0 – f0
Medida de desempenho: 7.6
```