

Primeiro Trabalho de IA/SI

Estimativa de término: 14/03/2016

24 de Fevereiro de 2016

Este trabalho não é para ser entregue nem apresentado. Será desenvolvido principalmente durante as aulas práticas, mas espera-se que o estudante complemente as horas de contato com trabalho extra-classe. Os testes e exames terão perguntas relacionadas com este trabalho. Para saber o método e critério de avaliação, por favor consulte a ficha da unidade curricular na página do sigarra.

O jogo dos 8 é representado por uma matriz 3x3 onde há 8 células numeradas e uma célula em branco. Variações deste jogo podem conter parte de uma imagem em cada célula. O problema consiste em partir de uma configuração inicial embaralhada das células e chegar a uma configuração final com uma ordenação determinada de algarismos (no caso da matriz de números) ou de imagens (no caso da matriz onde as células representam partes de uma imagem). Os movimentos/operadores possíveis para se chegar de uma configuração a outra são: 1) mover a célula em branco para cima, 2) mover a célula em branco para baixo, 3) mover a célula em branco para a direita e 4) mover a célula em branco para a esquerda.

Dada a descrição do problema acima, implemente uma solução usando:

- as estratégias de busca não informadas **busca em profundidade**, **busca em largura** e **busca em profundidade iterativa** e aplique a este problema.
- as estratégias de busca informadas **busca gulosa** e **busca A*** e aplique a este problema.

A implementação pode ser em qualquer linguagem.

Para cada estratégia, analise:

- tempo para chegar a uma solução,
- quantidade de espaço gasto (número de nós gerados e armazenados),
- completude (o algoritmo consegue encontrar uma solução?) e
- otimalidade (a solução encontrada corresponde ao menor número de passos para chegar da configuração inicial à final ou com menor custo utilizando pouco tempo e pouca memória? Qual é a profundidade da solução encontrada?)

Utilize as seguintes configurações iniciais:

| | | |
|---|---|---|
| 3 | 4 | 2 |
| 5 | 1 | 7 |
| 6 | | 8 |

| | | |
|---|---|---|
| 6 | 2 | 7 |
| 5 | | 3 |
| 8 | 1 | 4 |

A primeira configuração tem solução ótima em profundidade 23 (menor número de movimentos possível para chegar da configuração inicial à configuração final).

A segunda configuração inicial não leva à configuração final proposta. Explique o porque.

Apesar de propor apenas uma configuração inicial e final para teste, prepare o seu programa de forma a ser possível entrar com diversas configurações iniciais e finais. Portanto, escreva o seu código de forma que o utilizador possa escolher as configurações inicial e final. O seu código deve verificar se há solução para chegar do estado inicial ao estado final **antes** de iniciar a busca, e deve emitir uma mensagem de erro se não houver caminho entre a solução inicial e a final.

A configuração final de teste deve ser a seguinte:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | | 4 |
| 7 | 6 | 5 |

Utilize o algoritmo 1 como base para implementar **todas** as buscas:

Algorithm 1 Algoritmo Geral de Busca

```
1: GeneralSearchAlgorithm(QueueingFunction)
2: queue = configInicial
3: while queue not empty do
4:   node = removeFrontNodeFrom(queue)
5:   if node is solution then
6:     return Path to solution
7:   end if
8:   descendantList = MakeDescendants(node)
9:   insert(descendantList,queue,QueueingFunction)
10: end while
11: return "solution not found"
```

Este algoritmo recebe como parâmetro uma função de enfileiramento que será utilizada para ordenar a lista de nós (configurações) abertos (ainda não explorados).

Pontos a abordar

1. Introdução

- O que é um problema de busca/procura?
- Quais são os métodos utilizados para resolver problemas de procura?

2. Estratégias de Procura

(a) Procura não guiada (blind - "cega")

- Profundidade (DFS - Depth-First Search: como funciona, quando se aplica, qual é a complexidade temporal e espacial?)
- Largura (BFS - Breadth-First Search: como funciona, quando se aplica, qual é a complexidade temporal e espacial?)

- Busca Iterativa Limitada em Profundidade (como funciona, quando se aplica, qual é a complexidade temporal e espacial?)
- (b) Procura guiada (que usa alguma função **heurística** para orientar a procura)
- Gulosa
 - Como funciona e quando se aplica?
 - Qual foi a heurística utilizada para o problema a ser resolvido e por que esta heurística foi escolhida?
 - Busca A*
 - Como funciona e quando se aplica?
 - Qual foi a heurística utilizada para o problema a ser resolvido e por que esta heurística foi escolhida?

3. Descrição da Implementação

- Linguagem utilizada? Por que escolheu esta linguagem? Há alguma vantagem em utilizar esta linguagem para resolver este tipo de problema?
- Estruturas de dados utilizadas? Como foi que escolheu as estruturas de dados? São eficientes para manipular os dados do problema?

4. Resultados

Fazer tabela (ou curvas comparativas) com tempos de execução, utilização de memória e se encontrou a solução, para cada configuração, para cada estratégia, além da profundidade da solução encontrada. Se preferir utilizar uma tabela, esta poderá ter um sumário dos resultados organizados da seguinte forma:

| Estratégia | Tempo (segundos) | Espaço | Encontrou a solução? | Profundidade/Custo |
|------------|------------------|--------|----------------------|--------------------|
| DFS | ... | ... | ... | ... |
| BFS | ... | ... | ... | ... |
| IDFS | ... | ... | ... | ... |
| Gulosa | ... | ... | ... | ... |
| A* | ... | ... | ... | ... |

5. Comentários Finais e Conclusões

Comentar sobre as estratégias fazendo uma comparação entre o seu desempenho e eficácia para encontrar as soluções. Concluir dizendo qual foi a melhor estratégia para este problema.