

# Atividade de implementação 1 - Resolução de problemas usando busca

## 1 Introdução

Neste trabalho estudaremos a abordagem de resolução de problemas através de busca no espaço de estados, implementando métodos para solucionar o [jogo: régua-puzzle](#).

Os objetivos deste exercício-programa são:

- (i) compreender a abordagem de resolução de problemas baseada em busca no espaço de estados;
- (ii) estudar uma formulação de problema de busca para criar um jogador autônomo de régua-puzzle;
- (iii) implementar algoritmos de busca informada e não-informada e comparar seus desempenhos.

## 2 O jogo: régua-puzzle

Considere um jogo onde  $2N$  blocos são alinhados em uma régua com  $2N+1$  posições. Existem  $N$  blocos brancos (B),  $N$  blocos azuis (A) e uma posição vazia. Uma régua pode ser representada por um vetor  $R$  com posições no intervalo  $[1 .. 2N+1]$ .

B	A		A	B
---	---	--	---	---

O objetivo do jogo é movimentar os blocos (de acordo com as regras descritas a seguir) de forma a colocar todos os blocos brancos do lado esquerdo dos blocos azuis, ou seja, para todo  $i$ , se  $R[i] = B$  então  $R[j] \neq A$ , para todo  $0 < j < i$ . Uma solução ótima para esse jogo é:

- a solução de menor número de movimentos, considerando custo 1 para todo movimento ou
- a solução de custo mínimo, considerando o custo de movimentos conforme definido na próxima seção.

### 2.1 Regras e Custo de Movimentação de blocos

Definimos a distância entre duas posições  $i$  e  $j$ , como  $j - i$ , sendo  $0 < i < j \leq 2N + 1$ .

Blocos podem pular para a posição vazia quando a posição vazia estiver distante de no máximo  $N$  casas da posição do bloco. Desta maneira, existem no máximo  $2N$  movimentos legais (no caso do vazio estar localizado exatamente no meio da régua). O custo de um pulo é igual à distância entre a posição do bloco e a posição vazia. Por exemplo, considere uma régua-puzzle com  $N=2$ , em que o estado inicial é dado por:

B	A		A	B
---	---	--	---	---

Existem quatro estados sucessores para o estado inicial:

B		A	A	B
---	--	---	---	---

B	A	A		B
---	---	---	--	---

	A	B	A	B
--	---	---	---	---

B	A	B	A	
---	---	---	---	--

sendo que os dois primeiros movimentos tem custo 1 e os dois últimos, custo 2. Note ainda que para uma régua com  $N=2$ , existem cinco estados que satisfazem a meta:

B	B	A		A
---	---	---	--	---

B	B		A	A
---	---	--	---	---

B		B	A	A
---	--	---	---	---

	B	B	A	A
--	---	---	---	---

B	B	A	A	
---	---	---	---	--

Nesta atividade vamos considerar dois tipos de critério de otimalidade:

- Critério 1. a solução ótima é a de menor número de movimentos ou
- Critério 2. a solução ótima é a de caminho de menor custo.

## 2.2 Implementação

Você deverá implementar os seguintes algoritmos de busca discutidos em sala de aula:

1. Busca em Largura (BL) (usando busca em grafo). Nesse caso, a solução que deverá ser devolvida é aquela de menor número de passos (Critério 1).
2. Busca em Profundidade (BP) (também usando busca em grafo), para devolver a primeira solução encontrada. Nesse caso, a solução que deverá ser devolvida é a primeira encontrada pela busca.
3. Busca em Profundidade Limitada (BPL) (usando busca em árvore, ou seja, sem evitar nós repetidos). Nesse caso, a solução que deverá ser devolvida será a primeira encontrada pela busca, dentro do limite definido. Teste para 3 valores diferentes de limite,  $L_1$ ,  $L_2$  e  $L_3$ , com  $L_i \geq d$ , sendo  $d$  a profundidade da solução encontrada pela Busca em Largura. Essa busca garante devolver a solução ótima segundo o Critério 1?

4. Busca em Profundidade Iterativa (BPI) (com passo=1). Nesse caso, como na busca em largura, a solução devolvida deverá ser aquela de menor número de passos (Critério 1).
5. Busca de Custo Uniforme (BCU). Nesse caso, a solução devolvida deverá ser aquela de menor custo (Critério 2).

Além das buscas não-informadas descritas acima, você deve resolver os problemas com o seguinte algoritmo de busca informada (best-first search), usando [2 heurísticas diferentes](#):

6. Busca  $A^*$  (versão busca em grafo).

### 2.2.1 Entradas e saída

Seu programa deve considerar como entrada o nome de um arquivo contendo um jogo régua-puzzle.

Os arquivos contendo régua-puzzles consistem de duas linhas: a primeira corresponde ao número de blocos brancos (igual ao número de blocos azuis). Se a primeira linha contém o número  $N$ , então o puzzle consiste de  $N$  blocos brancos,  $N$  blocos azuis e  $2N+1$  posições. A segunda linha mostra o estado inicial, descrito como uma sequência de caracteres 'B' (blocos brancos), caracteres 'A' (blocos azuis) e 1 caractere '-' para indicar casa vazia. Exemplo de um problema régua-puzzle:

```
2
BA-AB
```

Além do arquivo contendo o jogo, a chamada do seu programa deve ser feita com um parâmetro de entrada que pode ser BL, BP, BPL, BPI, BCU ou  $A^*$ , especificando o tipo de busca que será executada. Para que você possa fazer a análise comparativa dos algoritmos de busca, seu programa deve computar (e imprimir na tela) as seguintes informações:

- se encontrou ou não uma solução (falha ou sucesso)
- a profundidade do estado meta (ou seja, o tamanho da solução);
- o custo da solução,  $g(G)$ , sendo  $G$  um estado meta;
- o número de nós explorados e o número total de nós gerados (explorados + borda);
- o fator de ramificação médio ( $b$ ), calculado como a razão entre o número
- total de nós sucessores gerados e o número de nós explorados; e
- o caminho da solução encontrada (sequência de estados da solução).

Essas informações deverão ser usadas para a construção do relatório que deverá ser entregue.

### 2.2.2 Linguagens de programação

Você poderá usar as seguintes linguagens de programação: C, C++, Java, Python ou Ruby.

### 2.2.3 Alguns exemplos de puzzles para testar os algoritmos

```
2
-ABAB
```

```
2
ABBA-
```

2  
-AABB

3  
-ABABAB

3  
ABBBAA-

3  
-AAABBB

4  
-ABABABAB

4  
ABBBBAAA-

4  
-AAAABBBB

### 3 Relatório

Após o desenvolvimento da parte prática, você deverá testar seus algoritmos e redigir um relatório claro e sucinto. Assim, você deverá:

- (a) compilar em tabelas e/ou gráficos as estatísticas das buscas implementadas em termos de nós expandidos e comprimento de plano;
  - tente rodar o seu programa com o maior número de blocos possíveis!
- (b) discutir os méritos e desvantagens de cada método, no contexto dos dados obtidos;
- (c) sugerir possíveis melhorias na sua implementação e relatar dificuldades.

### 4 Entrega

Você deve entregar um arquivo `T1-RA1_RA2.zip` contendo:

- (1) O código implementado;
- (2) relatório em formato PDF com a comparação e discussão dos resultados e uma breve explicação de como rodar seu projeto.