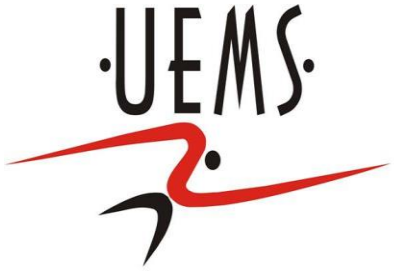


Busca Cega

Profa. Mercedes Gonzales Márquez

Tópicos

- Busca cega ou busca não informada
  - Busca em largura
  - Busca em profundidade
  - Busca em profundidade limitada
  - Busca com aprofundamento iterativo
  - Busca bidirecional

Estratégias de Busca Sem Informação (ou Busca Cega)

- Estratégias de busca sem informação não recebe qualquer indicação sobre a proximidade que um estado se encontra de sua meta. Veja o exemplo de Zarad no mapa (nos slides da aula anterior).
- As estratégias de busca sem informação se distinguem pela ordem em que os nós são expandidos.

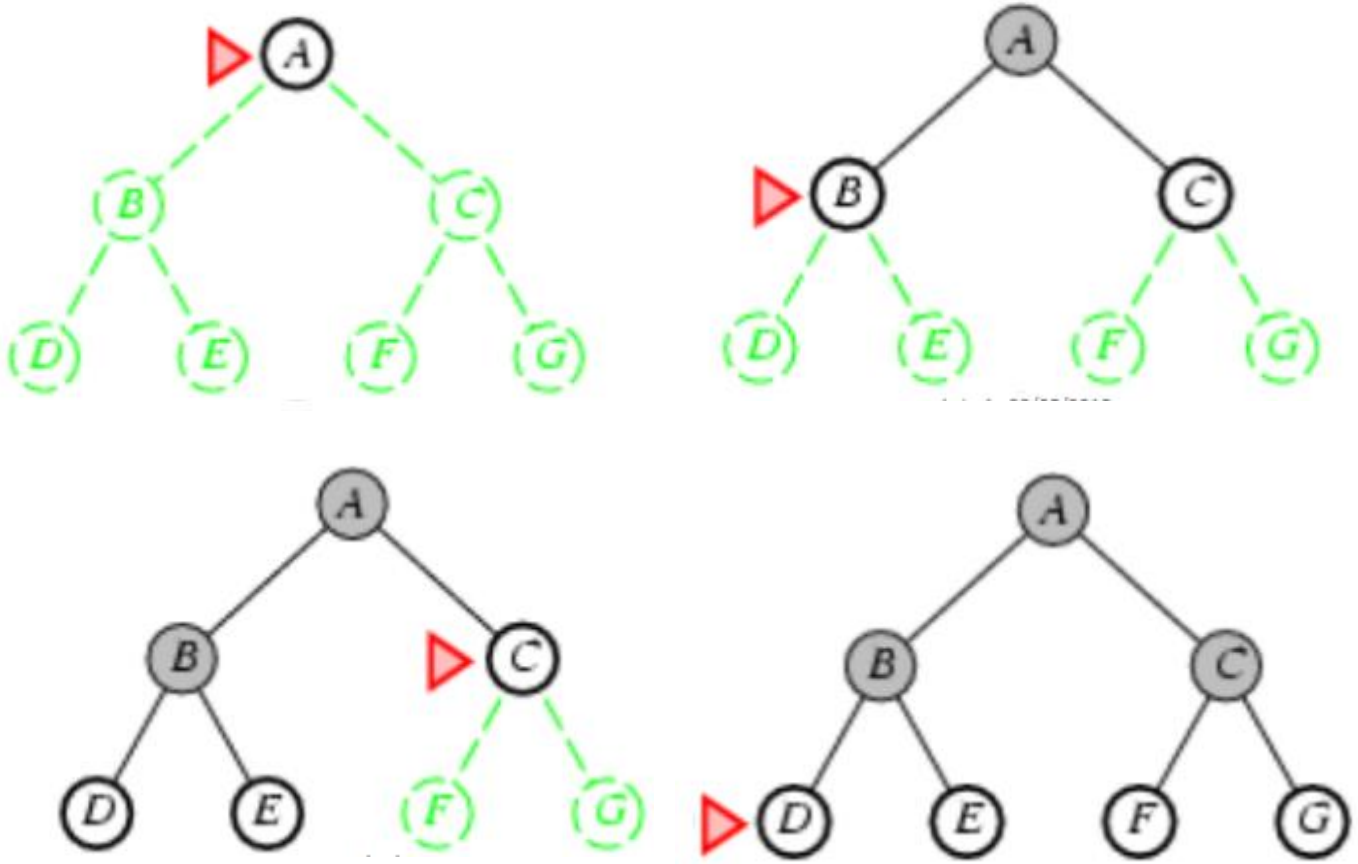
Estratégias de Busca
Sem Informação (ou Busca Cega)

- Busca em largura
- Busca em profundidade
- Busca em profundidade limitada
- Busca de aprofundamento iterativo.
- Busca de custo uniforme

Busca em largura

- Explora o espaço de estados **camada por camada**, visitando todos os nós em uma distância *k* antes de avançar para *k+1*.
- O nó raiz é expandido primeiro e, em seguida, todos os sucessores dele, depois todos os sucessores desses nós.
- Todos os nós em uma dada profundidade são expandidos antes de todos os nós do nível seguinte.

Busca em largura



Busca em largura

- Uma fila (FIFO) é usada : novos nós (que são sempre mais profundos do que seus pais) vão para o fim da fila, enquanto os antigos, que são mais rasos que os novos, são expandidos primeiro.
- A busca em largura sempre acha uma solução com um número mínimo de ações porque quando está gerando nós na profundidade d, ela já gerou todos os nós na profundidade d-1.

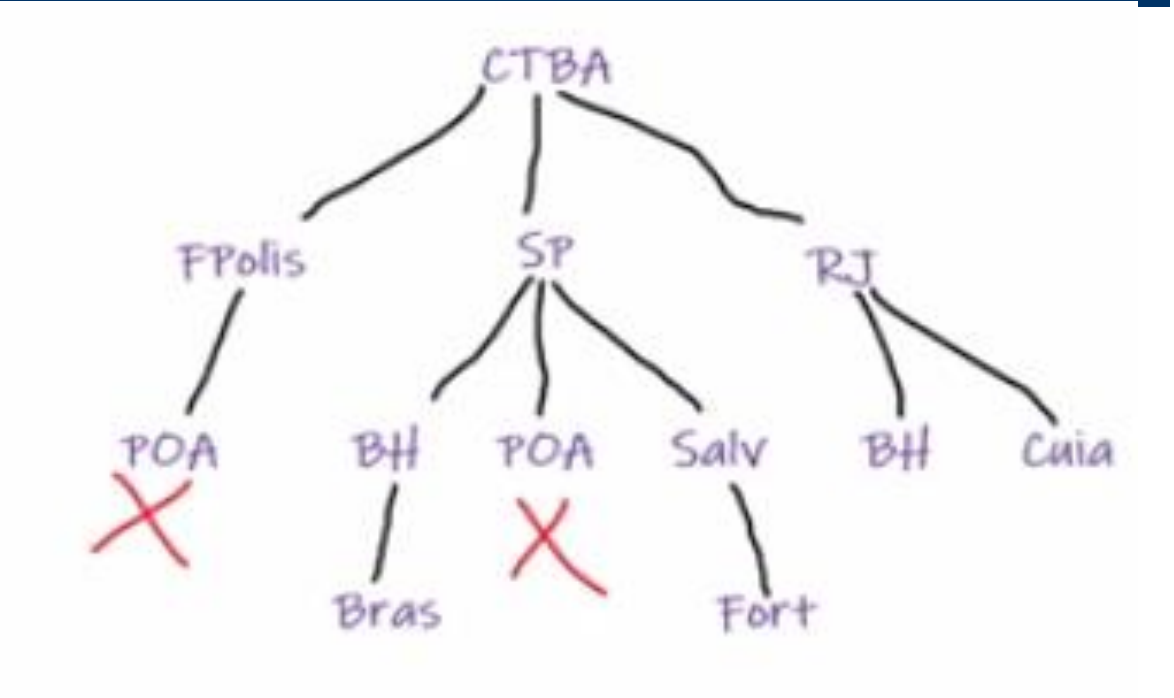
Busca em largura

- Propriedades da busca em largura, considere que cada estado tem b sucessores.
- Completa? Sim (se b é finito)
- Tempo? $1+b+b^2+b^3+\ldots +b^d = O(b^d)$
- Espaço? $O(b^d)$ (mantém todos os nós na memória)
- Ótima? Sim (se todas as ações tiverem o mesmo custo.

Exemplo 1

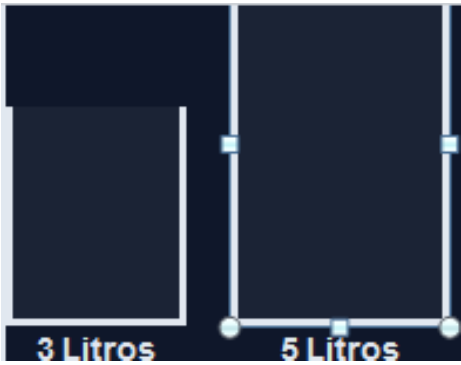
Queremos sair de Curitiba
e chegar em Fortaleza

Exemplo 1



Exemplo 2

O PROBLEMA DOS JARROS DE ÁGUA



Medir exatamente **4 litros** de água usando apenas dois jarros: um com capacidade de 3 litros e outro de 5 litros, sem marcações intermediárias. As ações permitidas são

- **Encher** um jarro completamente.
- **Esvaziar** um jarro completamente.
- **Transferir** de um jarro para outro.

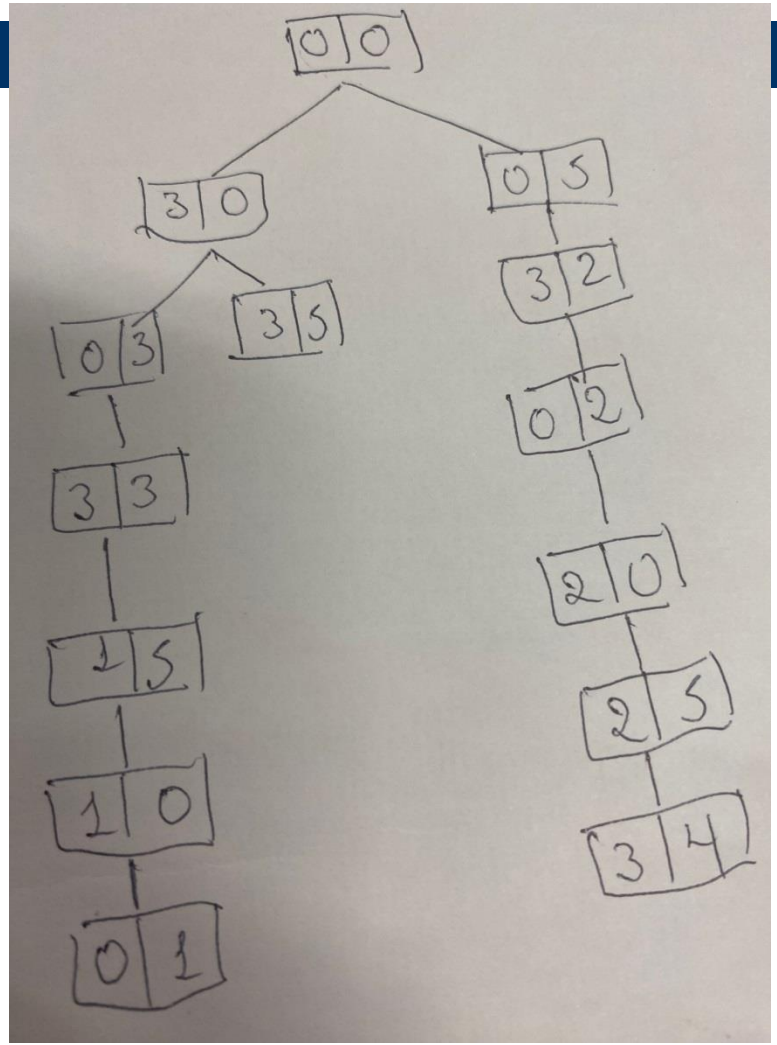
Exemplo 2

O PROBLEMA DOS JARROS DE ÁGUA

Cada estado é definido pelo par **(x, y)**:
x = água no jarro de 3L (0 a 3)
y = água no jarro de 5L (0 a 5)
Estado Inicial: **(0, 0)**

Estado Final: (*,4).

Busca em largura



Exemplo 3

O PROBLEMA DOS MISSIONÁRIOS E CANIBAIS

Um clássico quebra-cabeça de busca onde três missionários e três canibais precisam atravessar um rio usando um barco que comporta apenas duas pessoas.

Exemplo 3

Cada estado é definido pela tupla **(m, c, b)**:

- **m** = missionários na margem esquerda (0 a 3)
- **c** = canibais na margem esquerda (0 a 3)
- **b** = posição do barco (0 = esquerda, 1 = direita)

> **Inicial: (3, 3, 0) → Objetivo: (0, 0, 1)**

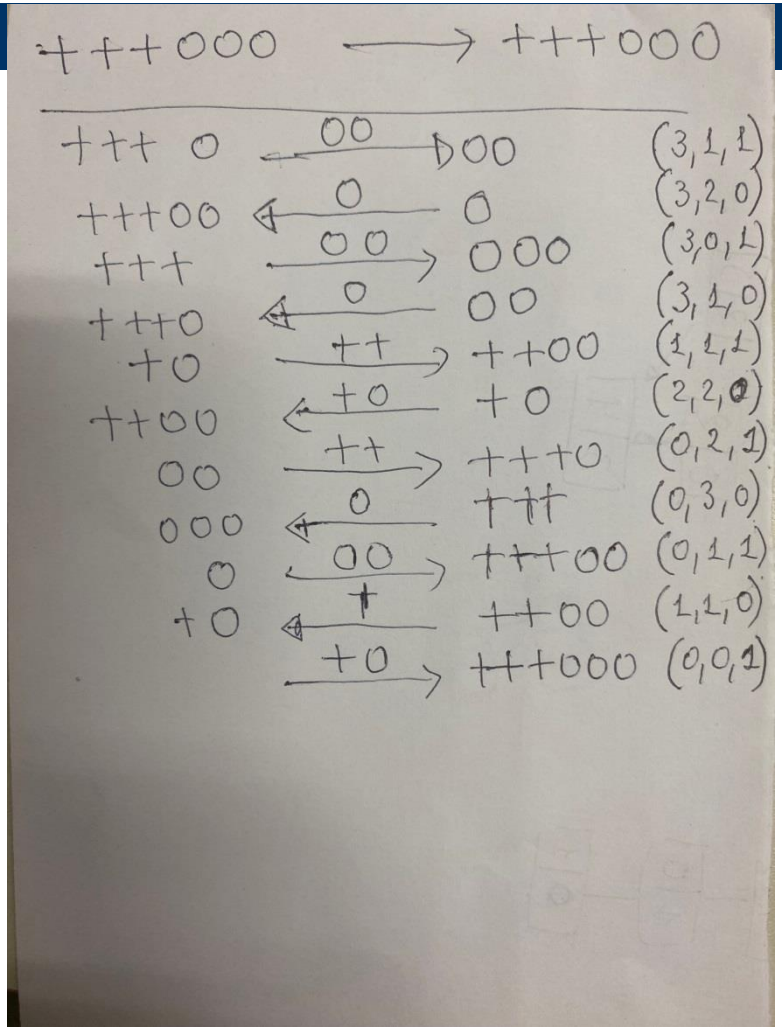
Restrições:

- O barco **não pode atravessar vazio** (requer 1 ou 2 pessoas).
- Em nenhuma margem os canibais podem superar os missionários (para evitar que sejam devorados).

Exemplo 3

- Inicia com o estado (3, 3, 0)
- Gera todos os movimentos válidos: (1, 0, 1), (2, 0, 1), (0, 1, 1), (0, 2, 1), (1, 1, 1)
- Para cada novo estado, verifica se é válido (restrição de canibais/missionários)
- Continua explorando camada por camada até alcançar (0, 0, 1)

Exemplo 3


+++000 ⟶ +++000
+++ 0 ⟶ 00 00 (3,1,1)
+++00 ⟵ 0 0 (3,2,0)
+++ ⟶ 00 000 (3,0,1)
+++0 ⟵ 0 00 (3,1,0)
+0 ⟶ ++ ++00 (1,1,1)
++00 ⟵ +0 +0 (2,2,0)
00 ⟶ ++ +++0 (0,2,1)
000 ⟵ 0 +++ (0,3,0)
0 ⟶ 00 +++00 (0,1,1)
+0 ⟵ + ++00 (1,2,0)
⟶ +0 +++000 (0,0,1)

Árvore de Solução – Problema dos Missionários e Canibais

Representação: (M, C, B)

M = Missionários, C = Canibais, B = Lado do barco (0 = esquerda, 1 = direita)

Legenda:

(M, C, B) = (Missionários, Canibais, Barco)

B = 0 → barco no lado esquerdo

B = 1 → barco no lado direito

Seta (rótulo) = pessoas que atravessam (direção do movimento)

→ : esquerda para direita

← : direita para esquerda

Representação:

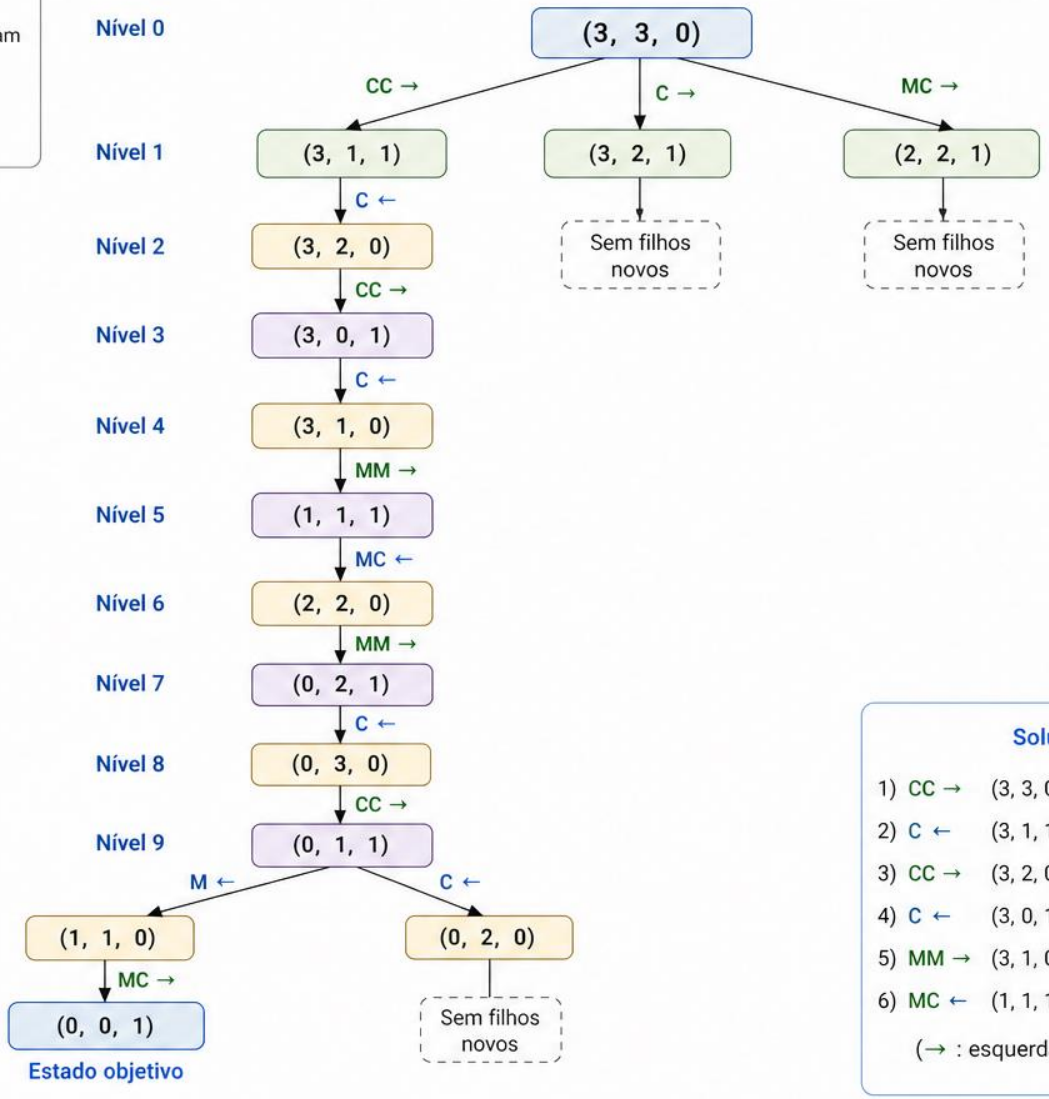
M = Missionários

C = Canibais

B = Lado do barco

0 = esquerda

1 = direita



Solução ótima (11 travessias):

1) CC →	(3, 3, 0) → (3, 1, 1)	7) MM →	(2, 2, 0) → (0, 2, 1)
2) C ←	(3, 1, 1) → (3, 2, 0)	8) C ←	(0, 2, 1) → (0, 3, 0)
3) CC →	(3, 2, 0) → (3, 0, 1)	9) CC →	(0, 3, 0) → (0, 1, 1)
4) C ←	(3, 0, 1) → (3, 1, 0)	10) M ←	(0, 1, 1) → (1, 1, 0)
5) MM →	(3, 1, 0) → (1, 1, 1)	11) MC →	(1, 1, 0) → (0, 0, 1)
6) MC ←	(1, 1, 1) → (2, 2, 0)		

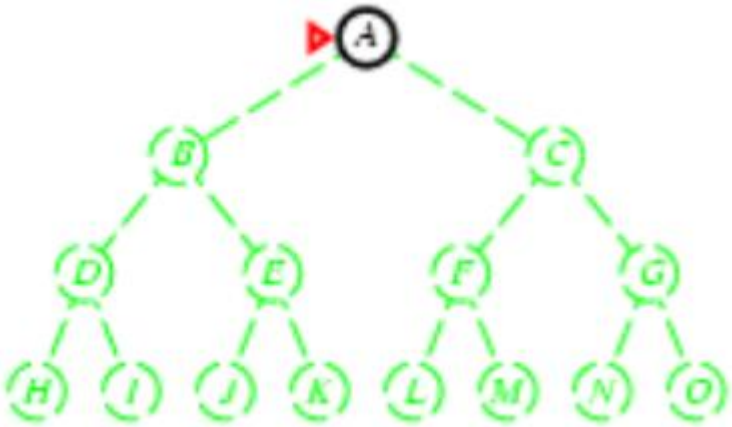
(→ : esquerda para direita, ← : direita para esquerda)

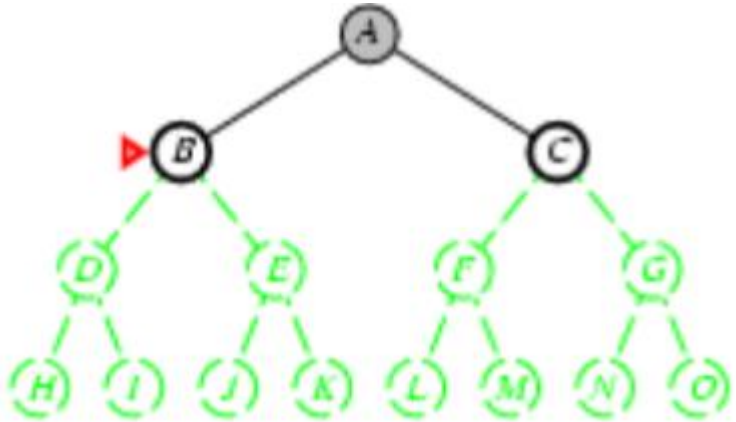
Busca em profundidade

- Segue um caminho até o final, explorando o máximo possível ao longo de cada ramo.
- A busca prossegue até o nível mais profundo da árvore, onde os nós não tem sucessores. Então a busca devolve ao nó seguinte mais profundo acima que ainda tem sucessores inexplorados.
- Uma fila (LIFO) é usada : novos nós (que são sempre mais profundos do que seus pais) são expandidos primeiro, enquanto os antigos, que são mais rasos que os novos, vão para o final da fila.

Busca em profundidade

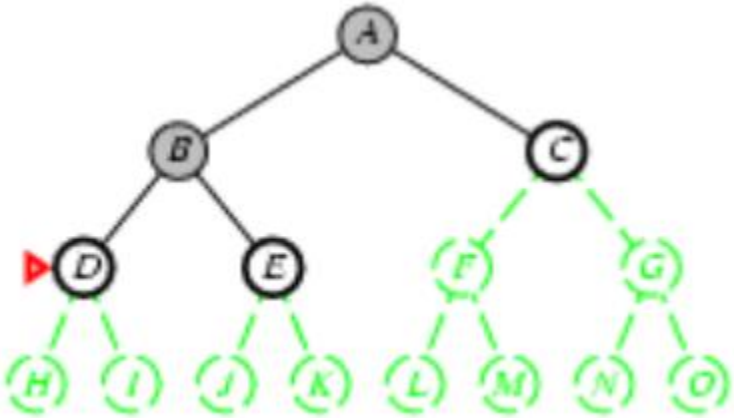
- Expande o nó não-expandido mais profundo.

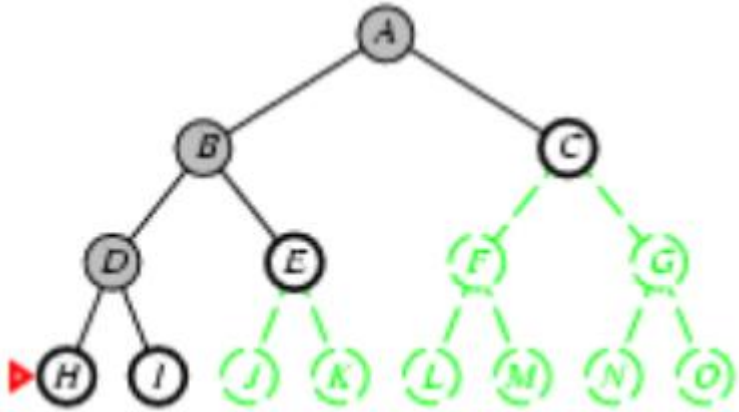




Busca em profundidade

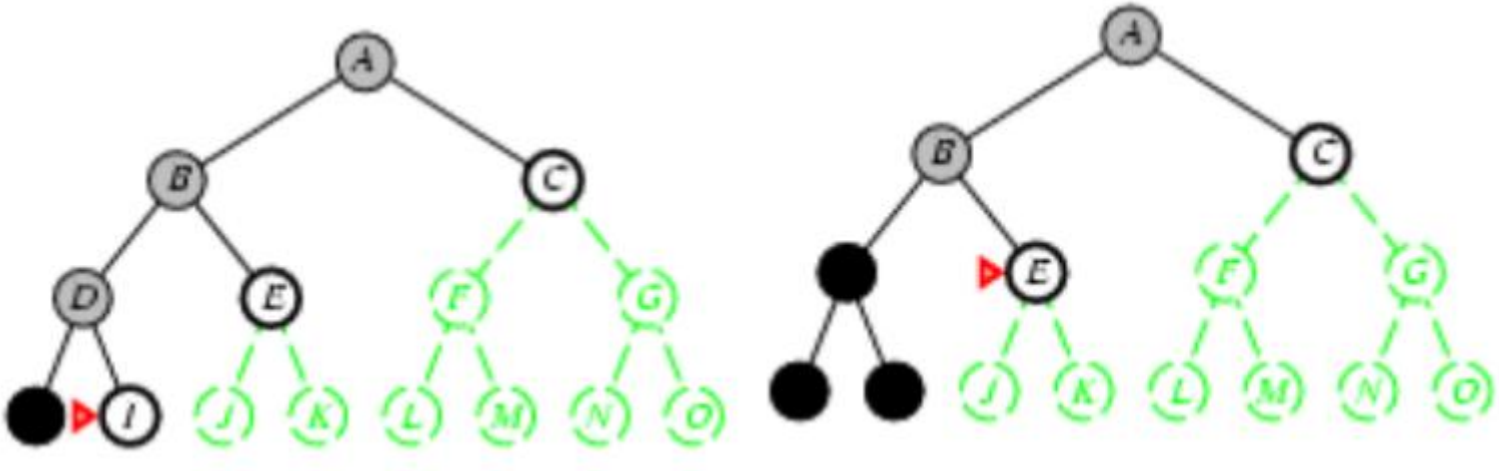
- Expande o nó não-expandido mais profundo.





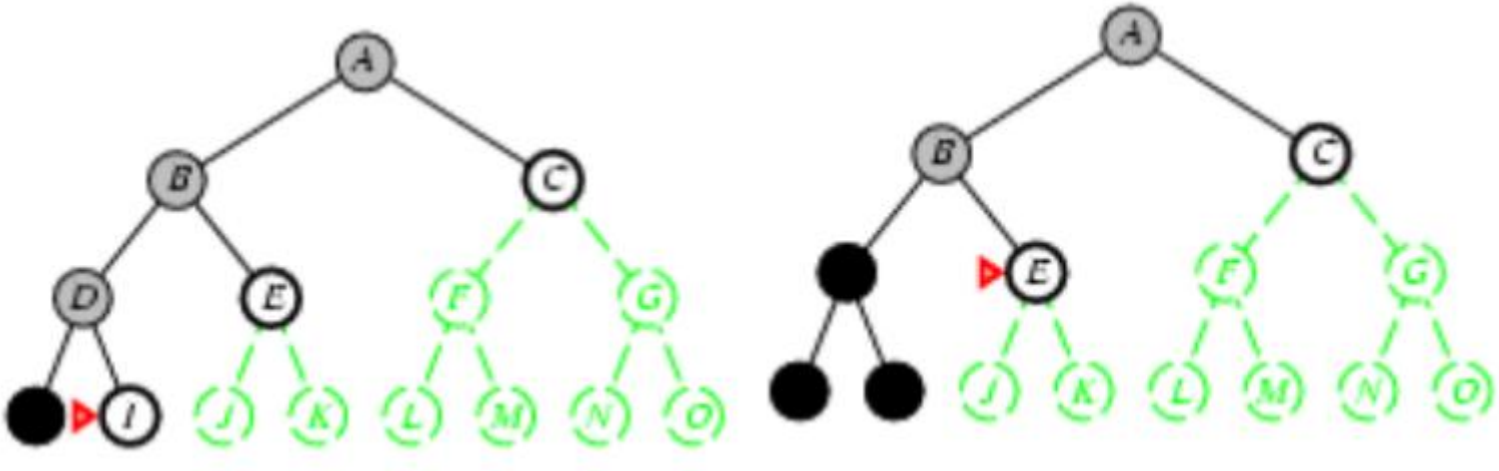
Busca em profundidade

- Expande o nó não-expandido mais profundo.



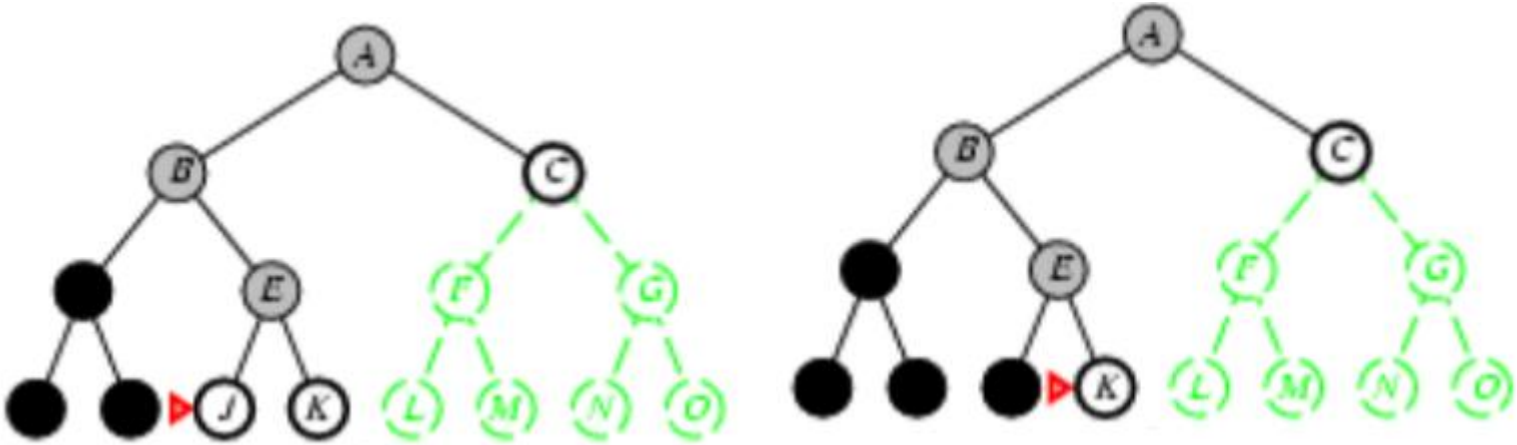
Busca em profundidade

- Expande o nó não-expandido mais profundo.



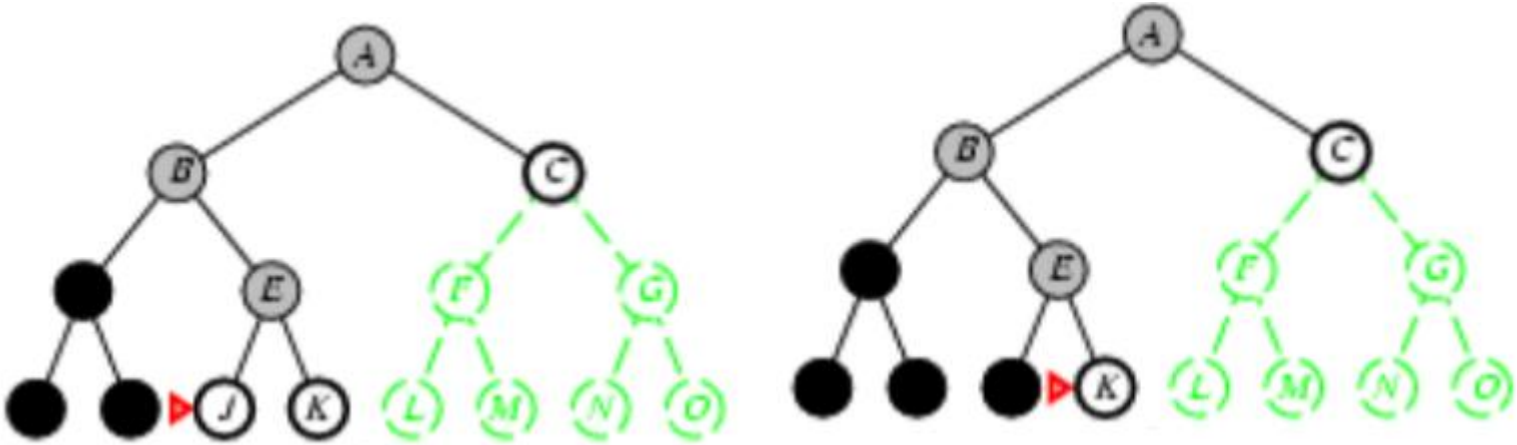
Busca em profundidade

- Expande o nó não-expandido mais profundo.



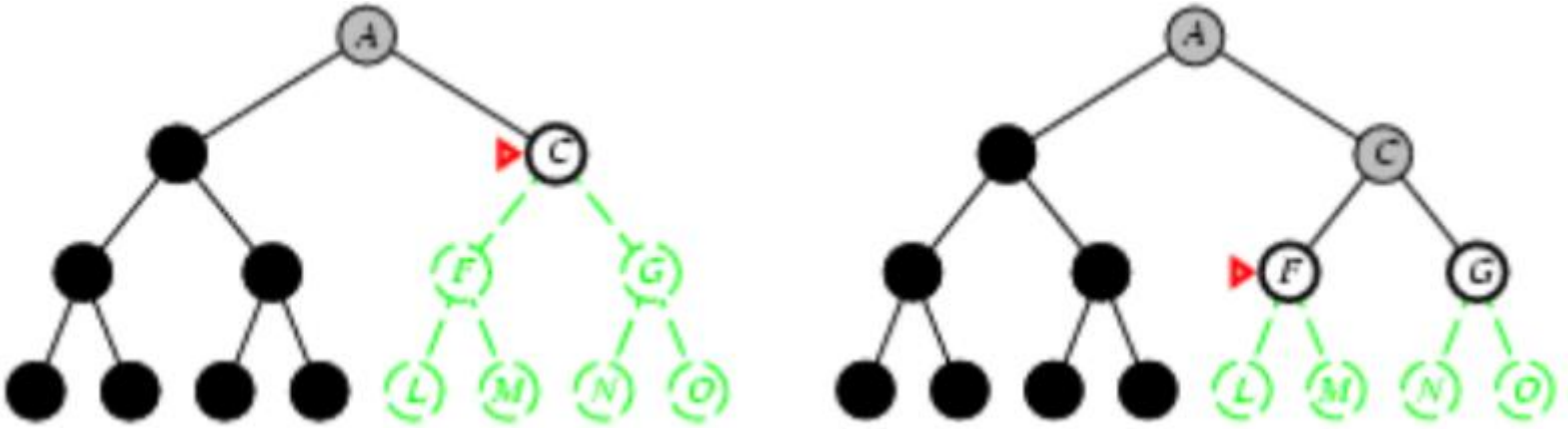
Busca em profundidade

- Expande o nó não-expandido mais profundo.



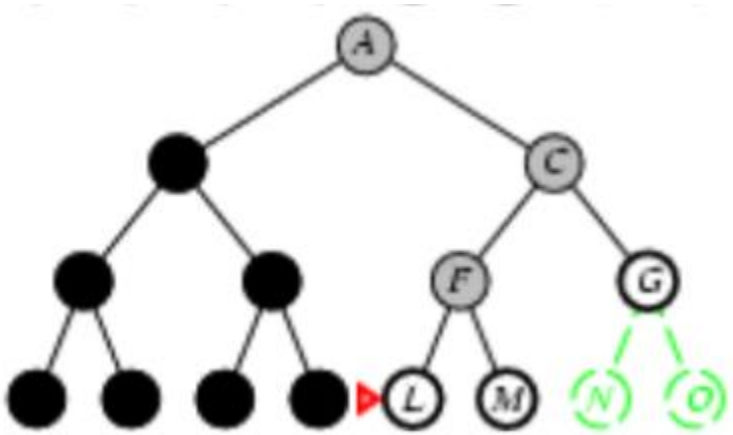
Busca em profundidade

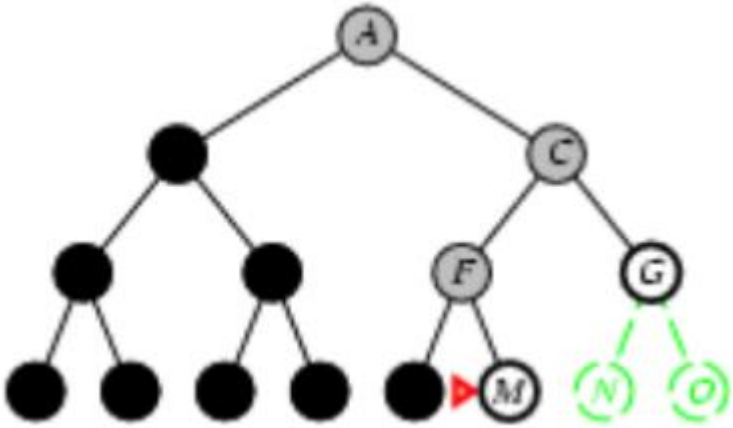
- Expande o nó não-expandido mais profundo.



Busca em profundidade

- Expande o nó não-expandido mais profundo.





Busca em profundidade

- Só precisa armazenar um único caminho da raiz até um nó folha, e os nós irmãos não expandidos;
- Nós cujos descendentes já foram completamente explorados podem ser retirados da memória;
- Faz muito menos uso de memória que a busca em largura. A fronteira da busca em largura pode ser vista como a superfície de uma esfera sempre em expansão, enquanto que a fronteira da busca em profundidade é apenas um raio dessa esfera.

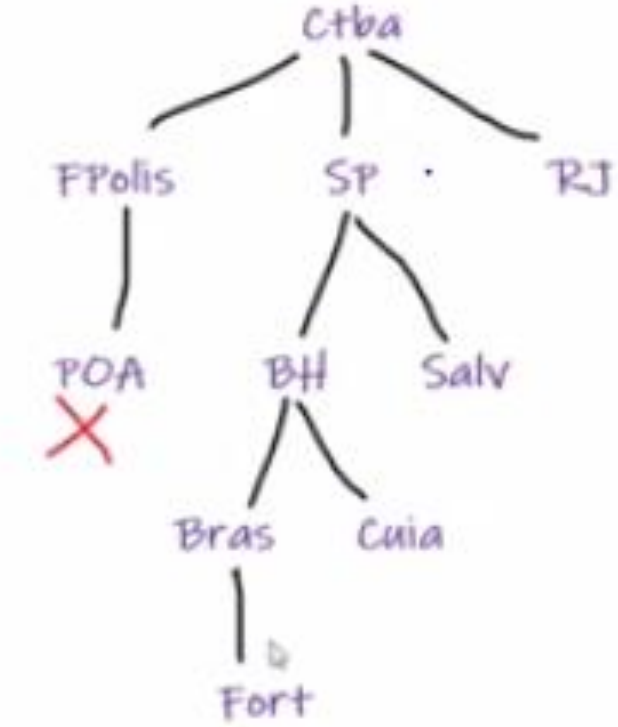
Busca em profundidade

- Logo para ramificação b e profundidade máxima m, a complexidade espacial é: O(bm)
- Propriedades da busca em profundidade.
- Completa? Sim em espaços com profundidade finita. Não completa em espaços com loops.
- Tempo? $O(b^m)$
- Espaço? O(bm)
- Ótima? Não.

Exemplo 1

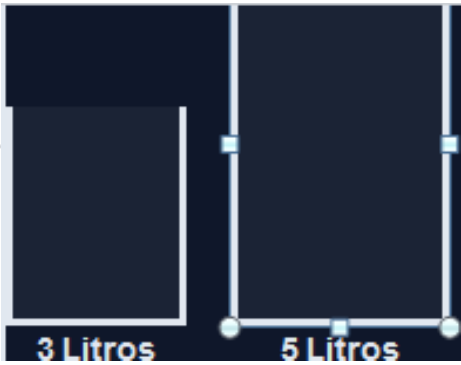
Queremos sair de Curitiba
e chegar em Fortaleza

Exemplo 1



Exemplo 2

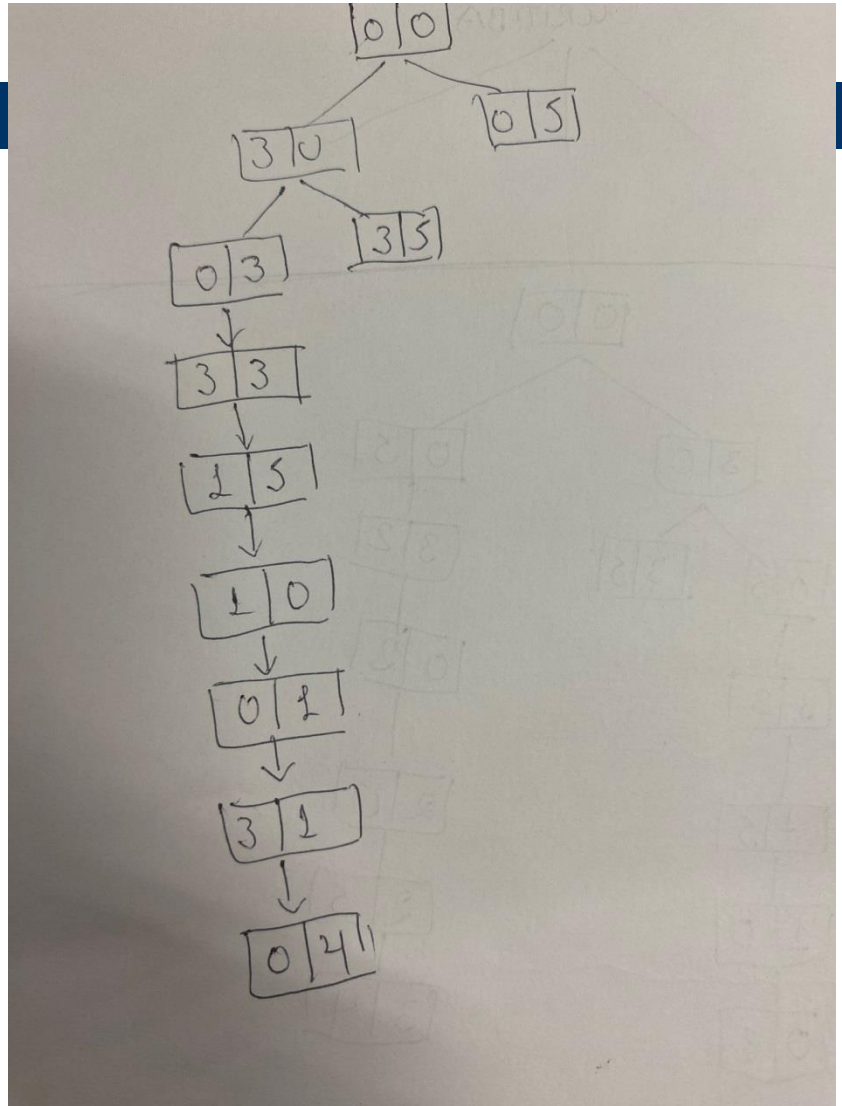
O PROBLEMA DOS JARROS DE ÁGUA



Medir exatamente **4 litros** de água usando apenas dois jarros: um com capacidade de 3 litros e outro de 5 litros, sem marcações intermediárias. As ações permitidas são

- **Encher** um jarro completamente.
- **Esvaziar** um jarro completamente.
- **Transferir** de um jarro para outro.

Exemplo 2



Exemplo 3

O PROBLEMA DOS MISSIONÁRIOS E CANIBAIS

Um clássico quebra-cabeça de busca onde três missionários e três canibais precisam atravessar um rio usando um barco que comporta apenas duas pessoas.

Exemplo 3

Legenda:

(M, C, B) = (Missionários, Canibais, Barco)

B = 0 → barco no lado esquerdo

B = 1 → barco no lado direito

Seta (rótulo) = pessoas que atravessam (direção do movimento)

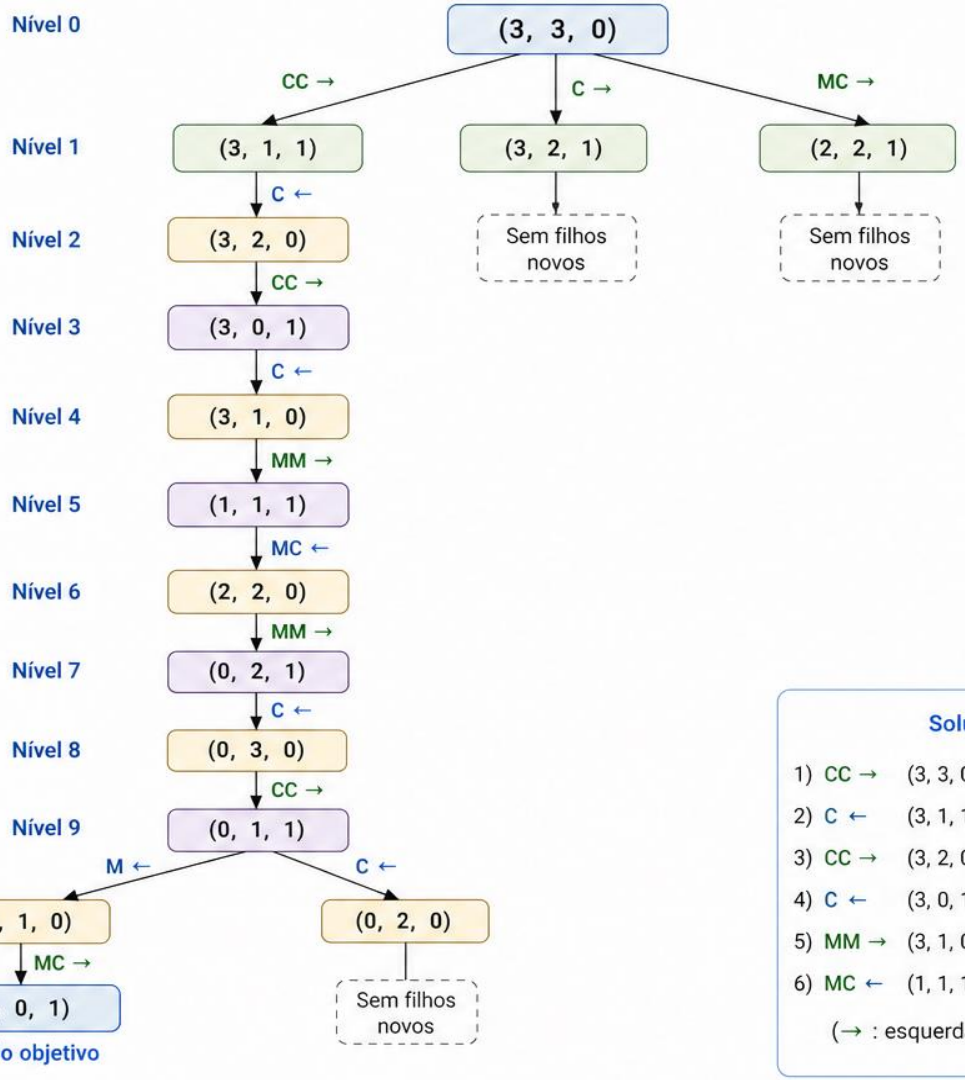
→ : esquerda para direita

← : direita para esquerda

Árvore de Solução – Problema dos Missionários e Canibais

Representação: (M, C, B)

M = Missionários, C = Canibais, B = Lado do barco (0 = esquerda, 1 = direita)



Representação:

M = Missionários

C = Canibais

B = Lado do barco

0 = esquerda

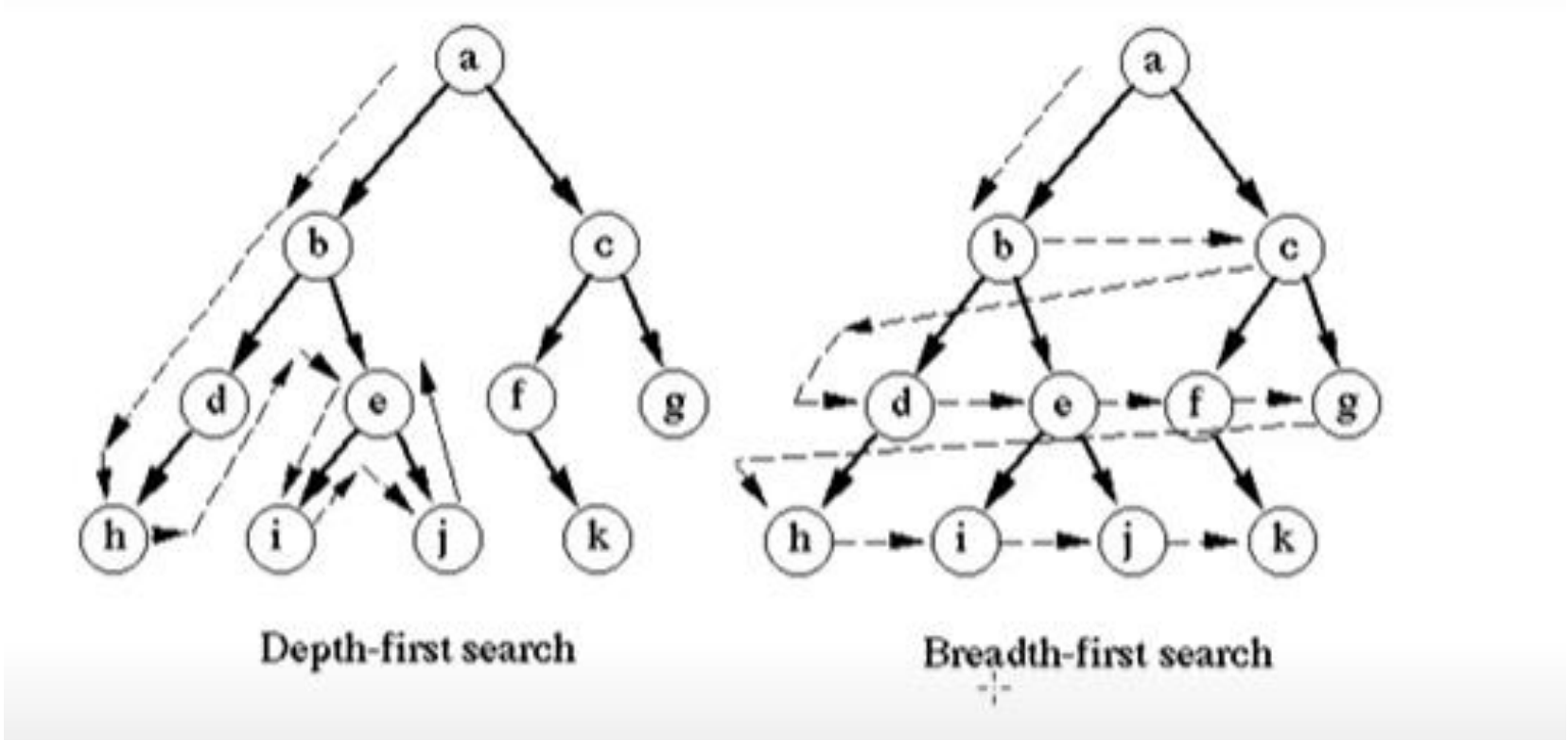
1 = direita

Solução ótima (11 travessias):

1) CC → (3, 3, 0) → (3, 1, 1)
2) C ← (3, 1, 1) → (3, 2, 0)
3) CC → (3, 2, 0) → (3, 0, 1)
4) C ← (3, 0, 1) → (3, 1, 0)
5) MM → (3, 1, 0) → (1, 1, 1)
6) MC ← (1, 1, 1) → (2, 2, 0)
7) MM → (2, 2, 0) → (0, 2, 1)
8) C ← (0, 2, 1) → (0, 3, 0)
9) CC → (0, 3, 0) → (0, 1, 1)
10) M ← (0, 1, 1) → (1, 1, 0)
11) MC → (1, 1, 0) → (0, 0, 1)

(→ : esquerda para direita, ← : direita para esquerda)

Busca em largura e profundidade



Busca em largura e profundidade

DFS

Depth-First Search: Mergulha o mais fundo possível em um ramo antes de retroceder.

Prioriza a profundidade.

BFS

Breadth-First Search: Varre todos os nós de um nível antes de avançar para o próximo.

Prioriza a largura/proximidade.

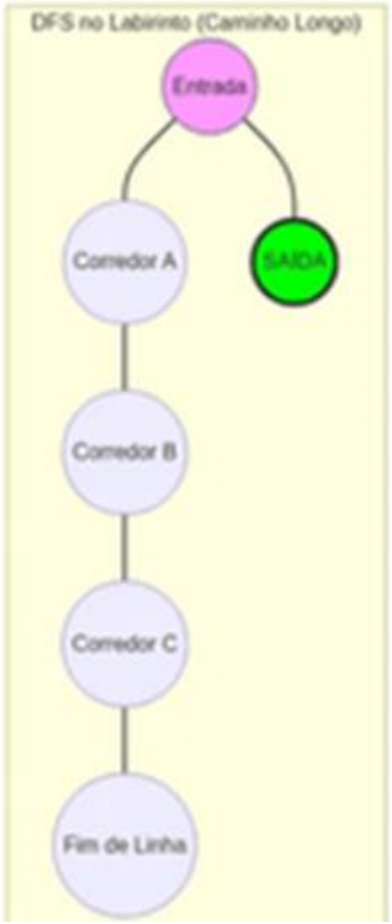
Busca em largura e profundidade

Exemplo 1: O Robô no Labirinto

Cenário: Um robô em um labirinto desconhecido buscando a saída.

Comportamento DFS: Escolhe uma direção e segue até bater na parede ou encontrar um beco sem saída.

Comportamento BFS: Explora todas as direções próximas simultaneamente, nível por nível.



Busca em largura e profundidade

Exemplo 1: O Robô no Labirinto

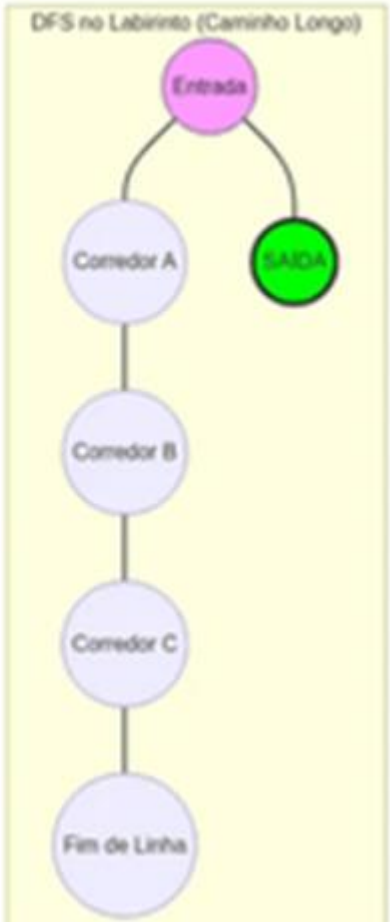
Desvantagem da DFS:

- Exploração Cega

Se a saída estiver à direita e a DFS escolher à esquerda, ela explora todo esse ramo antes de voltar.

- **Ineficiência de Percurso**

O robô percorre distâncias grandes e irrelevantes, gastando tempo e energia.



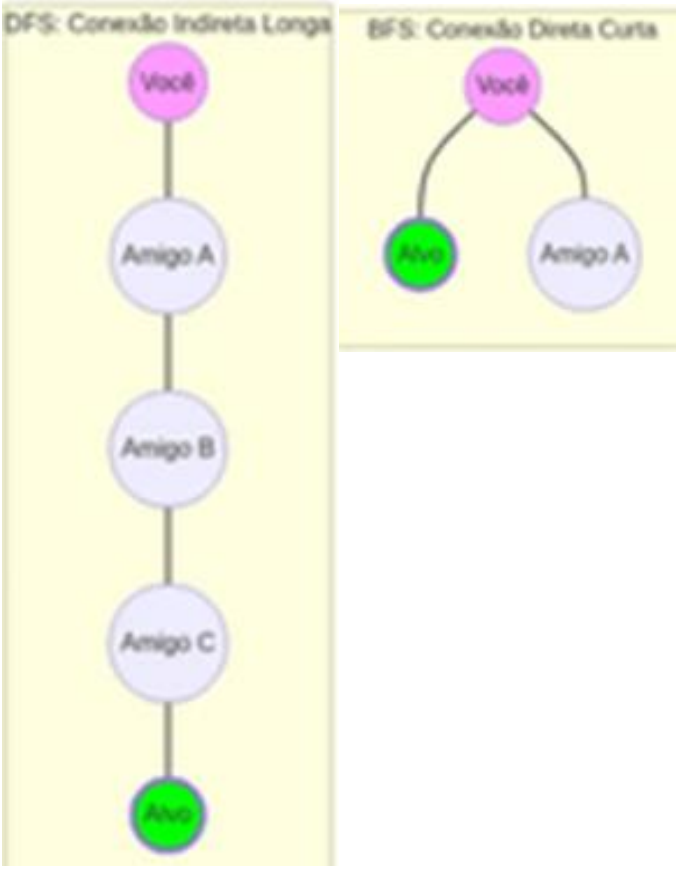
Busca em largura e profundidade

Exemplo 2: Conexões nas redes sociais

Cenário: encontrar pessoas conectadas a você

Comportamento DFS : A DFS pode ligar você a um vizinho através de 50 pessoas em outro continente, ignorando atalhos óbvios.

Comportamento BFS: A BFS garante que você veja "Amigo em Comum" ou "Conexão de 2º Grau" antes de qualquer conexão mais distante



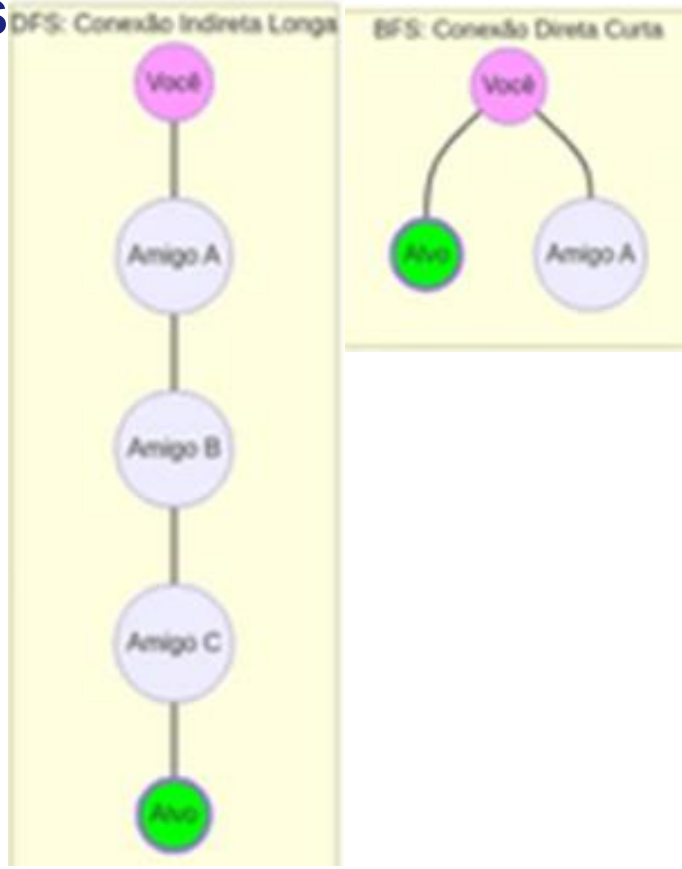
Busca em largura e profundidade

Exemplo 2: Conexões nas redes sociais

BFS prioriza as pessoas mais próximas de você socialmente, essencial para recomendação e busca.

BFS encontra conexões próximas muito mais rápido que a DFS, economizando processamento em larga escala.

Utilizado por: LinkedIn, Facebook e sistemas de recomendação modernos.



Exercícios

1. Escreva o pseudocódigo da busca em largura e da busca em profundidade
2. Nos slides anteriores apresentamos dois exemplos no qual a busca em largura é mais vantajosa que a busca em profundidade. Apresente dois exemplos nos quais a busca em profundidade seria mais adequada que a busca em largura.

Busca em profundidade limitada

- Para resolver o problema de busca em profundidade em árvores infinitas, um limite L restringe a busca, ou seja, nós na profundidade L são tratados como se não tivessem sucessores.
- Resolve caminhos infinitos, porém adiciona mais incompleteza, pois é difícil acharmos um límite adequado.

Busca em profundidade limitada

- Completa? Não; a solução pode estar além do limite.
- Tempo? $O(b^l)$
- Espaco? $O(b^l)$
- Ótima? Não

Busca em Aprofundamento Iterativo

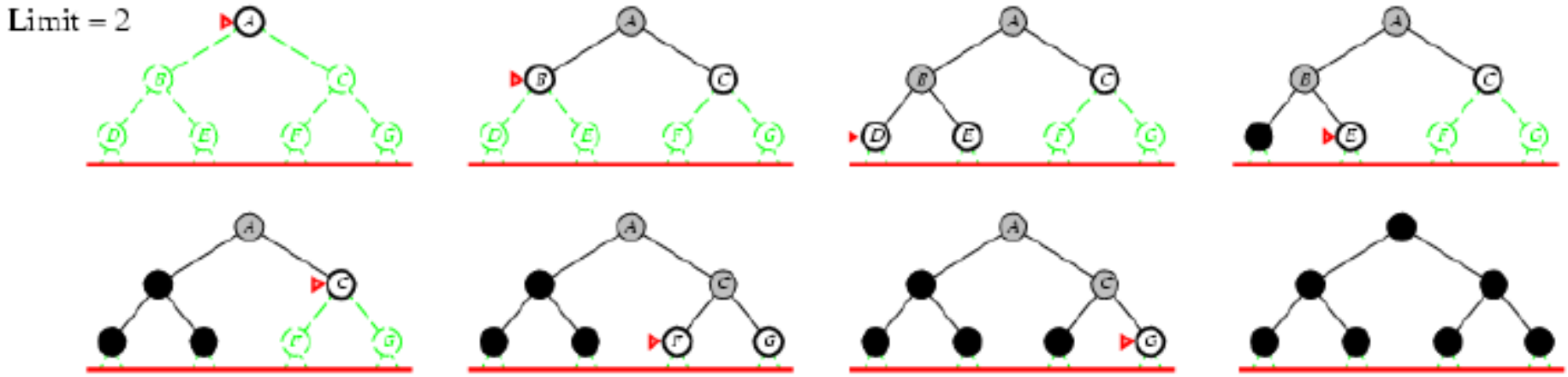
- Resolve o problema da escolha de um bom valor de l testando todos os valores: primeiro 0, depois 1, depois 2, e assim por diante.
- Combina muitos dos benefícios da busca em profundidade e da busca em largura.

Busca em Aprofundamento Iterativo

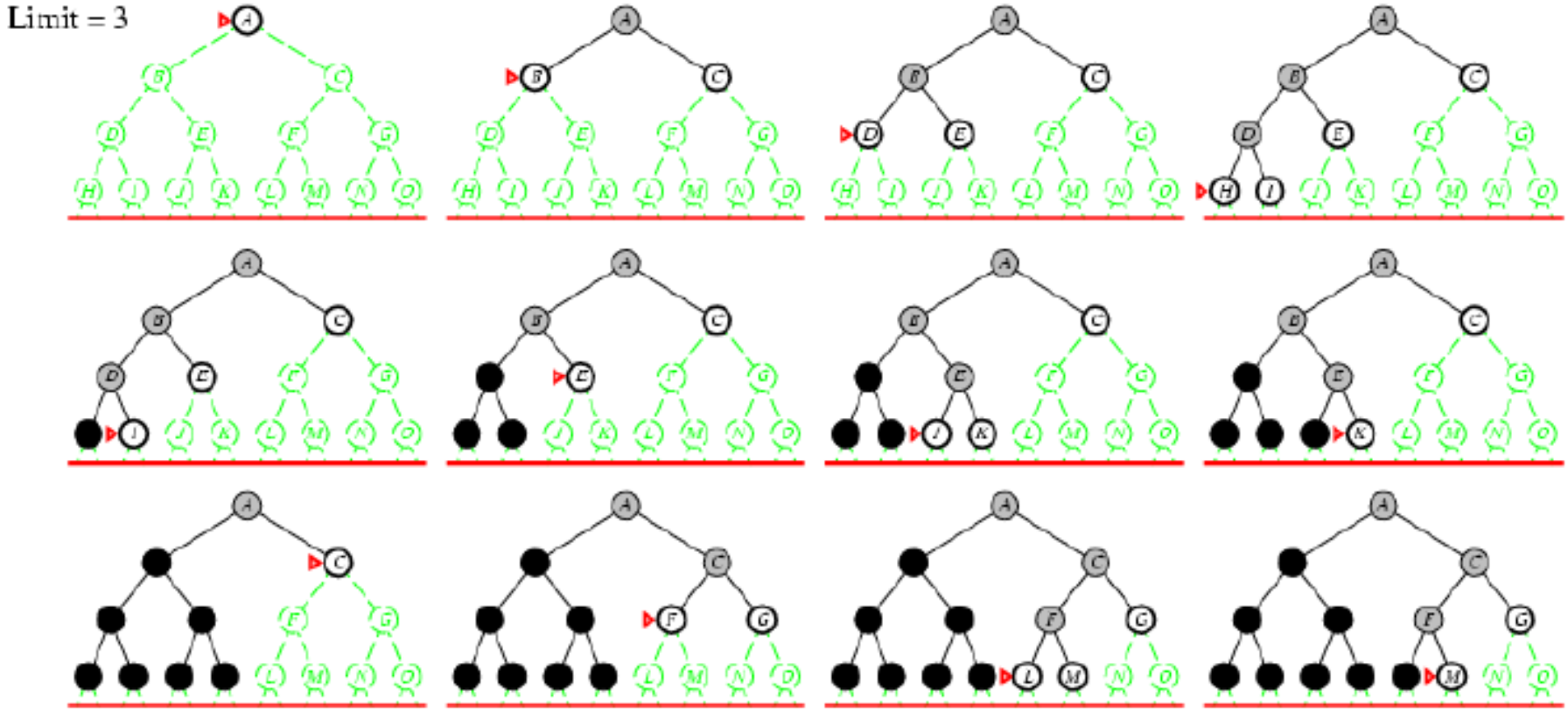
Limit = 0

Limit = 1

Busca em Aprofundamento Iterativo



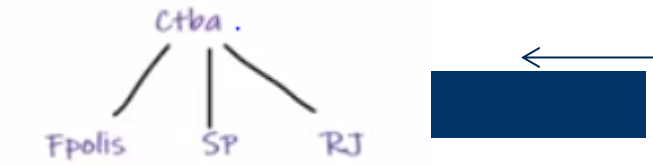
Busca em Aprofundamento Iterativo



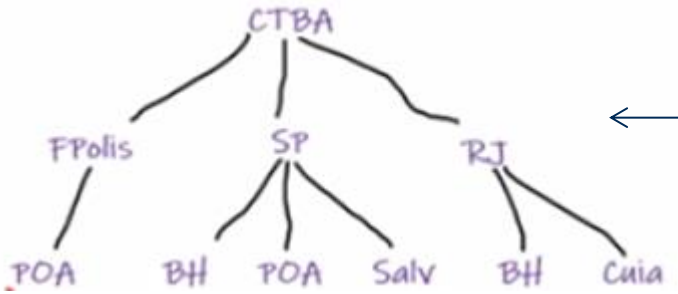
Busca em Aprofundamento Iterativo

- Ótima?
- Completa?
- Tempo?
- Espaço?

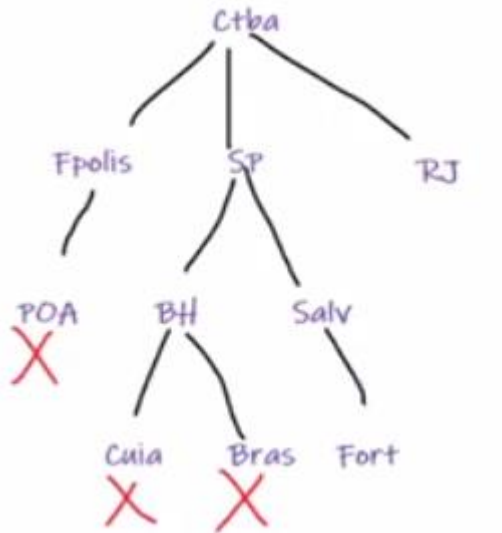
Busca em Aprofundamento Iterativo



Nível 1



Nível 2



Nível 3

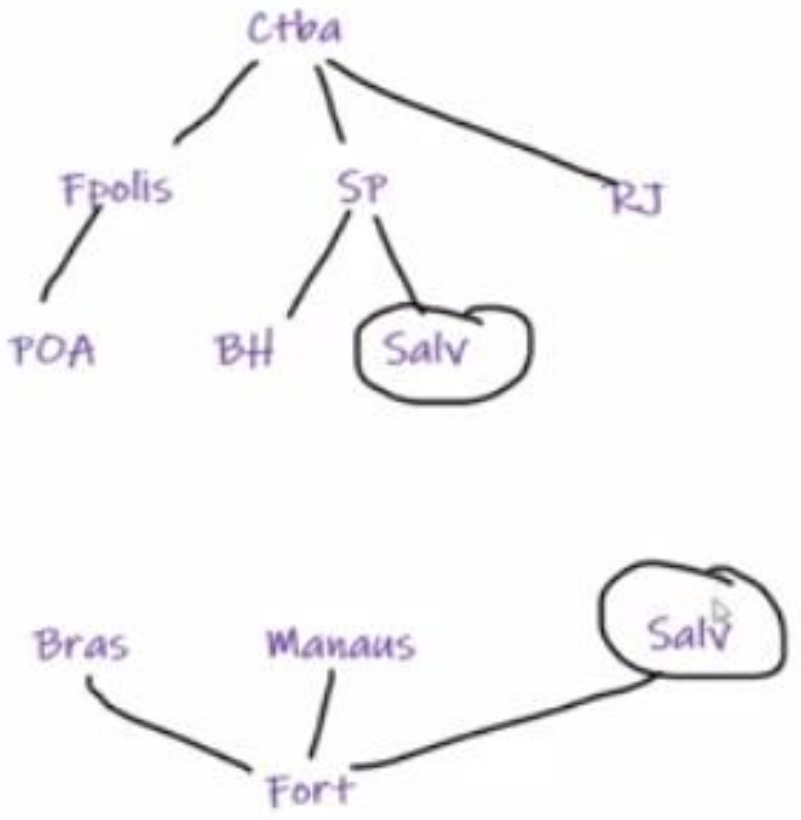
Busca Bidirecional

- Realiza duas buscas simultâneas – uma direta, a partir do estado inicial, e a outra inversa, a partir do estado meta, esperando que as duas buscas se encontrem em um ponto intermediário.

Busca Bidirecional

- Realize a busca bidirecional de Curitiba a Fortaleza

Busca Bidirecional



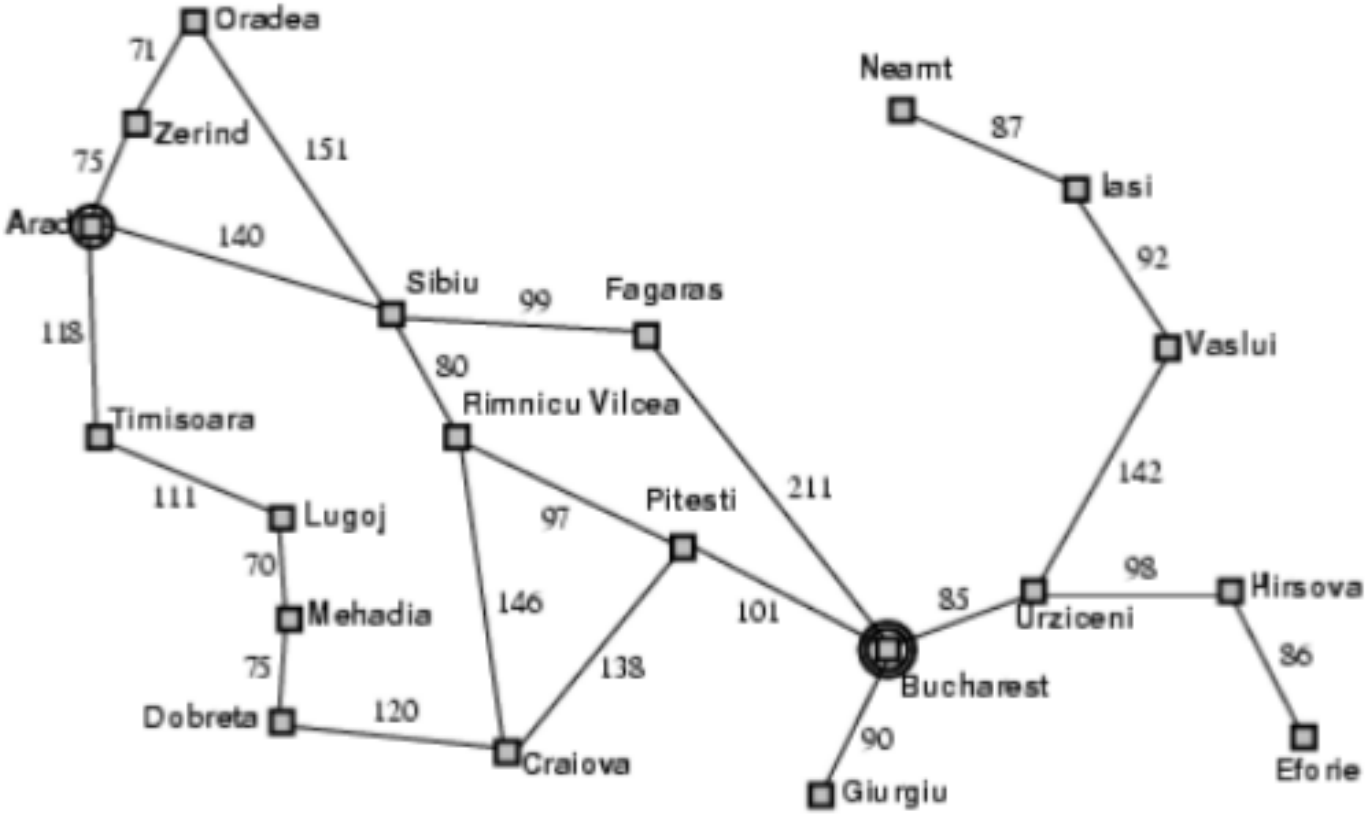
Busca Bidirecional

- Ótima?
- Completa?
- Tempo?
- Espaço?

Busca de custo uniforme

- Expande o nó n com custo de caminho g(n) mais baixo.
- Os nós da borda são armazenados em uma fila de prioridades (Heap).
- O teste de objetivo é aplicado quando o nó é selecionado para a expansão.
- Aplicar busca de custo uniforme para achar o caminho mais curto entre Sibiu e Bucareste.

Busca de custo uniforme



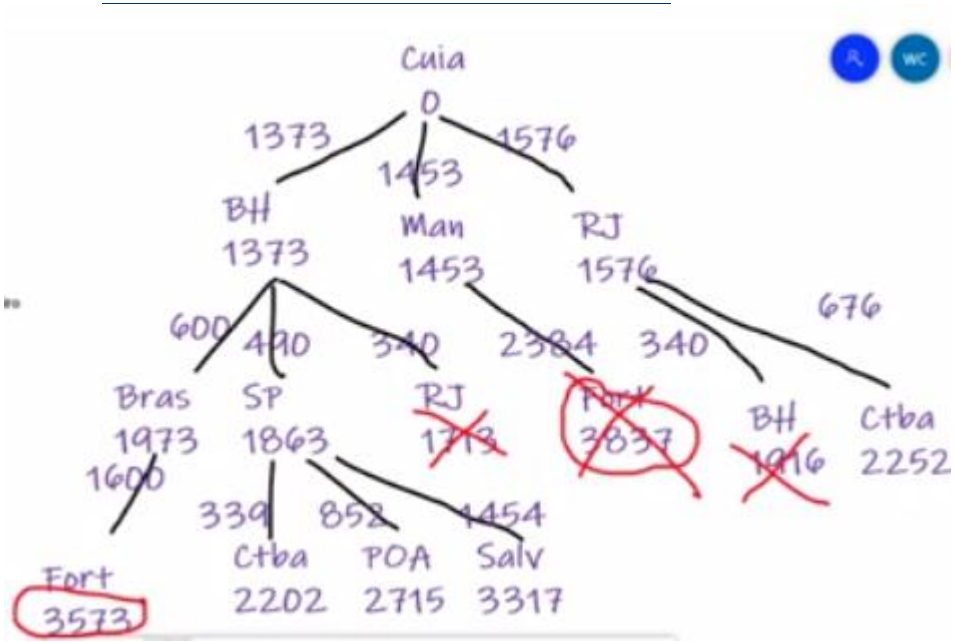
Busca de custo uniforme

- Ótima?
- Completa?
- Tempo?
- Espaço?

Outro exemplo

- Realize a busca de custo uniforme de Cuiaba a Fortaleza

Outro exemplo



A busca não finaliza aqui, precisamos expandir todos os nós restantes até que apareça um nó destino com custo inferior ao atual ou até que todos os nós tenham custo maior que o custo do nó destino atual.

Busca de custo uniforme

- Escreva o pseudocódigo da busca de custo uniforme.