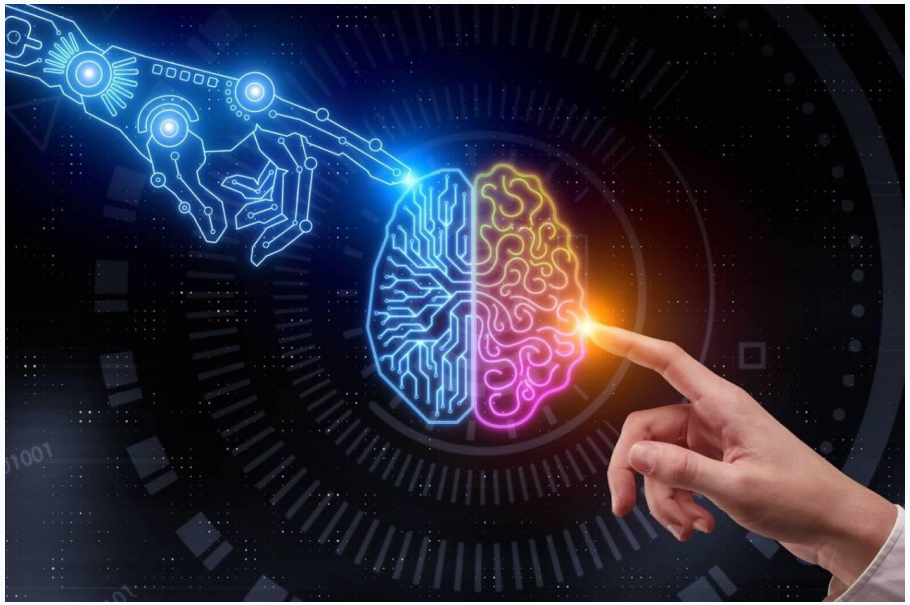


Inteligência Artificial

Profº - Dr. Thales Levi Azevedo Valente

thales.l.a.valente@gmail.com.br

Governo Federal

Grupo da turma 2024.2

https://chat.whatsapp.com/JFB6CgOI7lMCoYmoIKEK62

Sejam Bem-vindos !

Os celulares devem ficar no silencioso ou desligados

Pode ser utilizado apenas em caso de emergência

Boa tarde/noite, por favor e com licença DEVEM ser usados

Educação é essencial

Objetivos de hoje

Apresentar os principais conceitos de busca sem informação;

Ao final da aula, os alunos serão capazes de ter uma visão geral do funcionamento dos principais algoritmos de busca sem informação.

Relembrando a aula passada

- Um problema é definido por 4 itens:
  - ✓ **1. Estado inicial ex.: "em Arad"**
  - ✓ **2. Ações ou função sucessora S(x) = conjunto de pares estado-ação**
    - ✓ **– ex.: S(Arad) = {<Arad → Zerind, Zerind>, … }**
  - ✓ **3. Teste de objetivo, pode ser:**
    - ✓ **– explícito, ex.: x = "em Bucharest"**
    - ✓ **– implícito, ex.: Cheque-mate(x)**
  - ✓ **4. Custo de caminho (aditivo):**
    - ✓ **– ex.: soma das distâncias, número de ações executadas, etc.**
    - ✓ **– c (x, a, y) é o custo do passo, que deve ser sempre ≥ 0**

- Uma solução é uma sequência de ações que levam do estado inicial para o estado objetivo.
- Uma solução ótima é uma solução com o menor custo de caminho.

Relembrando a aula passada

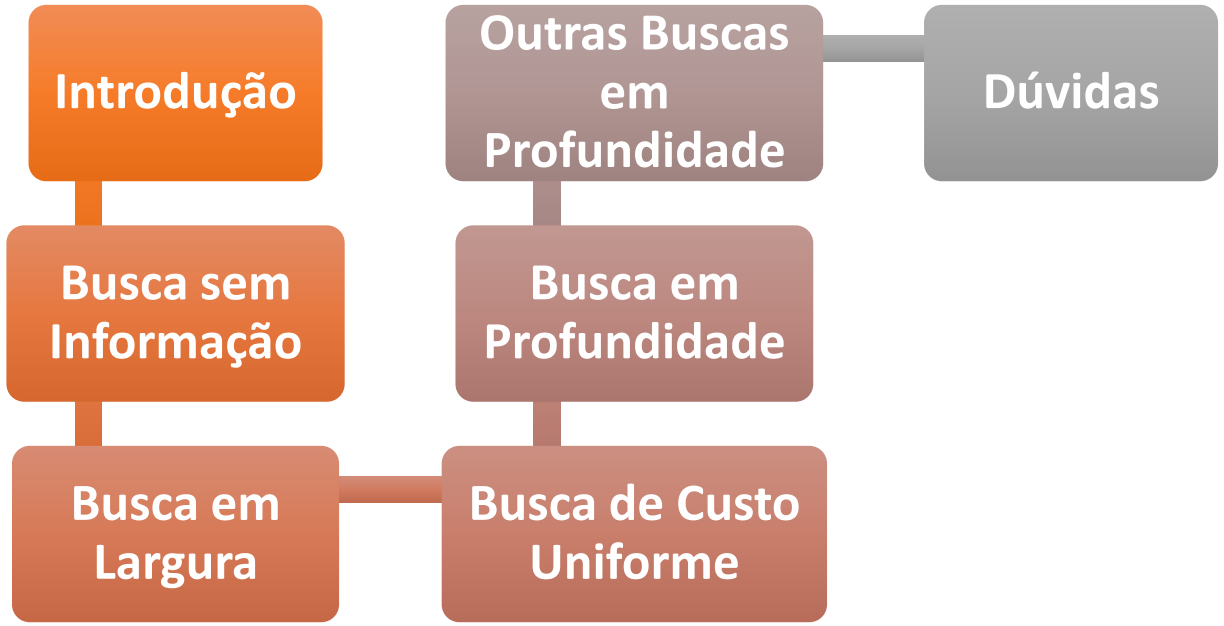
- **Uma estratégia de busca é definida pela escolha da ordem da expansão de nós**

- **Estratégias são avaliadas de acordo com os seguintes critérios**
  - ✓ **Completeza**: o algoritmo sempre encontra a solução se ela existe?
  - ✓ **Complexidade de tempo**: número de nós gerados
  - ✓ **Complexidade de espaço**: número máximo de nós na memória
  - ✓ **Otimização**: a estratégia encontra a solução ótima?

Relembrando a aula passada

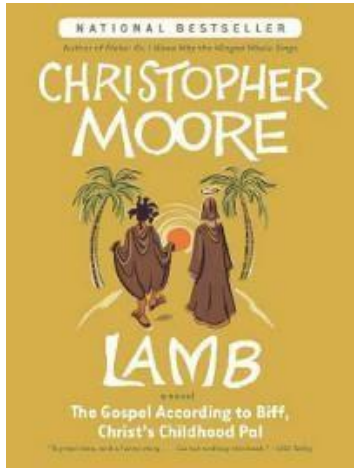
- **Complexidade de tempo e espaço são medidas em termos de**
- **Estratégias são avaliadas de acordo com os seguintes critérios**
  - ✓ **b**: máximo fator de ramificação da árvore (número máximo de sucessores de qualquer nó)
  - ✓ **d**: profundidade do nó objetivo menos profundo
  - ✓ **m**: o comprimento máximo de qualquer caminho no espaço de estados (pode ser ∞)

Roteiro: Busca sem Informação



Introdução

“Children see magic because they look for it.”
(Christopher Moore)



CHRISTOPHER MOORE
American Writer of Comic Fantasy.
Link:
https://www.chrismoore.com/biography/

Introdução

- **Reflexão sobre a forma como nossa perspectiva influencia o que percebemos no mundo ao nosso redor**.
  - ✓ As crianças têm uma curiosidade natural e disposição para explorar e buscar o extraordinário.
- **Idéia pode ser relacionada aos algoritmos de busca no contexto de inteligência artificial e ciência da computação da seguinte maneira**.
  - ✓ Busca como Exploração Intencional
  - ✓ Curiosidade e Estratégias de Busca
  - ✓ Enxergar Soluções Onde Não Parecem Existir

Introdução

- **Reflexão sobre a forma como nossa perspectiva influencia o que percebemos no mundo ao nosso redor**.
  - ✓ As crianças têm uma curiosidade natural e disposição para explorar e buscar o extraordinário.
- **Idéia pode ser relacionada aos algoritmos de busca no contexto de inteligência artificial e ciência da computação da seguinte maneira**.
  - ✓ Busca como Exploração Intencional
  - ✓ Enxergar Soluções Onde Não Parecem Existir
  - ✓ Curiosidade e Estratégias de Busca

Introdução

- **Busca como Exploração Intencional.**
  - ✓ Assim como as crianças veem magia porque estão dispostas a buscá-la, os algoritmos de busca encontram soluções porque são projetados para explorar o espaço de estados.
  - ✓ A eficiência e o sucesso de um algoritmo de busca dependem da maneira como ele "procura" a solução em um espaço de possibilidade.

- **Enxergar Soluções Onde Não Parecem Existir.**
  - ✓ Algumas abordagens, como algoritmos genéticos ou redes neurais, simulam a capacidade de encontrar padrões ou soluções que não são óbvias para os humanos, semelhante à capacidade infantil de ver "magia" onde outros não enxergam.

Introdução

- **Curiosidade e Estratégias de Busca.**
  - ✓ Busca Cega (como busca em largura ou profundidade):
    - ✓ Sem conhecimento prévio, o algoritmo explora todos os caminhos possíveis, semelhante à curiosidade infantil indiscriminada
  - ✓ Busca Heurística (como A*):
    - ✓ Aqui, o algoritmo "procura por magia" ao usar uma heurística para focar em caminhos mais promissores, tornando a busca mais direcionada e eficiente.
    - ✓ A heurística, nesse caso, é a "disposição para encontrar o extraordinário" no espaço de estados

Introdução

- **Estratégias de busca sem informação:**
  - ✓ Busca cega.
  - ✓ **Não** possui informação adicional sobre o estado, além daquelas fornecidas na definição do problema.
  - ✓ Geram sucessores e distinguem um estado objetivo de um estado não objetivo.
  - ✓ As estratégias de busca se distinguem pela ordem em que os nós são expandidos.

Estratégias de busca sem informação

- **Etapas básicas:**
  - ✓ Estado inicial;
  - ✓ Ações;
  - ✓ Teste de objetivos;
  - ✓ Uma função de custo.

Estratégias de busca sem informação

- **Tipos: se distinguem pela ordem em que os nós são expandidos**
  - ✓ Busca em largura (Breadth-first);
  - ✓ Busca de custo uniforme;
  - ✓ Busca em profundidade (Depth-first).

Busca em largura

- Também conhecido por breadth-first search (BFS) ou busca em amplitude.
- Examina, sistematicamente, todos os vértices de um grafo direcionado ou não-direcionado.
- Realiza uma busca exaustiva num grafo passando por todas as arestas e vértices.

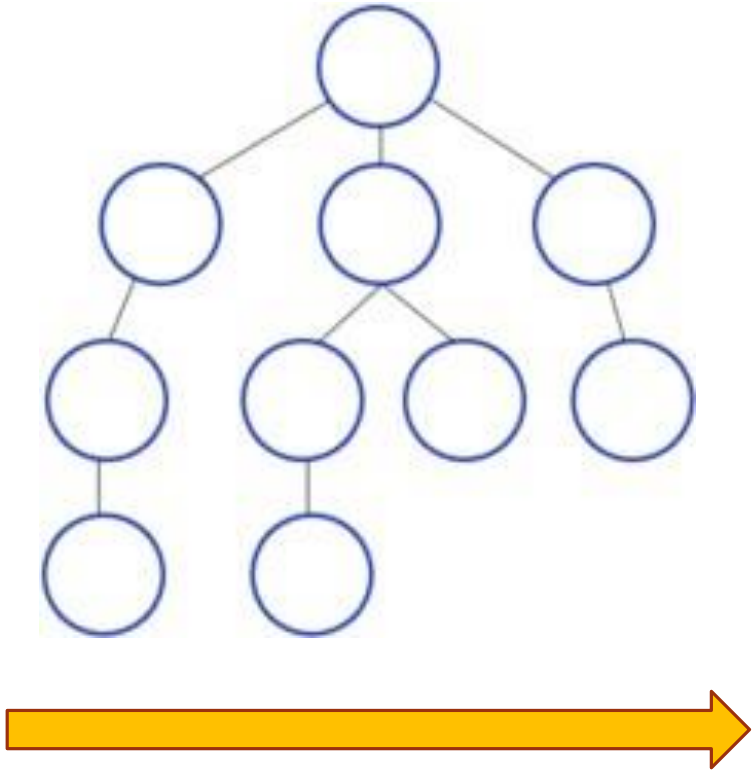
Busca em largura

- A busca começa por um vértice, digamos ***s***, especificado pelo usuário.
- O algoritmo visita ***s***, depois visita todos os vértices que estão à distância ***1 de s***, depois todos os vértices que estão à distância ***2 de s***, e assim por diante.
- Para implementar a ideia o algoritmo utiliza uma fila de vértices.
  - ✓ A "borda" é uma fila FIFO (first-in, first-out), isto é, novos itens entram no final.

Importante: a distância de um ***vértice s*** a um ***vértice t*** é o comprimento de um caminho mínimo de *s* a *t*.

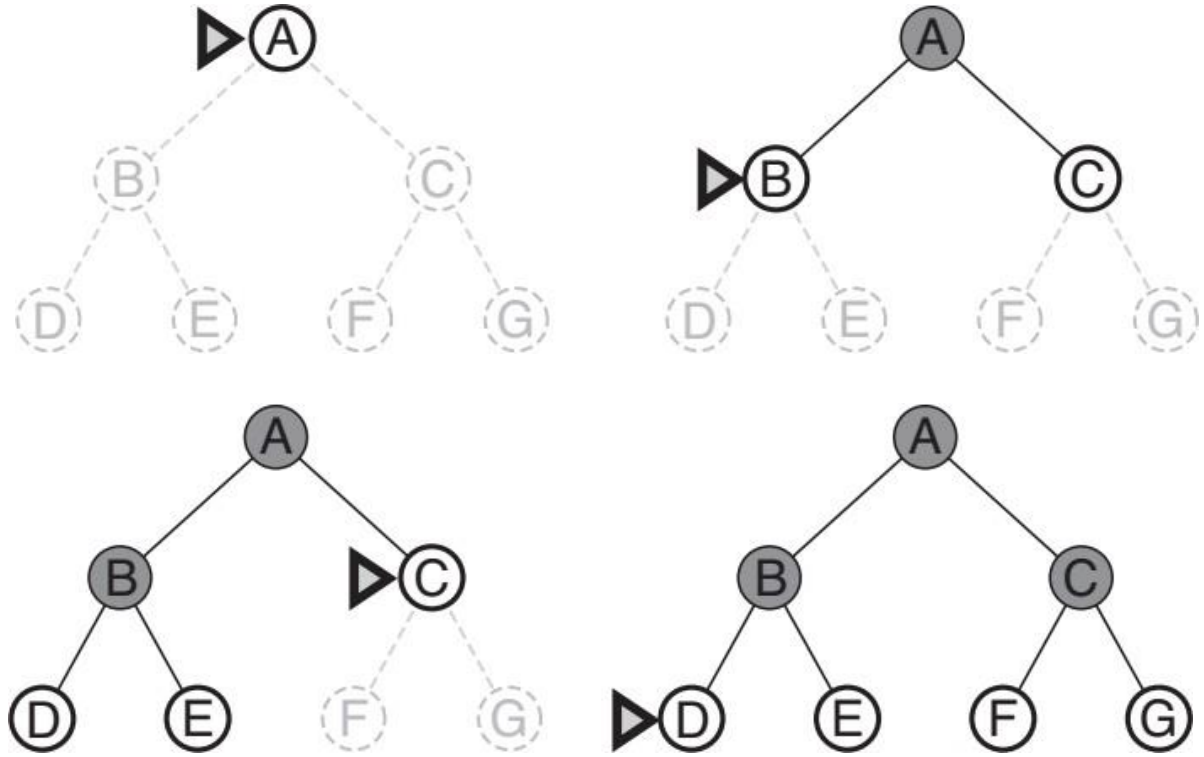
Busca em largura

- Exemplo ilustrativo:



Busca em largura

- Exemplo ilustrativo:



Busca em largura

- Completa? Sim (se b é finito)
- Tempo? $1+b+b^2+b^3+\ldots\ +b^d + b(b^d-1) = O(b^{d+1})$
- Espaço? $O(b^{d+1})$ (mantém todos os nós na memória)
- Ótima? Sim (se todas as ações tiverem o mesmo custo)

Busca em largura

- Esta estratégia só dá bons resultados quando a profundidade da árvore de busca é pequena.

Profundidade	Nós	Tempo	Memória
0	1	1 milissegundo	100 bytes
2	111	0.1 segundo	11 quilobytes
4	11111	11 segundos	1 megabytes
6	10^6	18 minutos	111 megabytes
8	10^8	31 horas	11 gigabytes
10	10^{10}	128 dias	1 terabyte
12	10^{12}	35 anos	111 terabytes
14	10^{14}	3500 anos	11111 terabytes

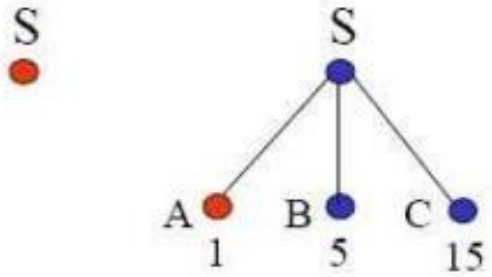
Busca de custo uniforme

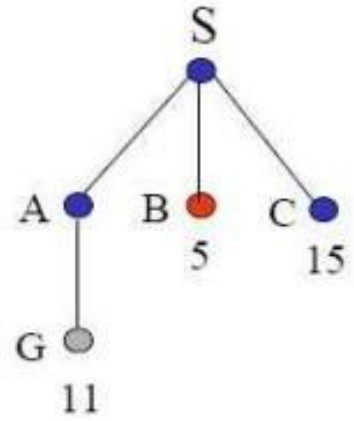
- Expande o nó não-expandido que tenha o caminho de custo mais baixo.
  - ✓ Equivalente a busca em extensão se os custos são todos iguais
- Implementação: fila ordenada pelo custo do caminho
- Completa? Sim, se o custo de cada passo $\geq \varepsilon$
- Tempo? número de nós com g $\leq$ custo da solução ótima
  - ✓ $O(b^{\lceil C^*/\varepsilon \rceil})$ onde C^* é o custo da solução ótima
- Espaço? de nós com $g \leq$ custo da solução ótima, $O(b^{\lceil C^*/\varepsilon \rceil})$
- Ótima? Sim pois os nós são expandidos em ordem crescente de custo total.

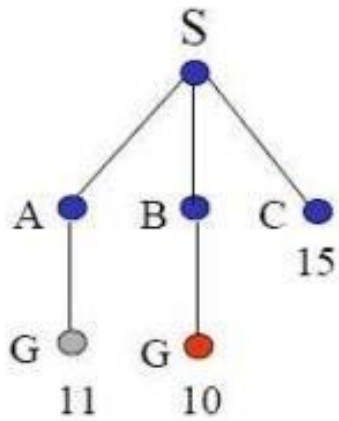
Busca em custo uniforme

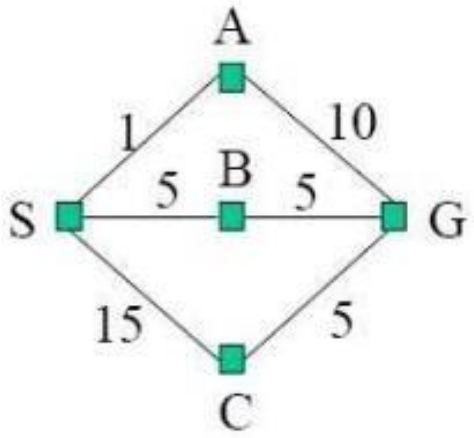
- A estratégia encontra a melhor solução se:
  - Se o custo do caminho nunca diminuir enquanto o mesmo for percorrido.
  - Esta restrição é obedecida se o custo do caminho é a soma dos custos dos operadores que geram o caminho e nenhum operador tem custo negativo.

Busca em custo uniforme



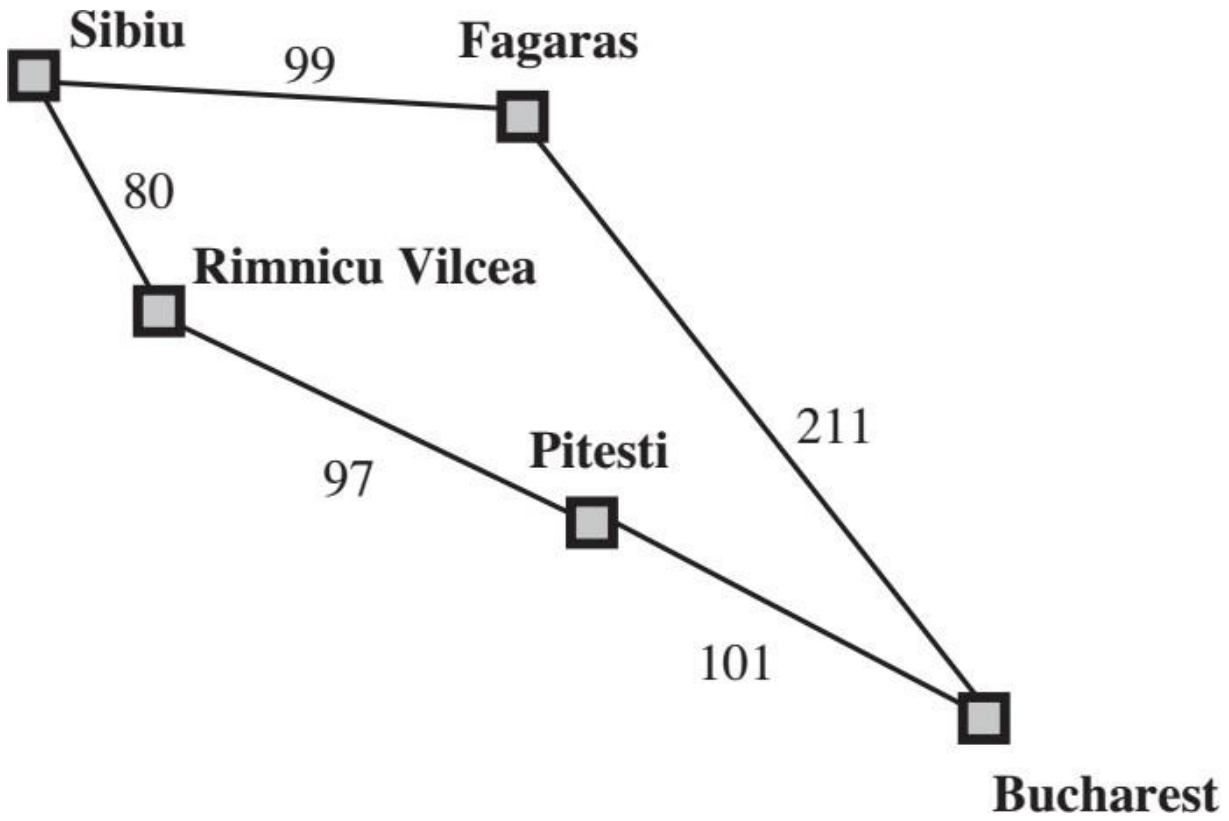






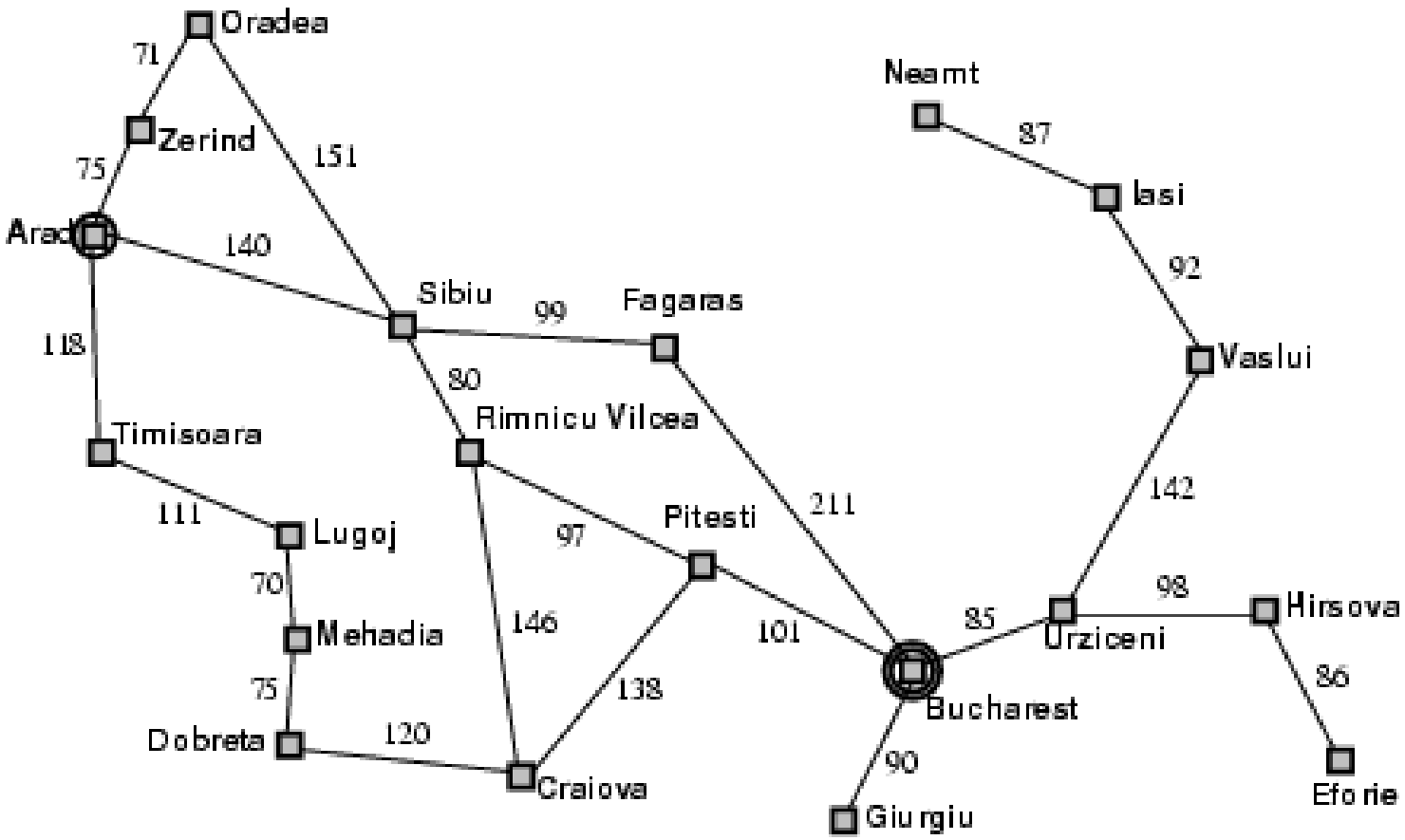
Exercício

- Aplicar busca de custo uniforme para achar o caminho mais curto entre Sibiu e Bucareste



Exercício

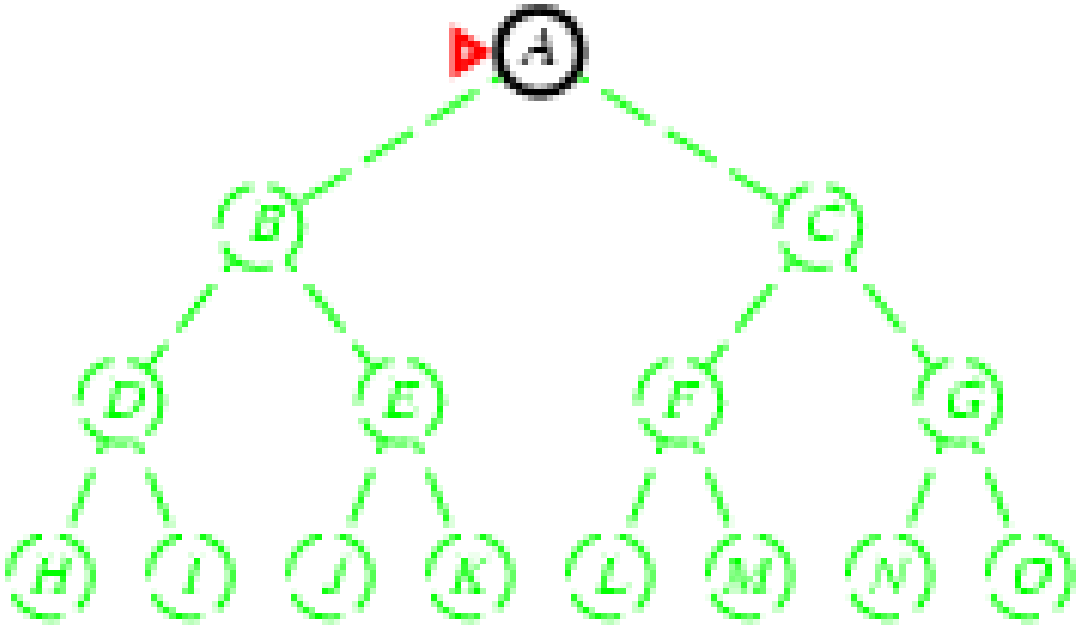
- Aplicar busca de custo uniforme para achar o caminho mais curto entre Arad e Bucareste



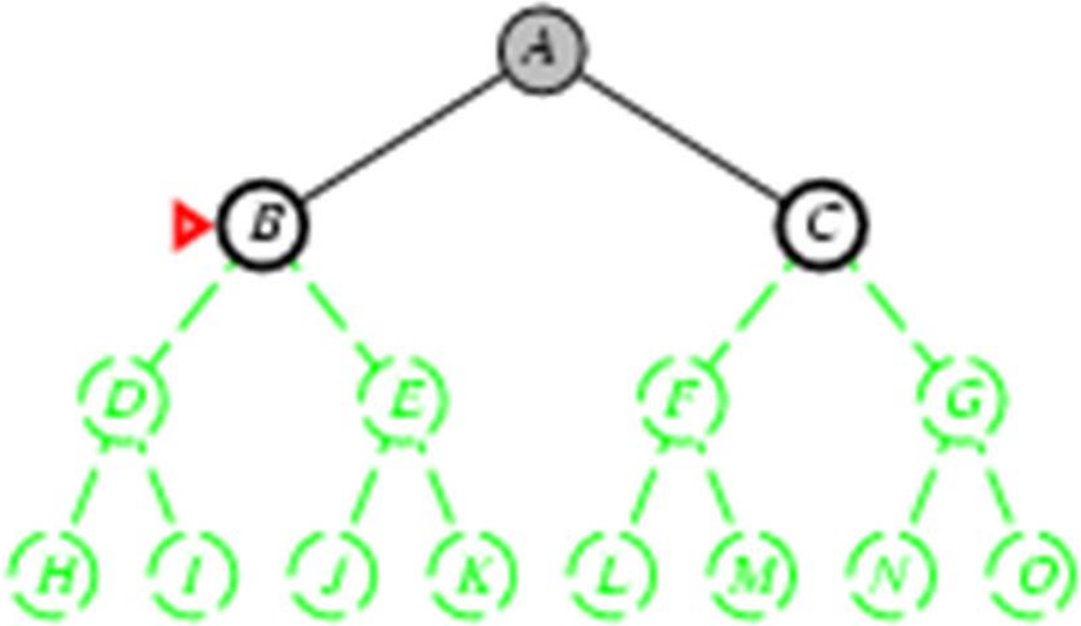
Busca em profundidade

- Também conhecido por depth-first seacrh (DFS).
- Expande sempre um dos nós de maior profundidade na árvore.
- Quando a busca encontra um nó que não pode ser expandido, retorna e expande os nós que estão pendentes.
- Para implementar a ideia o algoritmo utiliza uma pilha.

Busca em Profundidade



Busca em Profundidade

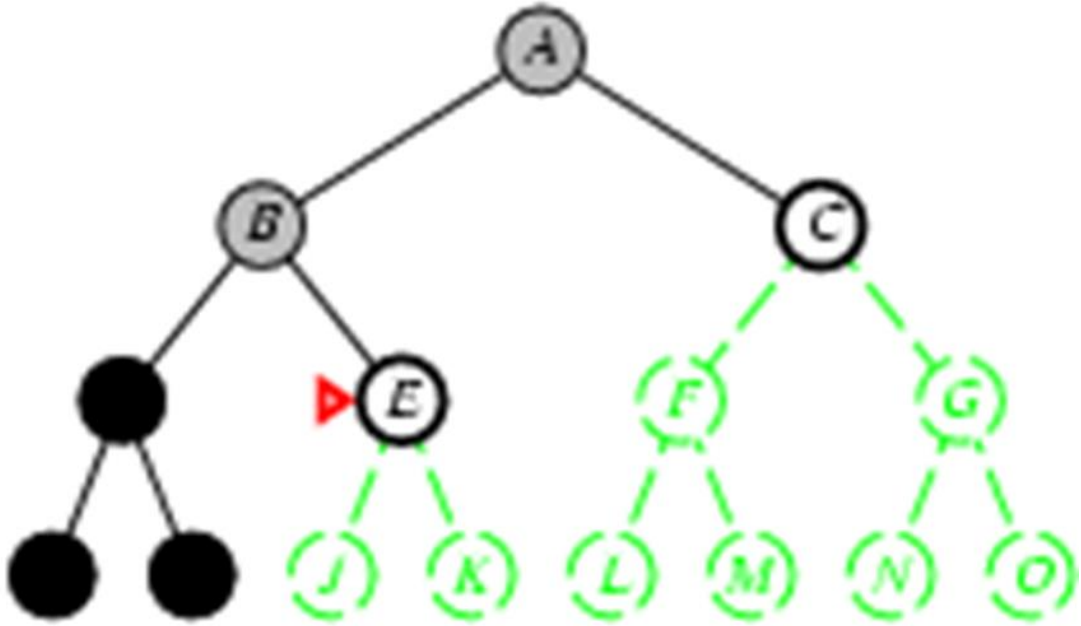


Busca em Profundidade

Busca em Profundidade

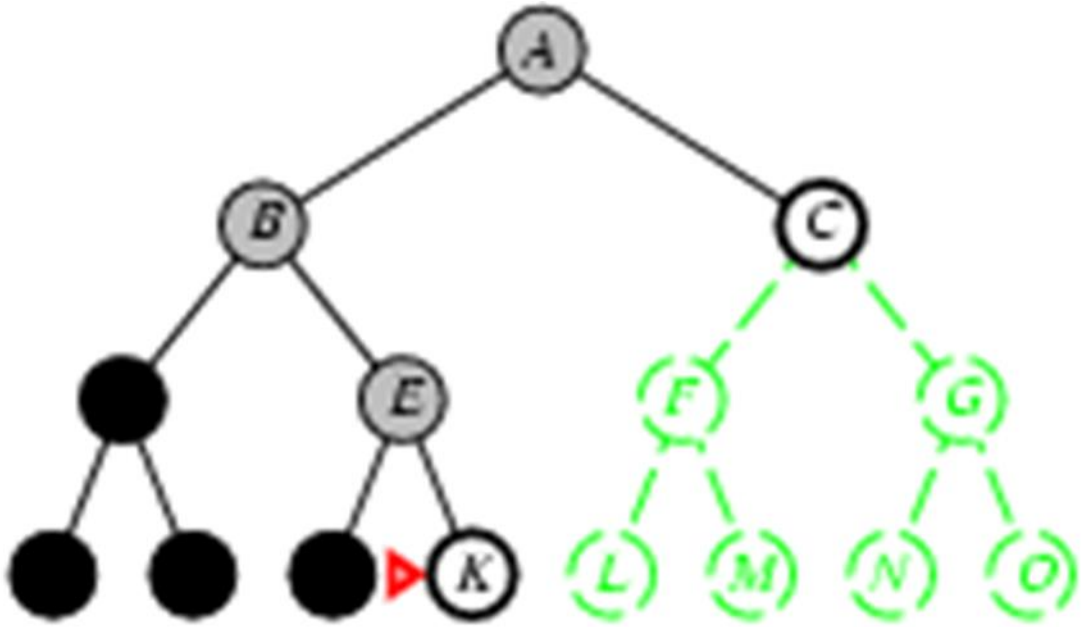
Busca em Profundidade

Busca em Profundidade

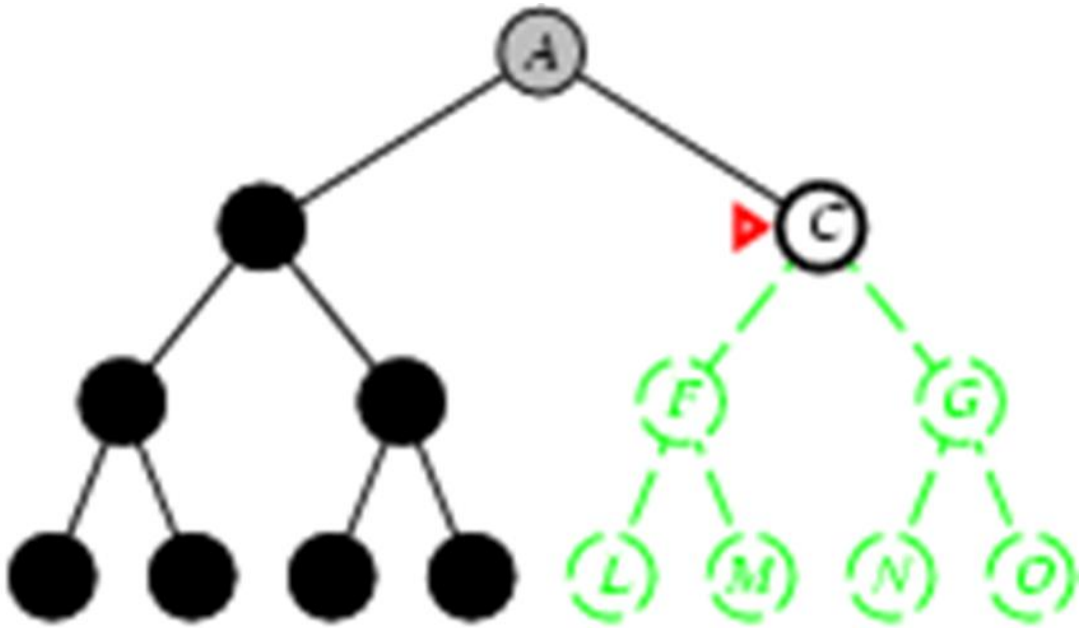


Busca em Profundidade

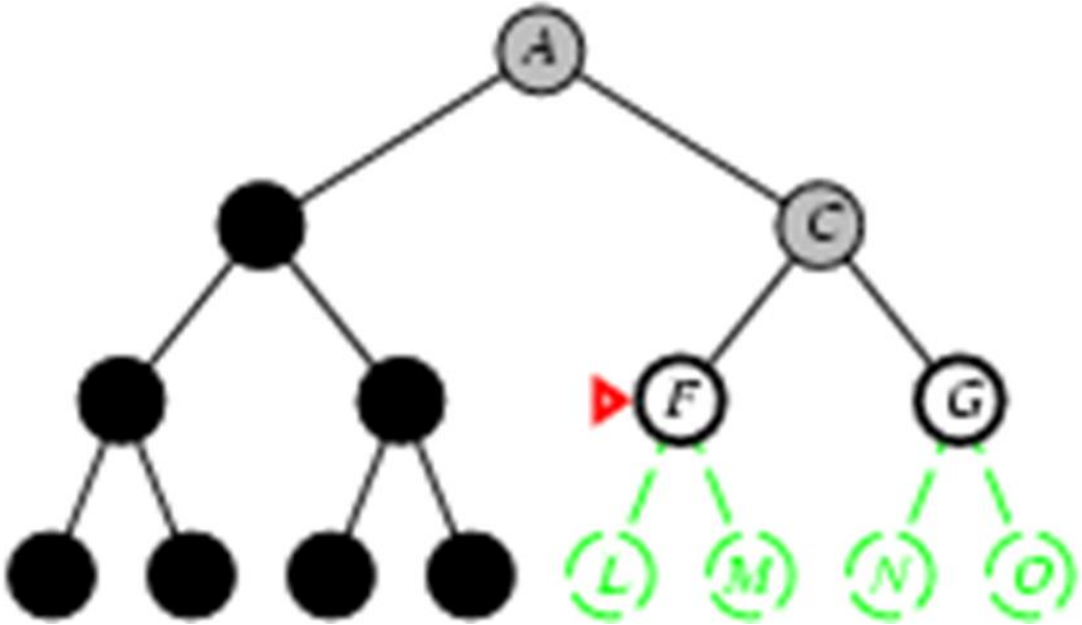
Busca em Profundidade



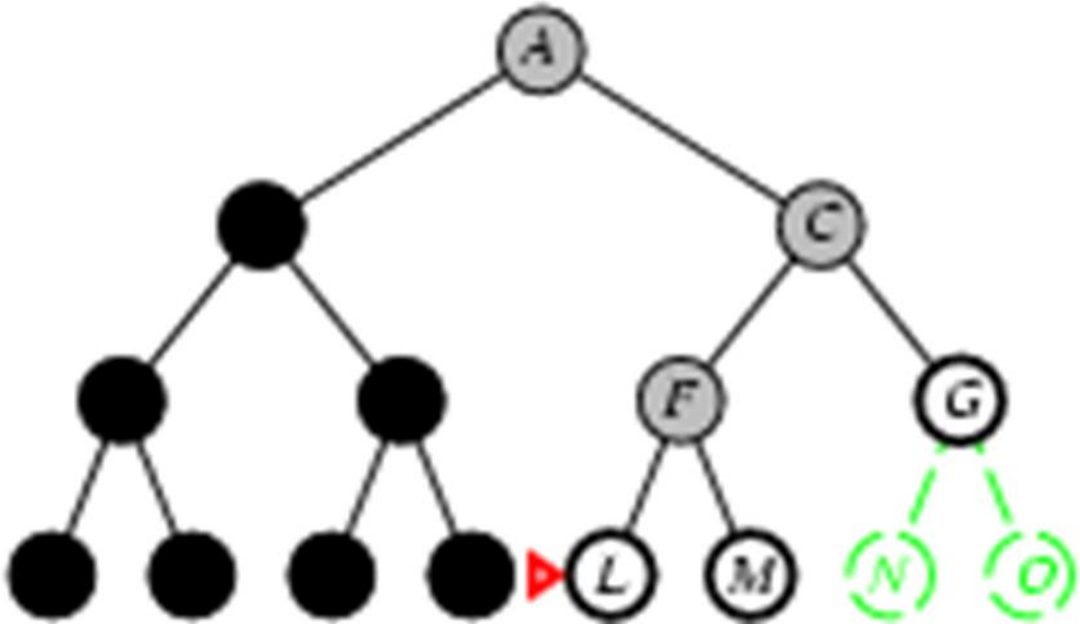
Busca em Profundidade



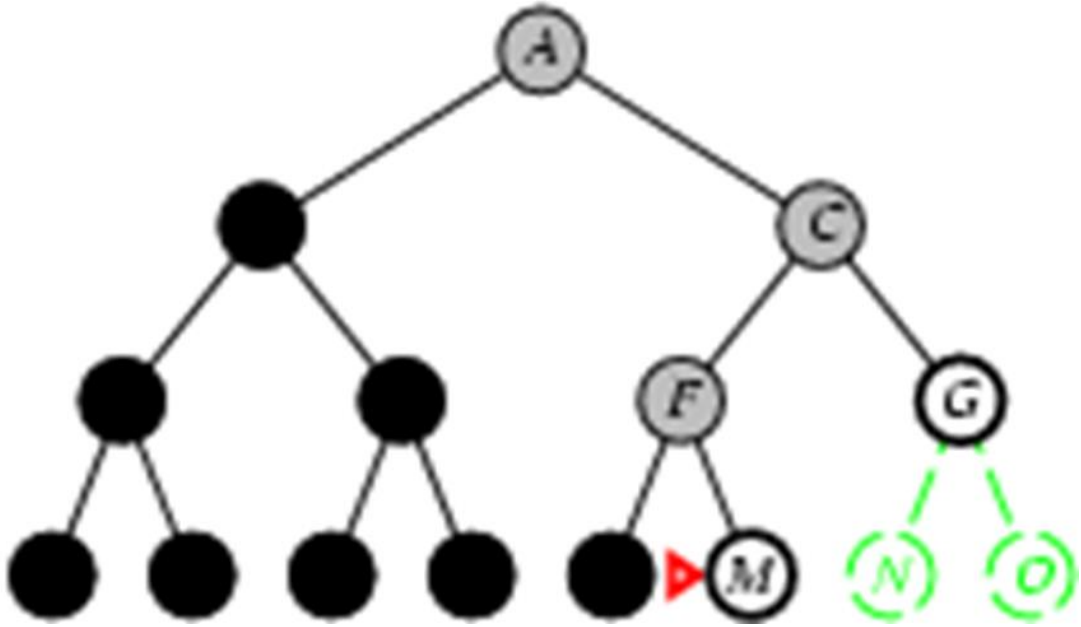
Busca em Profundidade



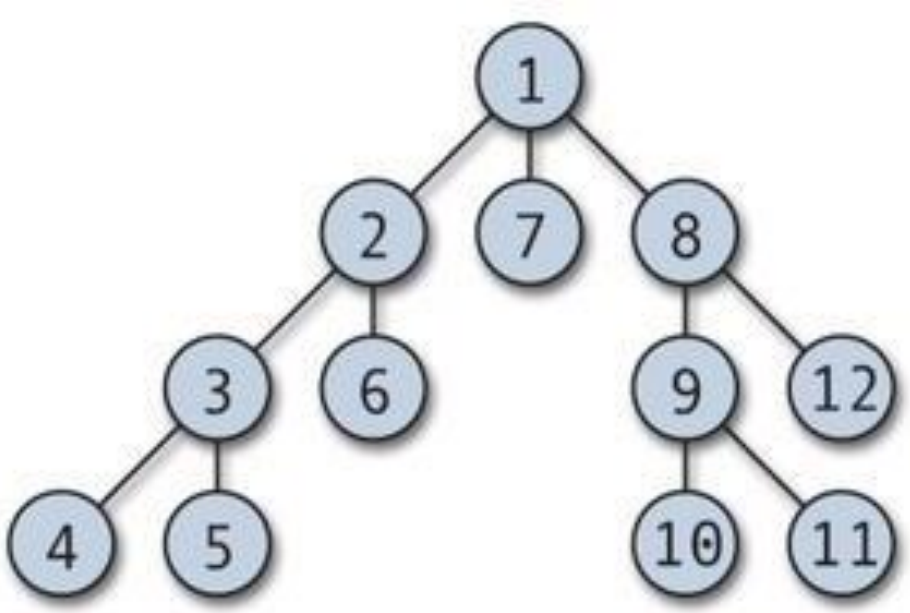
Busca em Profundidade

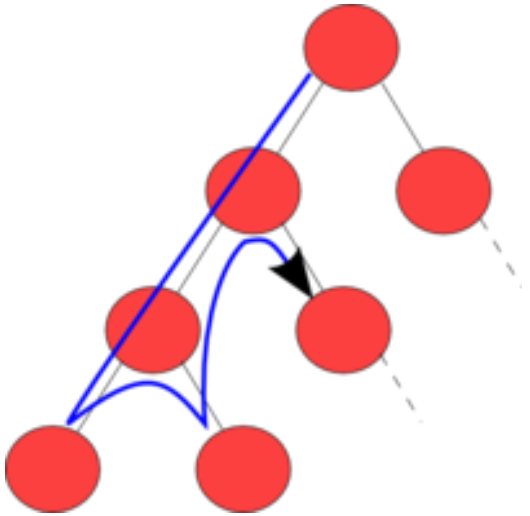


Busca em Profundidade



Busca em Profundidade (resumo)





Busca em profundidade

- **Vantagens:**
  - ✓ Econômico em memória.
  - ✓ É necessário armazenar apenas um único caminho da raiz até um nó-folha.
  - ✓ Problemas com muitas soluções: a busca em profundidade "tende" a ser melhor que a busca em largura.
- Desvantagens:
  - ✓ A estratégia pode forçar uma busca em um ramo que não há soluções.
  - ✓ Deve ser evitada para árvores de busca com profundidades muito grandes ou infinitas.

Busca em Profundidade Limitada

- Completa? Não: falha em espaços com profundidade infinita, espaços com loops. Se modificada para evitar estados repetidos é completa para espaços finitos.
- Tempo? $O(b^m)$: péssimo quando m é muito maior que d. Mas se há muitas soluções pode ser mais eficiente que a busca em extensão
- Espaço? $O(bm)$: *i.e., espaço linear!*
  - 118 kilobytes ao invés de 10 petabytes para busca com b=10, d=m=12
- Ótima? Não

Busca em Profundidade Limitada

- Busca em profundidade com limite de profundidade l, isto é, nós com profundidade l não tem sucessores
- Completa? Não; a solução pode estar além do limite.
- Tempo? $O(b^l)$
- Espaço? O(bl): i.e., espaço linear!
- Ótima? Não

Busca de Aprofundamento Iterativo

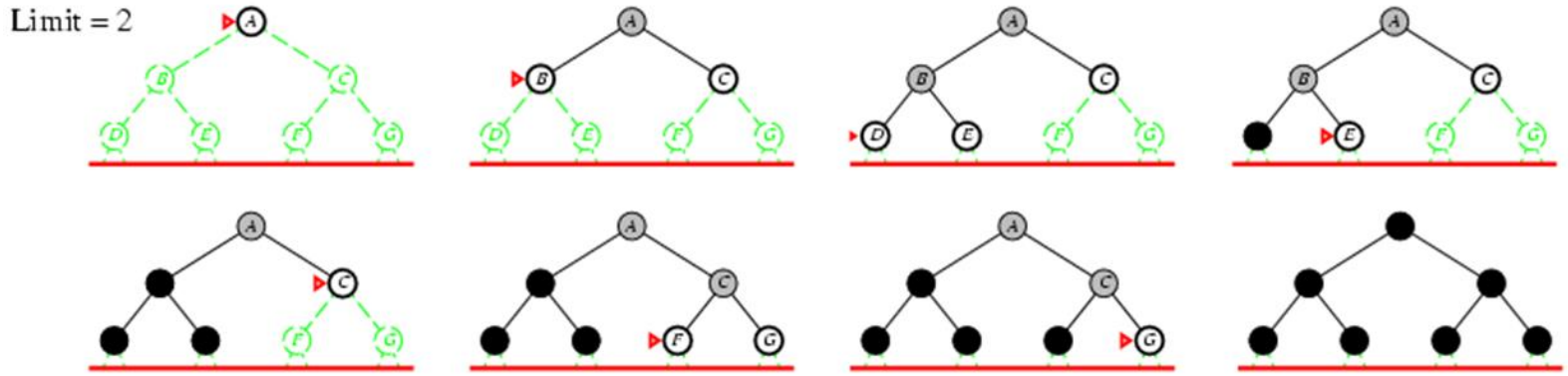
- Estratégia utilizada em conjunto com busca em profundidade para encontrar o melhor limite l
  - ✓ Aumentar gradualmente l até encontrar um estado objetivo
- Isto ocorre quando a profundidade alcançar d
  - ✓ (profundidade do objetivo mais raso)

Busca de Aprofundamento Iterativo em Profundidade l =0

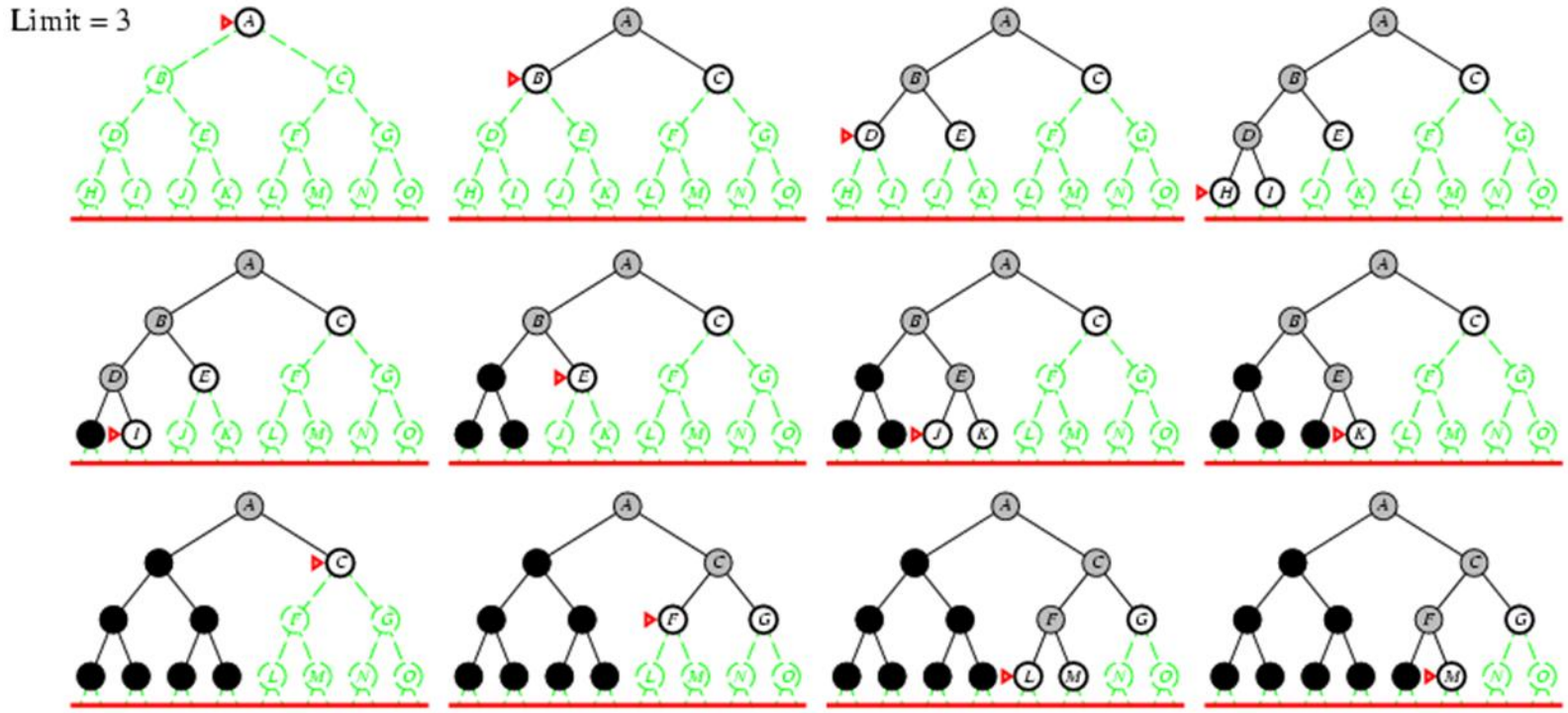
Limit = 0

Busca de Aprofundamento Iterativo em Profundidade l =1

Busca de Aprofundamento Iterativo em Profundidade l =2



Busca de Aprofundamento Iterativo em Profundidade l =3



Propriedades da busca de aprofundamento iterativo

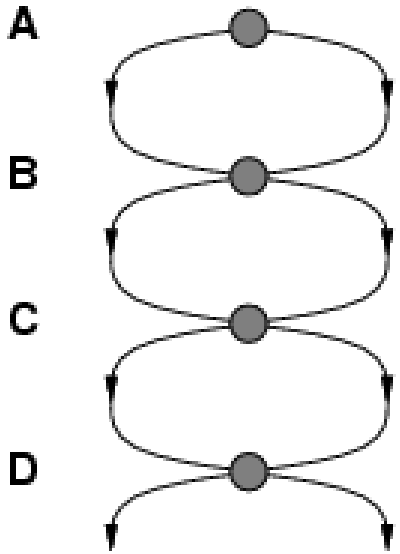
- Completa? Sim.
- Tempo? $(d+1)b^0 + d\,b^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$
- Espaço? $O(bd)$: *i.e., espaço linear!*
- Ótima? Sim, se custo de passo = 1

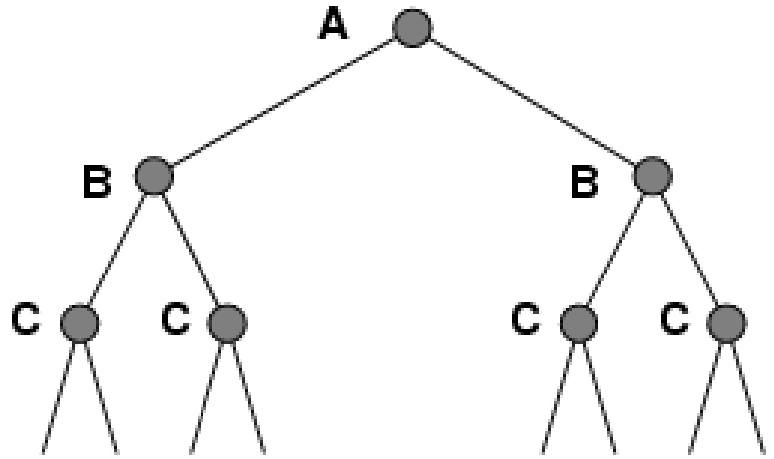
Resumo dos Algoritmos

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes	Yes	No	No	Yes

Detecção de estados repetidos

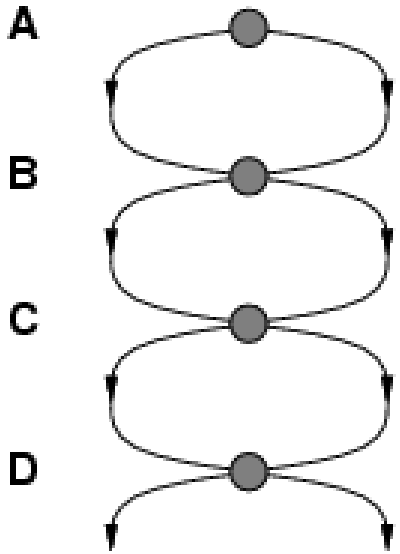
- O processo de busca pode perder tempo expandindo nós já explorados antes
  - ✓ Estados repetidos podem levar a loops infinitos
  - ✓ Estados repetidos podem transformar um problema linear em um problema exponencial

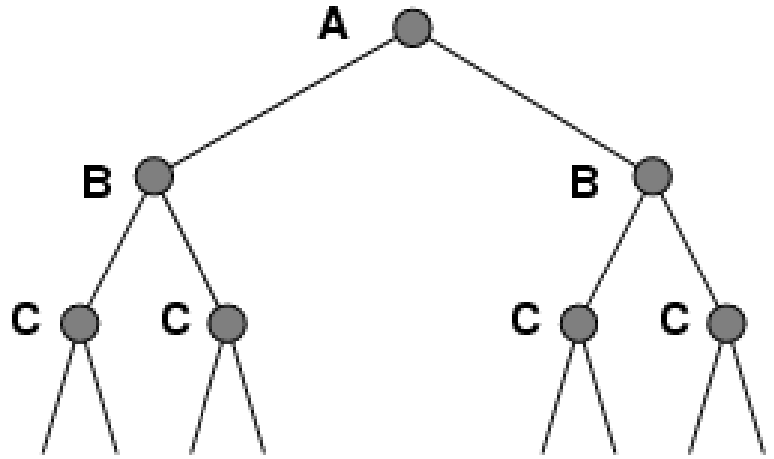




Detecção de estados repetidos

- Comparar os nós prestes a serem expandidos com nós já visitados
  - ✓ Se o nó já tiver sido visitado, será descartado.
  - ✓ Lista “closed” (fechado) armazena nós já visitados.
  - ✓ A busca percorre um grafo e não uma árvore.





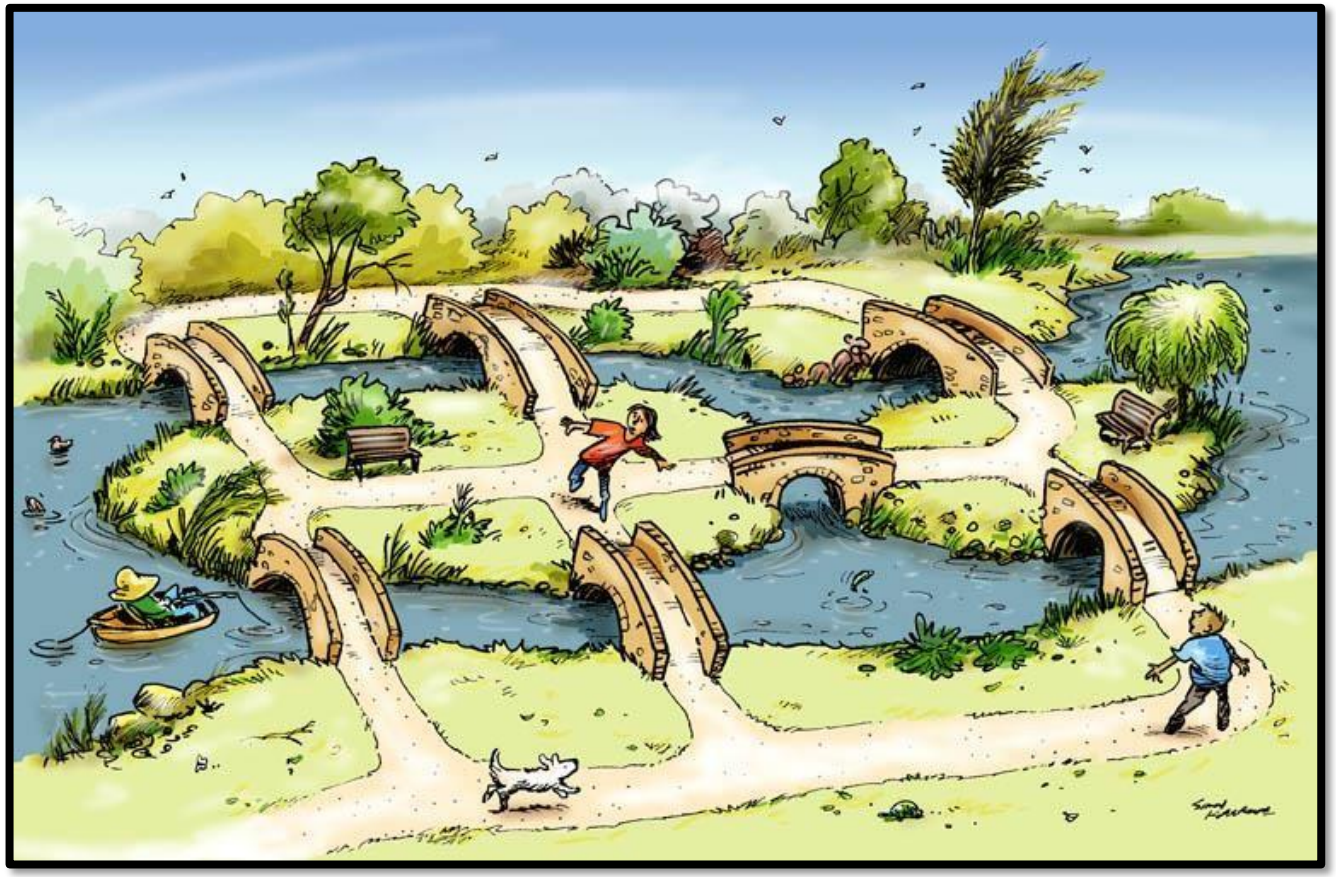
Conclusão

- A formulação de problemas usualmente requer a abstração de detalhes do mundo real para que seja definido um espaço de estados que possa ser explorado através de algoritmos de busca.

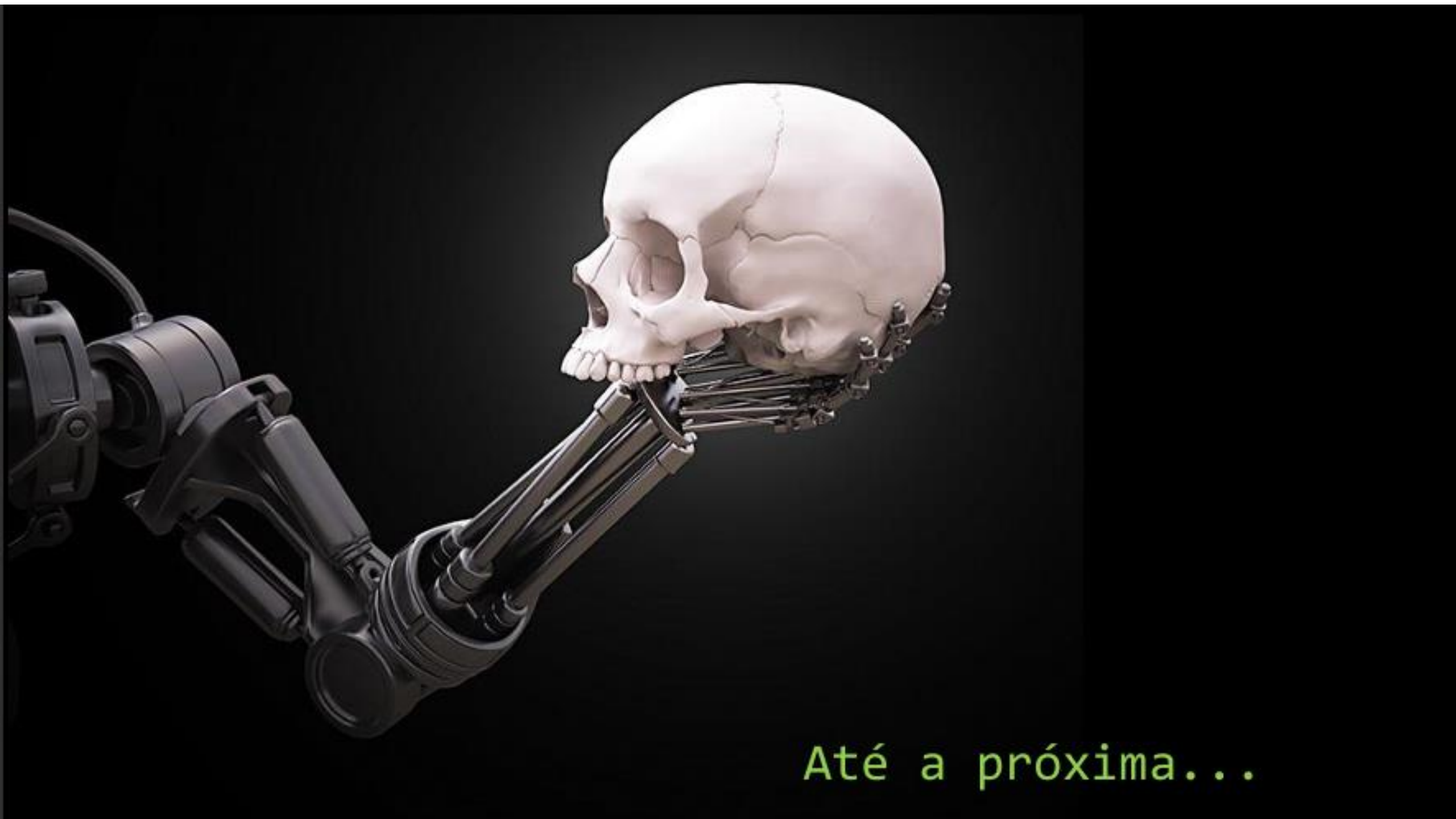
- Há uma variedade de estratégias de busca sem informação (ou busca cega).

- A busca de aprofundamento iterativo usa somente espaço linear e não muito mais tempo que outros algoritmos sem informação.

Obrigado!



Dúvidas?


Até a próxima...

Apresentador

Thales Levi Azevedo Valente

E-mail:
thales.l.a.valente@gmail.com

Referências

- Links referenciados nos respectivos slides.
- T.B. Borchartt . *Introdução à Inteligência Artificial*. 2024. 37 slides. Universidade Federal do Maranhão.
- A.O. B. Filho. *Inteligência Artificial - Introdução*. 2024. 31 slides. Universidade Federal do Maranhão.