Inteligência Artificial

Estratégia de Busca Parte 1

Prof. Jefferson Morais

Introdução

- **Estudaremos três estratégias de busca**
  - Busca sem informação (ou busca cega)
    - Não necessitam de nenhuma informação adicional sobre estados
  - Busca com informação (ou busca heurística)
    - Sabem se um estado não-objetivo é “mais promissor” que outro
  - Busca competitiva
    - Ocorre em ambientes onde os objetivos dos agentes estão em conflito (e.g., em jogos)

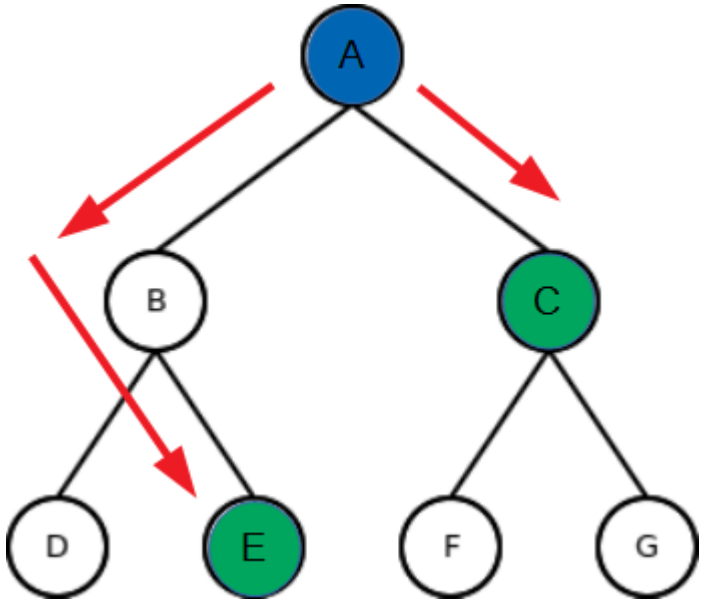
Desempenho da Busca

- Os critérios abaixo podem ser usados para comparar algoritmos
  - Completeza
    - O algoritmo garante encontrar uma solução se ela existir?
  - Otimização
    - A estratégia encontra a solução ótima?
  - Complexidade:
    - Tempo: expressão matemática “indicando” o tempo para encontrar uma solução?
    - Espaço: expressão matemática “indicando” quanta memória é necessária para executar a busca?

Desempenho da Busca

- A complexidade de tempo e espaço é representada por
  - **b**: número máximo de sucessores de qualquer nó
  - **d**: a profundidade do nó objetivo menos profundo (mais raso)
  - **m**: profundidade máxima da árvore

Ex.: o objetivo é encontrar um caminho de **A** até **C** ou **E**, temos que
b = 2
d = 1
m = 2

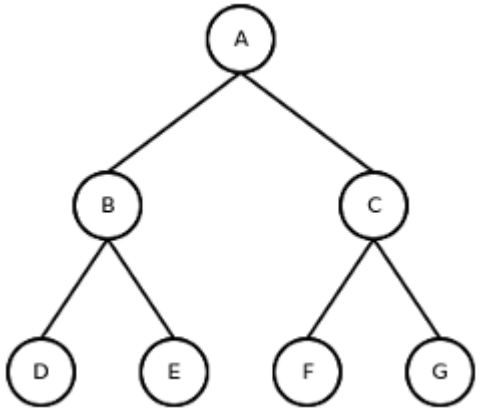


Algoritmos de Busca

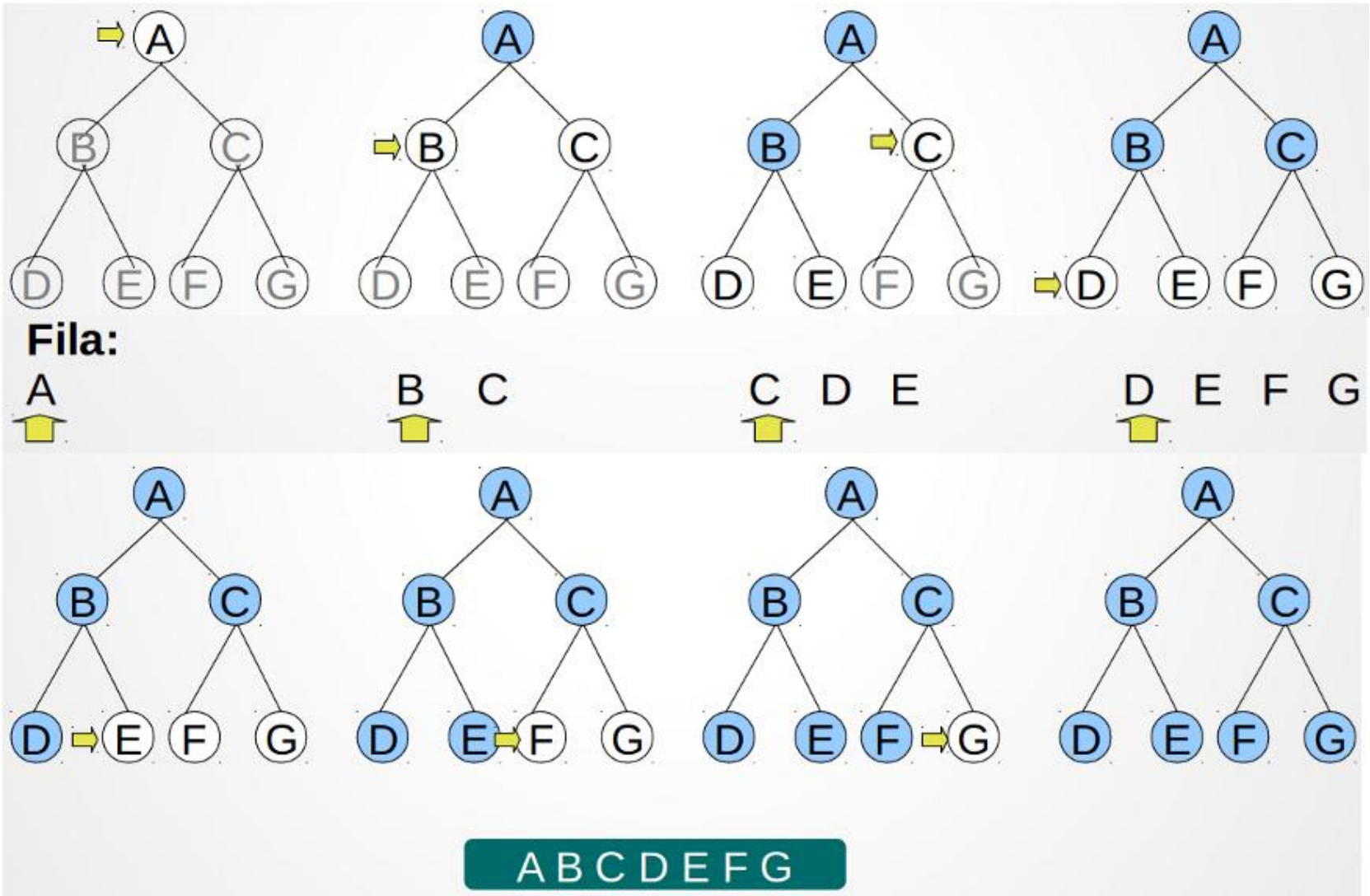
- Vamos estudar os seguintes algoritmos
  - Sem informação
    - Busca em largura
    - Busca em profundidade
    - Busca em profundidade limitada
    - Busca em profundidade iterativa
    - Busca com custo uniforme
  - Com informação
    - Busca gulosa
    - A*
  - Com competição
    - Algoritmo minimax
    - Poda alfa-beta

Sem Informação: Busca em Largura

- Estratégia: o nó raiz é expandido primeiro, em seguida todos os sucessores do nó raiz são expandidos, depois os sucessores desses nós, e assim por diante
- Utiliza-se uma fila (**FIFO (First In – First Out)**)para a borda
  - Novos nós vão para o fim da fila
  - Nós antigos são expandidos primeiro
- Ordem de expansão
  - 1) Nó raiz
  - 2) Todos os nós de profundidade 1
  - 3) Todos os nós de profundidade 2, etc

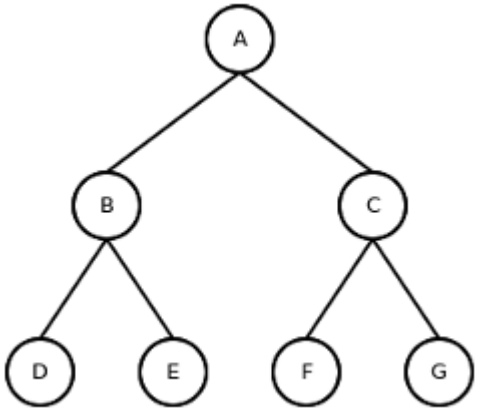


Sem Informação: Busca em Largura



Sem Informação: Busca em Largura

```
função BUSCA-EM-LARGURA(problema) retorna uma solução ou falha
  nó ← um nó com ESTADO = problema.ESTADO-INICIAL, CUSTO-DE-CAMINHO = 0
  se problema.TESTE-DE-OBJETIVO(nó.ESTADO) senão retorne SOLUÇÃO(nó),
  borda ← uma fila FIFO com nó como elemento único
  explorado ← conjunto vazio
  repita
    se VAZIO?(borda), então retorne falha
    nó ← POP(borda) / * escolhe o nó mais raso na borda */
    adicione nó.ESTADO para explorado
    para cada ação em problema.AÇÕES(nó.ESTADO) faça
      filho ← NÓ-FILHO(problema, nó, ação),
      se (filho.ESTADO)não está em explorado ou borda então
        se problema.TESTE-DE-OBJETIVO(filho.ESTADO) então retorne SOLUÇÃO(filho)
        borda ← INSIRA(filho, borda)
```



Sem Informação: Busca em Largura

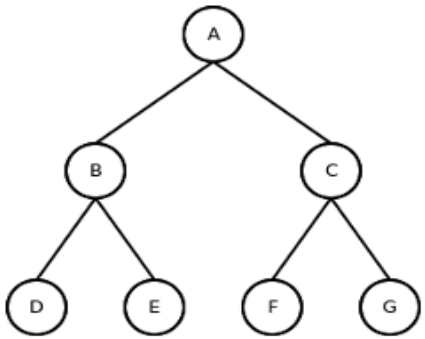
- Completa: Sim, quando o fator de ramificação (b) é finito
- Ótima: sim, desde que o custo de caminho cresça com a profundidade do nó
  - ∀ **n'** e **n**, profundidade(**n'**) ≥ profundidade(**n**) ⇒ custo de caminho(**n'**) ≥ custo de caminho (**n**)

Ex: quando todas as operações tiverem o mesmo custo, pois sempre explora profundidades **mais rasas primeiro**

- Pode não ser solução de menor custo de caminho, caso operadores tenham valores diferentes
  - ex. ir para uma cidade D passando por B e C pode ser mais perto do que passando só por E

Sem Informação: Busca em Largura

- **Complexidade de tempo**
  - Considere que cada estado tem b sucessores
    - Número de nós no nível **i** é $\mathbf{b^i}$
  - Suponha que a primeira solução está na profundidade **d**. No pior caso, é o último nó gerado naquele nível. Então, o número total de nós gerados é:
    - $b + b^2 + b^3 + ... + b^d = \boldsymbol{O(b^d)}$
  - Caso aplique o teste de objetivo para nós ao serem selecionados para a expansão, toda a camada de nós na profundidade **d** seria expandida antes que o objetivo fosse detectado e a complexidade de tempo seria
    - $\boldsymbol{O(b^{d+1})}$, isto é, exponencial



Sem Informação: Busca em Largura

- **Complexidade de espaço**
  - Para qualquer tipo de busca em grafos, que armazena todos os nós expandidos no conjunto explorado, a complexidade do espaço está sempre dentro de um fator de **b** da **complexidade do tempo**
  - Para busca em largura em grafos, por exemplo, cada nó gerado permanecerá na memória
    - Haverá $O(b^{d-1})$ nós no conjunto explorado
    - $O(b^{d})$ nós na borda (fronteira)
    - Assim a complexidade de espaço também será exponencia $O(b^{d})$

- Um limite de complexidade exponencial é **impraticável** para problemas grandes

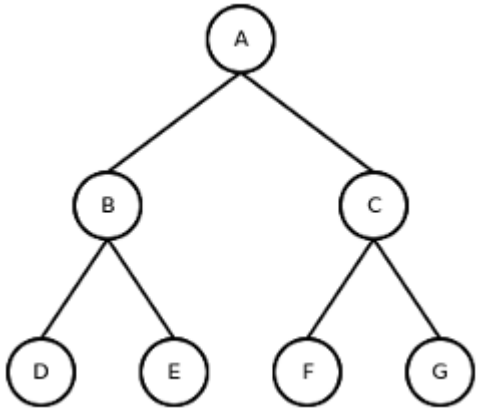
Sem Informação: Busca em Largura

- Exemplificando
  - Fator de ramificação b = 10
  - Processador que gera 1 milhão de nós por segundo
  - Cada nó ocupa 1000 bytes de armazenamento

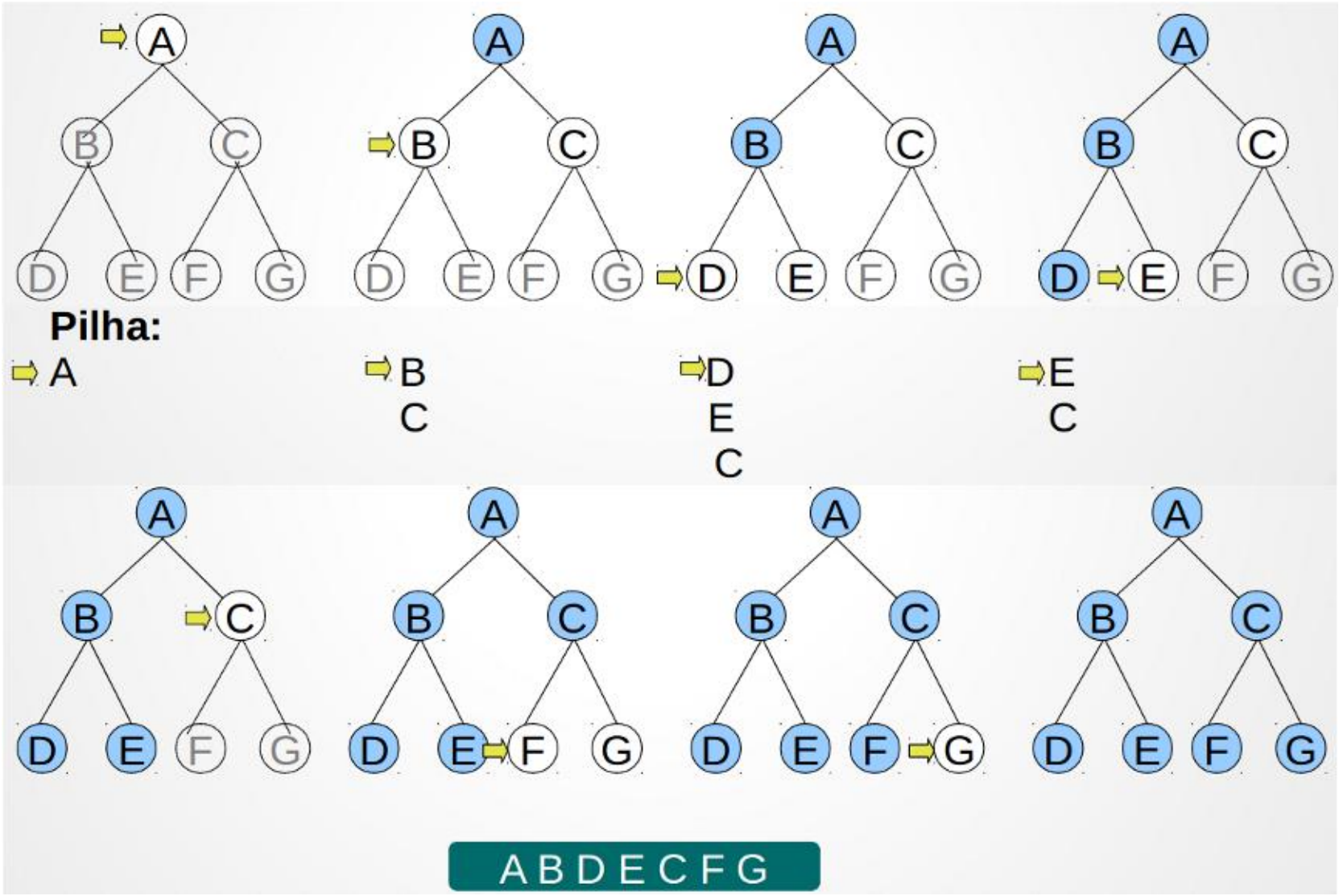
Profundidade	Nós	Tempo	Memória
2	110	0,11 milissegundo	107 kilobytes
4	11.110	11 milissegundos	10,6 megabytes
6	10^6	1,1 segundo	1 gigabyte
8	10^8	2 minutos	103 gigabytes
10	10^{10}	3 horas	10 terabytes
12	10^{12}	13 dias	1 petabyte
14	10^{14}	3,5 anos	99 petabytes
16	10^{16}	350 anos	10 exabytes

Sem Informação: Busca em Profundidade

- Estratégia: sempre expande o nó **mais profundo** na borda atual da árvore de busca
- Utiliza uma **Pilha LIFO (Last In – First Out)** para a borda
  - A busca prossegue até o nível mais profundo da árvore de busca, onde os nós não têm sucessores (folhas)
  - À medida que esses nós são expandidos, eles são retirados da borda
- Ordem de expansão
  - Nó raiz
  - Primeiro nó de profundidade 1
  - Primeiro nó de profundidade 2, etc

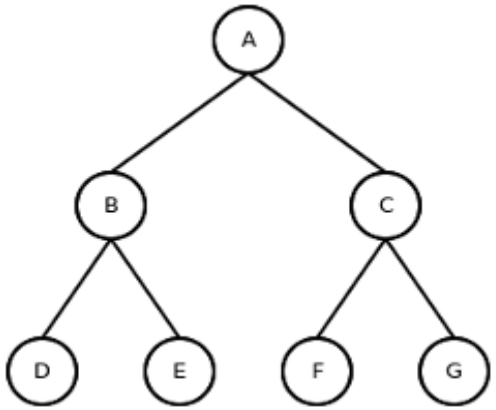


Sem Informação: Busca em Profundidade



Sem Informação: Busca em Profundidade

- Completa: Sim
- Ótima: não, pode retornar a solução com maior profundidade (e.g., C e E são objetivos → retorna E)
- Complexidade de tempo: **$O(b^m)$**
  - Irá gerar todos os nós e **m** (profundidade **máxima** da árvore de busca) pode ser muito maior que **d** (profundidade da **solução mais rasa**)
- Complexidade de espaço
  - O armazenamento é de apenas **O(bm)** nós (linear!)



Sem Informação: Busca em Profundidade

- **Próxima Aula:**

Continuação das Buscas Sem Informação:

Busca Em Profundidade Limitada

Busca em Profundidade Iterativa

Busca com Custo Uniforme