- Sumário:
  - Busca não informada
    - Busca em largura primeiro
    - Custo uniforme
    - Busca em profundidade primeiro

Objectivos

- Adquirir a noção de busca não informada
- Descrever alguns algoritmos de busca não informada: em largura primeiro, de custo uniforme, em profundidade primeiro

Formulação de problemas

- Consiste em decidir que acções e estados considerar dado um objectivo
- Um problema é definido formalmente por 4 componentes:
  - O ***estado inicial***
  - As possíveis ***acções*** do agente
    - Geralmente utiliza uma função sucessor
    - Estado inicial e função sucessor definem o ***espaço de estados***
  - ***Teste de satisfação do objectivo***
    - Utilizado para determinar se um estado constitui ou não o objectivo
  - O ***custo do caminho***
    - Função que atribui um custo numérico a cada trajectória

Busca em árvore: algoritmo

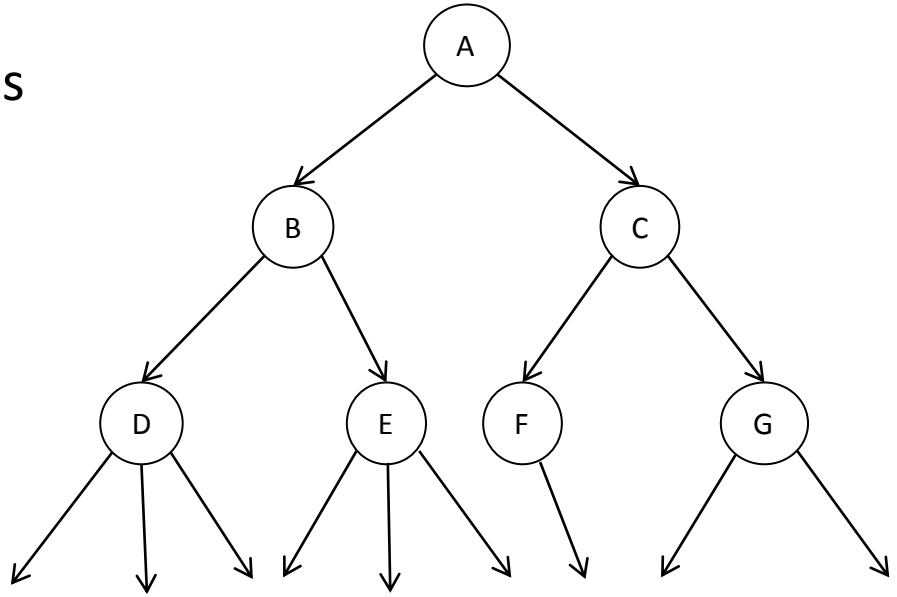
```
Função BuscaEmArvore(Problema, fronteira) retorna solução ou
   falha
Inicio
   fronteira ← InsereNaFila(FazNó(Problema[EstadoInicial]),
   fronteira)
   loop do
     se FilaVazia(fronteira) então
       retorna falha
     nó ← RemovePrimeiro(fronteira)
     se nó[Estado] for igual a Problema[EstadoFinal] então
       retorna Solução(nó)
     fronteira ← InsereNaFila(ExpandeFronteira(nó, Problema),
   fronteira)
Fim
```

Estratégia de busca não informada (cega)

- Estratégias de ***busca não informada*** **(*ou busca cega – blind search*)** usam apenas a informação disponível na definição do problema.
  - Apenas geram sucessores e verificam se o estado objetivo foi atingido
- As estratégias de busca não informada se distinguem pela ***ordem*** em que os nós são expandidos
  - Busca em largura primeiro (*breadth-first*)
  - Busca de custo uniforme (*uniform-cost*)
  - Busca em profundidade primeiro (*depth-first*)
  - Busca em profundidade limitada (*depth-limited*)
  - Busca de aprofundamento iterativo (*iterative deepening*)

Em largura primeiro

- Estratégia
  - Consiste em expandir todos os nós de um nível dado da árvore de busca, n, antes de qualquer nó correspondentes ao nível seguinte da árvore, n + 1
- Implementação
  - A fronteira é uma fila FIFO (first-in, first-out), os primeiros nós a ser incluídos são expandidos primeiro



Em largura primeiro: propriedades

- Completa?
  - Sim. Sempre encontra o objectivo que se encontra numa profundidade finita *d* (sempre que o factor de ramificação *b* seja finito)
- Óptima?
  - O nó objectivo mais raso nem sempre é o óptimo
  - É óptimo se o custo do caminho for uma função não decrescente da profundidade do nó

Em largura primeiro: propriedades

- Complexidade de tempo
  - Exponencial
  - $O(b^{d+1})$, considerando que cada estado tem b sucessores e a solução se encontra a uma profundidade d
- Complexidade de espaço
  - Exponencial
  - $O(b^{d+1})$, todos os nós gerados são mantidos em memória

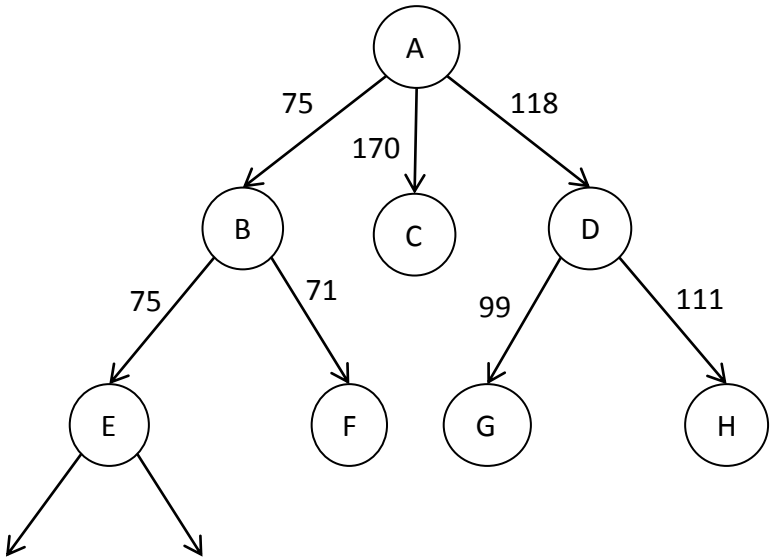
Em largura primeiro: requisitos de tempo e memória

- Considerando
  - Factor de ramificação, $b = 10$
  - Podem ser gerados 10.000 nós por segundo
  - Cada nó ocupa 1000 bytes

Profundidade	Nós	Tempo	Memória
2	1100	0.11 segundo	1 megabyte
4	111.100	11 segundos	106 megabytes
6	10^7	19 minutos	10 gigabytes
8	10^9	31 horas	1 terabyte
10	10^{11}	129 dias	101 terabytes
12	10^{13}	35 anos	10 petabytes
14	10^{15}	3.523 anos	1 exabyte

Custo uniforme

- Estratégia
  - Expande sempre o nó de menor custo de caminho
  - Se os custos de todos os passos são iguais, método é idêntico à busca em largura primeiro
- Implementação
  - Fronteira -> fila ordenada pelo custo do caminho



Custo uniforme: propriedades

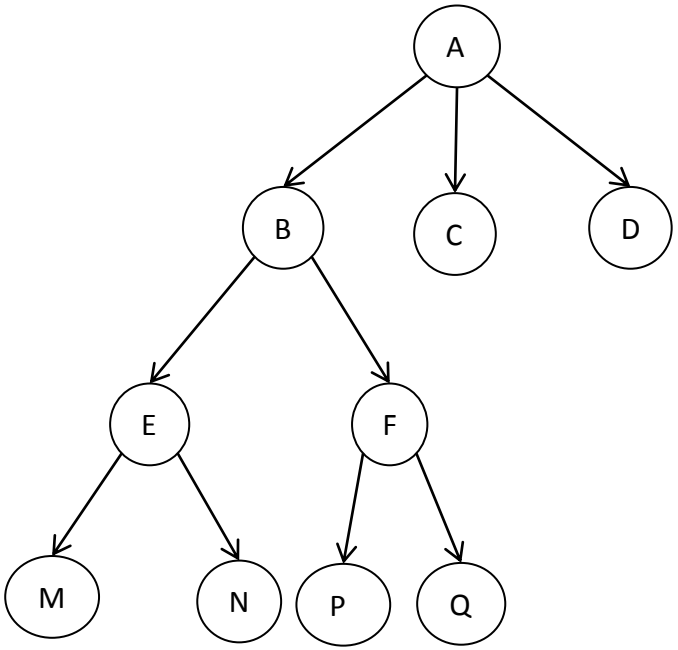
- Completa?
  - Sim, caso o custo de cada passo seja maior ou igual que um valor pequeno ϵ, constante e positivo
  - Condição evita a existência de ciclos infinitos ao expandir estados com custo de caminho nulo
- Óptima?
  - Sim, sob a mesma condição

Custo uniforme: propriedades

- Complexidade temporal
  - Exponencial, $O(b^{1+(C^*/\epsilon)})$
    - C^* - custo da solução óptima
    - ϵ - custo mínimo de uma acção
- Complexidade espacial
  - Exponencial, $O(b^{1+(C^*/\epsilon)})$

Em profundidade primeiro

- Estratégia
  - Expande o nó mais profundo da fronteira da árvore de busca
- Implementação
  - Fronteira -> fila LIFO (last-in, first-out) ou pilha



Em profundidade primeiro: propriedades

- Complexidade espacial
  - 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
  - Para um factor de ramificação *b* e uma profundidade máxima *m*, *O(mb)* –> **complexidade linear!**

Em profundidade primeiro: propriedades

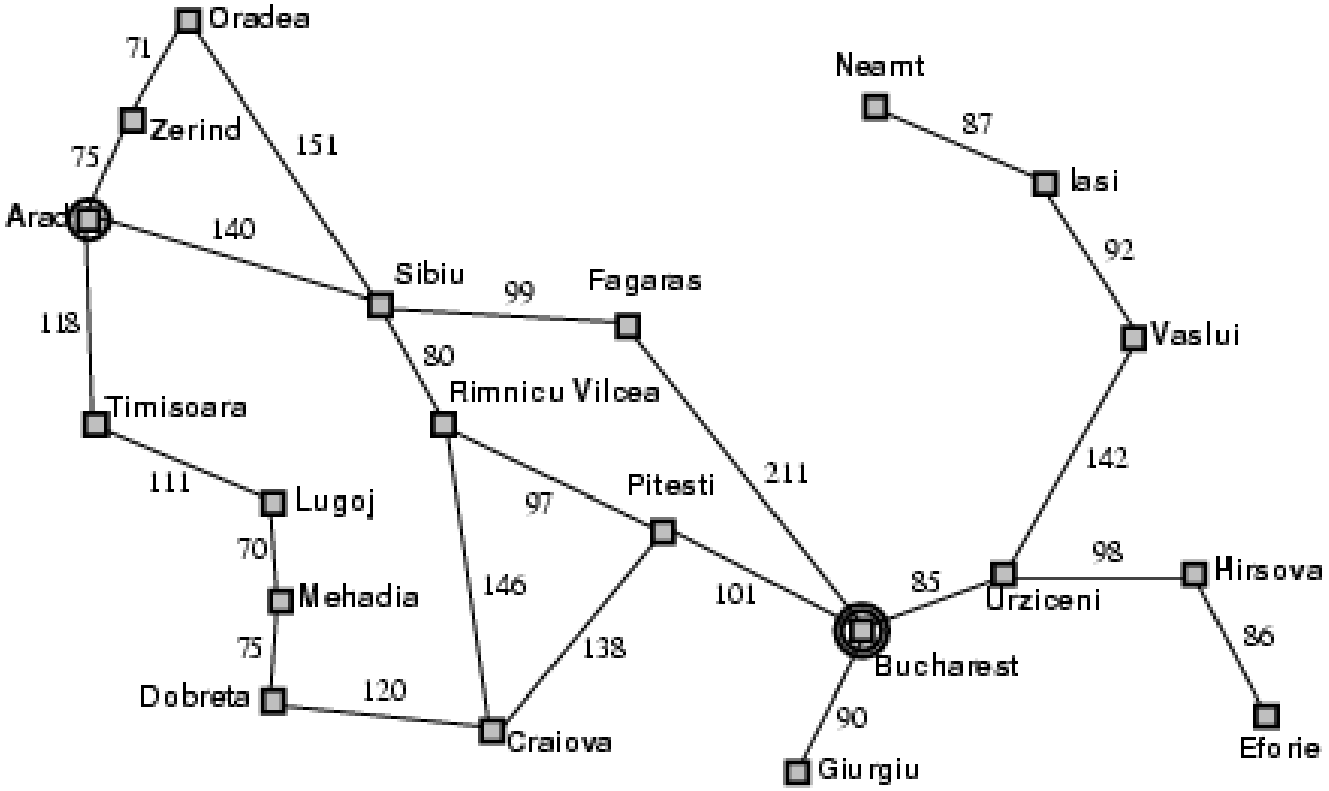
- Complexidade temporal
  - Exponencial
    - No pior dos casos todos os nós são gerados, $O(b^m)$
    - Péssimo quando *m* é muito maior que *d*
    - Mas se há muitas soluções pode ser mais eficiente que a busca em largura primeiro

Em profundidade primeiro: propriedades

- Completa?
  - Não
    - A busca não termina caso haja caminhos com profundidade infinita
- Óptima?
  - Não
    - Pode seleccionar um caminho errado e devolver uma solução profunda enquanto existem outras mais próxiimas da raiz

Tarefa

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



Bibliografia

- Russell & Norvig, pg. 73 – 77
- Costa & Simões, pg. 78 – 93
- Palma Méndez & Marín Morales, pg. 315 – 320