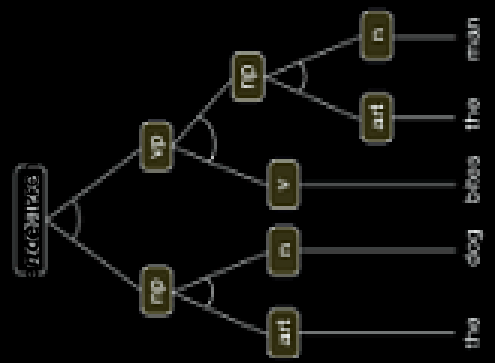
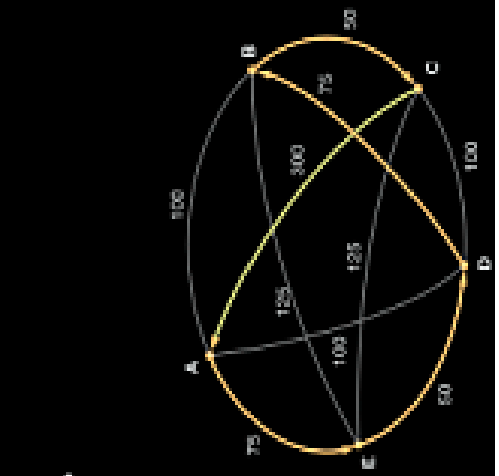
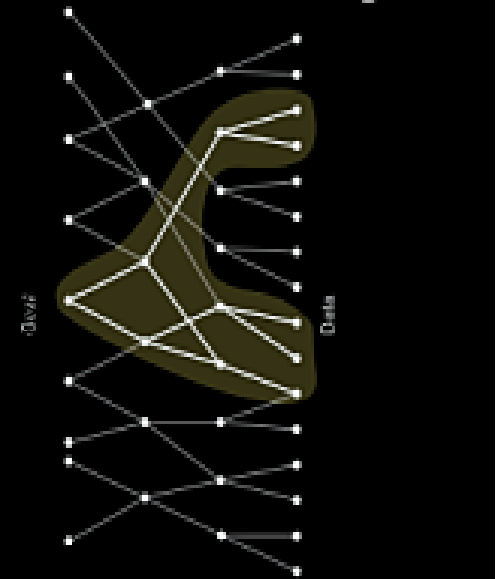






Universidade Federal de Campina Grande

Departamento de Sistemas e Computação

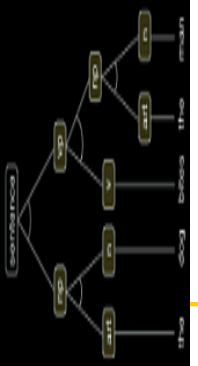
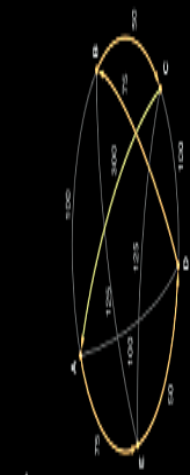
Curso de Pós-Graduação em Ciência da Computação

Inteligência Artificial

Resolução de Problemas (Parte II)

Prof.ª Joseana Macêdo Fechine Régis de Araújo

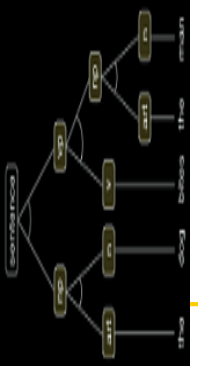
joseana@computacao.ufcg.edu.br

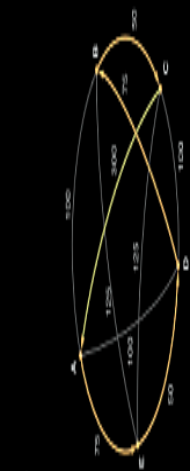




Em Busca de Soluções

Tópico

- Em Busca de Soluções



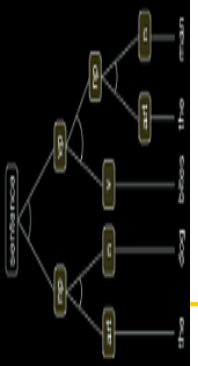
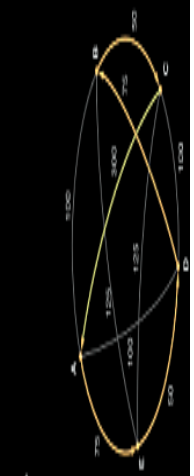


Em Busca de Soluções

Busca em todo o espaço de estados

- Uso de uma ***Árvore* de busca** explícita – gerada pelo estado inicial e pela função sucessor.

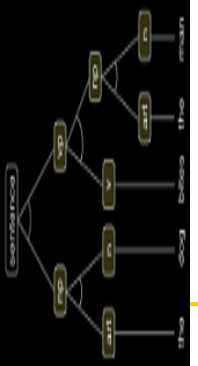
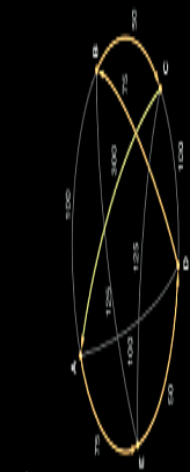
- Uso de um ***grafo* de busca** (substituindo a árvore de busca) – o mesmo estado pode ser alcançado a partir de vários caminhos.


Espaço de Estados do Problema

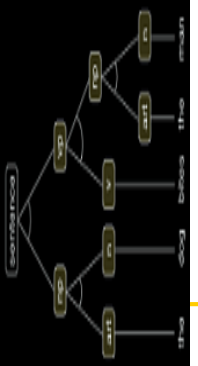
- Um problema pode ser visto como uma tripla: {***I***,***O***,***B***}
  - ***I*** = estados iniciais
  - ***O*** = conjunto de operações
  - ***B*** = estados objetivo

- Uma solução para o problema é uma seqüência finita de operações que permite sair de um elemento em ***I*** e chegar a um elemento em ***B***.


Espaço de Estados do Problema

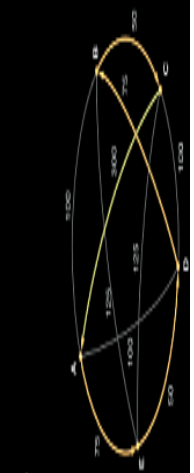
- **Um sistema de resolução de problemas comporta**:
  - Um conjunto de estruturas de dados organizada em um grafo;
  - Um conjunto de operadores caracterizados por suas condições de aplicação e sua ação;
  - Uma estrutura de controle implementando a estratégia de resolução.

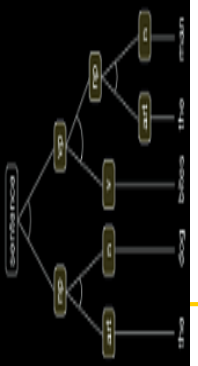
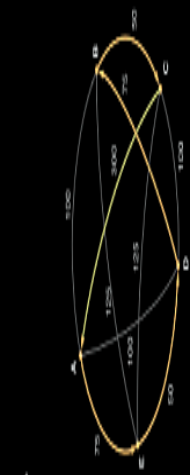


Estratégias de Busca

Abordagens de busca básicas num espaço de estados:

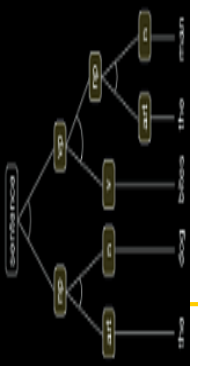
- **Busca Cega** (Sem informação/Não informada)
  - Não tem informação sobre qual sucessor é mais promissor para atingir a meta.
- **Busca Heurística** (Busca Com Informação/Informada)
  - Possui informação (estimativa) de qual sucessor é mais promissor para atingir a meta.
  - É uma busca cega com algum guia ou orientação.
- **Todas as estratégias de busca se distinguem pela ordem em que os nós são expandidos.**

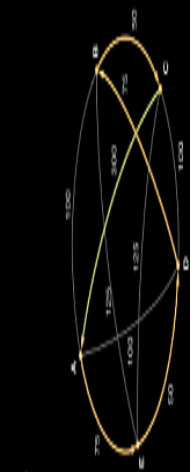



Estratégias de Busca Cega

- **Busca em Largura**
- Busca de Custo Uniforme
- **Busca em Profundidade**
- Busca em Profundidade Limitada
- Busca em Profundidade com Aprofundamento Iterativo
- Busca Bidirecional
- Evitando Estados Repetidos
- Busca com Conhecimento Incompleto





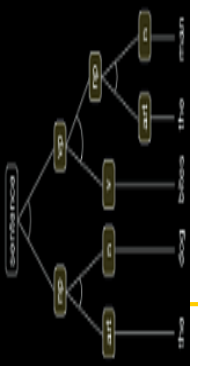
Busca em Profundidade ...

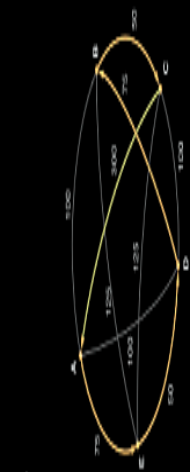
- **Ordem de expansão dos nós**:

  1. Nó raiz

  2. Primeiro nó de profundidade 1

  3. Primeiro nó de profundidade 2, etc …



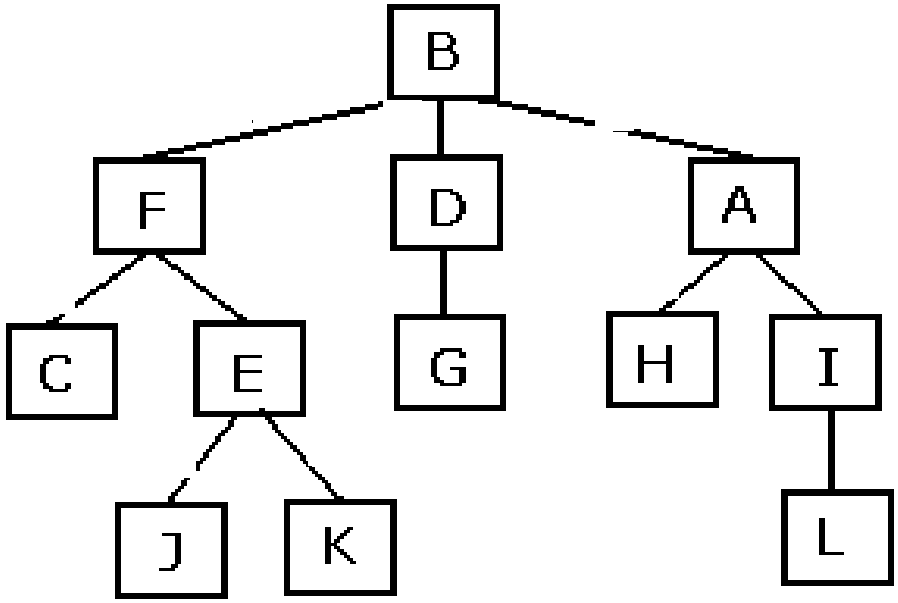


Busca em Profundidade ...

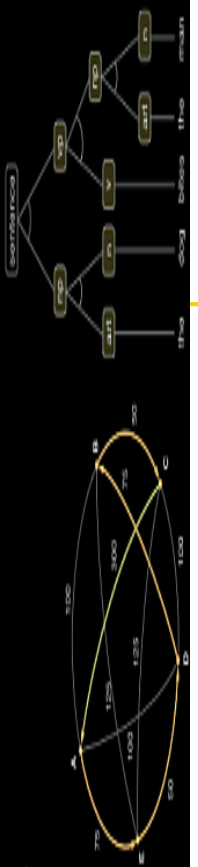
- Começa na raiz e avança para baixo em níveis cada vez mais profundos;
- Um operador é aplicado a um nó para gerar o próximo nó mais profundo na seqüência;
- O processo continua até que uma solução é encontrada ou um retrocesso é forçado ao atingir-se um nó terminal que não é solução.

Busca em Profundidade ...

- Faz uma busca sistemática em cada filho de um nó até encontrar a meta.
- **Exemplo**: O caminho para se chegar ao nó **G**, usando busca em profundidade: **Caminho = { B, F, C, E, J, K, D, G }**



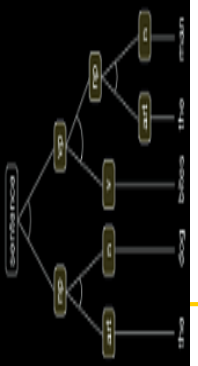
DSC/CCT/UFCG

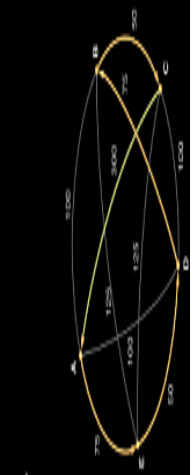


Busca em Profundidade ...

Problema

- Garante uma solução, mas a busca pode ser muito demorada.
- **Motivo**: muitas ramificações diferentes podem ter que ser consideradas até o nível mais profundo antes de uma solução ser atingida.





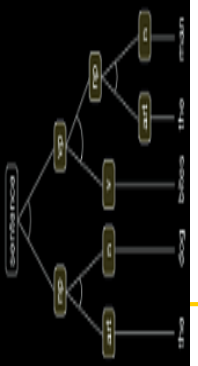
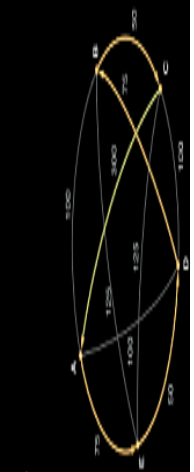
Busca em Largura ...

- **Ordem de expansão dos nós**:

  1. Nó raiz

  2. Todos os nós de profundidade 1

  3. Todos os nós de profundidade 2, etc …

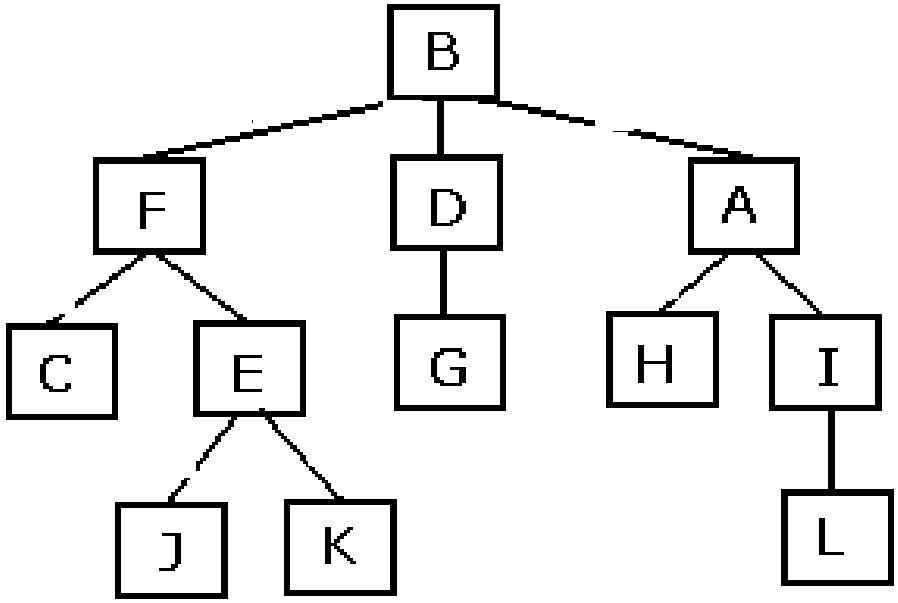



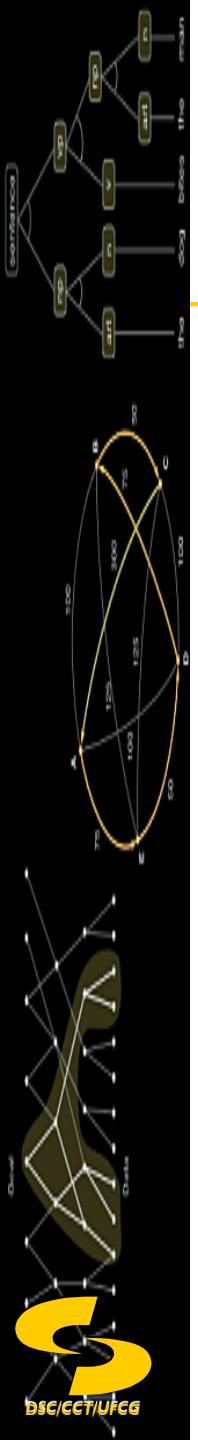

Busca em Largura ...

- Os nós em cada nível da árvore são completamente examinados antes de se mover para o próximo nível.
- Uma busca em largura sempre encontrará o menor caminho entre o estado inicial e o estado-objetivo.
- O menor caminho é o caminho com o menor número de passos (não confundir com o caminho de menor custo).

Busca em Largura ...

- Faz uma busca sistemática examinando primeiro os módulos próximos à raiz.
- **Exemplo**: Caminho para encontrar o nó **G**, usando a Busca em Largura: **Caminho = { B, F, D, A, C, E, G }**





Observações ...

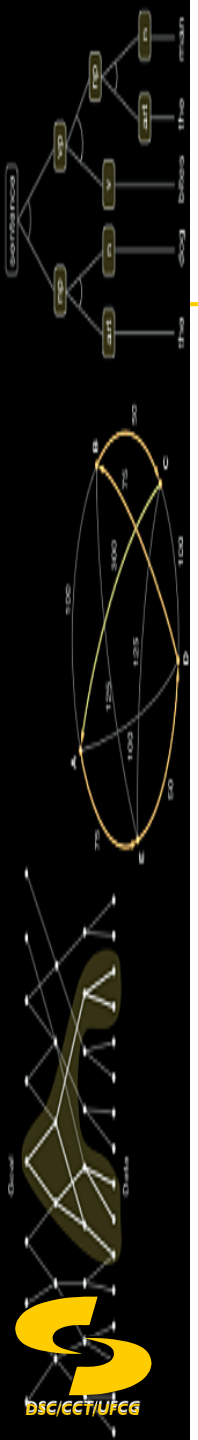
- Critérios importantes na análise de um algoritmo de busca:
  - Completeza: O algoritmo oferece a garantia de encontrar uma solução quando ela existir?
  - Otimização: A estratégia encontra a solução ótima (tem o menor custo de caminho entre todas as soluções)?
  - Complexidade de tempo: Quanto tempo ele leva para encontrar uma solução?
  - Complexidade de espaço: Quanto de memória é necessário para executar a busca?

Observações ...

Critério	Busca em Largura	Busca em Profundidade
Completa?	Sim[a]	Sim[a,b]
Ótima?	Sim[c]	Não
Tempo	$O(b^{d+1})$	$O(b^m)$
Espaço	$O(b^{d+1})$	$O(bm)$

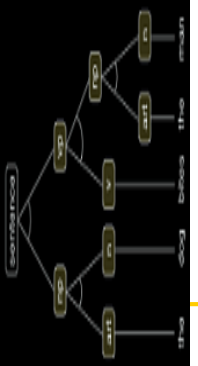
b - fator de ramificação; ***d*** - profundidade da solução mais "rasa"; ***m*** - profundidade máxima da árvore de busca; ***l*** - limite de profundidade.

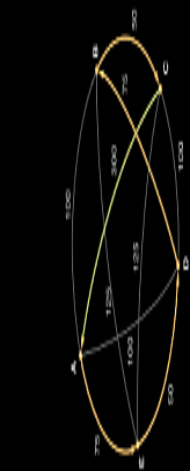
Anotações sobrescritas: ***a*** - completa se ***b*** é finito; ***b*** - completa se o custo do passo é ≥ ∈ positivo; ***c*** - ótima se os custos dos passos são todos idênticos; ***d*** - se ambos os sentidos utilizam busca em extensão.



Observações ...

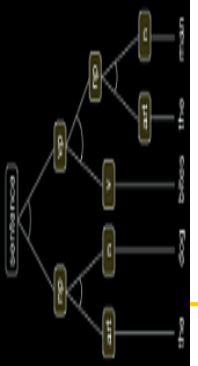
- **Explosão combinatorial**: quando o número de alternativas a serem exploradas é tão grande que o problema de complexidade torna-se crítico.
- **Exemplo**: Se cada nó no espaço de estados tem ***N*** sucessores, então o número de caminhos de comprimento ***C*** a partir do nó inicial é N^C (assumindo que não há ciclos).
- O número de caminhos candidatos à solução é exponencial com relação ao seu comprimento.
- As estratégias de busca em profundidade e em largura não fazem nada para combater esta complexidade: todos os caminhos candidatos são tratados como igualmente relevantes.

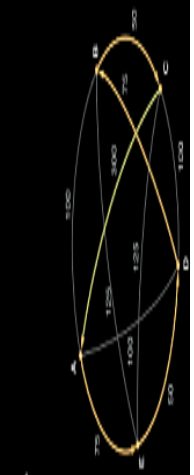




Observações ...

- As buscas em profundidade e em largura não precisam ser realizadas em uma ordem específica;
- Em se tratando de memória utilizada, na busca em profundidade é preciso armazenar todos os filhos não visitados de cada nó entre nó atual e nó inicial.
- Na busca em largura, antes de examinar nó a uma profundidade ***d***, é necessário examinar e armazenar todos os nodos a uma profundidade ***d*** - 1;





Observações ...

- Busca em profundidade utiliza menos memória;
- Quanto ao tempo, a busca em profundidade é geralmente mais rápida.
- Métodos de busca cega não examinam a árvore de forma ótima, o que poderia minimizar o tempo gasto para resolver o problema.

Observações

- As buscas em largura e profundidade não fazem uso de nenhum conhecimento para encontrar sua solução, fazendo uma busca exaustiva dentro do seu espaço. Para contornar este problema, pode-se usar os **métodos heurísticos**.

Demonstrações

- Luger, G. F., ***Artificial Intelligence: Structures and Strategies for Complex Problem Solving***, 5 Ed., 2005.
  - Busca em Largura (= *breadth-first search* = ***BFS***)
  - Busca em Profundidade (= *depth-first search* = ***DFS***)