

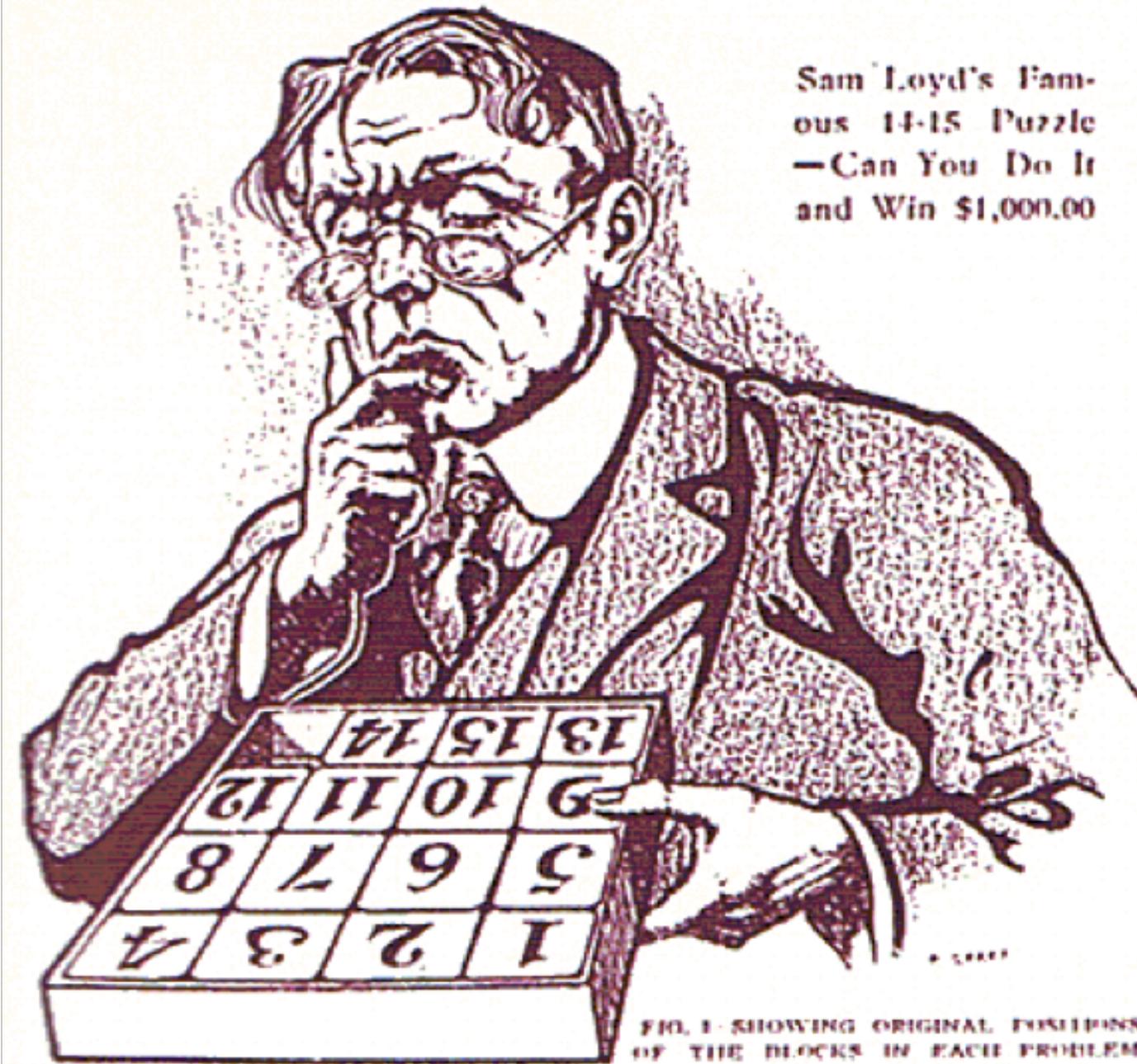
COMP0271: Inteligência Artificial

Busca cega



Professor: Hendrik Macedo
Universidade Federal de Sergipe, Brasil

A \$1,000.00 Cash Prize Puzzle



Sam Loyd's Famous 14-15 Puzzle
—Can You Do It
and Win \$1,000.00

FIG. I SHOWING ORIGINAL POSITIONS
OF THE BLOCKS IN EACH PROBLEM

Agentes Reativos

- Ações baseadas na percepção atual
- Não considera futuras consequências dos seus atos



The screenshot shows the Pydev interface in Eclipse. The left pane displays a list of search demos:

- 1 search demo maze
- 2 search -- greedy bad
- 3 search -- greedy good
- 4 search demo empty
- 5 search demo umaze
- 6 search -- plan slow
- 7 search -- plan fast
- 8 search -- reflex optimal
- 9 search -- reflex loop
- 1st class -- pacman

The right pane is empty. Below the interface is a terminal window showing Pacman search results:

```
<terminated> C:\Documents and Settings\Dan Klein\Eclipse\workspace\pacman_old\marwold\demo\demo.py

solution found.
Solution cost: 33.0
Number of nodes expanded: 51
Number of unique nodes expanded: 51

11:21 AM 8/28/2012
```

The screenshot shows the Pydev interface in Eclipse. The terminal window displays the same Pacman search results as the previous screenshot:

```
<terminated> 1.5
Pacman emerges victorious! Score: 671
{'numKills': 0, 'results': ['Win'], 'numMoves': 19, 'scores': [671]}

11:21 AM 8/28/2012
```

The status bar at the bottom of the screen shows a message: "FA12 cs188 lecture 2 -- uninformed search.pptx was updated to the latest version. (click to view)".

Agentes Reativos

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
```

```
persistent: rules, a set of condition-action rules
```

```
state  $\leftarrow$  INTERPRET-INPUT(percept)
```

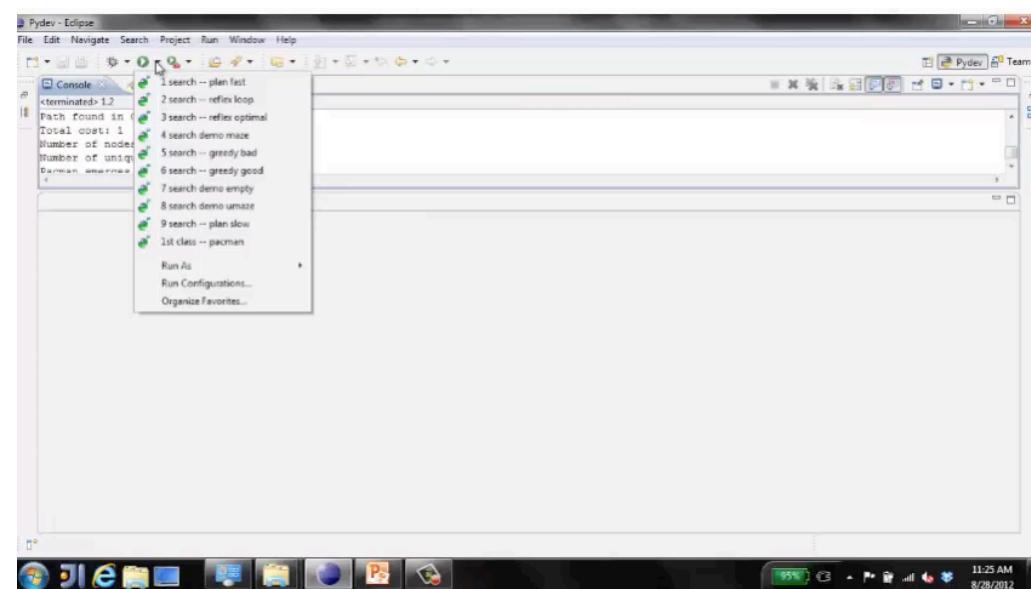
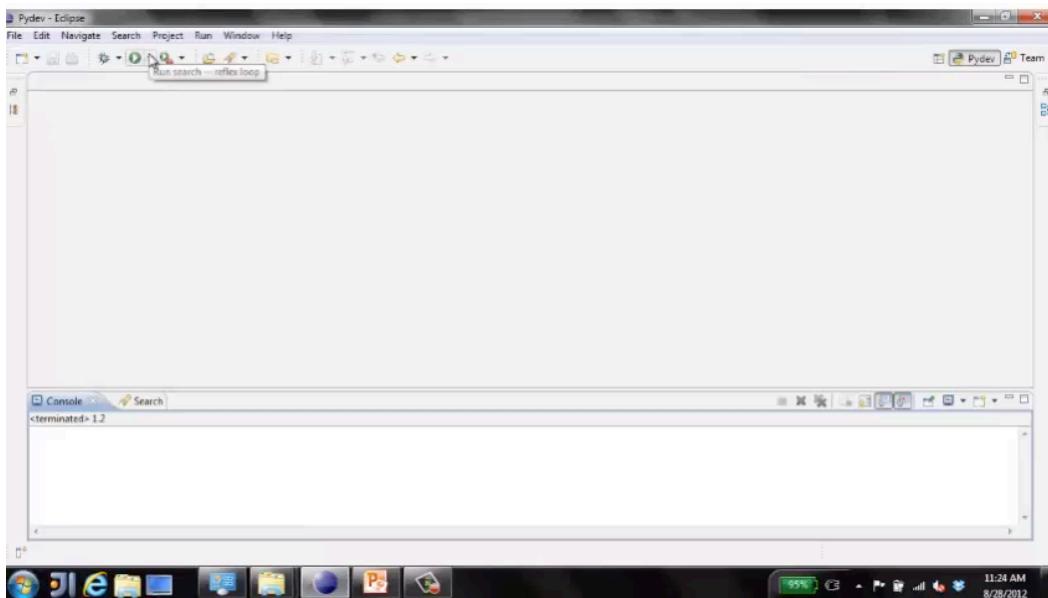
```
rule  $\leftarrow$  RULE-MATCH(state, rules)
```

```
action  $\leftarrow$  rule.ACTION
```

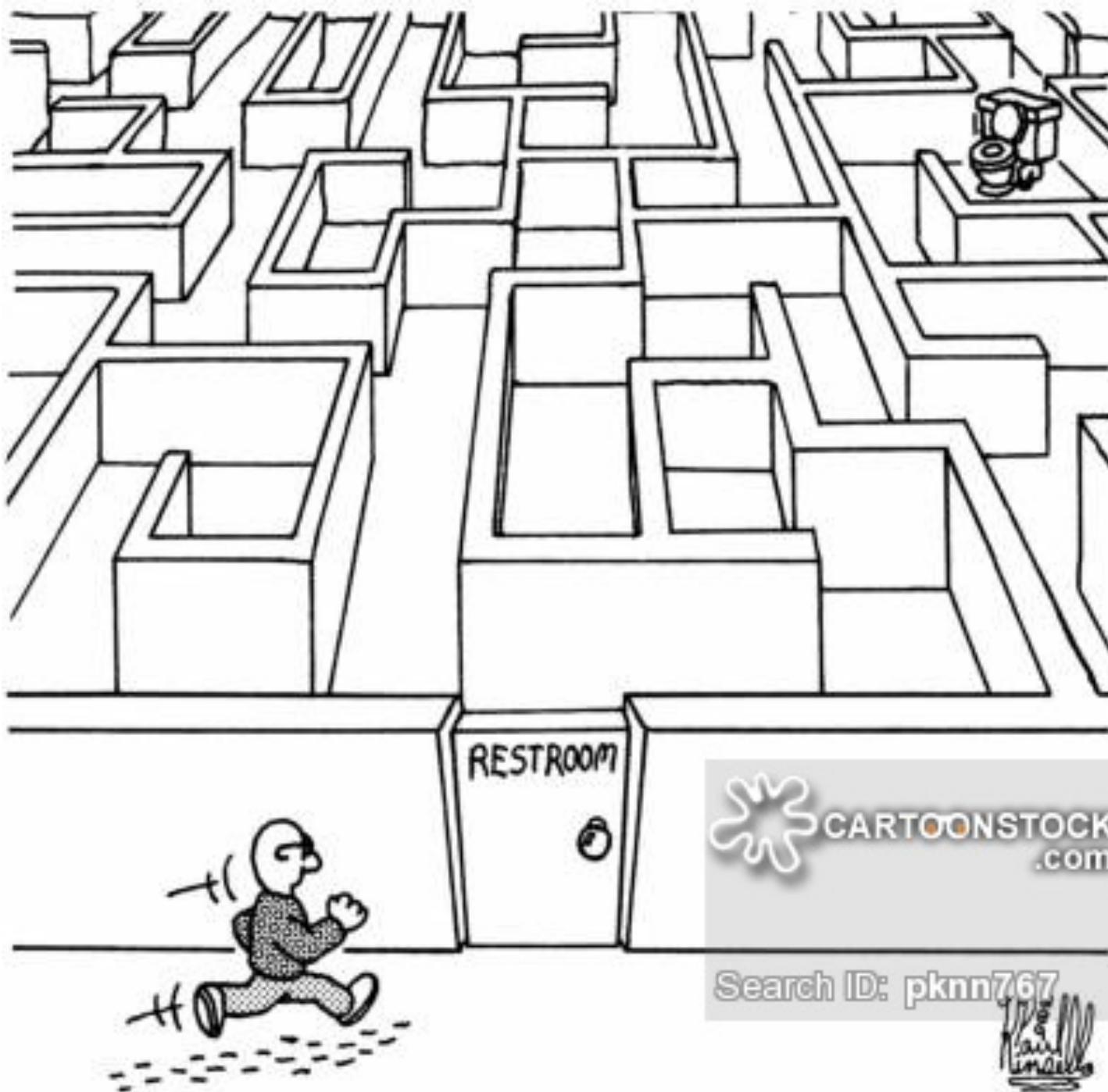
```
return action
```

Agentes com planejamento

- Decisões baseadas em hipóteses levantadas sobre as consequências de seus atos
- Mantém modelo de como o mundo evoluirá
- Deve formular um objetivo (em forma de teste)
- **Completure vs. Optimalidade**

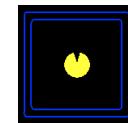
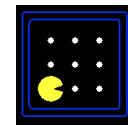
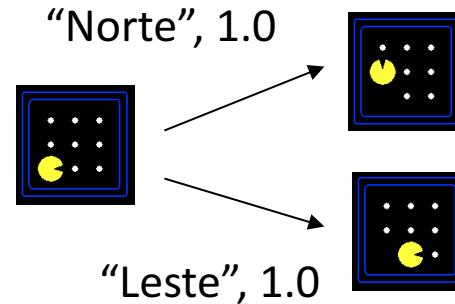
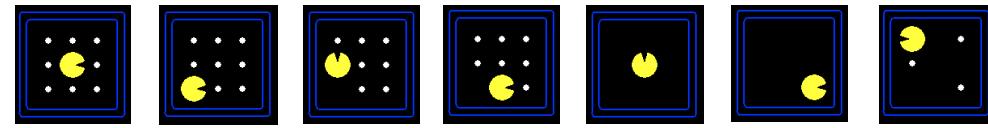


Problemas de busca



Formulação de um problema de busca

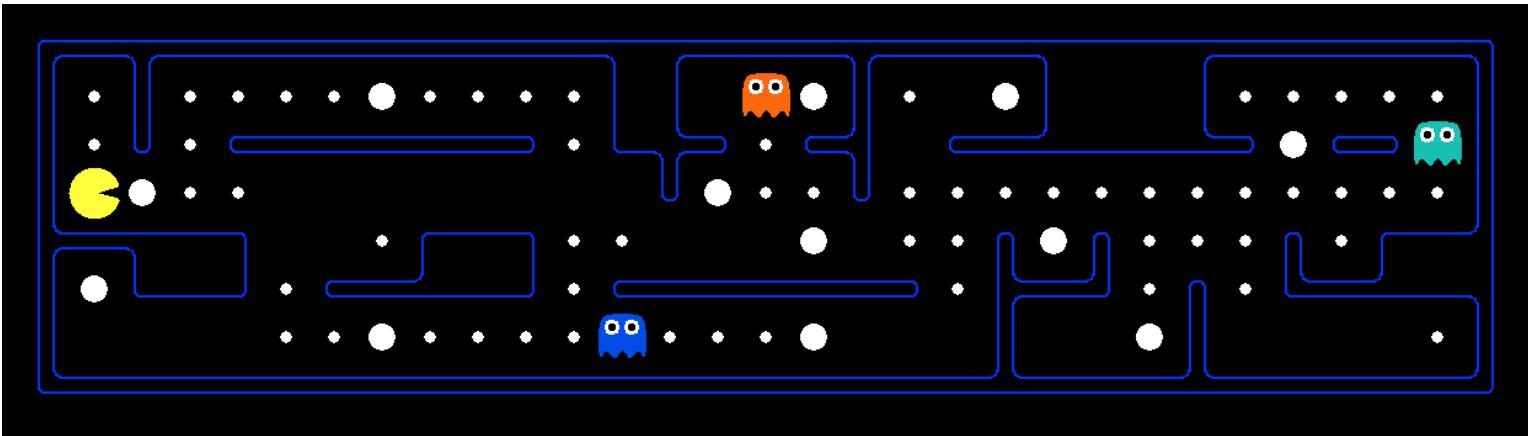
- ESPAÇO DE ESTADOS
- FUNÇÃO SUCESSOR
(com ações e custos)
- ESTADO INICIAL
- ESTADO OBJETIVO



Uma **solução** é uma sequência de ações (plano) que transforma um estado inicial em um estado objetivo

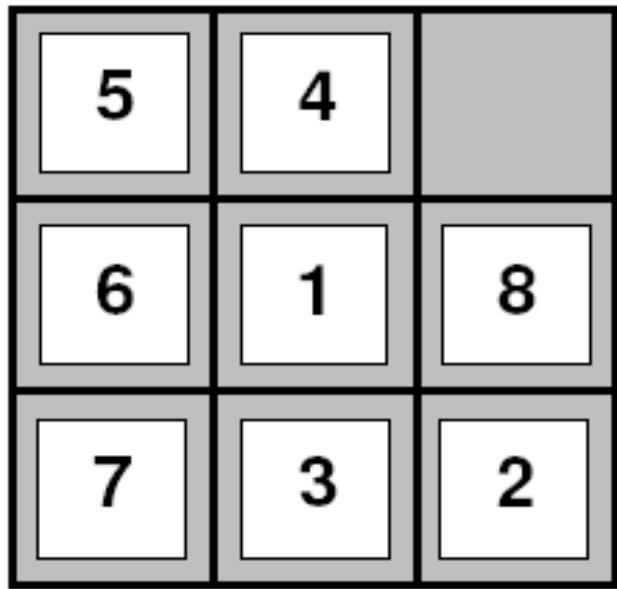


Como formular?



o **Espaço de busca** deve conter apenas o **necessário** para o planejamento (abstração)

- Problema: percurso
 - Espaço de Estados: posições (x,y)
 - Função sucessor: NSLO
 - Estado inicial: $(0, 0)$
 - Estado objetivo: $(x,y) = \text{FIM} ?$
- Problema: Comer todos os pontos
 - Espaço de Estados: ?
 - Função sucessor: ?
 - Estado inicial: ?
 - Estado objetivo: ?

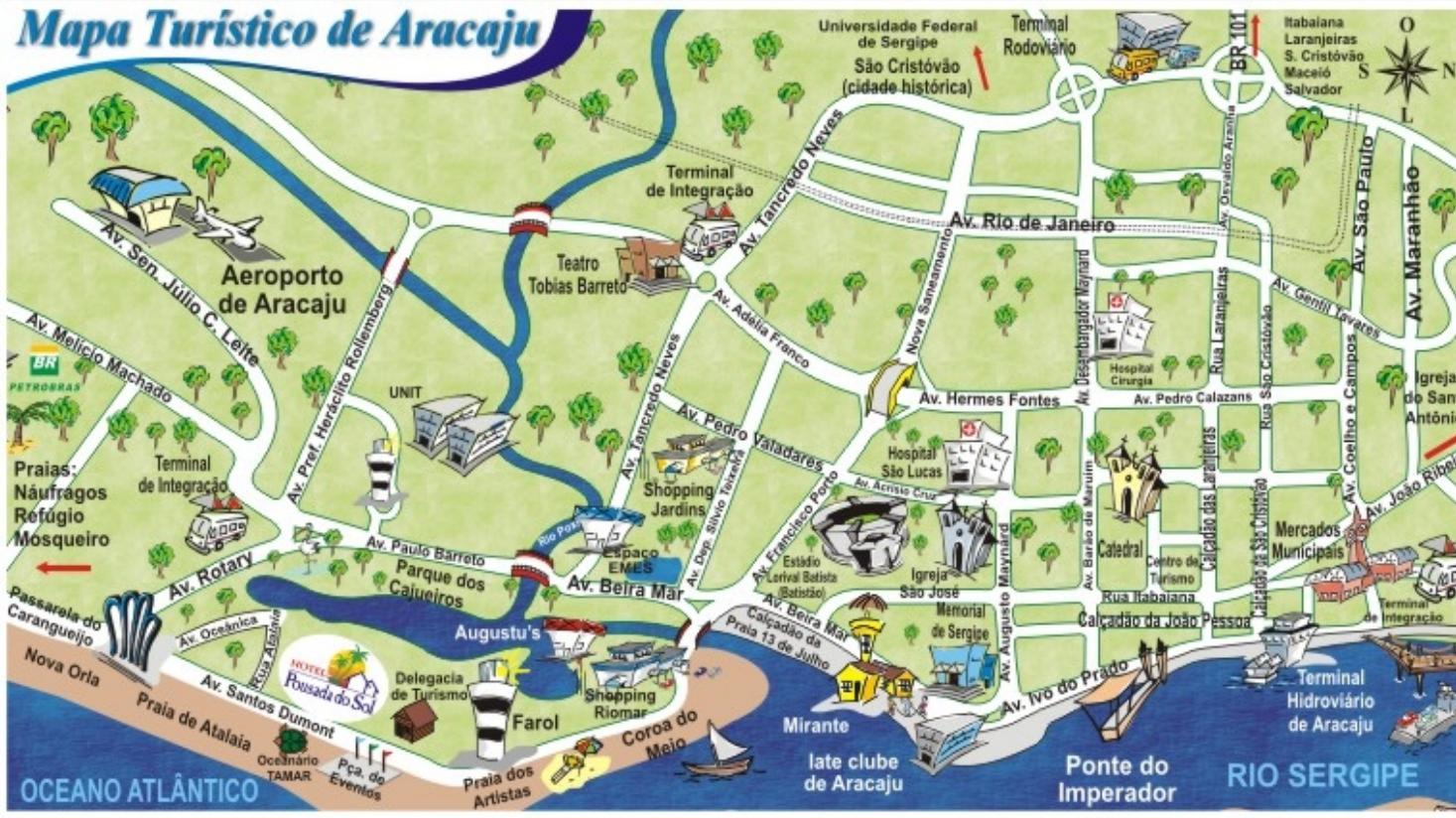


- Problema: 8-puzzle
 - Espaço: ?
 - Sucessor: ?
 - Inicial: ?
 - Objetivo: ?

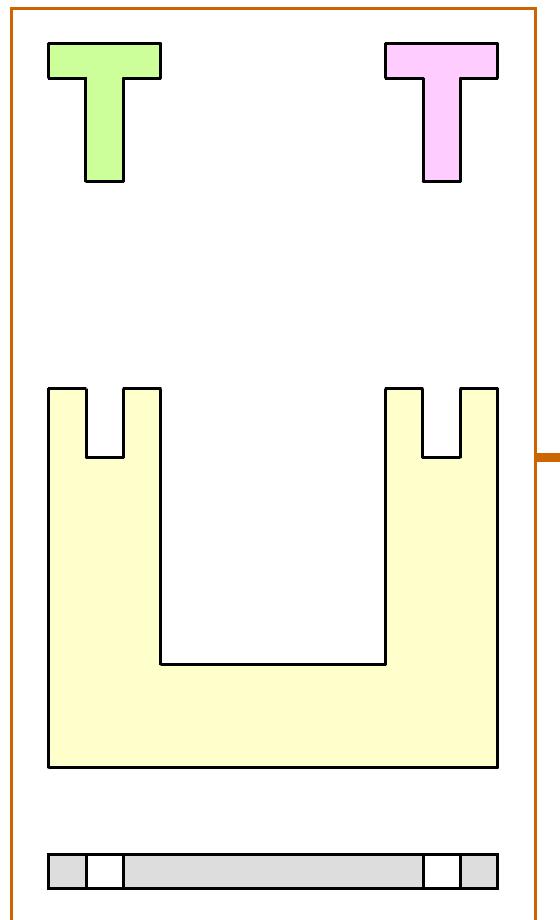
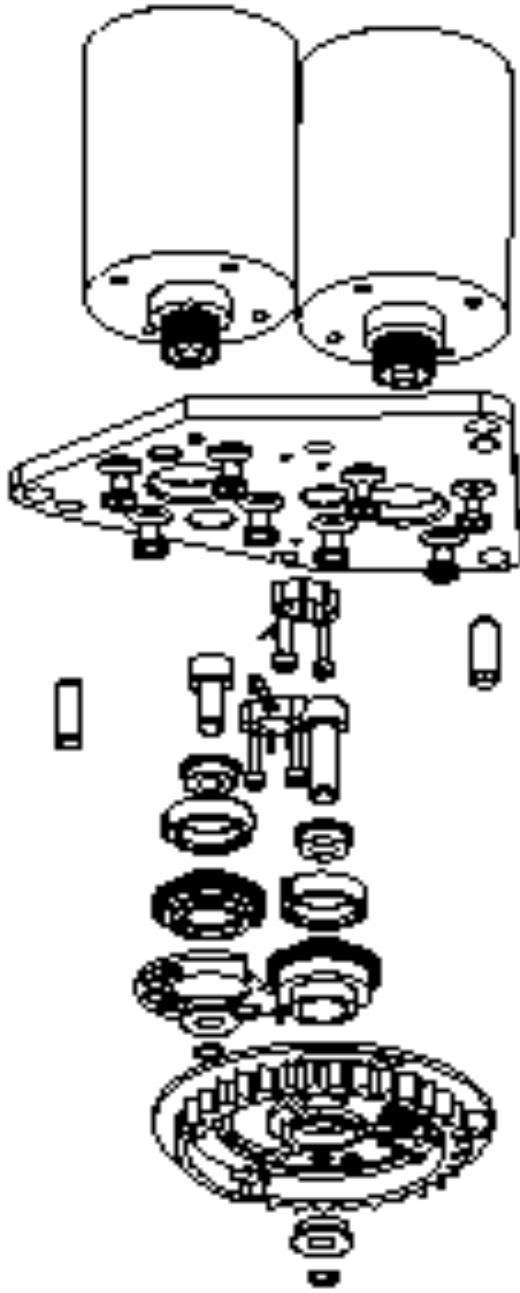


- Problema: Torre de Hanoi
 - Espaço: ?
 - Sucessor: ?
 - Inicial: ?
 - Objetivo: ?

Mapa Turístico de Aracaju



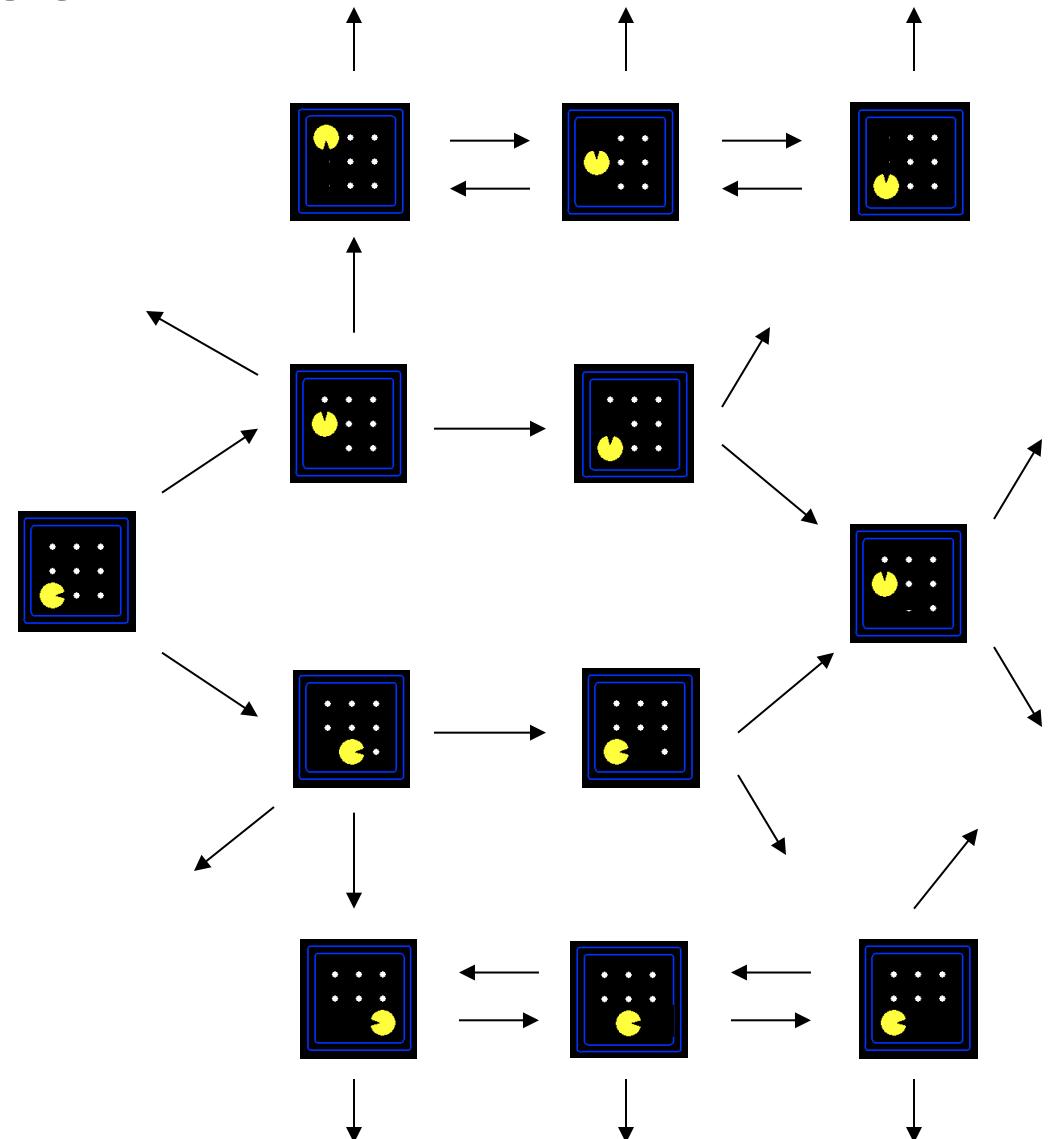
- Problema: Percurso na cidade
 - Espaço: ?
 - Sucessor: ?
 - Inicial: ?
 - Objetivo: ?



- Problema: processo de construção
 - Espaço: ?
 - Sucessor: ?
 - Inicial: ?
 - Objetivo: ?

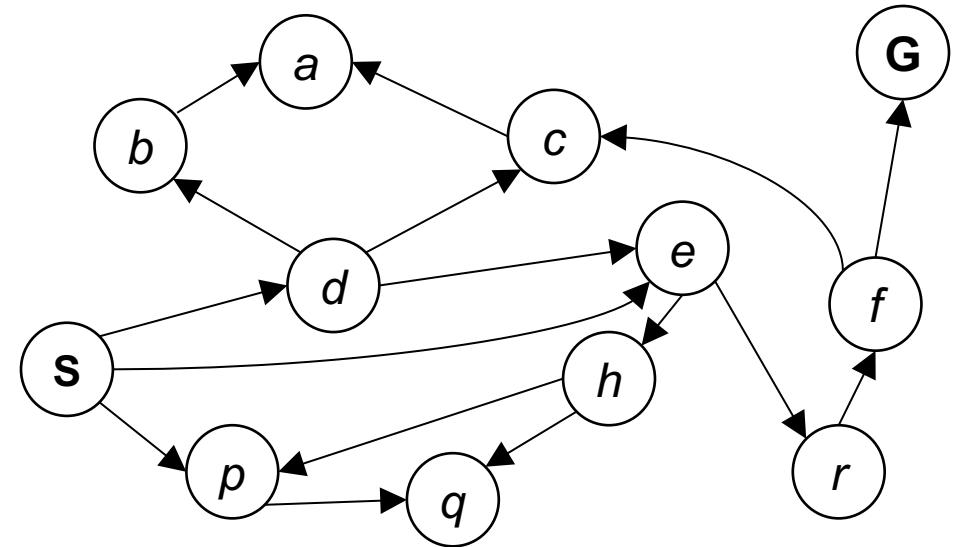
Grafos do espaço de estados

- Representação matemática do espaço de busca
 - Nós são estados
 - Arcos são sucessores
 - Cada estado ocorre apenas uma vez!
 - Usualmente não conseguimos manter todo o grafo na memória

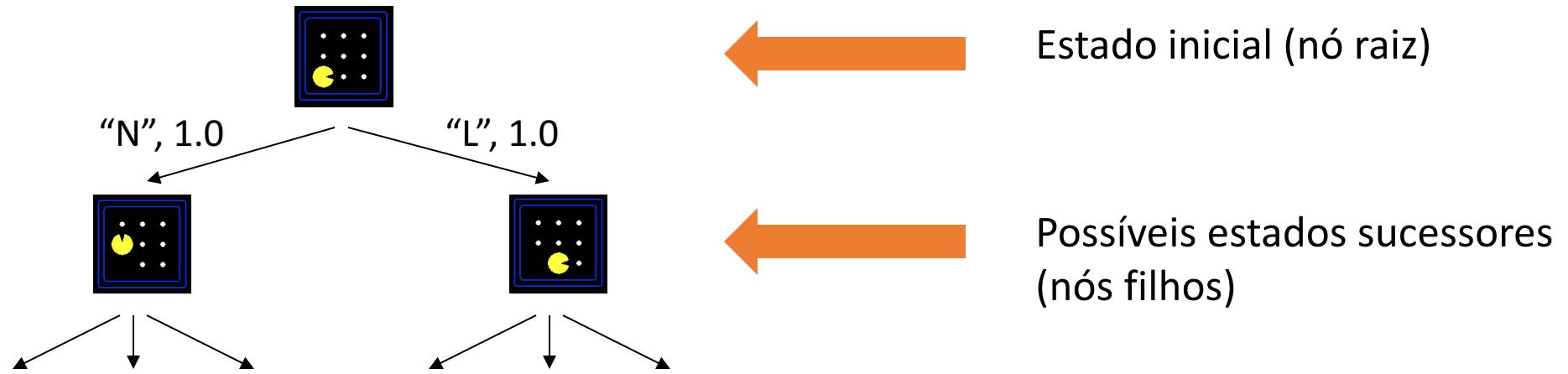


Grafos do espaço de estados

- Representação matemática do espaço de busca
 - Nós são estados
 - Arcos são sucessores
 - Cada estado ocorre apenas uma vez!
 - Usualmente não conseguimos manter todo o grafo na memória



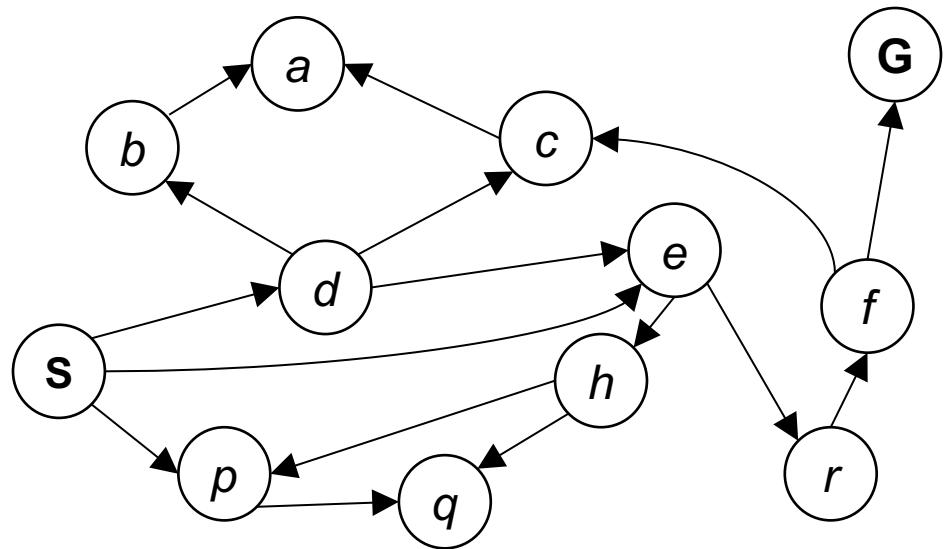
Árvores do espaço de estados



- Nós representam estados mas correspondem também a PLANOS que geraram estes estados. Ou seja, **diferentes planos podem gerar estados iguais**.
- Para a maioria dos problemas, nunca conseguimos construir toda a árvore

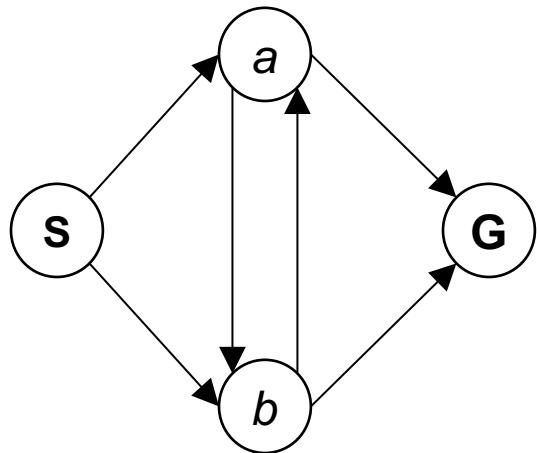
Q: Grafos vs. Árvores

Árvore a partir de S ?

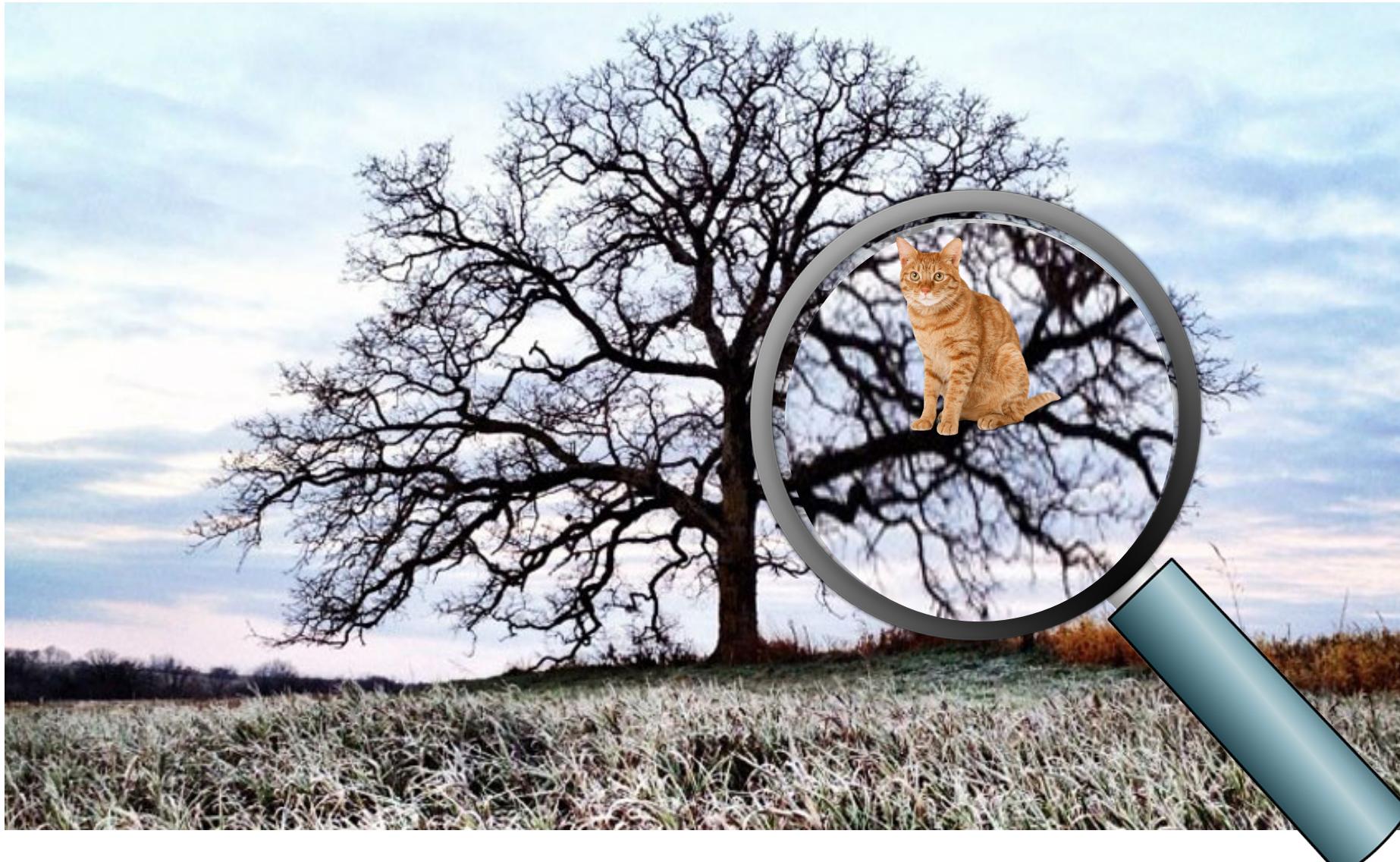


Q: Grafos vs. Árvores

Árvore a partir de S ?



Busca em árvore



Algoritmo de busca genérico

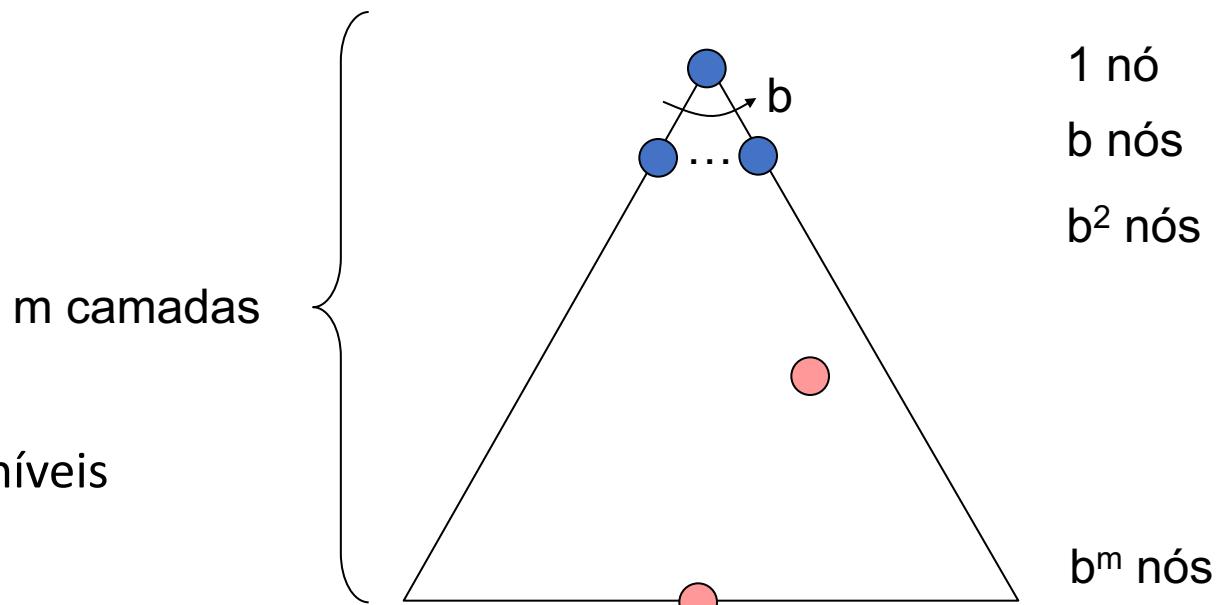
```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

fronteira

Que nó da *fronteira* escolher para **expandir**?

Propriedades do algoritmo de busca

- Completude: garante encontrar uma solução se existir ?
- Ottimalidade: garante encontrar o caminho de menor custo?
- Complexidade de tempo?
- Complexidade de espaço?
- Aspectos da árvore de busca:
 - b = fator de ramificação
 - m = maior profundidade
 - Objetivos podem existir em vários níveis
- Número de nós total na árvore?
 - $1 + b + b^2 + \dots + b^m = O(b^m)$



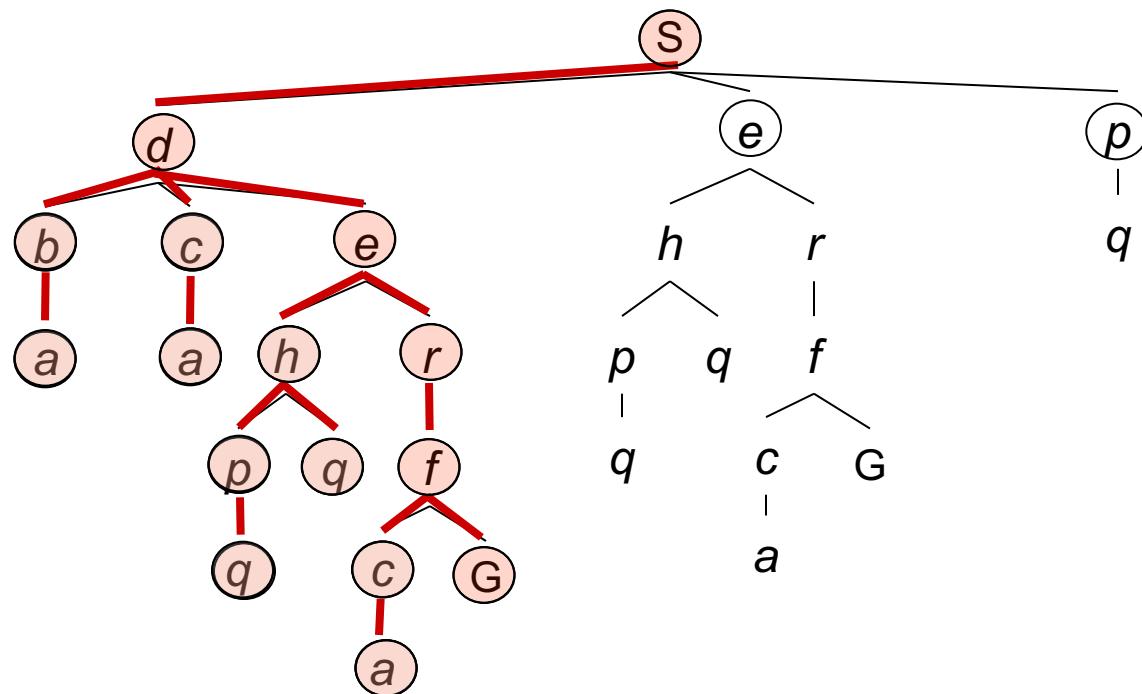
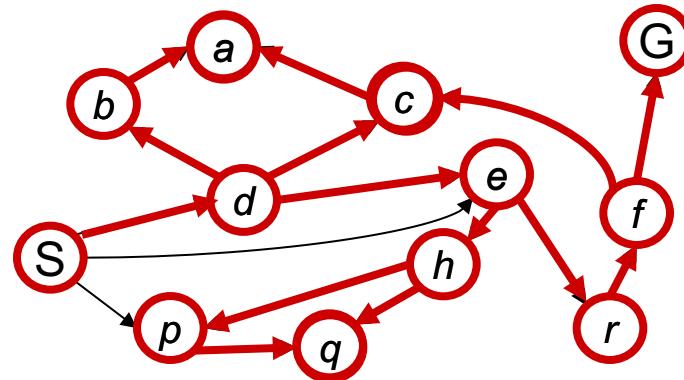
Depth-First Search (DFS)



Estratégia: expandir primeiramente cada uma das ramificações até sua profundidade máxima

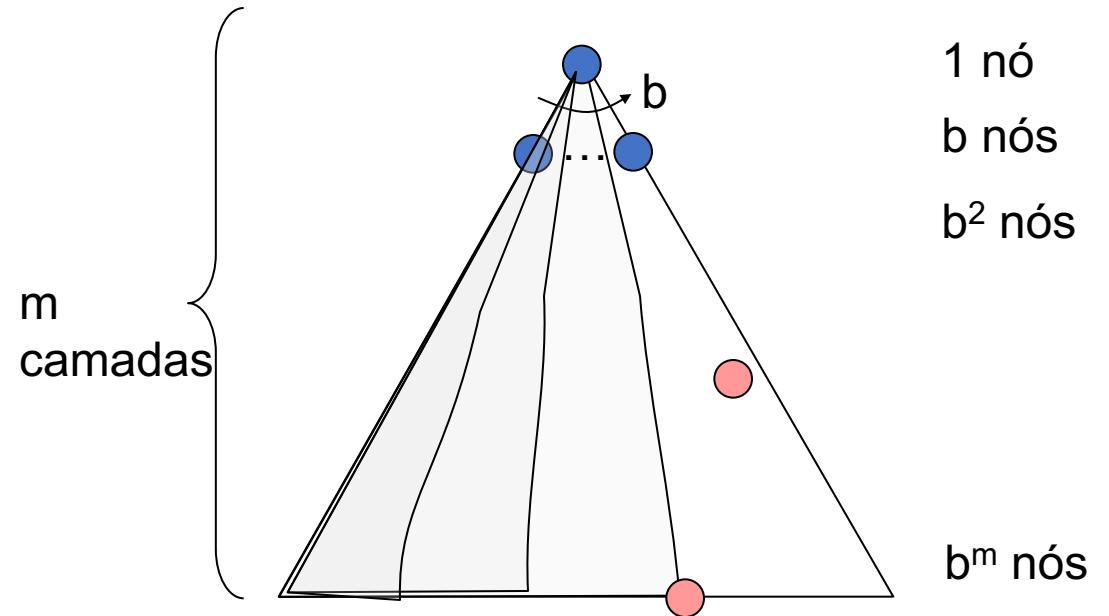
Implementação: fronteira é uma estrutura LIFO

Depth-First Search (DFS)



Depth-First Search (DFS) .. propriedades ..

- Quais nós DFS expande?
 - Esquerda para direita da árvore
 - Pode processar toda a árvore
 - Se m é finito, tempo = $O(b^m)$
- Quanto espaço a fronteira ocupa?
 - Apenas irmãos no caminho para a raiz, espaço = $O(bm)$
- Completude?
 - Apenas se prevenirmos ciclos
- Optimalidade?
 - Não. Encontra a solução mais à esquerda independentemente da profundidade e do custo.



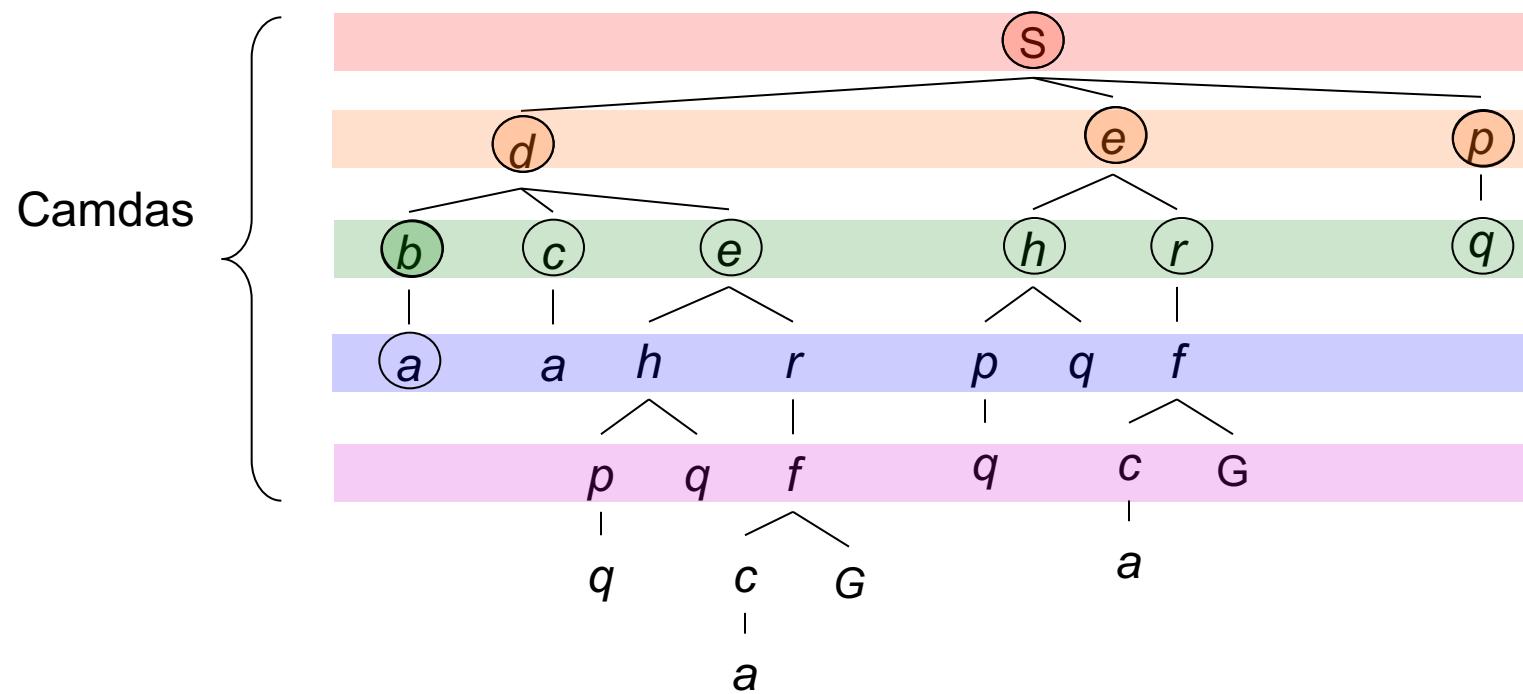
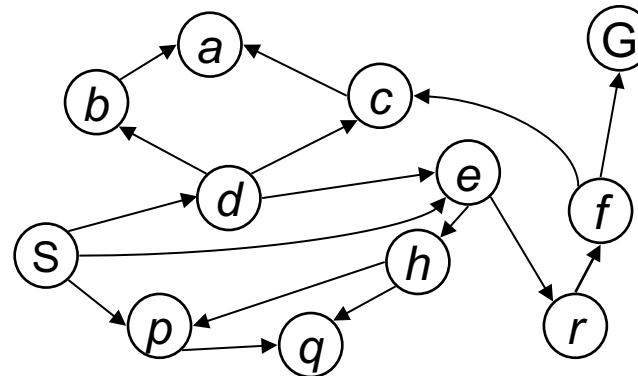
Breadth-First Search (BFS)



*Estratégia: expandir
primeiramente todos os
filhos do nível mais acima*

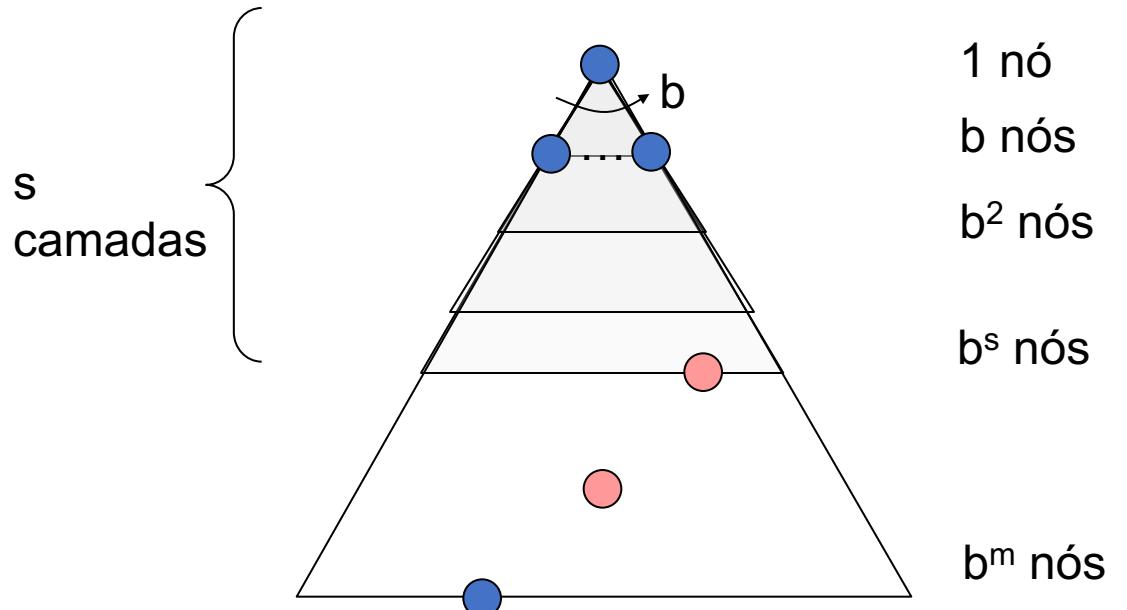
*Implementação: fronteira
é uma estrutura ?*

Breadth-First Search



Breadth-First Search (BFS) .: propriedades:.

- Que nós expande?
 - Todos os nós acima do nível do nó objetivo mais raso
 - Se nível do nó objetivo for s
 - busca leva tempo = $O(b^s)$
- Espaço?
 - Todos os nós da última camada $\rightarrow O(b^s)$
- Completude?
 - Se solução existe, sim!
- Otimalidade?
 - Apenas se custos forem todos de igual valor



Q: DFS vs BFS; Quem é quem abaixo?



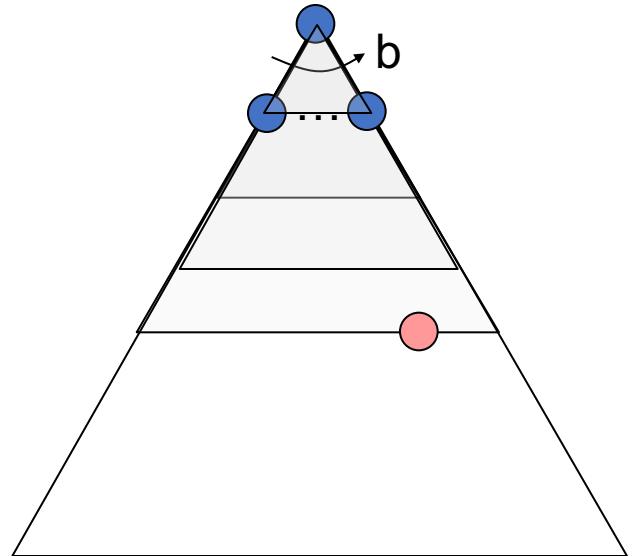
Q: DFS vs BFS; Quando um será melhor que outro?

Iterative Deepening

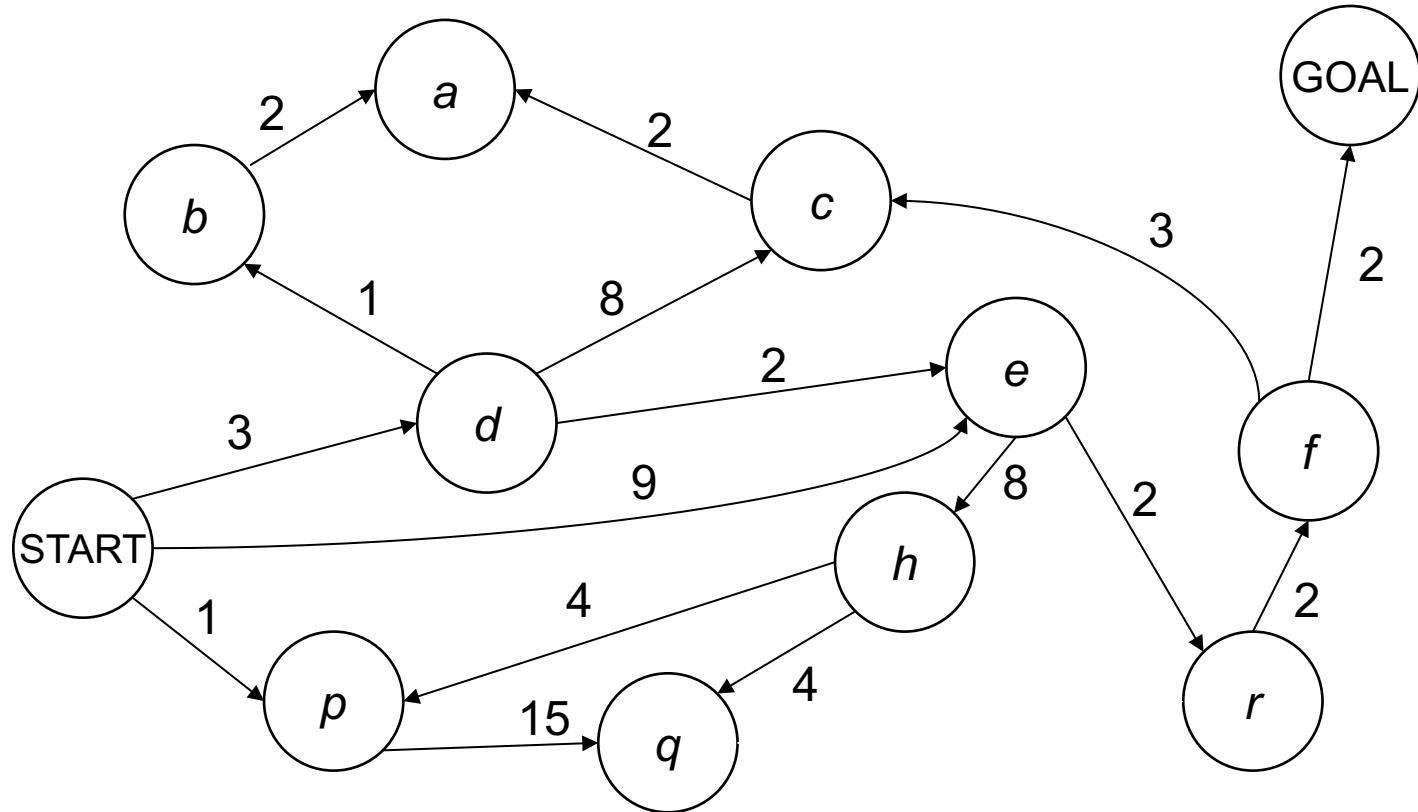


Iterative Deepening

- Ideia: espaço DFS + tempo BFS
 - Executa DFS com profundidade 1...
 - Executa DFS com profundidade 2...
 - Executa DFS com profundidade 3...



Considerando custos diferenciados na busca...



BFS encontra o melhor caminho em termos de número de ações, não em termos de menor custo!

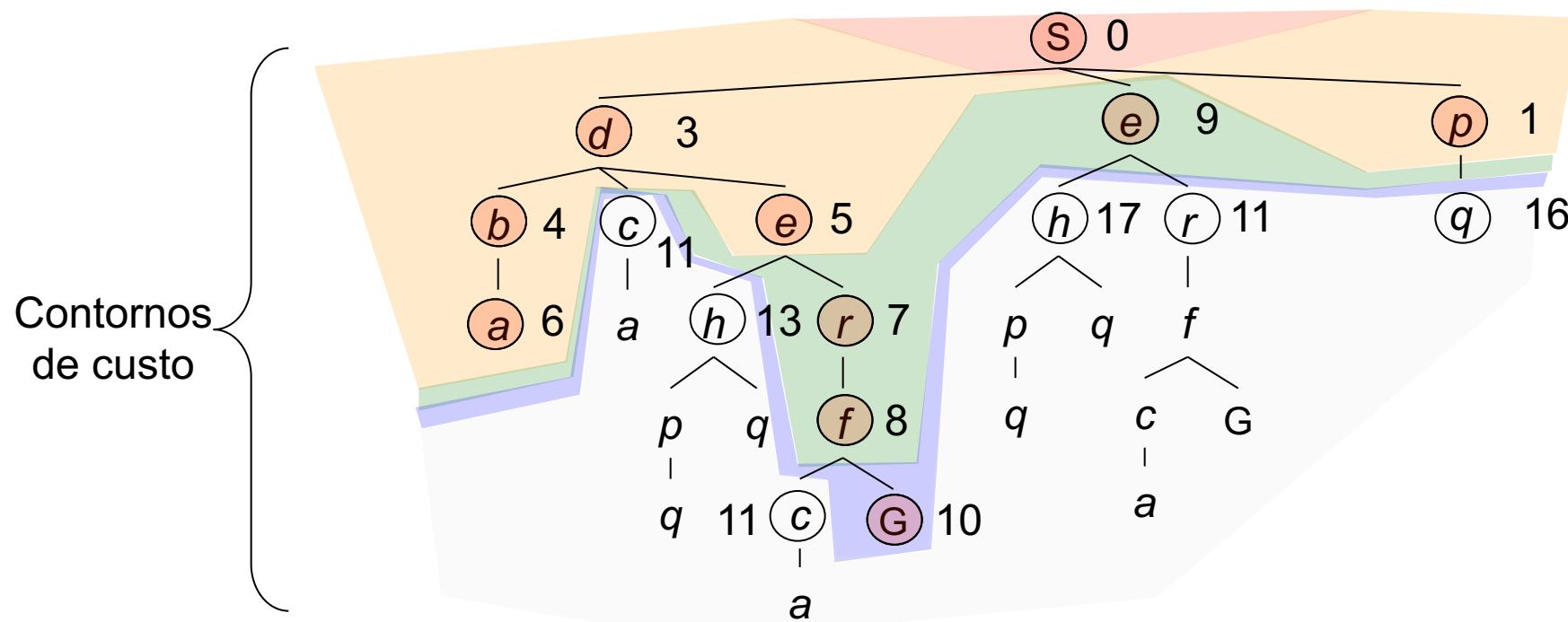
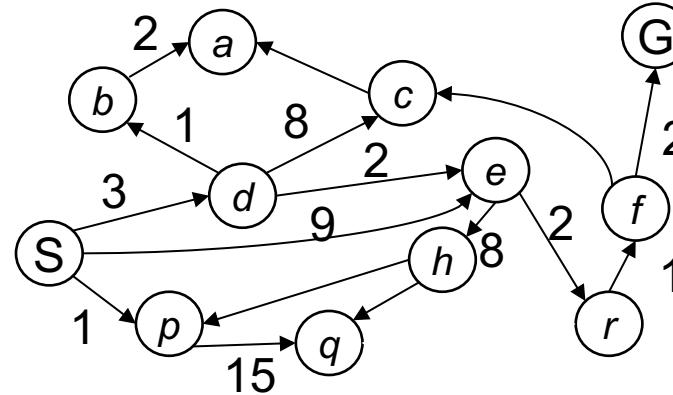
Uniform Cost Search



Estratégia: expandir primeiramente um nó menos custoso

*Fronteira é uma Fila de prioridade
(prioridade = custo acumulado)*

Uniform Cost Search



Uniform Cost Search (UCS) .:propriedades:.

- Expansão?

- Todos os nós com custo menor do que da solução mais barata.
- Se solução custa C^* e arcos custam pelo menos ε , \rightarrow profundidade efetiva $= C^*/\varepsilon$
- Tempo = $O(b^{C^*/\varepsilon})$

- Espaço?

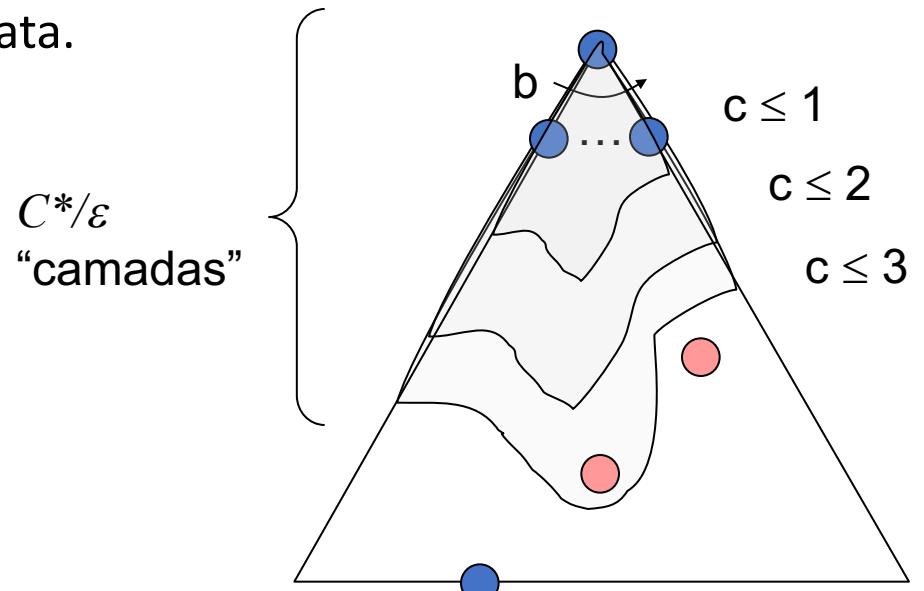
- Apenas última camada $\rightarrow O(b^{C^*/\varepsilon})$

- Completude?

- Sim!

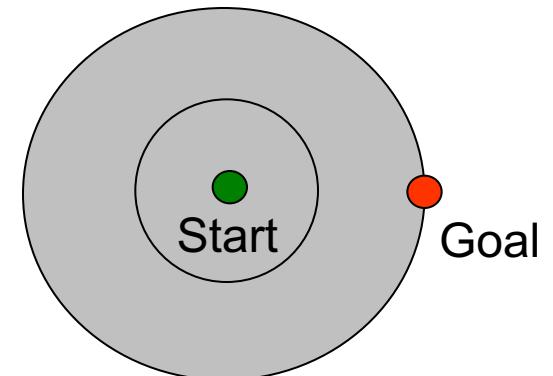
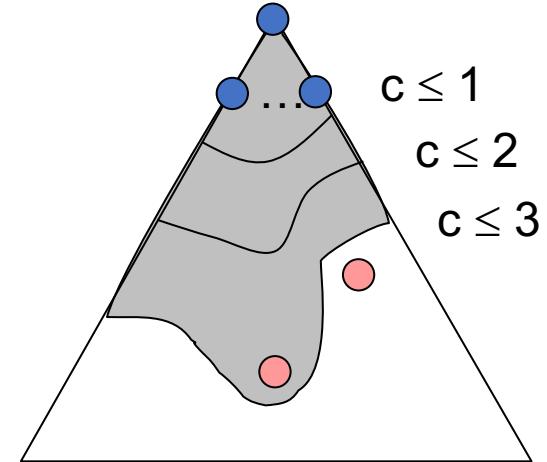
- Optimalidade?

- Sim!



Uniform Cost : observações

- UCS explora contornos de custos crescentes
- UCS é completo e ótimo (BOM)
- UCS explora opções em todas as direções (não há dica sobre local do estado objetivo) (RUIM)



Q: DFS vs. BFS vs. UCS; Quem é quem?



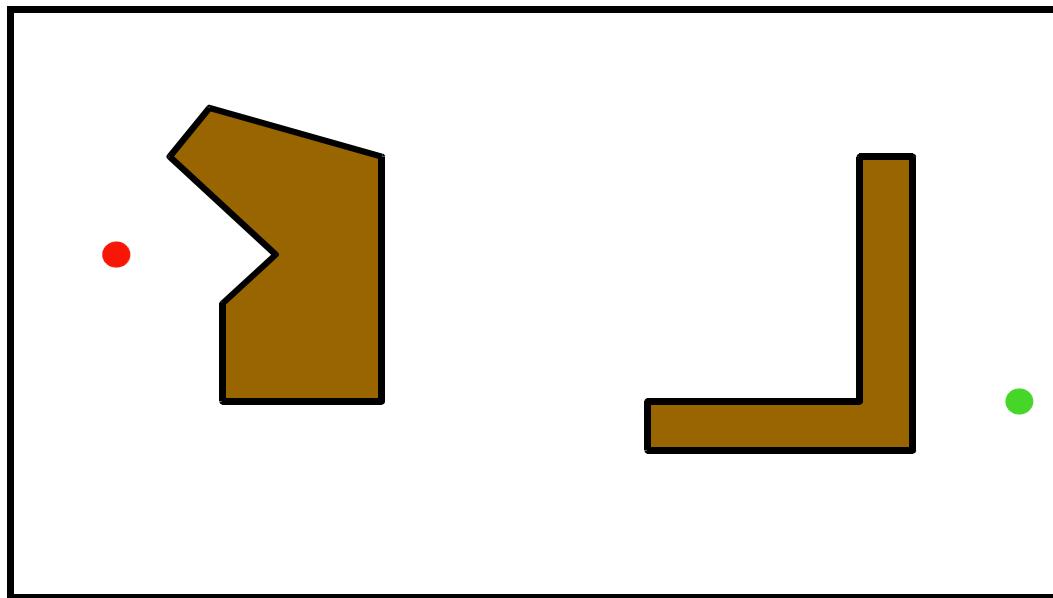
Busca age sobre modelos do mundo

- O agente não tenta todos os possíveis planos para aquele problema!
- A busca é tão boa quanto for seu modelo do mundo

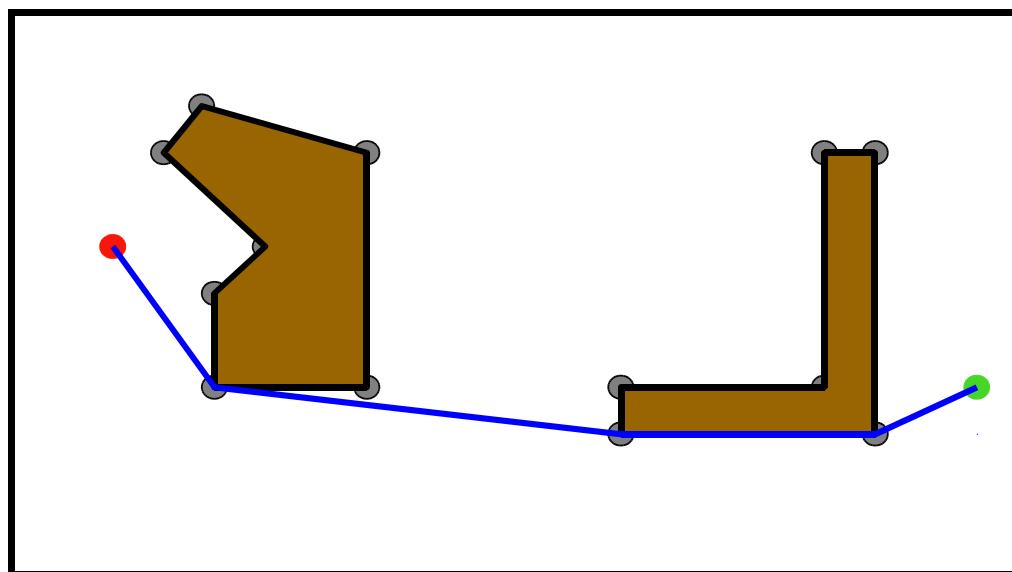
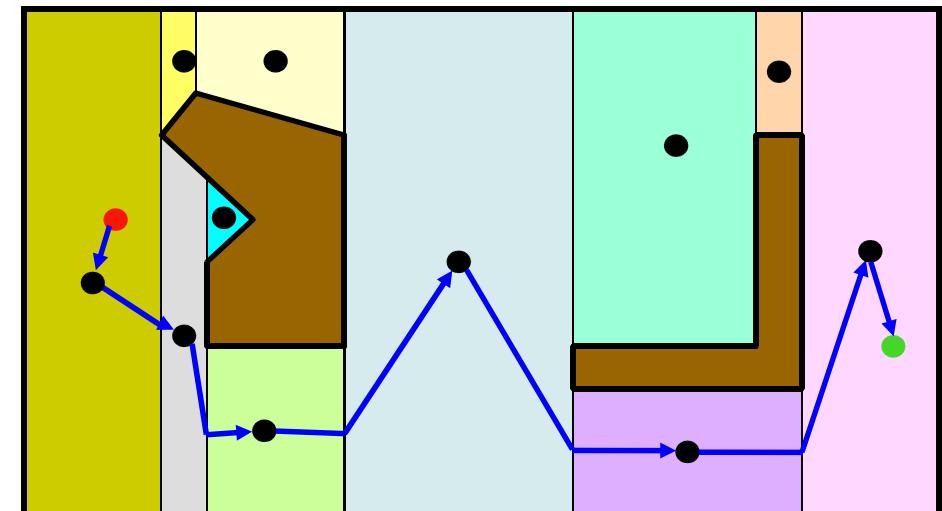
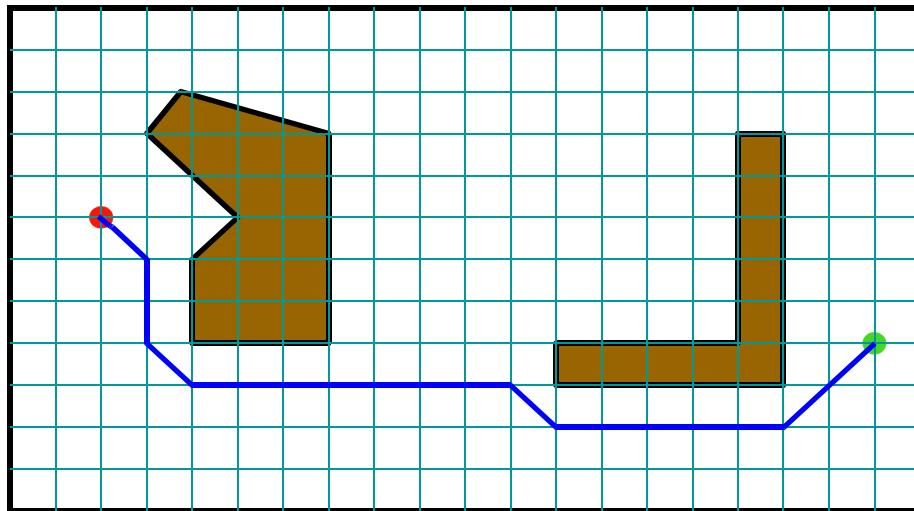


Podem existir diferentes modelos para o mesmo problema

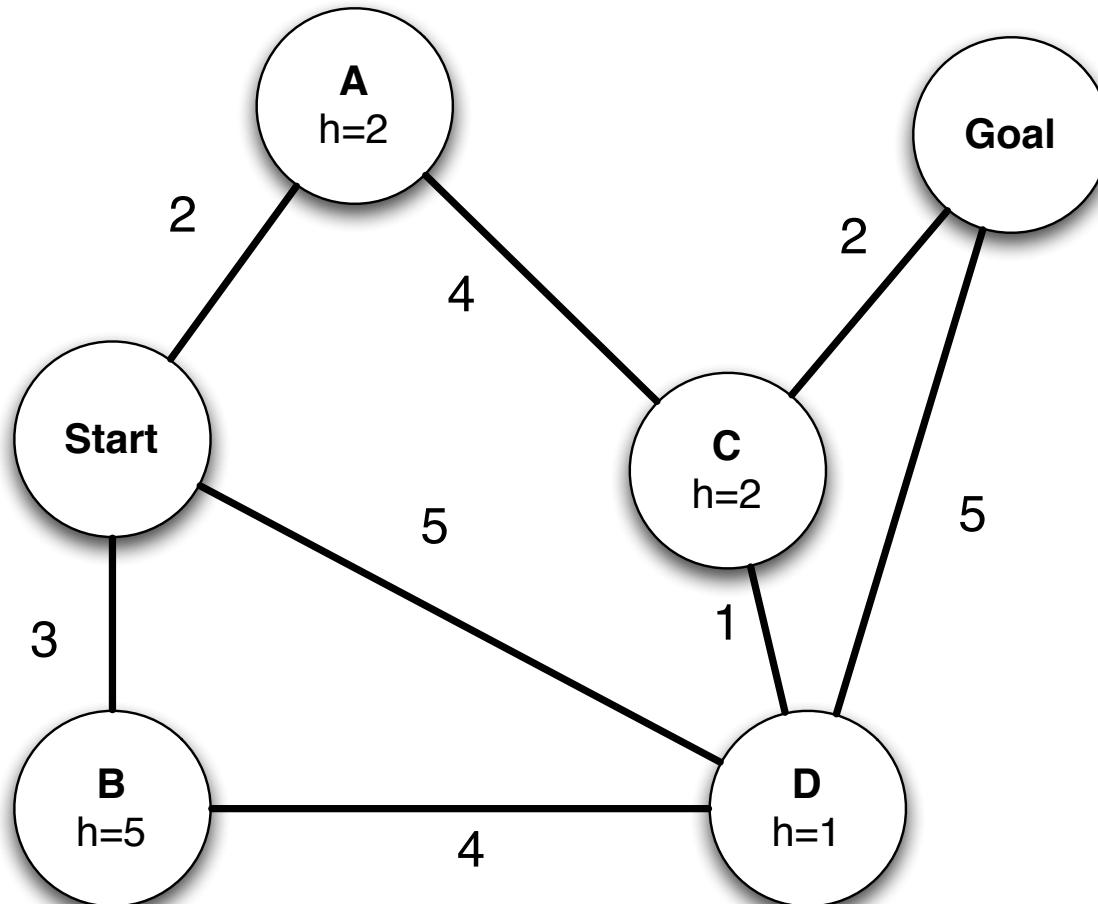
Qual o Espaço de Estados?



Diferentes espaços de estado

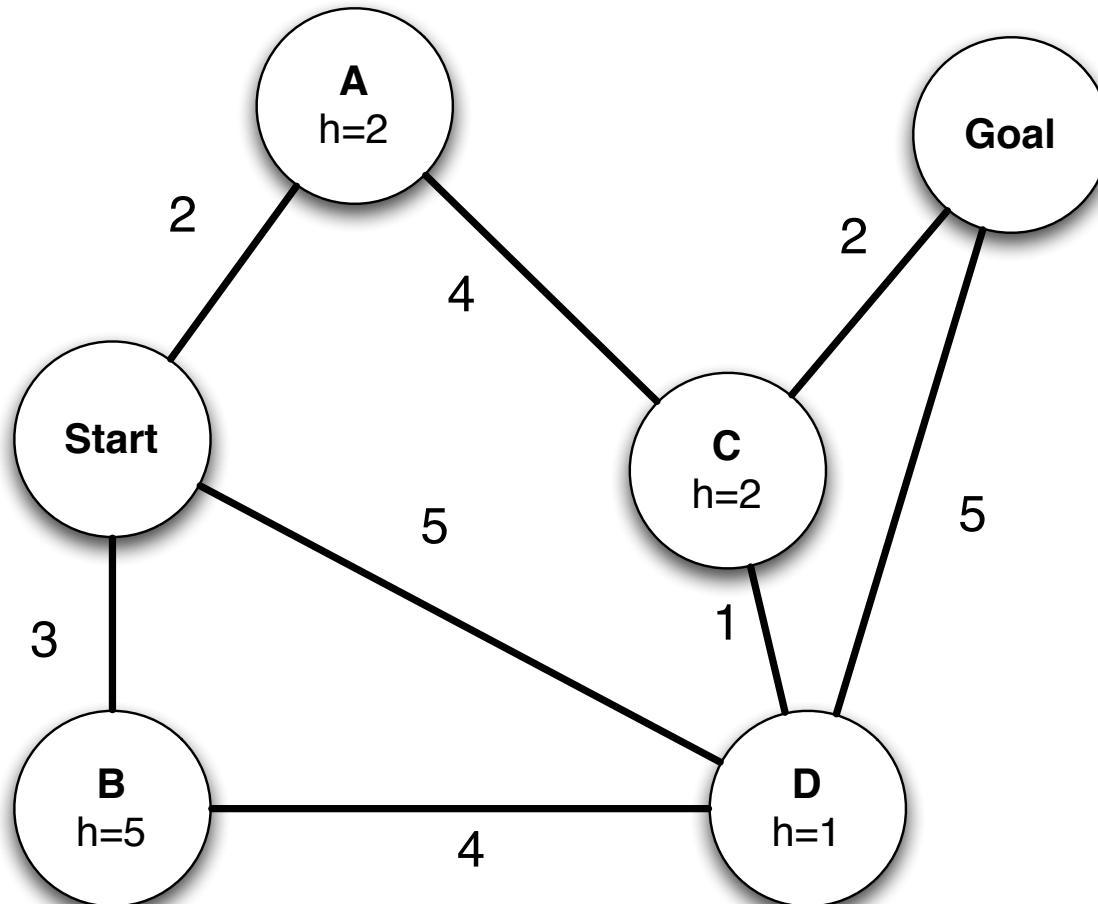


Q: Para cada estratégia de busca em grafo, mostre a ordem em que os estados são expandidos e o caminho da solução retornado pela busca (empates resolvem-se em ordem alfabética)



- a) Depth-first search
Estados expandidos: ??
Caminho retornado: ??
- b) Breadth First Search
Estados expandidos: ??
Caminho retornado: ??
- c) Uniform-Cost Search
Estados expandidos: ??
Caminho retornado: ??

Q: Para cada estratégia de busca em grafo, mostre a ordem em que os estados são expandidos e o caminho da solução retornado pela busca (empates resolvem-se em ordem alfabética)



a) Depth-first search.

Expandidos: Start, A, C, D, B, Goal

Retornados: Start-A-C-D-Goal

b) Breadth First Search

Expandidos: Start, A, B, D, C, Goal

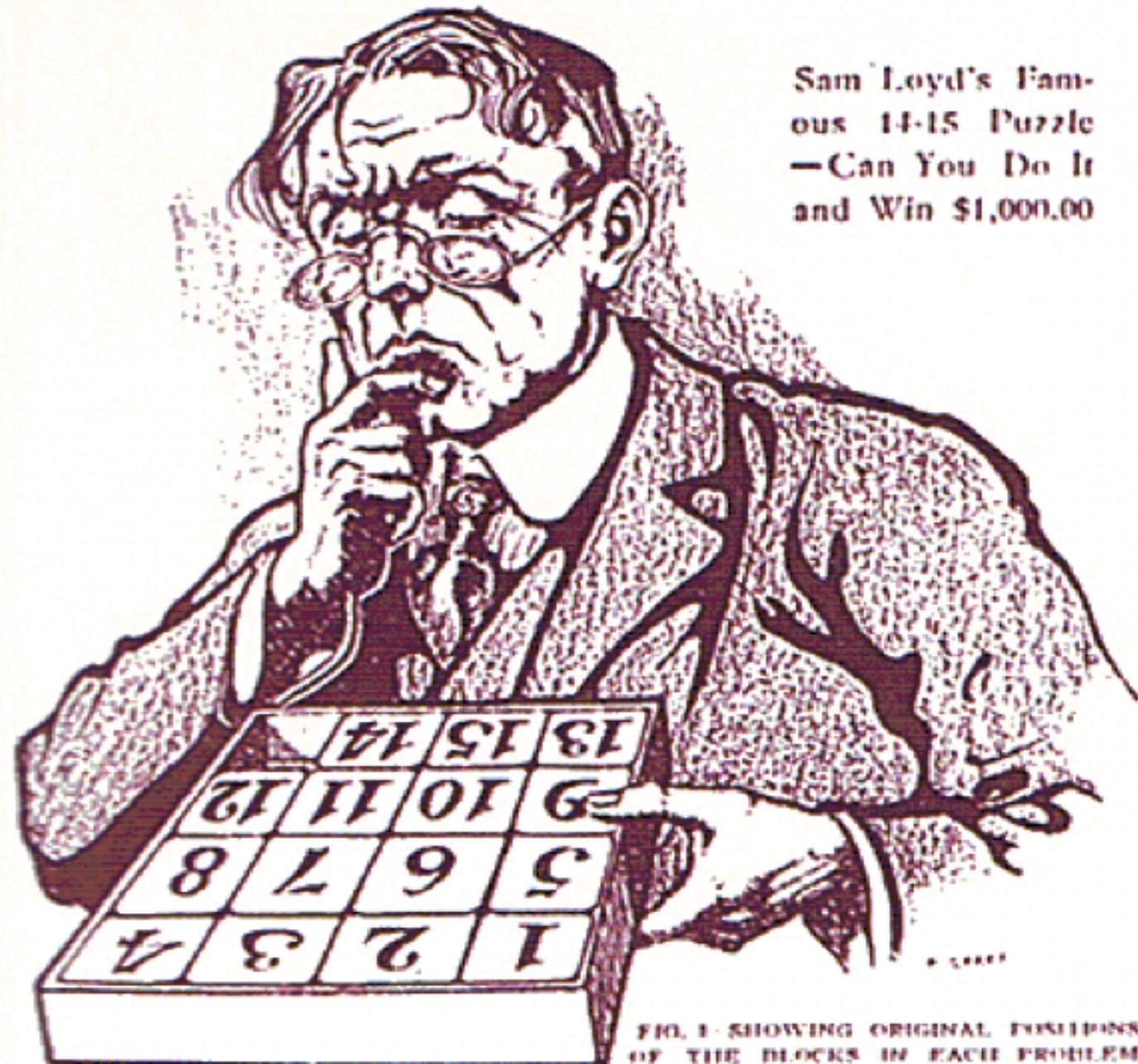
Retornados: Start-D-Goal

c) Uniform-Cost Search

Expandidos: Start, A, B, D, C, Goal

Retornados: Start-A-C-Goal

A \$1,000.00 Cash Prize Puzzle



Sam Loyd's Famous 14-15 Puzzle
—Can You Do It
and Win \$1,000.00

FIG. 1 SHOWING ORIGINAL POSITIONS
OF THE BLOCKS IN EACH PROBLEM



Hendrik Macedo

Escreve sobre Inteligência Artificial no Saense.

<http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/>