# DCA0121 – INTELIGÊNCIA ARTIFICIAL APLICADA
# Aula 9 – Métodos de Busca Informada

Prof. Marcelo Augusto Costa Fernandes

mfernandes@dca.ufrn.br

# Introdução

- Também chamada de busca heurística
- Possuem duas estratégias fundamentais
  - Funções Heurísticas
  - Funções para cálculo de custo
- Um dos pontos importantes da utilização de heurísticas é podar ou eliminar ramos do espaço de busca.
- Função de avaliação

$$f(n) = g(n) + h(n)$$

$$g(n): \text{Função custo}$$

$$h(n): \text{Função heurística}$$

# Busca pela melhor escolha (*BEST-FS - Best-First Search*)

- O espaço de busca (espaço de estados) é criado através de uma função de avaliação (*evaluation function*)
- Pode ser implementado por uma fila com prioridades, semelhante a busca uniforme (não informada)
- Função de avaliação

$$f(n) = h(n)$$

$$h(n): \text{Função heurística}$$

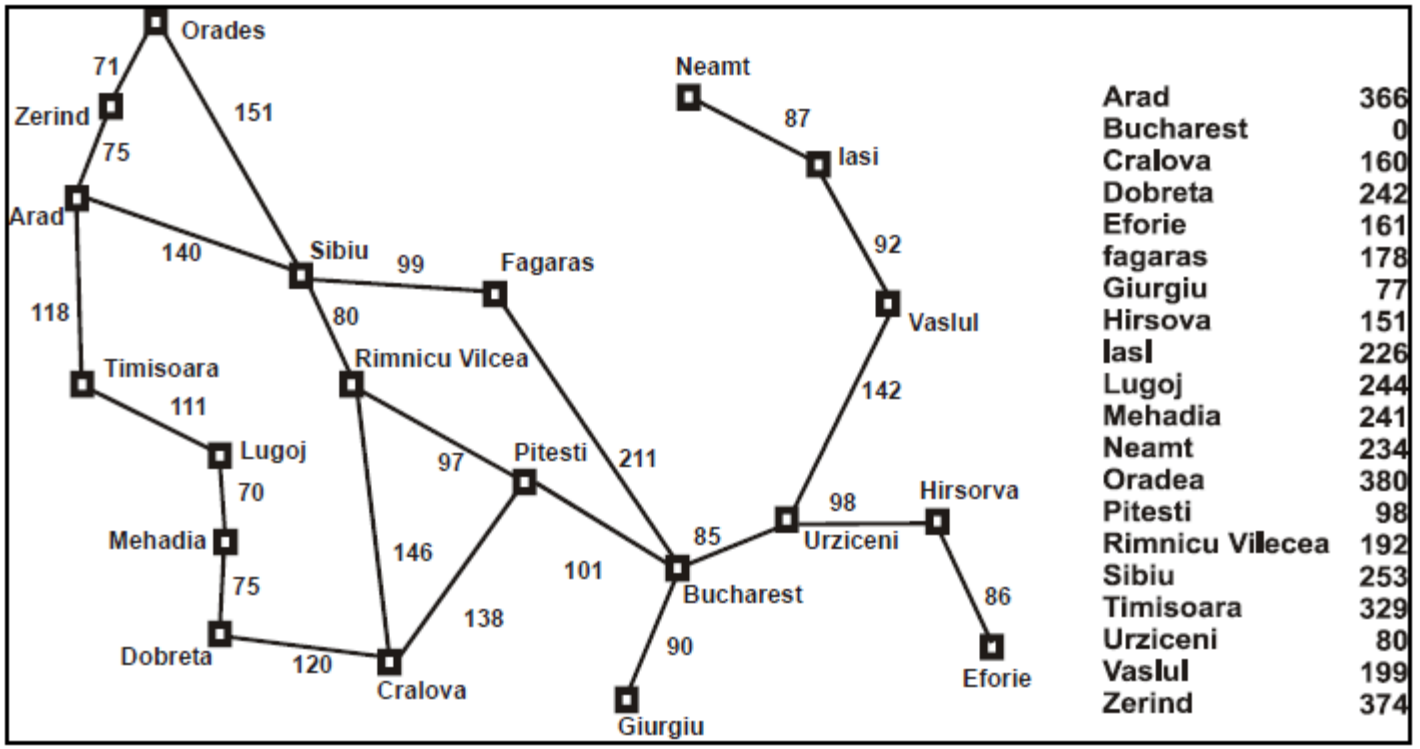
# Busca pela melhor escolha (*BEST-FS - Best-First Search*)

- Expande o estado que possui o menor custo estimado até o estado final
- O algoritmo mantém duas listas. A primeira, aberta, representa o conjunto de nós que ainda podem ser pesquisados e utilizados na composição do percurso.
- Já a lista fechada contem todos nós que já foram visitados, e que não precisam ser examinados novamente.
- Utiliza uma função heurística que é caracterizada pelo percurso mais econômico do *n*-ésimo estado ao estado final.
- Algoritmo de busca não completo e não ótimo
  - Depende da função heurística

# Busca pela melhor escolha (*BEST-FS - Best-First Search*)

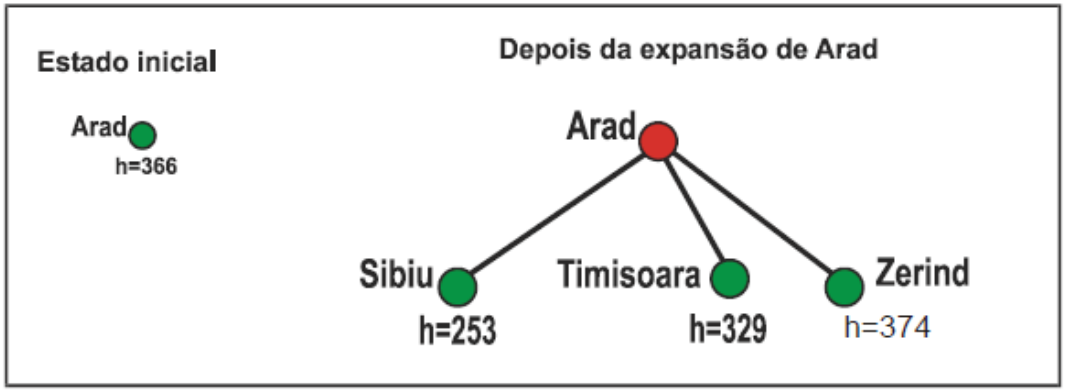
- Complexidade espacial: $O(b^d)$
- Complexidade temporal: $O(b^d)$
- d - Profundidade da solução encontrada
- b - Fator de Ramificação (Branching Factor)
  - número máximo de sucessores de um nó
- Implementação
  - Pode-se implementar com uma Fila de prioridade
  - Os estados são ordenados pelo valor de $h(n)$

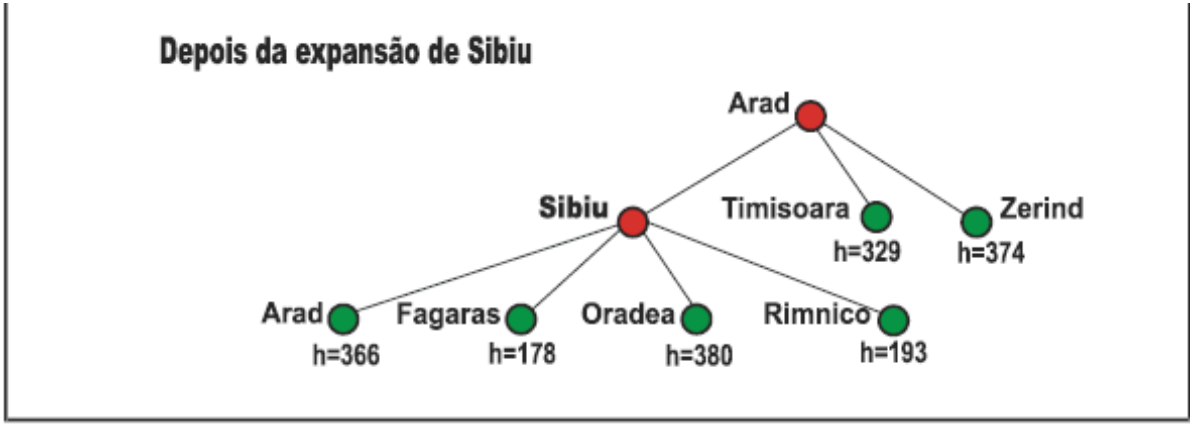
# Exemplo

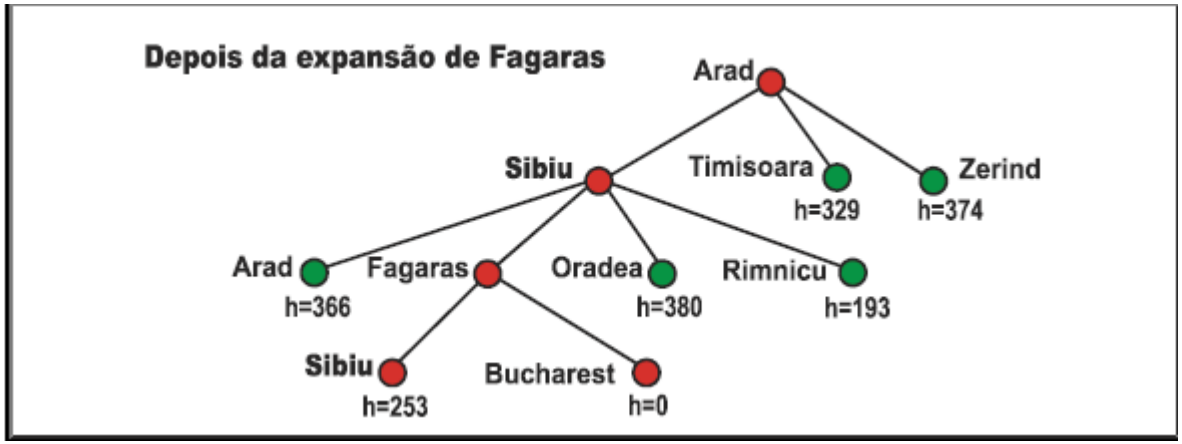



- Utiliza a heurística da distância em linha reta

# Exemplo


Estado inicial
Arad
h=366
Depois da expansão de Arad
Arad
Sibiu
h=253
Timisoara
h=329
Zerind
h=374


Depois da expansão de Sibiu
Arad
Sibiu
Timisoara
h=329
Zerind
h=374
Arad
h=366
Fagaras
h=178
Oradea
h=380
Rimnico
h=193


Depois da expansão de Fagaras
Arad
Sibiu
Timisoara
h=329
Zerind
h=374
Arad
h=366
Fagaras
Oradea
h=380
Rimnicu
h=193
Sibiu
h=253
Bucharest
h=0

# Algoritmo

1. Crie uma lista, OPEN(aberta), na qual o nó inicial é inserido.
2. Se OPEN está vazia, então algoritmo termina.
3. Remova de OPEN o n-ésimo nó em que o valor de $f(n)$ seja o menor possível, e o adicione em uma lista CLOSED(fechada).
4. Expanda o n-ésimo nó.
5. Se qualquer um dos nós expandidos a partir do n-ésimo é o nó de destino, retorne sucesso e a solução (fazendo o caminho de volta do estado inicial até o n-ésimo nó).
6. 6. Para cada um dos nós expandidos:
    1. Calcule o valor de $f(n)$;
    2. Se o nó expandido não se encontra em quaisquer uma das listas, adicione-o à lista OPEN.
7. Vá para 2.

# Algoritmo de Dijkstra

$$f(n) = g(n)$$

$$g(n): \text{Função custo}$$

- A função de avaliação é composta apenas pela função custo, g(n), que leva em consideração o quanto foi percorrido entre o nó inicial e o n-ésimo nó.
- Algoritmo semelhante ao BFS sendo modificado apenas no cálculo de $f(n)$
- No passo 6 do algoritmo do Dijkstra, verificar se o nó expandido já se encontra na lista aberta, mas apresenta um valor menor do que o que encontrado antes, devendo portanto ser substituído.

# Algoritmo de Dijkstra

- Diferentemente do BFS, o algoritmo de Dijkstra sempre encontra o percurso ótimo entre o estado inicial e o estado objetivo.
  - Porém o custo não pode ser negativo
  - Devem existir memória e tempo suficientes para a busca

# Algoritmo A* (A-Star)

- Junção entre o BFS e o Dijkstra
- Utiliza a função custo e a função heurística

$$f(n) = g(n) + h(n)$$

$$g(n): \text{Função custo}$$

$$h(n): \text{Função heurística}$$

$g(n)$ é o custo acumulado para atingir o nó e $h(n)$ é uma estimativa heurística da distância real até o nó destino.

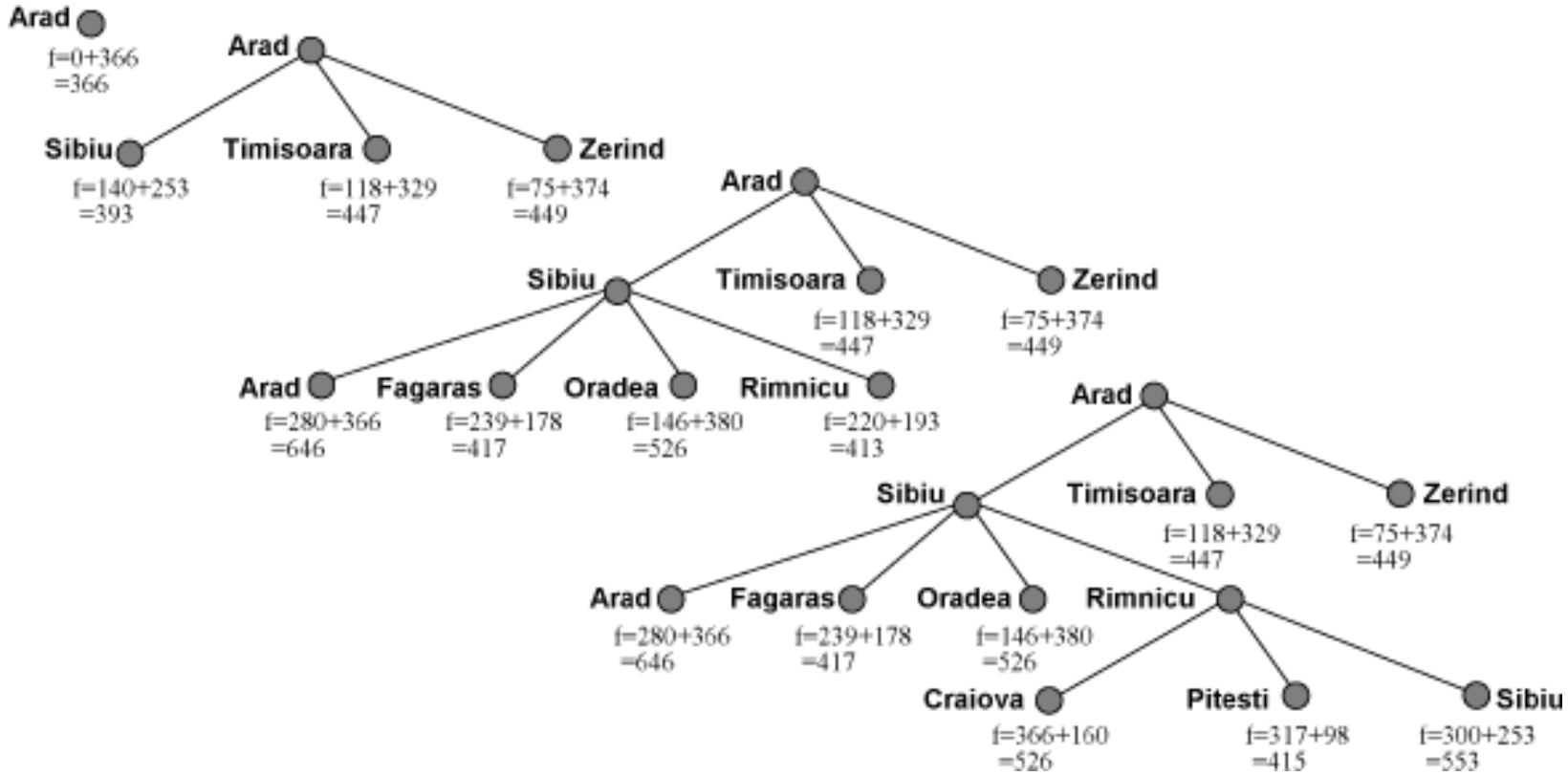
# Algoritmo A* (A-Star)

- Considera os custos do nó inicial até o nó corrente e o custo do nó corrente até o nó final.
- Algoritmo capaz de encontrar a melhor solução, onde o custo depende do contexto em que está sendo aplicado, que por exemplo pode ser quantidade de nós do caminho ou tempo gasto para se chegar ao destino.
- Pode ser implementado com duas listas (Aberta e Fechada)

# Algoritmo A* (A-Star)

1. Adicione o nó inicial à lista OPEN.
2. Enquanto a lista OPEN não estiver vazia, faça:
    1. O nó corrente será o nó de menor custo da lista OPEN.
    2. Se o nó corrente for igual ao nó destino, então o caminho está completo e o algoritmo termina.
    3. Se não, mova o nó corrente da lista OPEN para a lista CLOSED e examine cada nó adjacente ao nó corrente.
    4. Para cada nó adjacente, faça:
        1. Calcule o valor de $f(n)$
        2. Se o nó não está na lista OPEN e não está na lista CLOSED
            1. Mova-o para a lista OPEN
        3. Se o nó esta na lista OPEN ou CLOSED e tem menor custo $g(n)$
            1. Retire-o da lista OPEN ou CLOSED
            2. Mova-o para a lista OPEN

# Algoritmo A* (A-Star)



# Algoritmo A* (A-Star)

- Quando o algoritmo termina, a lista FECHADOS contém todos os nós acessados pelo algoritmo.
- Estratégia completa e ótima
- Complexidade temporal: Depende a função heurística
- Complexidade espacial: $O(b^d)$

# Tópicos sobre heurísticas

- A função heurística informa a estimativa do menor custo n-ésimo nó até o destino, sendo importante para um bom funcionamento do A* a escolha de uma boa função heurística.
- Se a heurística for igual a zero, o algoritmo A* irá funcionar de forma semelhante ao algoritmo de Dijkstra
- Se considerarmos uma heurística onde o seu valor é relativamente alto ao valor de $g(n)$, o algoritmo se torna um BFS.
- Assim é necessário escolher uma heurística admissível, onde nunca se superestime o custo para se chegar ao destino.

# Distância de Manhattan

- D representa o custo para se mover de um nó n para um nó adjacente.
  - Assim o valor total será D vezes a soma das diferenças absolutas das coordenadas do nó corrente e destino.
- $x(\mathrm{n})$ e $y(n)$ correspondem a posição atual.
- $x_d(n)$ e $y_d(n)$ correspondem a posição destino.

$$f(n) = D\left(\left|x(n) - x_d(n)\right| + \left|y(n) - y_d(n)\right|\right)$$

# Distância Euclidiana

$$f(n) = D\sqrt{\left(x(n) - x_d(n)\right)^2 + \left(y(n) - y_d(n)\right)^2}$$

# Outros métodos

- IDA* (Iterative Deepening A*)
  - igual ao aprofundamento iterativo, porém seu limite é dado pela função de avaliação (f) , e não pela profundidade (d). necessita de menos memória do que A*
- SMA* (Simplified Memory-Bounded A*)
  - O número de nós guardados em memória é fixado previamente

# Bibliografia


- Capítulo 3
  - Jones , M. Tim. Artificial Intelligence - A Systems Approach. Jones & Bartlett Publishers. 2007.
- Capítulo 4
  - Russell, Stuart. Artificial Intelligence: A Modern Approach. Prentice Hall. 2009.