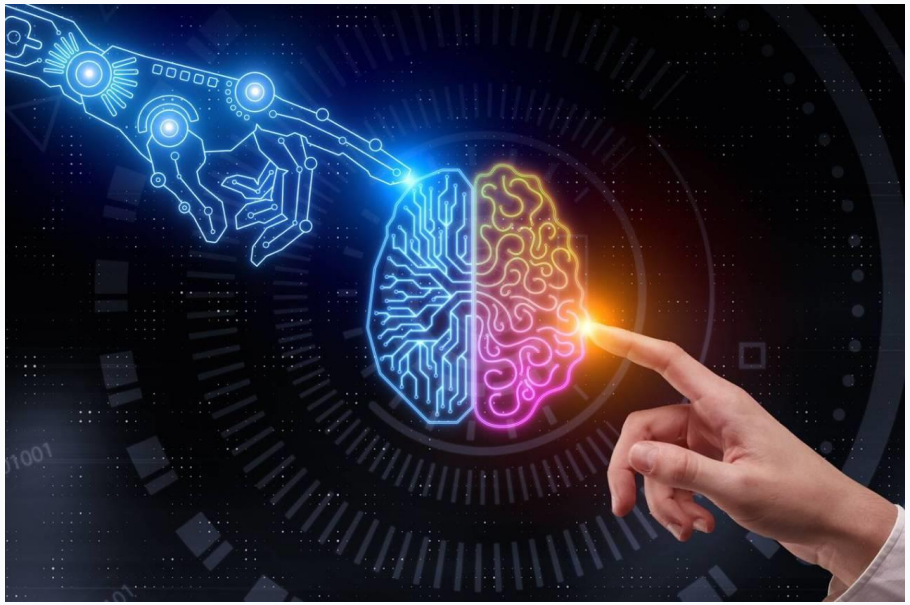


Inteligência Artificial

Profº - Dr. Thales Levi Azevedo Valente

thales.l.a.valente@gmail.com.br

Governo Federal

Grupo da turma 2024.2

https://chat.whatsapp.com/JFB6CgOI7lMCoYmoIKEK62

Sejam Bem-vindos !

Os celulares devem ficar no silencioso ou desligados

Pode ser utilizado apenas em caso de emergência

Boa tarde/noite, por favor e com licença DEVEM ser usados

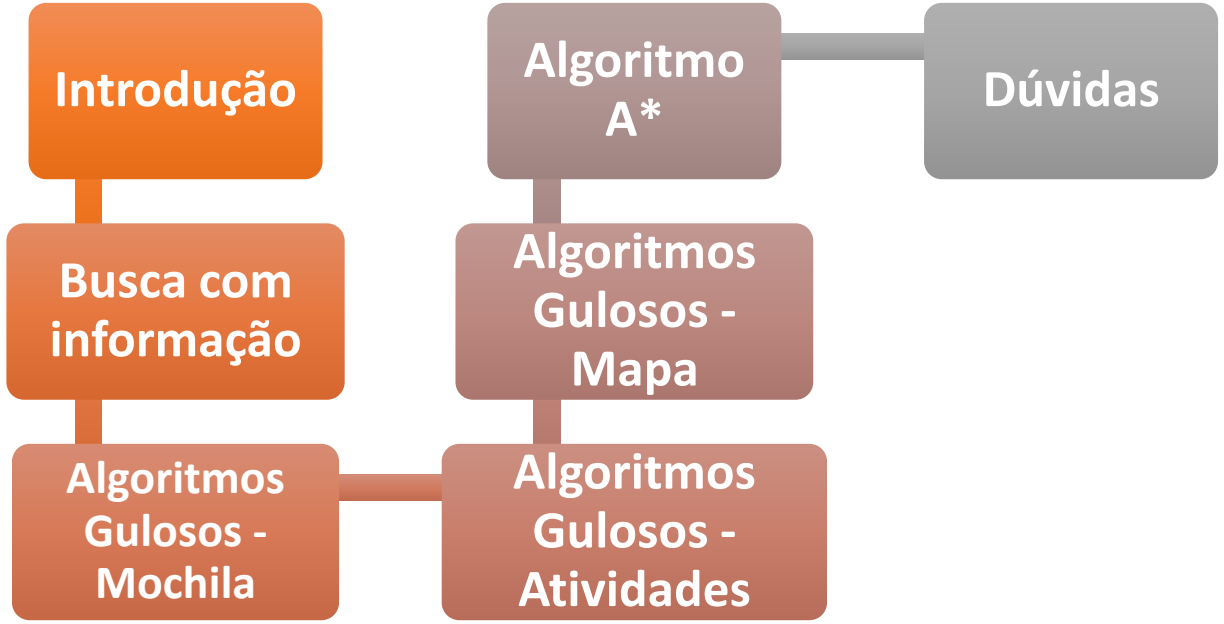
Educação é essencial

Objetivos de hoje

Apresentar os principais conceitos de busca sem informação;

Ao final da aula, os alunos serão capazes de ter uma visão geral do funcionamento dos principais algoritmos de busca cem informação.

Roteiro: Metodologias de busca



Introdução

- **Na área de otimização um problema comum é determinar uma solução que seja a melhor possível (a solução ótima)**
  - ✓ Considera-se um conjunto de restrições
  - ✓ A solução ótima também é chamada de solução global ou melhor escolha

- **Podemos fazer a seguinte pergunta: como chegar a melhor escolha?**
  - ✓ Devo usar estratégias exaustivas (que exploram todos os caminhos possíveis)?
  - ✓ Devo reduzir o espaço de busca ?
  - ✓ Devo considerer escolhas locais ?

Busca com informação (ou heurística)

- **Utiliza conhecimento específico sobre o problema para encontrar soluções de forma mais eficiente do que a busca cega**
  - ✓ Conhecimento específico além da definição do problema

- **Abordagem geral: busca pela melhor escolha**
  - ✓ Utiliza uma função de avaliação para cada nó
  - ✓ Expande o nó que tem a função de avaliação mais baixa
  - ✓ Dependendo da função de avaliação, a estratégia de busca muda

Busca pela melhor escolha

- **Ideia: usar uma função de avaliação f(n) para cada nó**
  - ✓ Estimativa do quanto aquele nó é desejável
  - ✓ Expandir nó mais desejável que ainda não foi expandido

- **Implementação**
  - ✓ Ordenar nós na borda em ordem decrescente de acordo com a função de avaliação

Algoritmos Gulosos - greedy algorithms

- **Ideia: em cada etapa do processo de construção da solução, faz-se a escolha local que parece ser a melhor naquele momento**
  - ✓ Esperança de que a soma dessas boas escolhas locais resulte em uma solução globalmente ótima
  - ✓ Mesmo quando não há garantia de otimalidade, o guloso pode fornecer uma solução boa o suficiente em tempo reduzido, sendo útil em situações práticas em que tempo é mais crítico do que alcançar a perfeição

- **Definição**
  - ✓ Um algoritmo guloso é uma estratégia de solução de problemas que constrói a resposta passo a passo, sempre fazendo a escolha que parece trazer o maior benefício ou a menor perda imediata, sem reconsiderar decisões anteriores

Algoritmos Gulosos - greedy algorithms

- **Passos**
  1. Identifica-se o conjunto de escolhas possíveis para resolver uma parte do problema
  2. Escolhe-se aquela que parece a melhor do ponto de vista local (ou seja, a que traz a maior vantagem imediata)
  3. Essa escolha é incorporada à solução parcial
  4. Passa-se então para o próximo subproblema, repetindo o processo até não haver mais o que decidir

- **Particularidade**
  - ✓ Algoritmo guloso não volta atrás em suas decisões. Cada escolha é definitiva.
  - ✓ Normalmente mais simples e rápido, porém não é garantido que encontre sempre a solução globalmente ótima

Algoritmos Gulosos - greedy algorithms

Exemplo - O Problema da Mochila Fracionária

- **Considere o seguinte cenário clássico: você tem uma mochila com uma certa capacidade de peso e uma coleção de itens, cada um com um valor monetário e um peso. Você quer encher a mochila para obter o maior valor possível. Porém, neste exemplo, é permitido colocar partes (frações) de um item na mochila.**
  - ✓ Esse é o chamado problema da mochila fracionária.
- **Enunciado. Suponha que você tenha uma mochila que aguenta 50 kg. Há três itens disponíveis:**
  - ✓ Item A: valor = 60 reais, peso = 10 kg
  - ✓ Item B: valor = 100 reais, peso = 20 kg
  - ✓ Item C: valor = 120 reais, peso = 30 kg

Algoritmos Gulosos - greedy algorithms

Exemplo - O Problema da Mochila Fracionária

- **Qual seria a idéia gulosa para resolver o problema? O objetivo é maximizar o valor dentro da mochila**
  - ✓ Carga mochila: 50 kg
  - ✓ Item A: valor = 60, peso = 10 kg
  - ✓ Item B: valor = 100, peso = 20 kg
  - ✓ Item C: valor = 120, peso = 30 kg

Algoritmos Gulosos - greedy algorithms

Exemplo - O Problema da Mochila Fracionária

- **Qual seria a idéia gulosa para resolver o problema?**
  - ✓ Carga mochila: 50 kg
  - ✓ Item A: valor = 60, peso = 10 kg
  - ✓ Item B: valor = 100, peso = 20 kg
  - ✓ Item C: valor = 120, peso = 30 kg

- O algoritmo guloso diz: "**Escolha primeiramente o item que lhe dá o maior valor por quilo**".
  - ✓ Calculemos a razão valor/peso

Algoritmos Gulosos - greedy algorithms

Exemplo - O Problema da Mochila Fracionária

- O algoritmo guloso diz: "**Escolha primeiramente o item que lhe dá o maior valor por quilo**".
  - ✓ Calculemos a razão valor/peso
  - ✓ Item A: 60/10 = 6 por kg
  - ✓ Item B: 100/20 = 5 por kg
  - ✓ Item C: 120/30 = 4 por kg
- O item A apresenta a melhor razão (6), seguido do item B (5) e depois o item C (4)
- **Com essa informação, qual a solução algorítmica?**

Algoritmos Gulosos - greedy algorithms

Exemplo - O Problema da Mochila Fracionária

- **Começamos então colocando o Item A inteiro na mochila**
  - ✓ Agora, a mochila tem 10 kg preenchidos, restam 40 kg de capacidade

- **Próxima escolha**: o Item B, com a segunda melhor razão. Colocamos o Item B inteiro (20 kg).
  - ✓ A mochila agora contém 30 kg (10 do A + 20 do B). Há ainda 20 kg disponíveis.

- **Finalmente, o Item C**: não é possível colocar ele inteiro (30 kg) porque só temos capacidade para 20 kg restantes
  - ✓ Porém, como é a versão fracionária do problema, podemos colocar 20/30 do Item C. Isso nos dá 2/3 do valor do C, ou seja, 2/3 de 120 = 80.

Algoritmos Gulosos - greedy algorithms

Exemplo - O Problema da Mochila Fracionária

- **O valor total obtido**
  - ✓ A: valor completo = 60
  - ✓ B: valor completo = 100
  - ✓ C: 2/3 de 120 = 80
  - ✓ Valor total = 60 + 100 + 80 = 240.

- **Esta estratégia gulosa, no caso da mochila fracionária, é ótima. Ou seja, a melhor escolha local (itens com maior razão valor/peso primeiro) leva à solução globalmente ótima. Este exemplo ilustra um caso em que o paradigma guloso funciona perfeitamente**

Algoritmos Gulosos - greedy algorithms

Comparando com outras abordagens

- **Versus Busca Cega ou Força Bruta**
  - ✓ A busca cega tentaria enumerar todas as combinações possíveis, encontrando com certeza a solução ótima, mas gastando muito mais tempo.
  - ✓ O guloso encontra rapidamente uma solução, embora não necessariamente a melhor, exceto em problemas muito bem estruturados.

- **Versus Programação Dinâmica (PD)**
  - ✓ Enquanto o algoritmo guloso faz uma escolha imediata e definitiva, a programação dinâmica geralmente requer armazenar soluções de subproblemas, compará-las e então montar uma solução global ótima

Algoritmos Gulosos - greedy algorithms

Quando o Algoritmo Guloso é Adequado?

- **Subestrutura Ótima**
  - ✓ A solução ótima global pode ser construída a partir de soluções ótimas de subproblemas.
  - ✓ Na mochila fracionária, a melhor distribuição final é composta por escolhas ótimas em cada etapa (selecionar o item com melhor valor/peso primeiro)

- **Escolha Gulosa Segura**
  - ✓ Sempre há uma escolha local que faz parte de alguma solução ótima global
  - ✓ Assim, no caso da mochila fracionária, escolher o item com maior valor/peso não inviabiliza a solução ótima final

Algoritmos Gulosos - greedy algorithms

Exemplo - Seleção de Atividades

- **Considere um conjunto de atividades, cada uma com um horário de início e término, e você deseja selecionar o máximo número possível de atividades que não se sobreponham no tempo. O algoritmo guloso para esse problema é simples e ótimo:**
  - ✓ Ordene as atividades pelo seu horário de término.
  - ✓ Escolha a atividade que termina mais cedo.
  - ✓ Em seguida, escolha a próxima atividade que começar depois que a primeira tiver terminado.
  - ✓ Repita o processo até não haver mais atividades compatíveis.

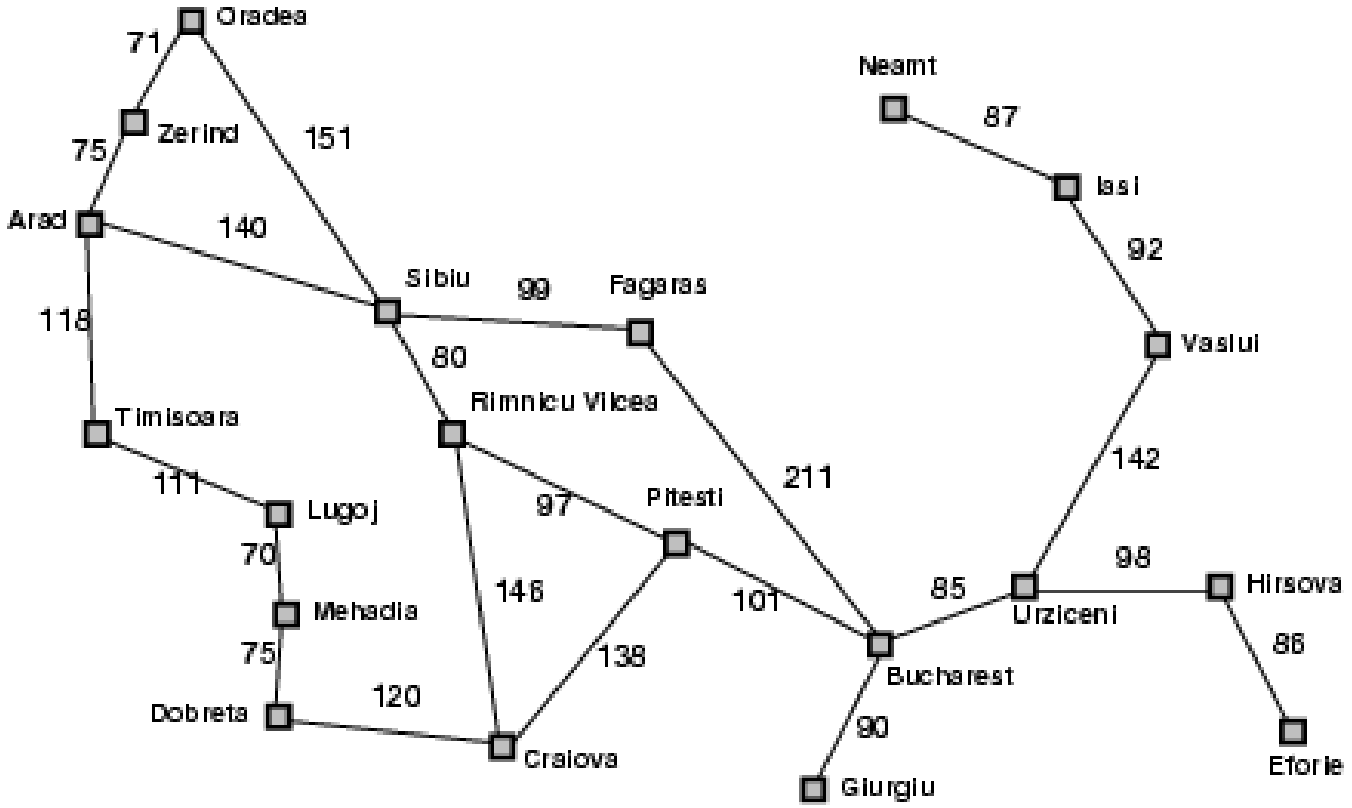
Algoritmos Gulosos - greedy algorithms

Exemplo - Seleção de Atividades

- **Essa estratégia gulosa — escolher sempre a atividade que termina mais cedo — é provavelmente ótima. Ou seja, os algoritmos gulosos não apenas são heurísticas: em diversos problemas eles são a melhor solução, simples e elegante**
- **Função de avaliação f(n) = h(n) (heurística) = estimativa do custo de n até o objetivo**
- **Busca gulosa pela melhor escolha expande o nó que parece mais próximo ao objetivo de acordo com a função heurística local.**

Algoritmos Gulosos - greedy algorithms

Exemplo: movimento no mapa com heurística

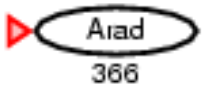


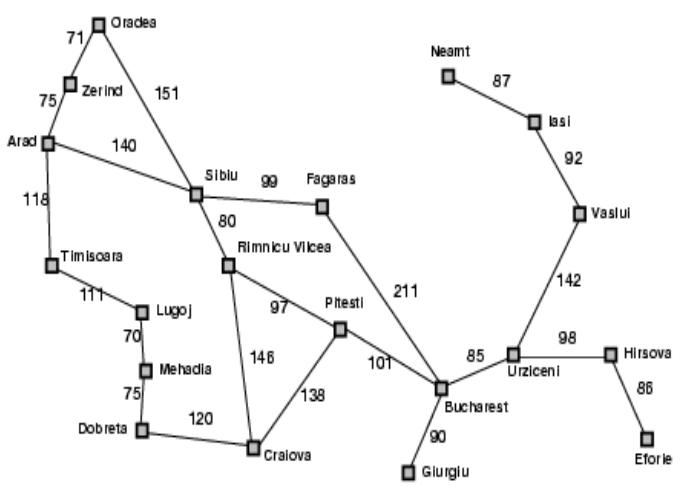
Distância em linha reta para Bucareste

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmos Gulosos - greedy algorithms

Exemplo: movimento no mapa com heurística

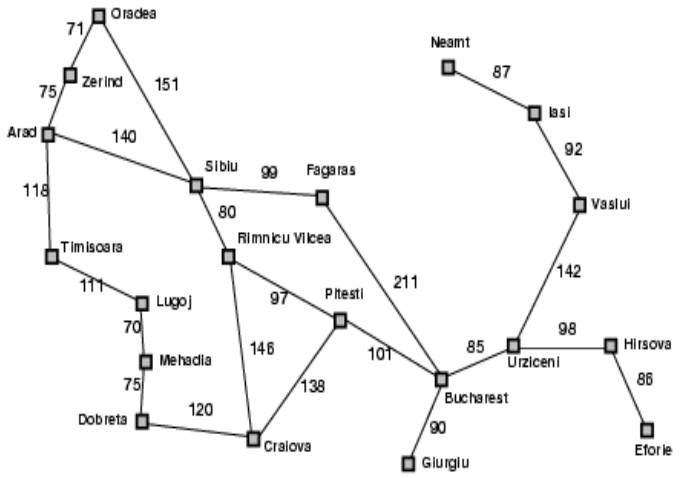




Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmos Gulosos - greedy algorithms

Exemplo: movimento no mapa com heurística

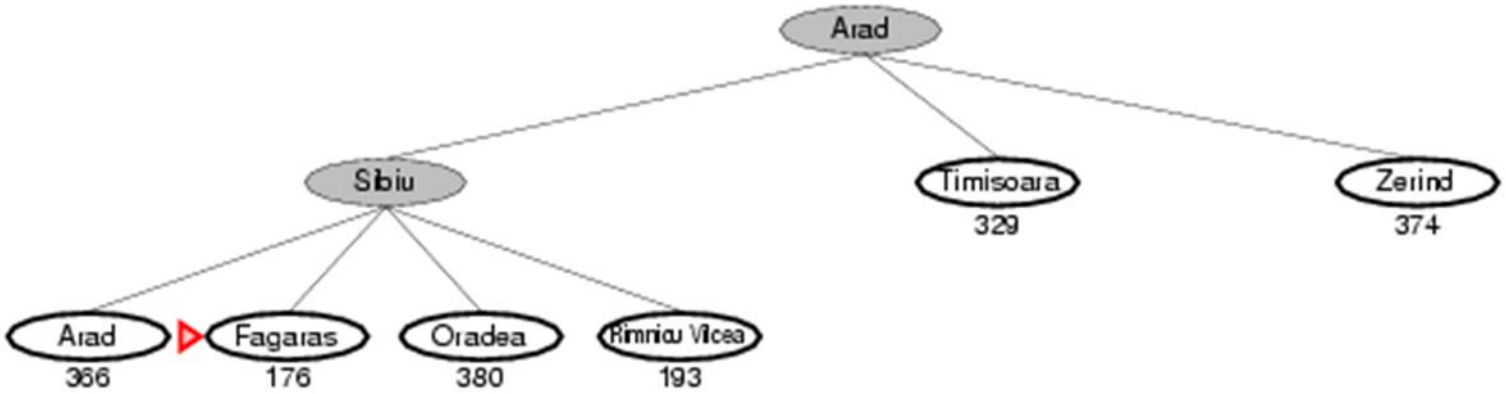


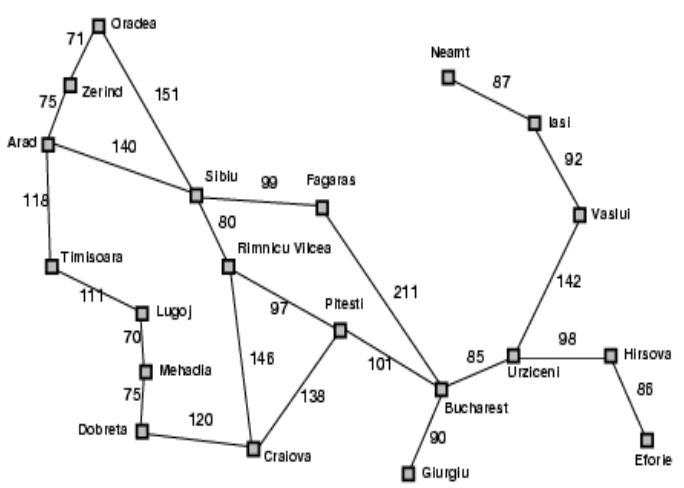
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Distância em linha reta para Bucareste

Algoritmos Gulosos - greedy algorithms

Exemplo: movimento no mapa com heurística



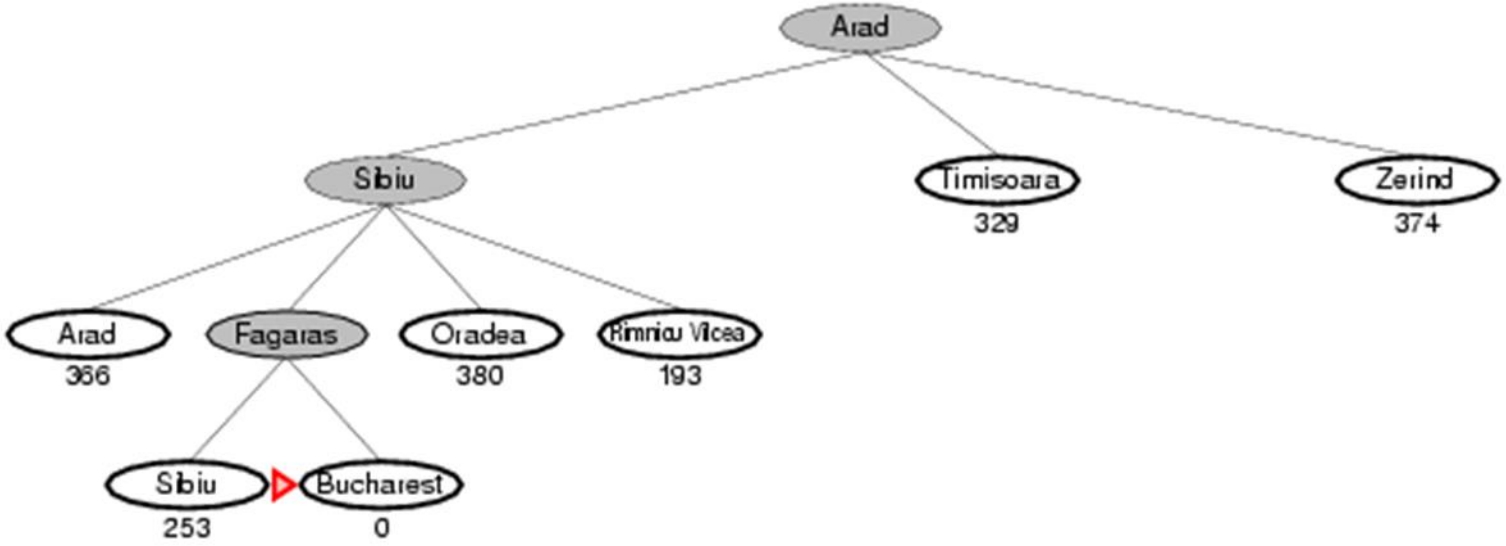


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

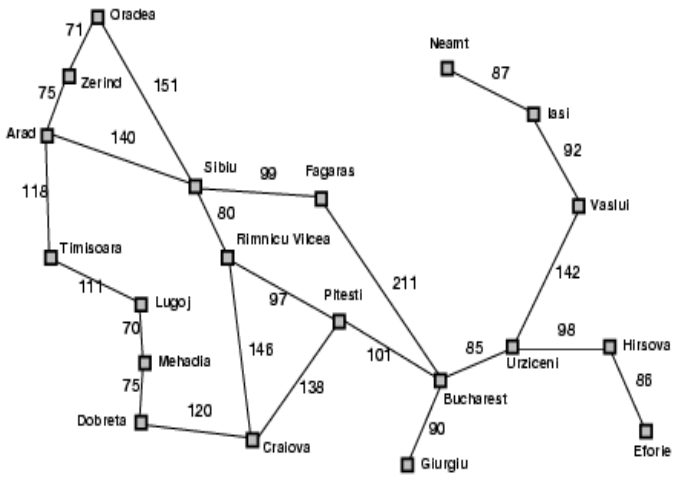
Distância em linha reta para Bucareste

Algoritmos Gulosos - greedy algorithms

Exemplo: movimento no mapa com heurística



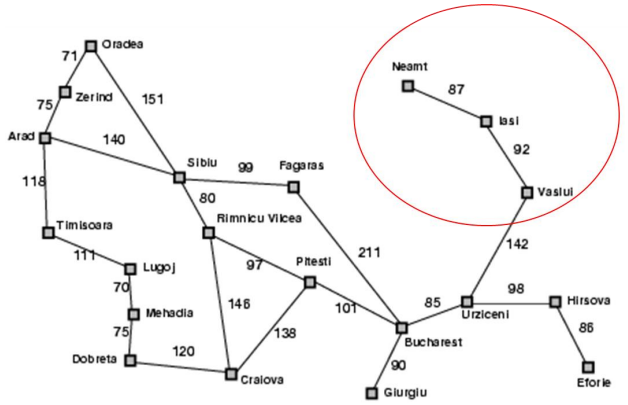
Distância em linha reta para Bucareste



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

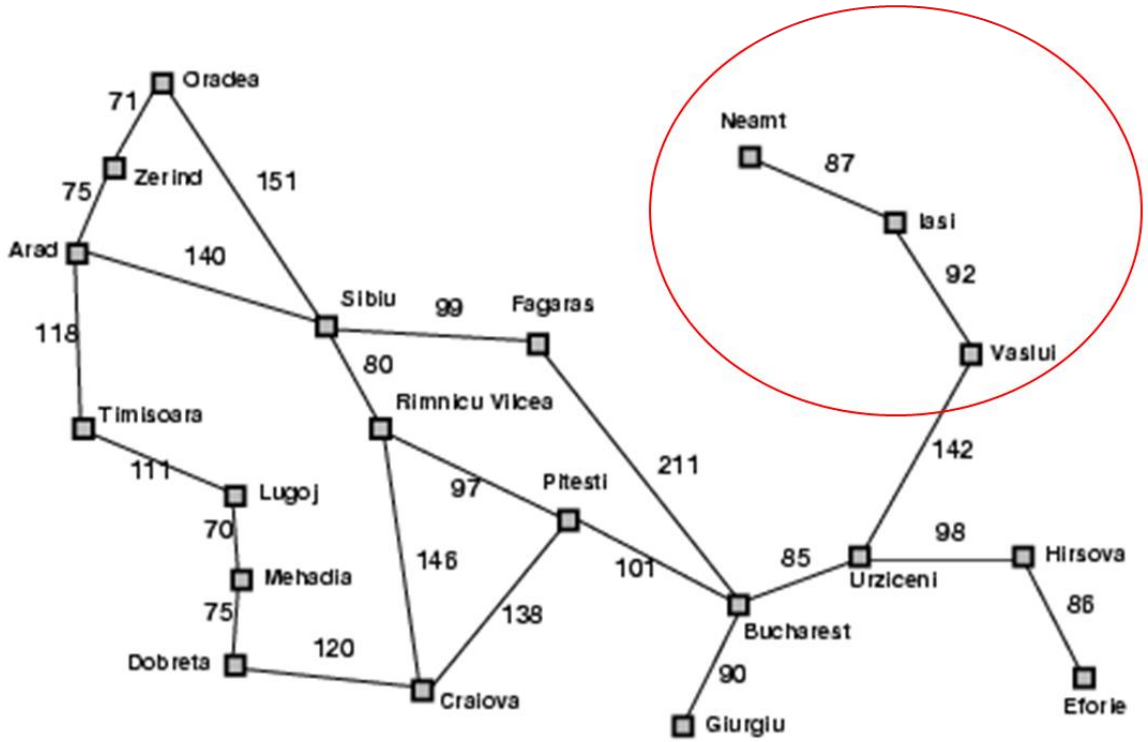
Algoritmos Gulosos - greedy algorithms

- **Em alguns casos, não é ótima, pois segue o melhor passo considerando somente o estado atual**
  - ✓ Pode haver um caminho melhor seguindo algumas opções piores em alguns pontos da árvore de busca
- **Minimizar h(n) é suscetível a falsos inícios**
  - ✓ Ex. Ir de Iasi a Fagaras
  - ✓ Heurística sugerirá ir a Neamt, que é um beco sem saída.
  - ✓ Se repetições não forem detectadas a busca entrará em loop



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmos Gulosos - greedy algorithms

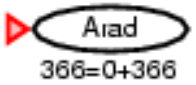


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmo A*

- **Idéia: evitar expandir caminhos que já são caros**
- **Função de avaliação f(n) = g(n) + h(n)**
  - **g(n) = custo até o momento para alcançar n**
  - **h(n) = custo estimado de n até o objetivo**
  - **f(n) = custo total estimado do caminho através de n até o objetivo**

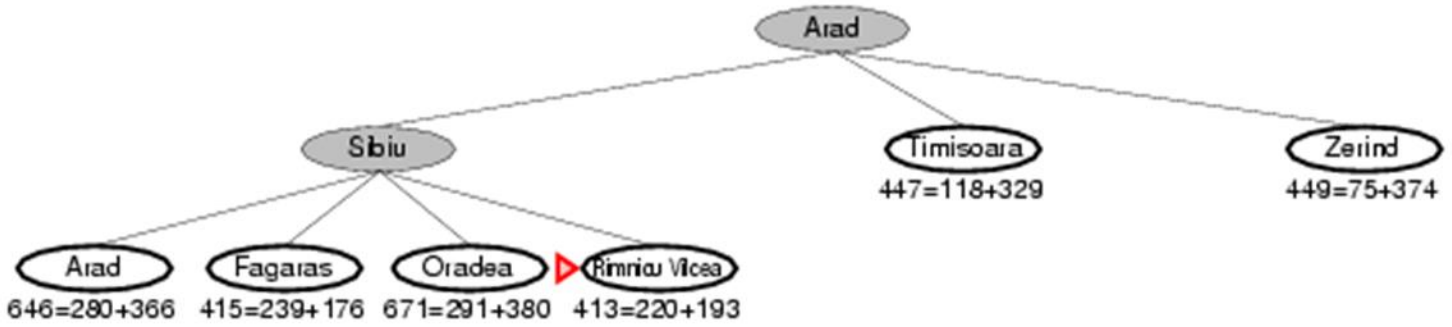
Algoritmo A*



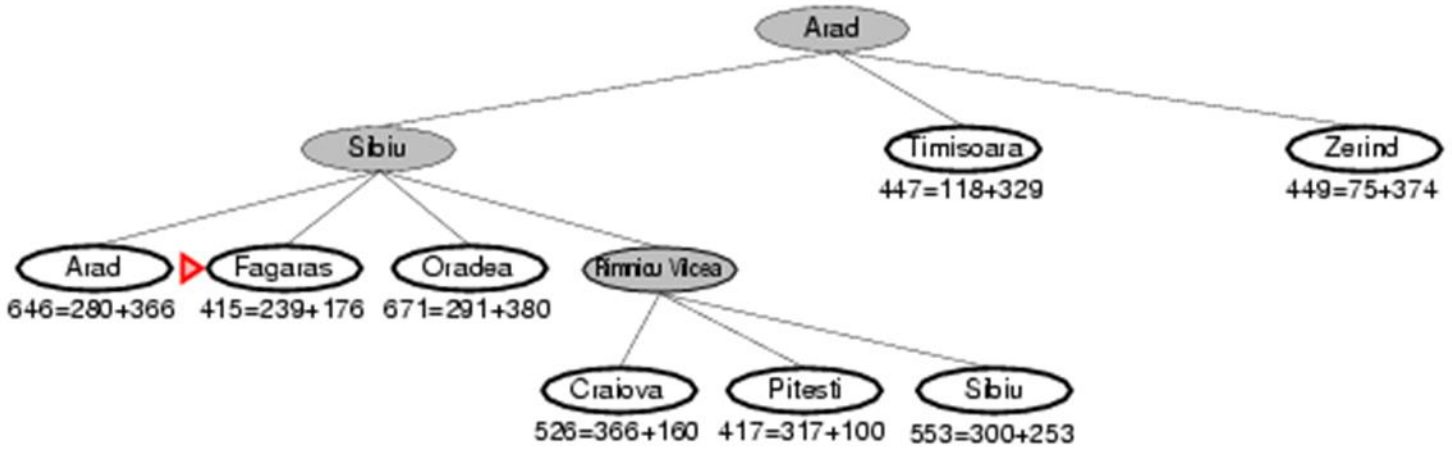
Algoritmo A*


Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374

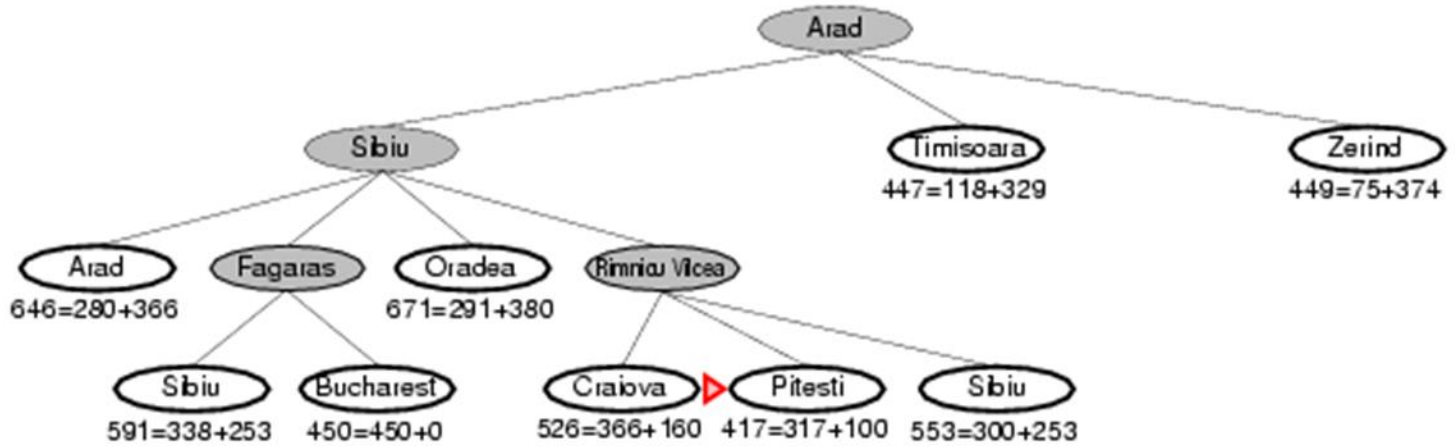
Algoritmo A*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
413=220+193

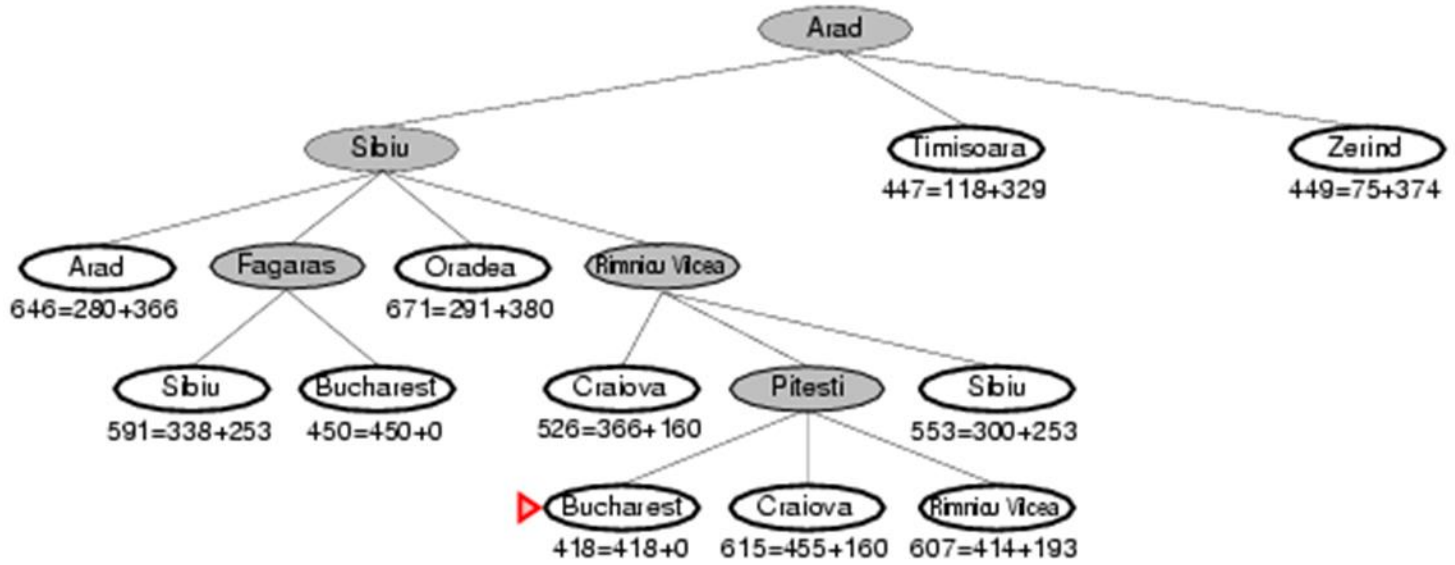
Algoritmo A*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

Algoritmo A*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

Algoritmo A*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193

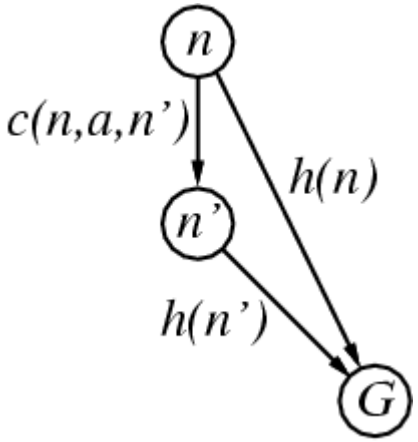
Algoritmo A* - heurística admissível

- **Uma heurística h(n) é admissível se para cada nó n, $h(n) \leq h^*(n)$, onde $h^*(n)$ é o custo verdadeiro de alcançar o estado objetivo a partir de n.**
- **Uma heurística admissível nunca superestima o custo de alcançar o objetivo, isto é, ela é otimista.**
- **Exemplo: hDLR(n) (distância em linha reta nunca é maior que distância pela estrada).**

Algoritmo A* - heurística consistente

- **Uma heurística é consistente (ou monotônica) se para cada nó n, cada sucessor n' de n gerado por qualquer ação a, apresenta:**
  - **$h(n) \leq c(n,a,n') + h(n')$**
- **Se h é consistente, temos**

$$f(n') = g(n') + h(n')$$
$$= g(n) + c(n,a,n') + h(n')$$
$$\geq g(n) + h(n)$$
$$f(n') \geq f(n)$$

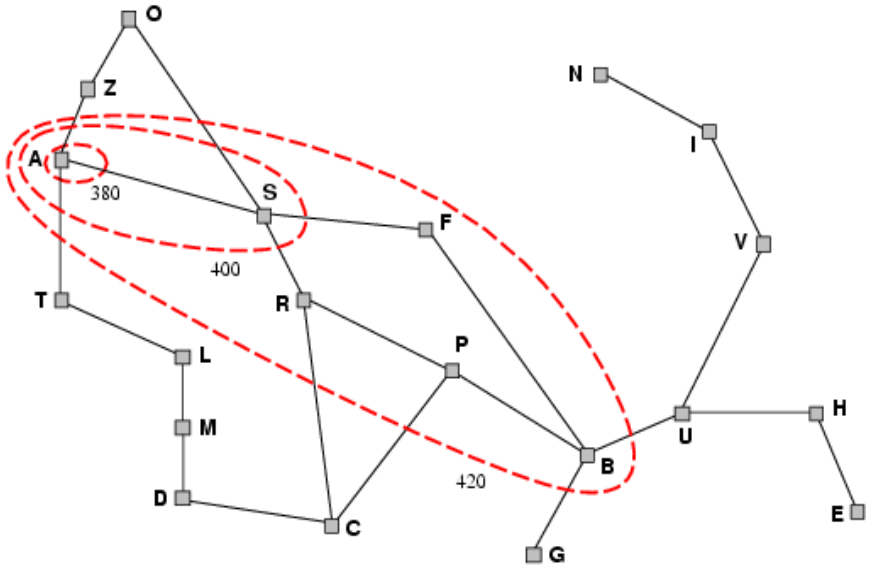


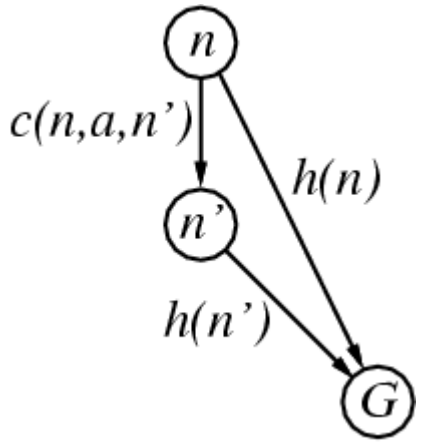
- **Isto é, f(n) é não-decrescente ao longo de qualquer caminho. O custo total aumenta.**
- **Teorema: Se h(n) é consistente, A* usando BUSCA-EM-GRAFOS é ótima**

Algoritmo A* - otimicidade

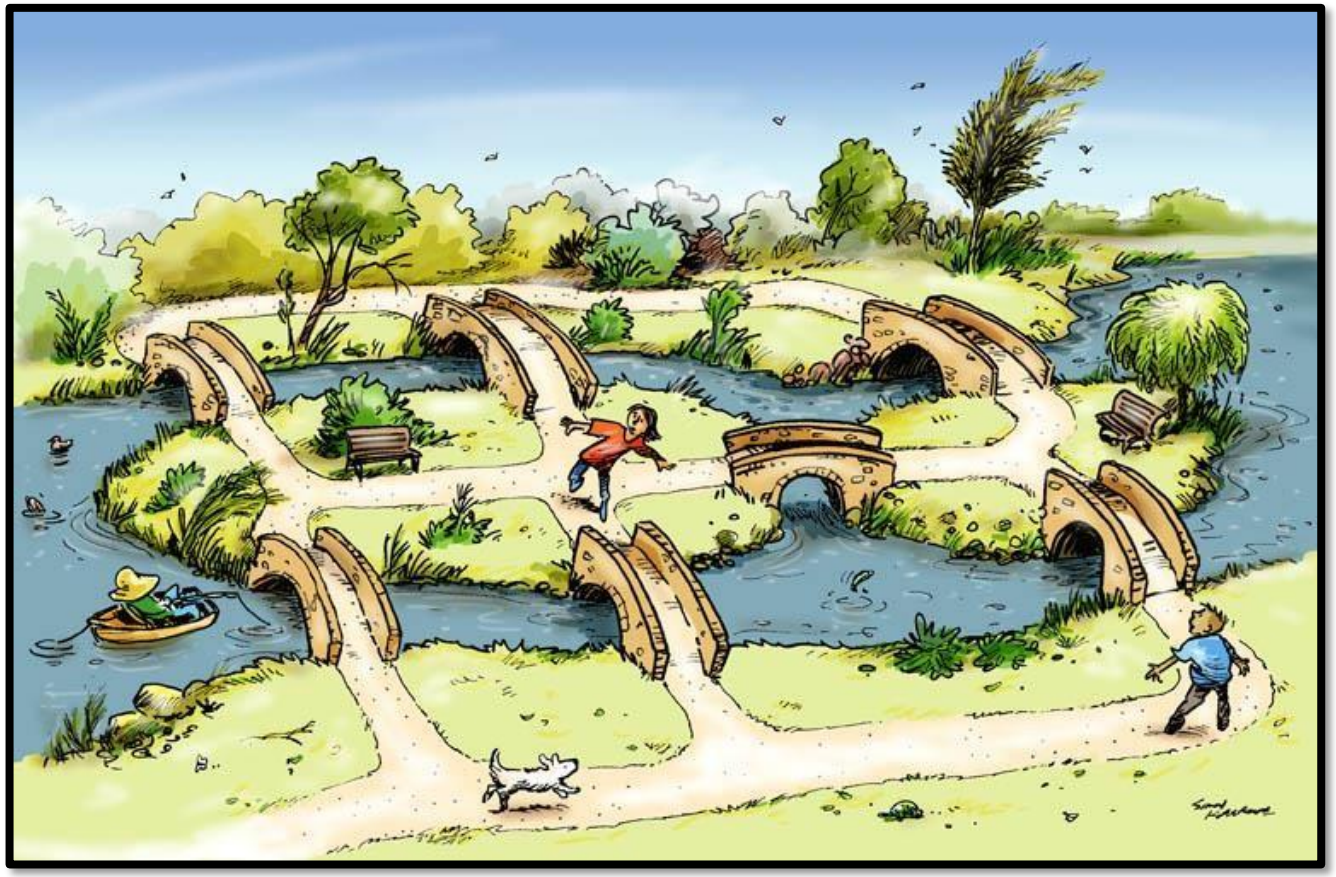
- **Nenhum outro algoritmo de busca ótimo tem garantia de expandir um número de nós menor que A*. Isso porque qualquer algoritmo que não expande todos os nós com f(n) < C* corre o risco de omitir uma solução ótima**

- Quanto melhor a heurística mais direcionados ao objetivo será o algoritmo

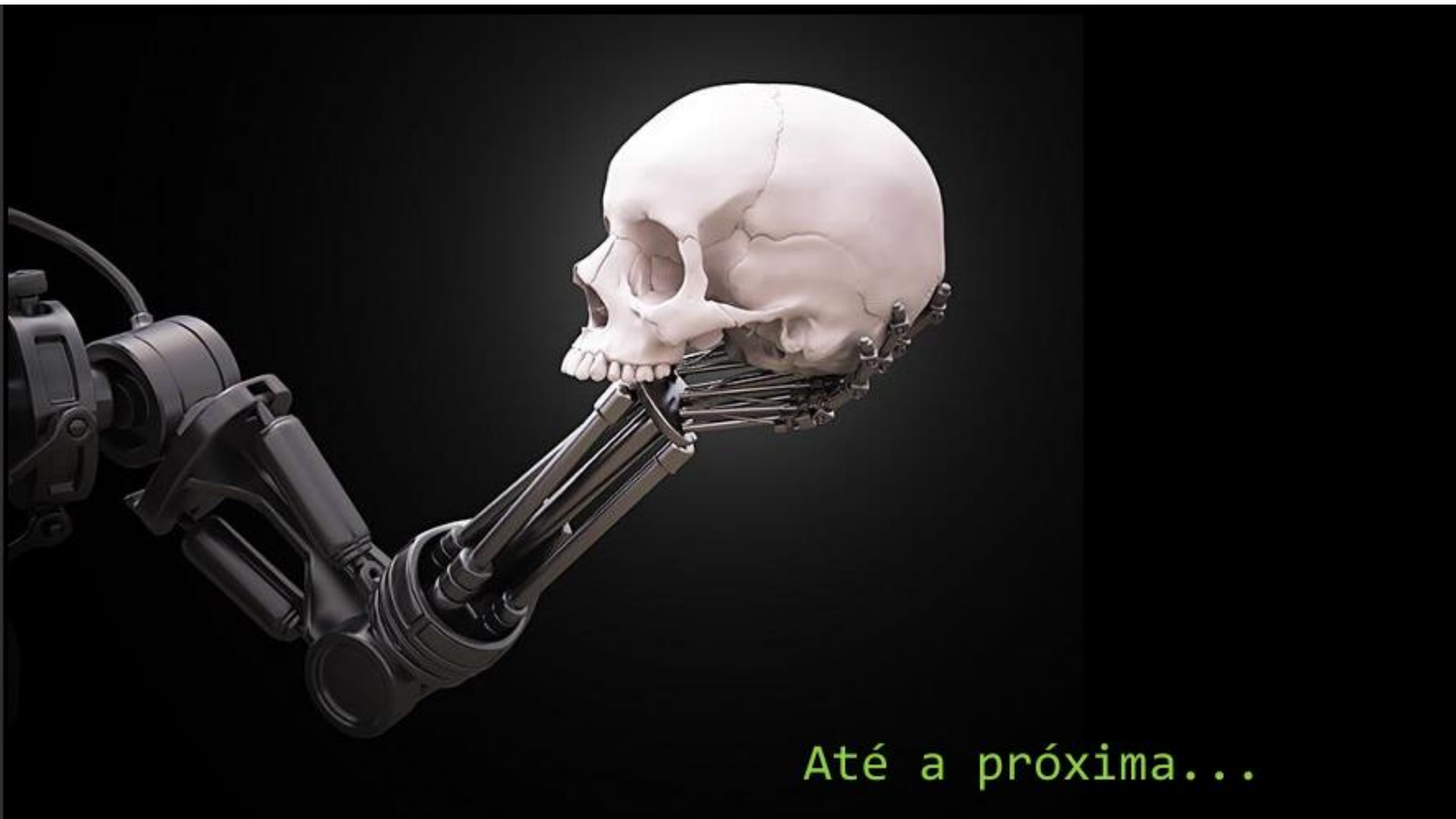




Obrigado!



Dúvidas?


Até a próxima...

Apresentador

Thales Levi Azevedo Valente

E-mail:
thales.l.a.valente@gmail.com

Referências

- Links referenciados nos respectivos slides.
- T.B. Borchartt . *Introdução à Inteligência Artificial*. 2024. 37 slides. Universidade Federal do Maranhão.
- A.O. B. Filho. *Inteligência Artificial - Introdução*. 2024. 31 slides. Universidade Federal do Maranhão.