- Sumário:
  - Busca informada
    - Busca gulosa pela melhor escolha
    - Busca A*

Objectivos

- Adquirir a noção de busca informada
- Descrever alguns algoritmos de busca informada: busca gulosa pela melhor escolha, busca A*

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
- são ineficientes na maioria dos casos:
  - No caso de um espaço de estados grande a busca pode ser impraticável por demorar muito para encontrar a solução
  - Limitações de memória necessária podem limitar a profundidade em que se realiza a busca

Busca informada (heurística)

- Utiliza conhecimento específico sobre o problema para guiar o processo de busca, para além da informação contida na definição do problema
- Permite:
  - Encontrar soluções mais rápido
  - Encontrar soluções mesmo na presença de limitações de tempo
  - Geralmente encontram melhores soluções

Busca pela melhor escolha (*best first search*) (I)

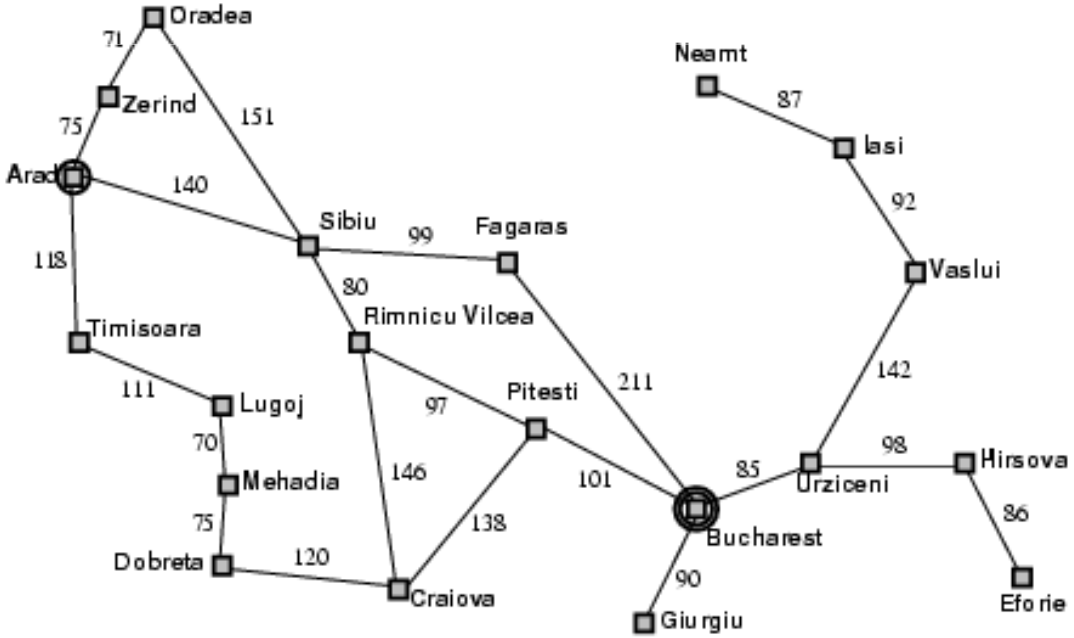
- Abordagem geral ao problema da busca informada
- Constitui uma variante dos algoritmos de busca em árvore ou em grafos
  - Os nós são seleccionados para expansão com base numa **função de avaliação**, ***f(n)***
  - Expande o nó correspondente ao menor valor da função de avaliação
  - Dependendo da função de avaliação, a estratégia de busca muda

Busca pela melhor escolha (*best first search*) (II)

- Elemento chave
  - Utilização de uma função heurística, ***h(n)***
    - ***h(n)*** → custo estimado do caminho mais barato desde um nó até ao objectivo
      - *h(n)* >= 0, se *n* não é um nó objectivo
      - *h(n)* = 0, se *n* é um nó objectivo
      - *h(n)* = ∞, se é impossível atingir o objectivo a partir de *n*
  - São específicas para cada problema

Busca pela melhor escolha (*best first search*) (III)

- Exemplo
  - No mapa da Roménia se pode estimar o custo do percurso mais barato através da distância em linha recta entre Arad e Bucareste (h_{DLR})



Busca pela melhor escolha (*best first search*) (IV)

- Implementação
  - Ordenar nós na fronteira em ordem decrescente de acordo com a função de avaliação
  - Se utiliza para a fronteira uma lista ordenada de acordo aos valores da função de avaliação

Busca pela melhor escolha (*best first search*) (V)

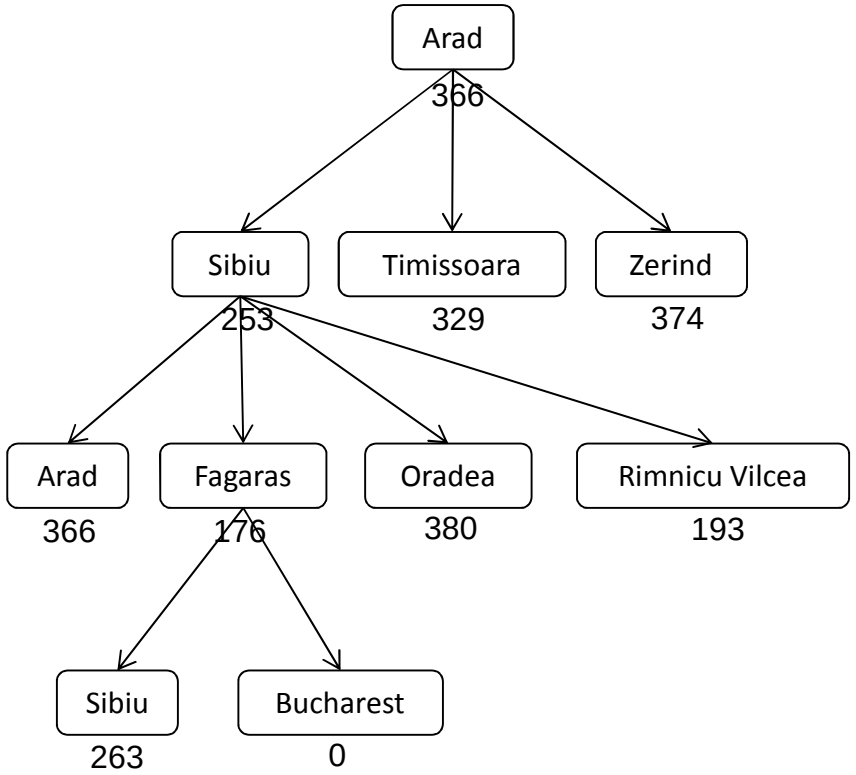
- Existem vários métodos incluídos nesta categoria:
  - Busca gulosa pela melhor escolha (*greedy best first search*)
  - Busca A*

Busca gulosa pela melhor escolha

- Expande o nó que ***parece*** mais próximo ao objetivo *de acordo com a função heurística*
- Ao avaliar os nós utiliza somente a função heurística
  - $f(n) = h(n)$
- Exemplo
  - Ir de Arad a Bucareste, usando como heurística a distância em linha recta
  - $h(n) = h_{DLR}$

Busca gulosa pela melhor escolha: exemplo

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374
Hirsova	151	Urziceni	80



Busca gulosa pela melhor escolha: análise (I)

- Completa?
  - Não
    - Pode tentar desenvolver um caminho infinito
    - Pode ficar presa em ciclos se não forem detectadas expansões de estados repetidos, ex., Iasi → Neamt → Iasi → Neamt
- Óptima?
  - Não
    - Segue o melhor passo ***considerando somente o estado actual***
    - Pode haver um caminho melhor seguindo algumas opções piores em alguns pontos da árvore de busca

Busca gulosa pela melhor escolha: análise (II)

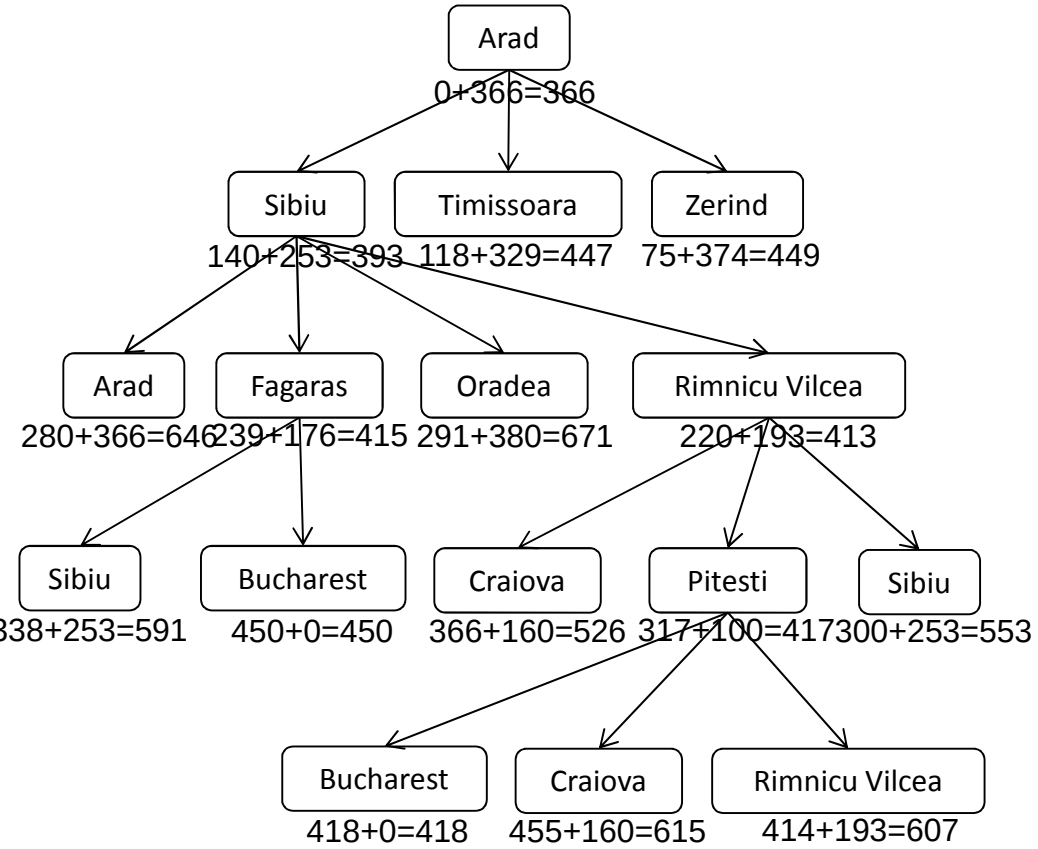
- Complexidade temporal
  - Exponencial, $O(b^m)$, no pior caso
    - m -> profundidade máxima do espaço de estados
- Complexidade espacial
  - Exponencial, $O(b^m)$
  - Mantém todos os nós na memória
- Uma boa função heurística pode levar a uma redução substancial

Busca A* (I)

- Tenta minimizar o custo total da solução combinando:
  - Busca Gulosa: económica, porém não é completa nem óptima
  - Busca de Custo Uniforme: ineficiente, porém completa e óptima
- Função de avaliação: $f(n) = g(n) + h(n)$
  - $g(n)$: custo do caminho desde o nó inicial até ao n-ésimo nó
  - $h(n)$: custo estimado desde o nó n até ao objectivo
  - $f(n)$: custo estimado da solução mais barata através do nó n

Busca A* (II)

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374
Hirsova	151	Urziceni	80



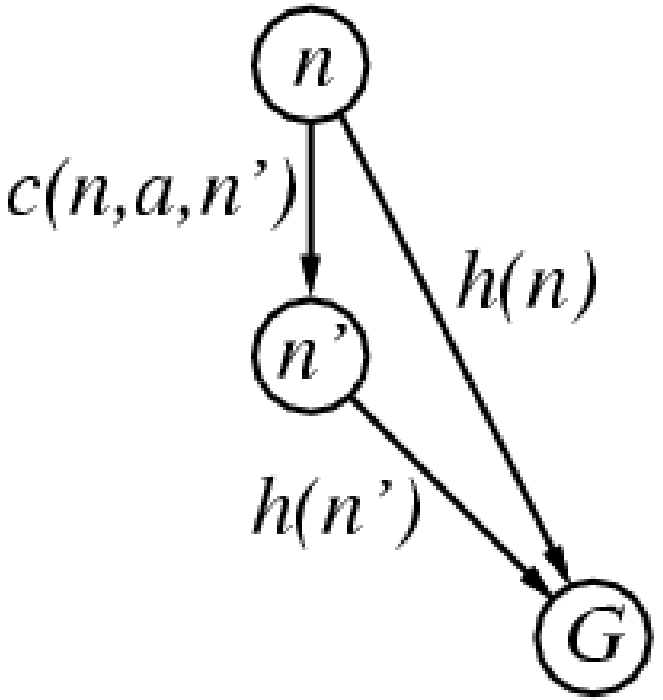
Heurística admissível

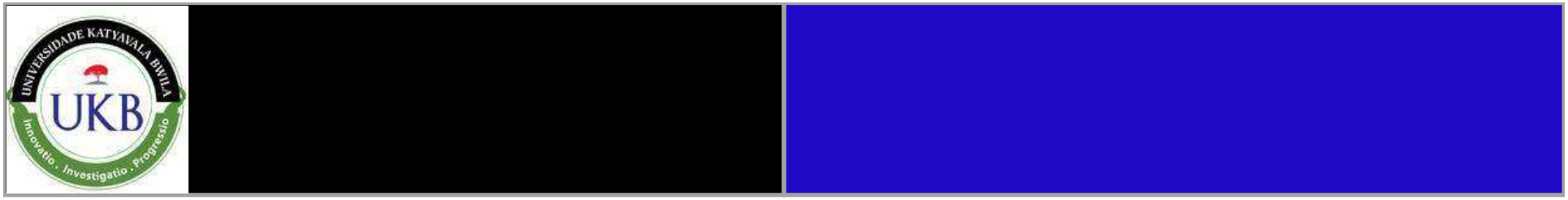
- Uma heurística *h(n)* é admissível se para qualquer nó *n*, *h(n)* ≤ *h*(n)*, onde *h*(n)* é o custo verdadeiro de alcançar o estado objectivo a partir de *n*
  - Nunca sobrestima o custo de alcançar o objectivo (é optimista)
  - Exemplo: $h_{DLR}(n)$ (distância em linha recta nunca é maior que distância pela estrada)
- Se ***h(n)* é admissível**, **A*** usando o algoritmo `BuscaEmArvore` **é óptimo**

Heurística consistente (I)

- Uma heurística é ***consistente*** (ou ***monotónica***) se para cada nó *n*, cada sucessor *n'* de *n* gerado por qualquer acção *a*, o custo estimado de alcançar o objectivo a partir de *n* não é maior que a soma do custo do passo de alcançar *n'* a partir de *n* mais o custo estimado de alcançar o objectivo a partir de *n'*

  $h(n) \leq c(n,a,n') + h(n')$





Heurística consistente (II)

- Se ***h(n)*** **é consistente**, **os valores de** ***f(n)*** ao longo de qualquer trajectória são **não decrescentes**
- Se ***h(n)*** **é consistente**, **A*** usando o algoritmo `BuscaEmGrafo` **é óptimo**

Busca A*: análise (I)

- Completa?
  - Sim
- Óptima?
  - Sim
- Óptimamente eficiente
  - Nenhum outro algoritmo de busca óptimo garante expandir menos nós que A*

Busca A*: análise (II)

- Complexidade temporal
  - Exponencial, $O(b^m)$, no pior caso
    - m -> profundidade máxima do espaço de estados
- Complexidade espacial
  - Exponencial, $O(b^m)$
  - Mantém todos os nós na memória
- Uma boa função heurística pode levar a uma redução substancial

Busca A*: variantes

- Principal problema -> memória
- Algumas variantes recentes permitem ultrapassar o problema do espaço de memória, sem sacrificar o carácter óptimo nem a completitude, a custo de um ligeiro incremento no tempo de execução
  - A* com aprofundamento progressivo (IDA*)
  - Busca pela melhor escolha recursiva (RBFS)
  - A* com memória limitada (MA*)
  - A* com memória limitada simplificado (SMA*)

Bibliografia

- Russell & Norvig, pg. 94 – 105
- Costa & Simões, pg. 97 - 108
- Palma Méndez & Marín Morales, pg. 339 – 353, 362 – 366