CSCM603130: Sistem Cerdas
Informed Search

Fariz Darari, Aruni Yasmin Azizah

Fakultas Ilmu Komputer
Universitas Indonesia

2019/2020 • Semester Ganjil

Outline

Outline

Best-first search

Prinsip best-first search

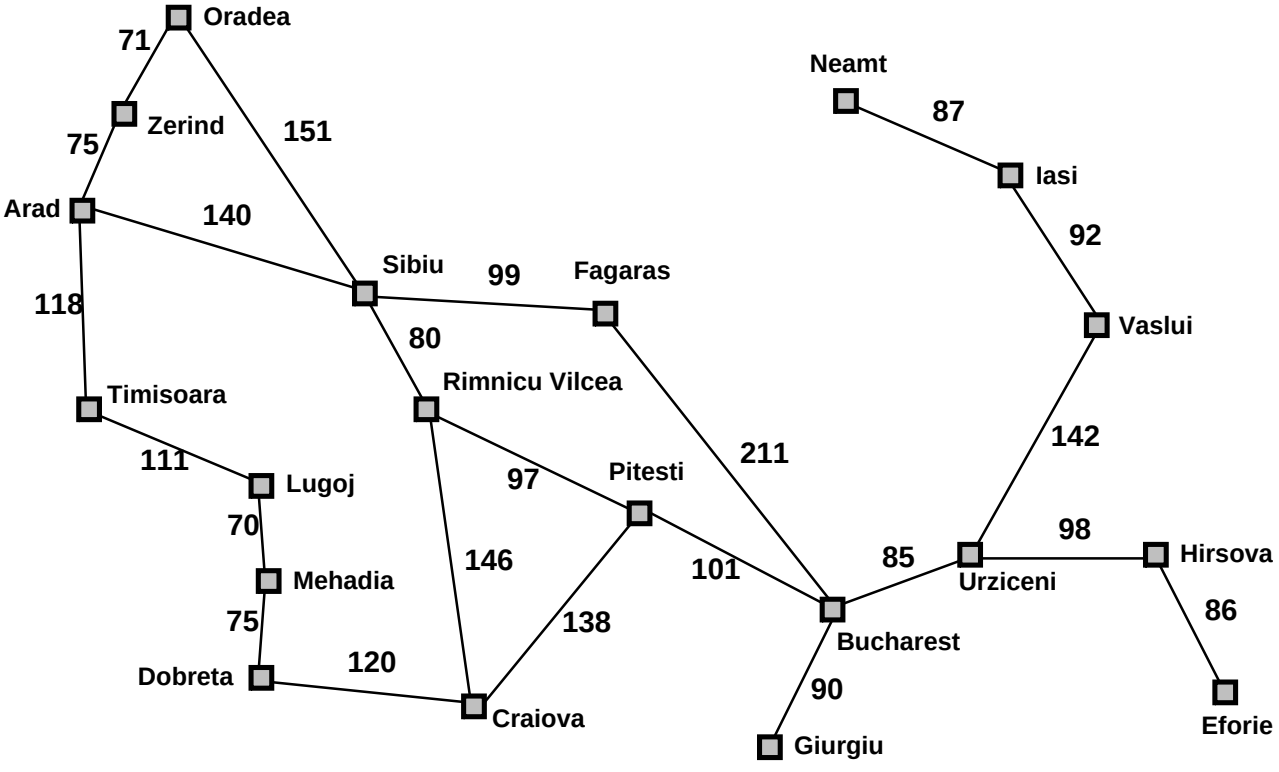
Lakukan node expansion terhadap node di *frontier* yang nilai $f(n)$-nya paling kecil.

- Ide dasar: $f(n)$ adalah sebuah *evaluation function* $\rightarrow$ fungsi yang menyatakan perkiraan seberapa "bagus" sebuah node n.
- Implementasi: *frontier* adalah sebuah *priority queue* di mana node disortir berdasarkan $f(n)$.
- Contoh:
  - Greedy best-first search
  - A^* search

Heuristic function

- Kunci keberhasilan *best-first search* terletak di *heuristic function*.
- Heuristic adalah:
  - *rule of thumb*
  - *"kiat-kiat sukses", "tips-tips keberhasilan"*
  - *informasi tambahan bagi si agent (agar lebih sukses)*
    $\rightarrow$ *informed search*
- *Heuristic function* $h(n)$ adalah fungsi yang menyatakan estimasi cost dari node n ke *goal state*.
- Ada banyak kemungkinan *heuristic function* untuk sebuah masalah.

Contoh heuristic function



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Sebuah heuristic function untuk agent turis Rumania

$h_{SLD}(n) =$ *straight-line distance* dari kota n ke Bucharest.

Outline

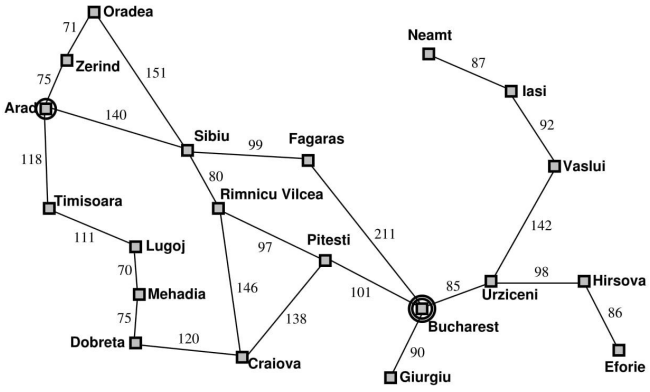
Greedy best-first search

Prinsip greedy best-first search

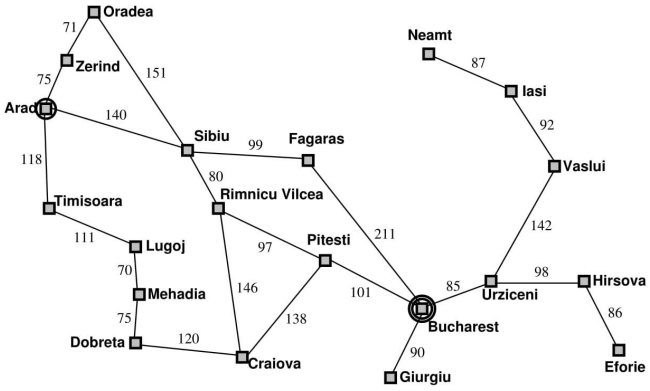
Lakukan node expansion terhadap node di *frontier* yang nilai $h(n)$-nya paling kecil.

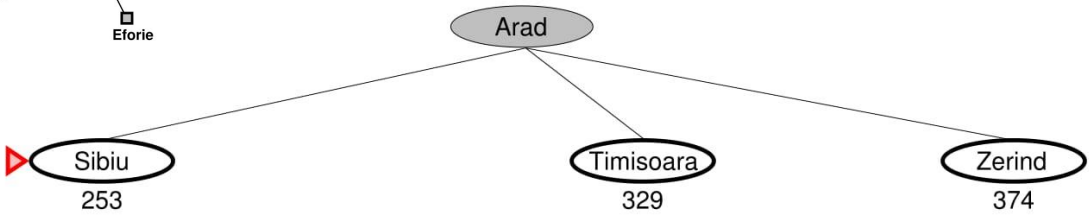
Greedy best-first search selalu memilih node yang kelihatannya paling dekat ke *goal*.

Greedy best-first search

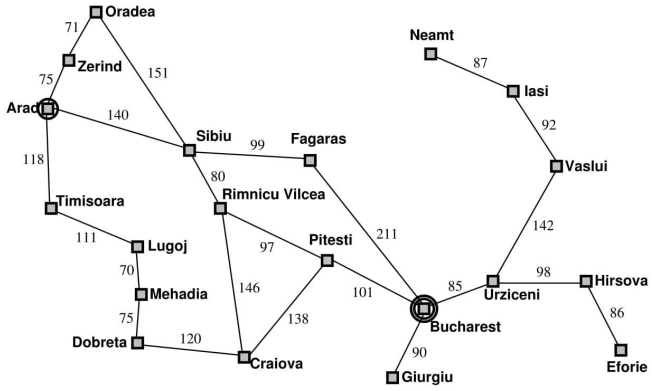


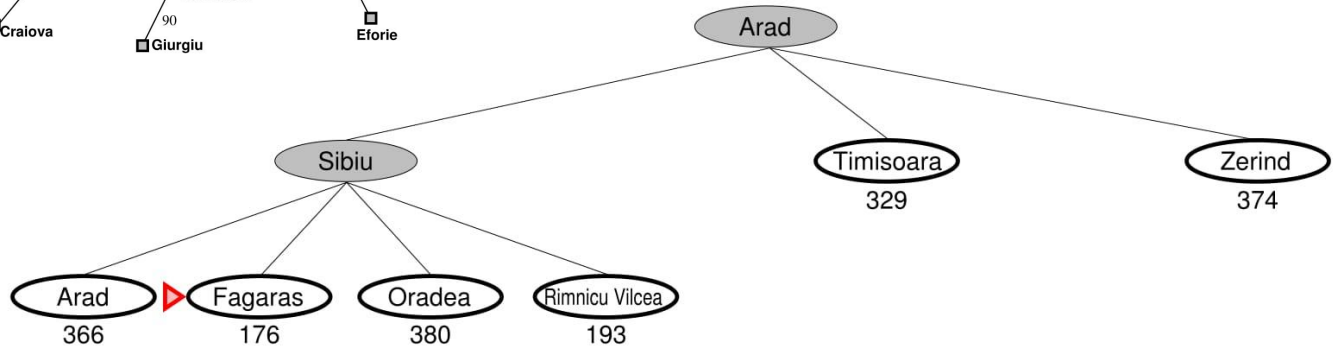
Greedy best-first search



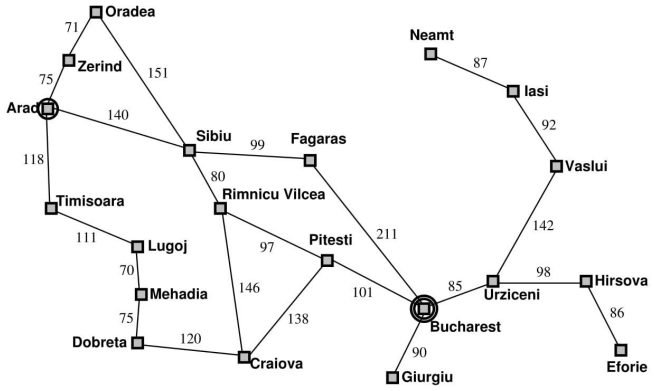


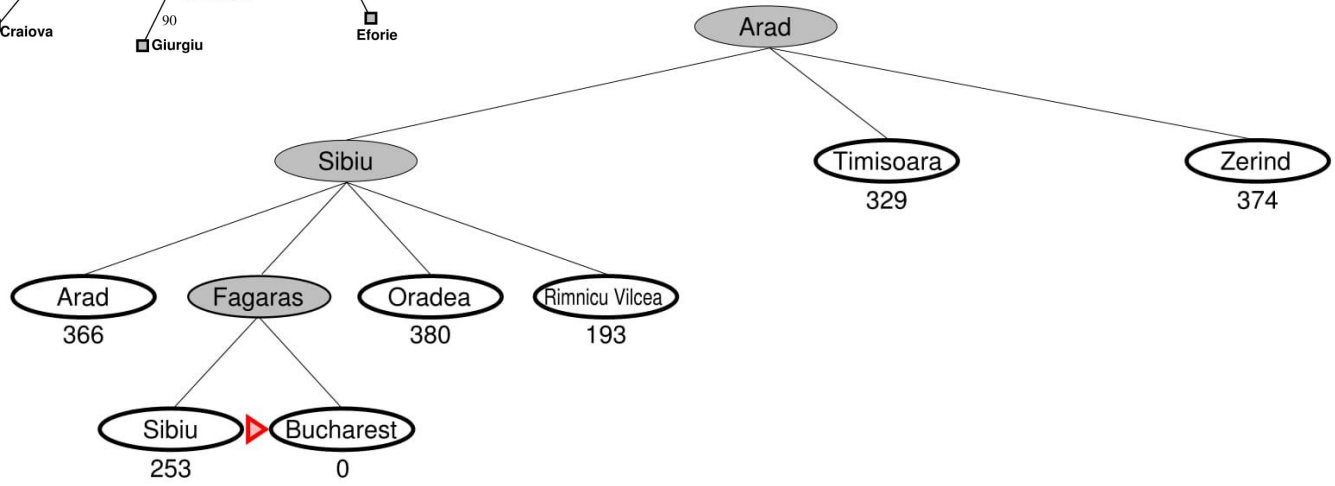
Greedy best-first search





Greedy best-first search





Sifat Greedy Best First Search

- Secara umum tidak complete, kecuali jika *state space* terbatas dan pengulangan state ditangani.
  - Mis: pencarian rute Iasi $\rightarrow$ Oradea.
- Optimal?

Sifat Greedy Best First Search

- Secara umum tidak complete, kecuali jika *state space* terbatas dan pengulangan state ditangani.
  - Mis: pencarian rute Iasi $\rightarrow$ Oradea.
- Optimal? Tidak.

Outline

A^* search

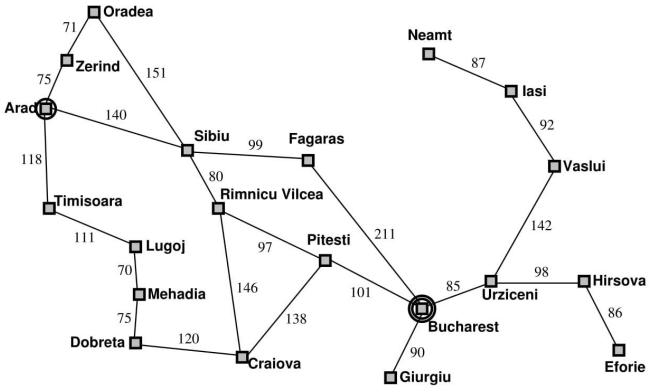
Prinsip A^* search

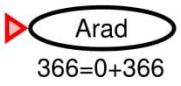
Hindari node yang berada di *path* yang "mahal"

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

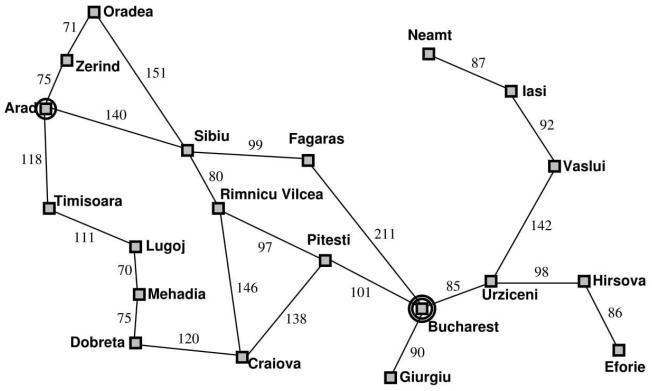
- $g(n)$ = Path cost ke n
- $h(n)$ = Estimasi path cost dari n ke *goal*
- $f(n)$ = Estimasi total cost melalui n

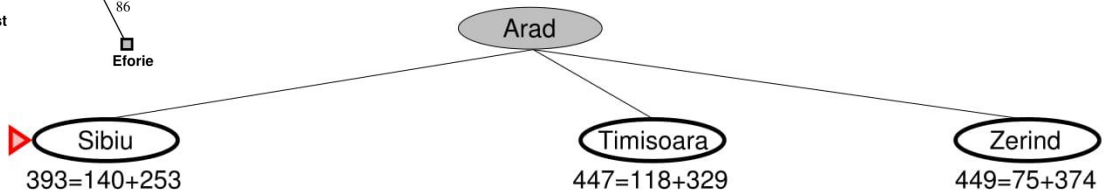
Contoh penelusuran A^* search



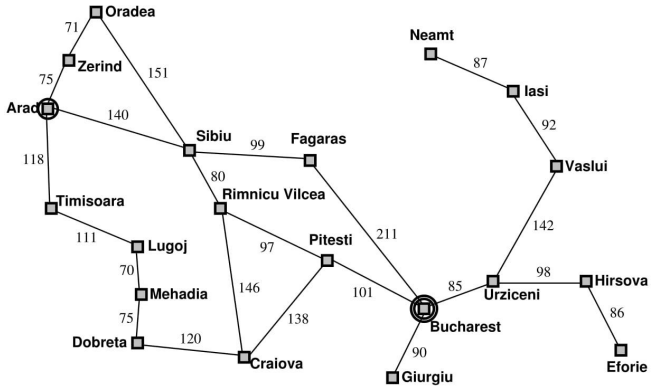


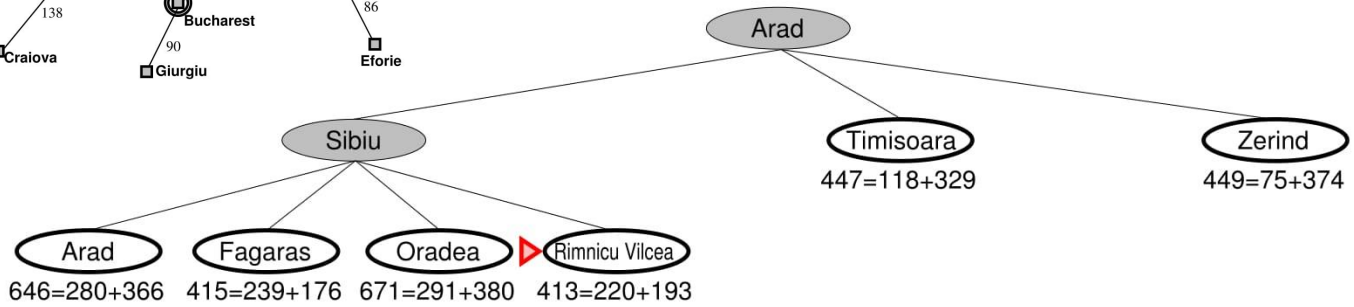
Contoh penelusuran A^* search



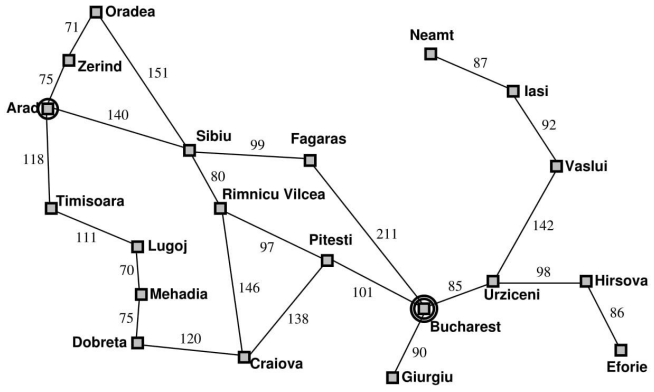


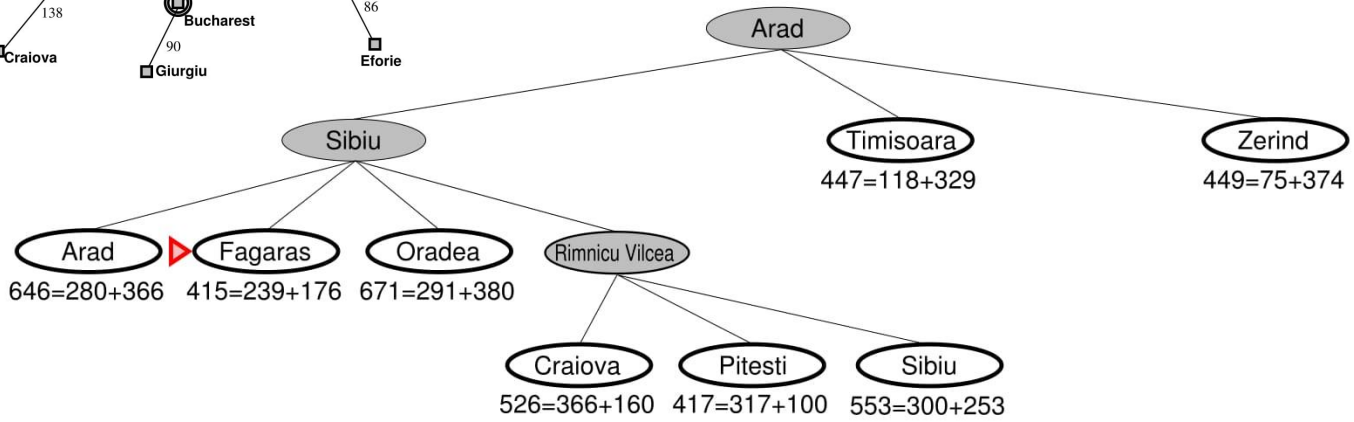
Contoh penelusuran A^* search



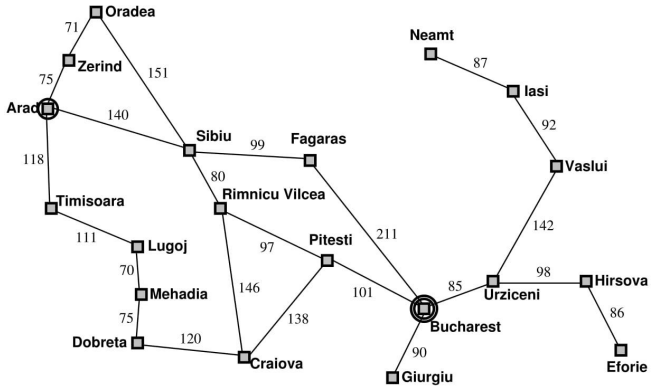


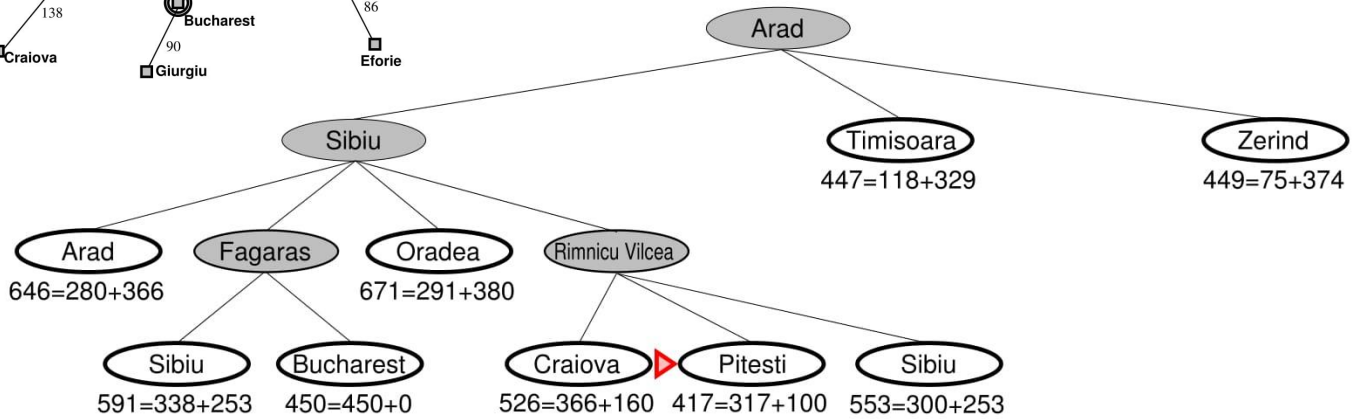
Contoh penelusuran A^* search



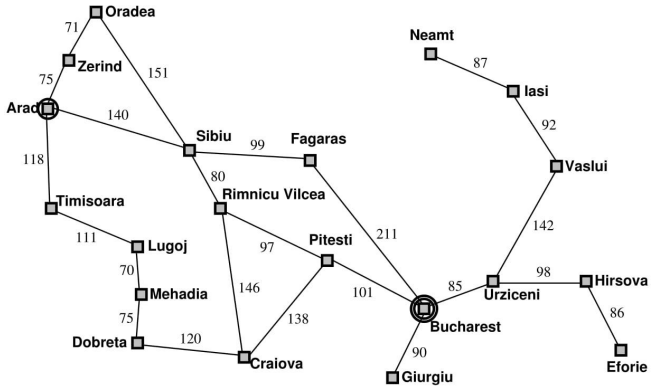


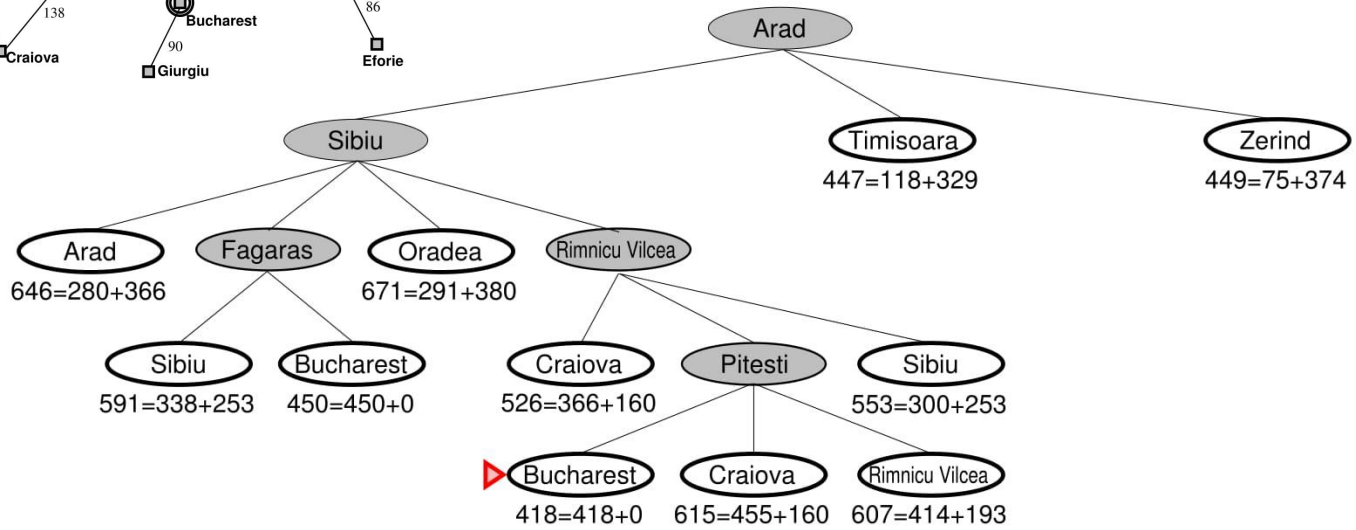
Contoh penelusuran A^* search





Contoh penelusuran A^* search





Admissibility sebuah heuristic – TreeSearch

Nilai sebuah heuristic function tidak pernah melebihi *cost* ke *goal* yang sebenarnya.

Heuristic yang *admissible*

$0 \leq h(n) \leq h^*(n)$, di mana $h^*(n)$ adalah *cost* dari n ke *goal* yang sebenarnya.

Contoh: $h_{SLD}(n)$

Theorem

A^* search (TreeSearch) adalah optimal, jika $h(n)$ admissible.

Consistency sebuah heuristic – GraphSearch

Heuristic yang *consistent*

$h(n) \leq c(n, a, n') + h(n')$, untuk setiap node n dan successor-nya (yaitu n') yang dicapai dengan action a.

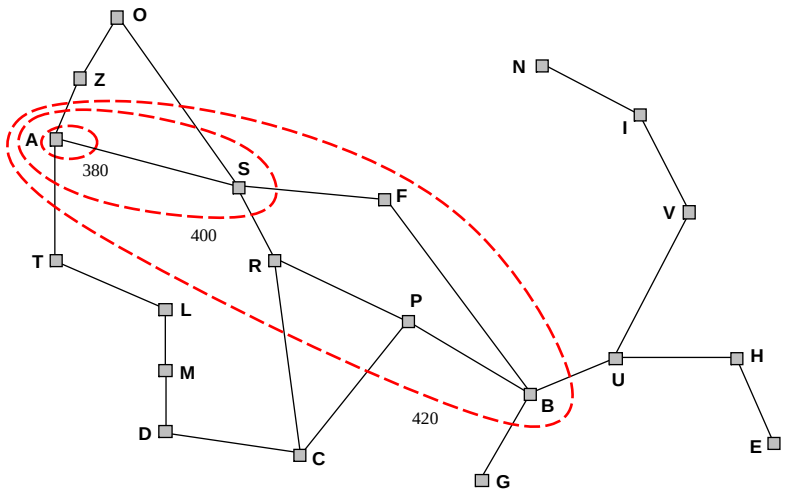
- Jika h konsisten, maka:

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

- Jika $h(n)$ consistent, maka nilai $f(n)$ tidak pernah turun (nondecreasing) pada sembarang path.

Optimalitas A^* oleh Consistent Heuristic

- *Node expansion* A^* berdasarkan urutan nilai f.
- Bayangkan penelusuran *state space* yang dilakukan A^* membentuk f-contour.



- Bandingkan dengan strategi *uniform-cost*.

Theorem

A^* search (GraphSearch) adalah optimal, jika $h(n)$ consistent.

Sifat A^* search

- Complete, jika banyaknya node n di mana $f(n) \leq C^*$ berhingga.

Sifat A^* search

- Complete, jika banyaknya node n di mana $f(n) \leq C^*$ berhingga.
- Optimal?

Sifat A^* search

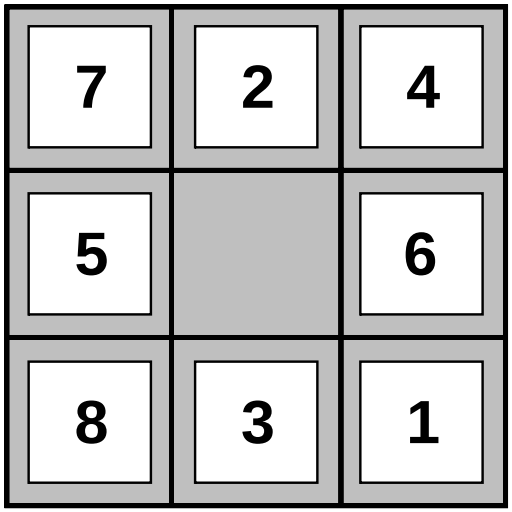
- Complete, jika banyaknya node n di mana $f(n) \leq C^*$ berhingga.
- Optimal? Ya.
  - A^* search (TreeSearch) adalah optimal, jika $h(n)$ admissible.
  - A^* search (GraphSearch) adalah optimal, jika $h(n)$ consistent.

Outline

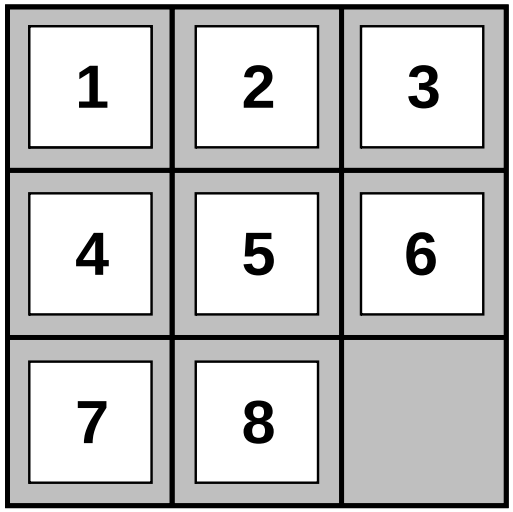
Contoh admissible heuristic

$h(n)$ untuk 8-puzzle

$h_1(n)$: jumlah angka yang salah posisi
$h_2(n)$: jumlah jarak semua angka dari posisi yang benar



Start State



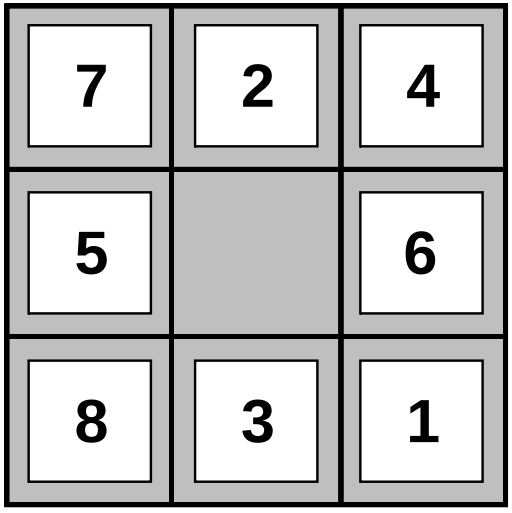
Goal State

$h_1(s) =$
$h_2(s) =$

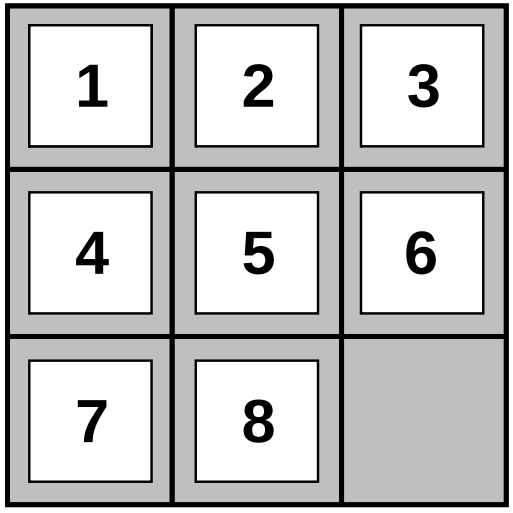
Contoh admissible heuristic

$h(n)$ untuk 8-puzzle

$h_1(n)$: jumlah angka yang salah posisi
$h_2(n)$: jumlah jarak semua angka dari posisi yang benar



Start State



Goal State

$h_1(s) = 6$
$h_2(s) = 4{+}0{+}3{+}3{+}1{+}0{+}2{+}1{=}14$

Membandingkan dua heuristic

- h_1 dan h_2 sama-sama *admissible*. Mana yang lebih baik? Bandingkan jumlah node yang di-generate:

d	IDS	$A^*(h_1)$	$A^*(h_2)$
12	3,644,035	227	73
24	-	39,135	1,641

- Jika $h_2(n) \geq h_1(n)$ untuk semua n (dan keduanya *admissible*), dikatakan bahwa h_2 mendominasi h_1 dan lebih baik untuk *search*.
  - Semakin besar nilai $h(n)$,
    semakin dekat ke $h^*(n)$,
    semakin banyak node yang di-*prune* (= dipangkas),
    semakin efisien *search*-nya!
- Catatan: Heuristic yang dominan tersebut consistent dan tidak membutuhkan waktu komputasi yang lama.

Merancang admissible heuristic

- *Admissible heuristic* dapat diperoleh dari *solution cost* yang sebenarnya dari variasi masalah yang dipermudah (*relaxed*).
- Contoh:
  - Andaikan masalah 8-puzzle dipermudah sehingga sebuah angka bisa dipindahkan ke mana saja. *Cost* dari solusinya = h_1.
  - Andaikan masalah 8-puzzle dipermudah sehingga sebuah angka bisa dipindahkan ke tetangga mana saja (kosong atau tidak). *Cost* dari solusinya = h_2.
- *Optimal solution cost* dari masalah yang dipermudah memberikan heuristic yang admissible untuk masalah yang sebenarnya.
- *Admissible heuristic* bisa juga diperoleh dari sub-masalah.