Route/Path Planning

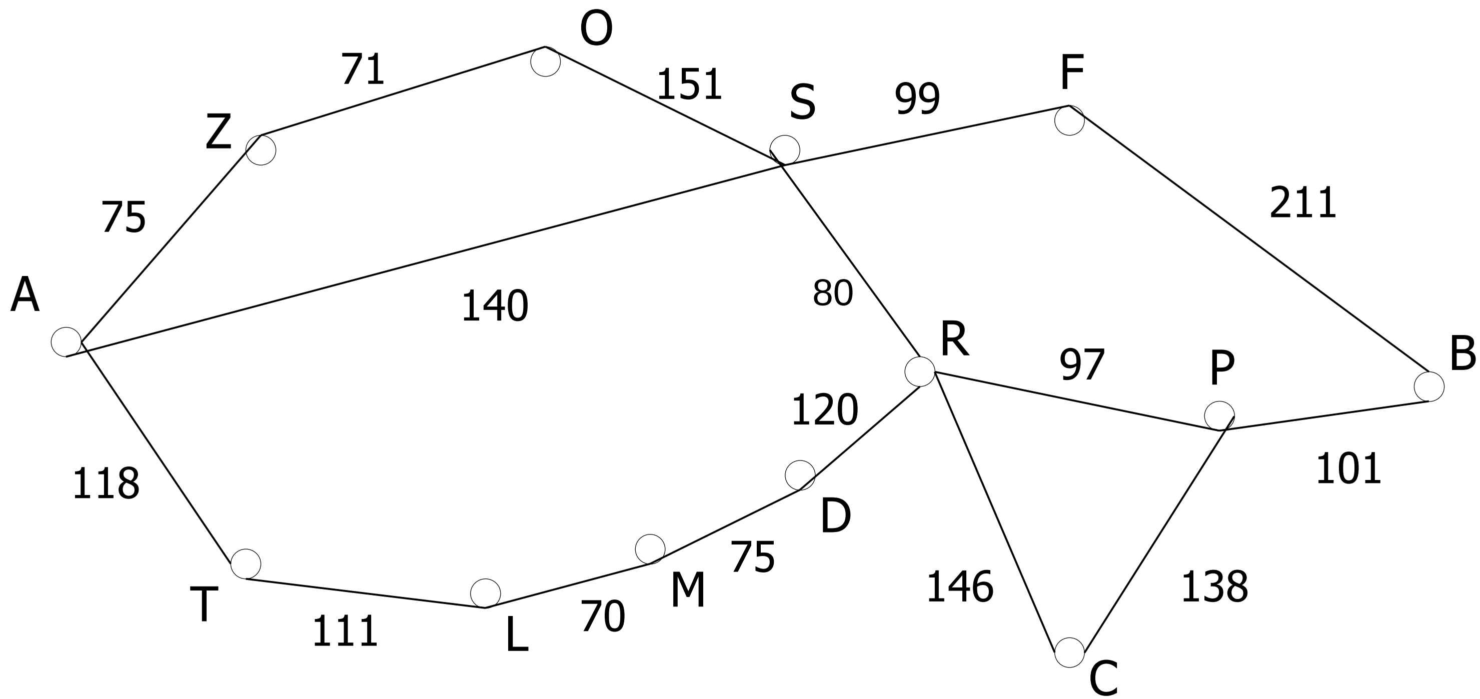
Referensi

- Materi kuliah IF3170 Inteligensi Buatan Teknik Informatika ITB, Course Website:
  http://kuliah.itb.ac.id → STEI → Teknik Informatika → IF3170
- Stuart J Russell & Peter Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice-Hall International, Inc, 2010, Textbook
  Site: http://aima.cs.berkeley.edu/ (2nd edition)
- Free online course materials | MIT OpenCourseWare Website:
  Site: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/
- Lecture Notes in Informed Heuristic Search, ICS 271 Fall 2008,
  http://www.ics.uci.edu/~dechter/courses/ics-271/fall-08/lecture-notes/4.InformedHeuristicSearch.ppt

Route Planning

Source: Russell's book

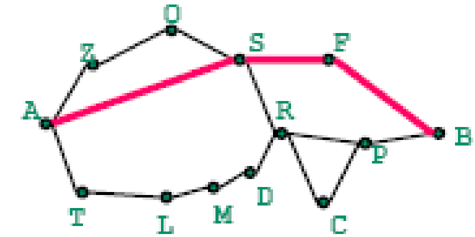
Search



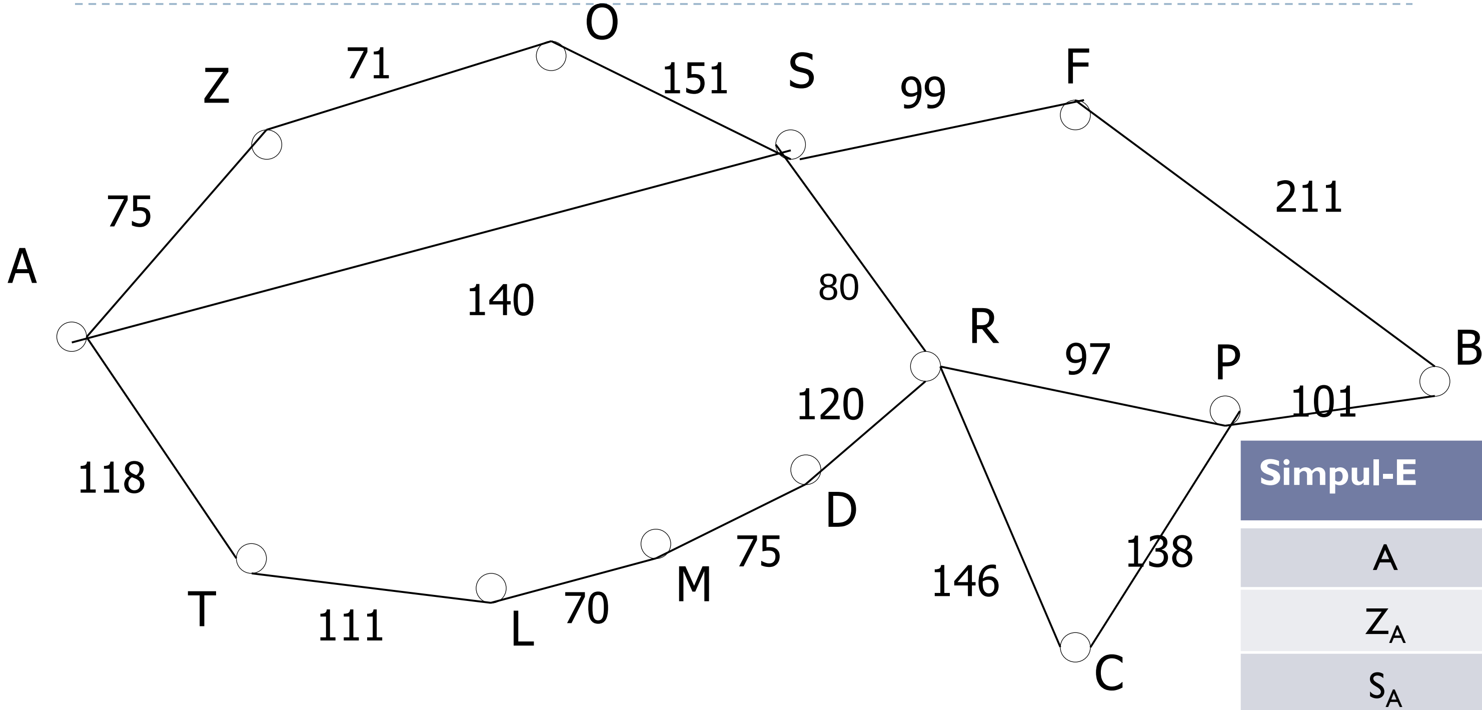
S: set of cities
i.s: A (Arad)
g.s: B (Bucharest)
Goal test: s = B ?
Path cost: time ~ distance

Uninformed Search

Breadth-First Search (BFS)



Treat agenda as a queue (FIFO)

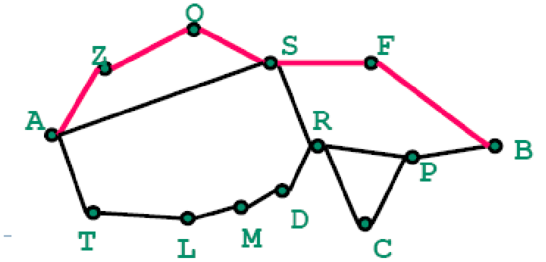


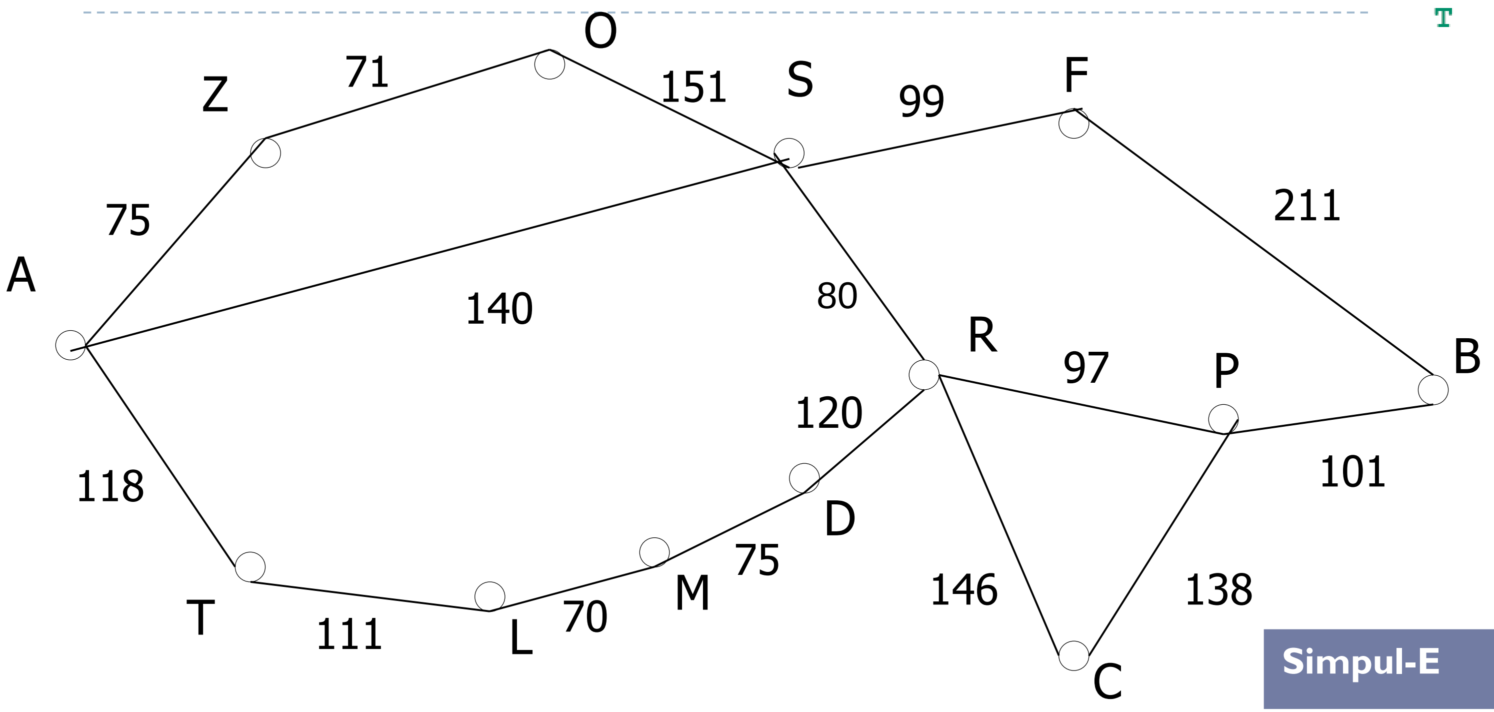
Path: A→ S → F → B,
Path-cost = 450

Simpul-E	Simpul Hidup
A	Z_A,S_A,T_A
Z_A	S_A,T_A,O_{AZ}
S_A	T_A,O_{AZ},O_{AS},F_{AS},R_{AS}
T_A	O_{AZ},O_{AS},F_{AS},R_{AS},L_{AT}
O_{AZ}	O_{AS},F_{AS},R_{AS},L_{AT}
O_{AS}	F_{AS},R_{AS},L_{AT}
F_{AS}	R_{AS},L_{AT},B_{ASF}
R_{AS}	L_{AT},B_{ASF},D_{ASR},C_{ASR},P_{ASR}
L_{AT}	B_{ASF},D_{ASR},C_{ASR},P_{ASR},M_{AT}
B_{ASF}	Solusi ketemu

Depth-First Search (DFS)

Treat agenda as a stack (LIFO)



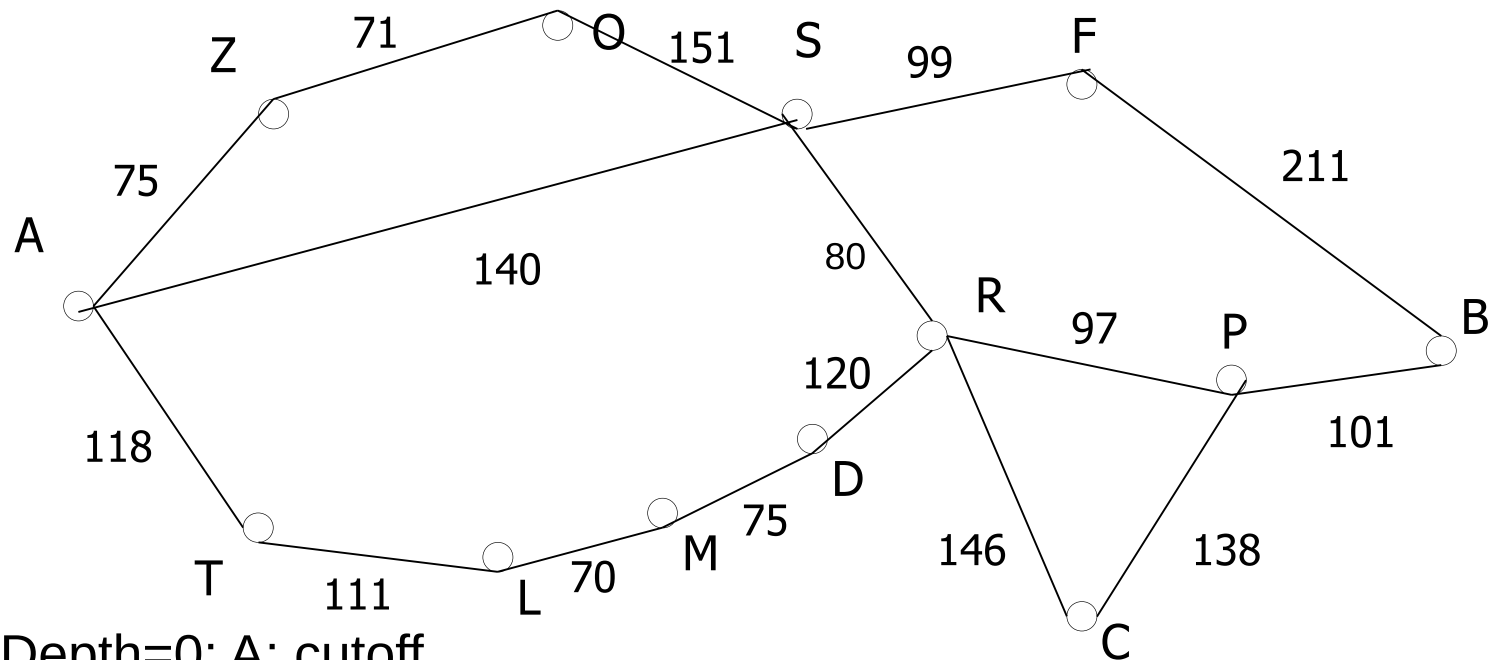


Simpul-E	Simpul Hidup
A	Z_A, S_A, T_A
Z_A	O_{AZ}, S_A, T_A
O_{AZ}	S_{AZO}, S_A, T_A
S_{AZO}	$F_{AZOS}, R_{AZOS}, S_A, T_A$
F_{AZOS}	$B_{AZOSF}, R_{AZOS}, S_A, T_A$
B_{AZOSF}	Solusi ketemu

Path: A→ Z → O → S → F → B

Path-cost = 607

IDS



Depth=0: A: cutoff

Depth=1: A → Z_A, S_A, T_A → Z_A: cutoff, S_A: cutoff, T_A: cutoff

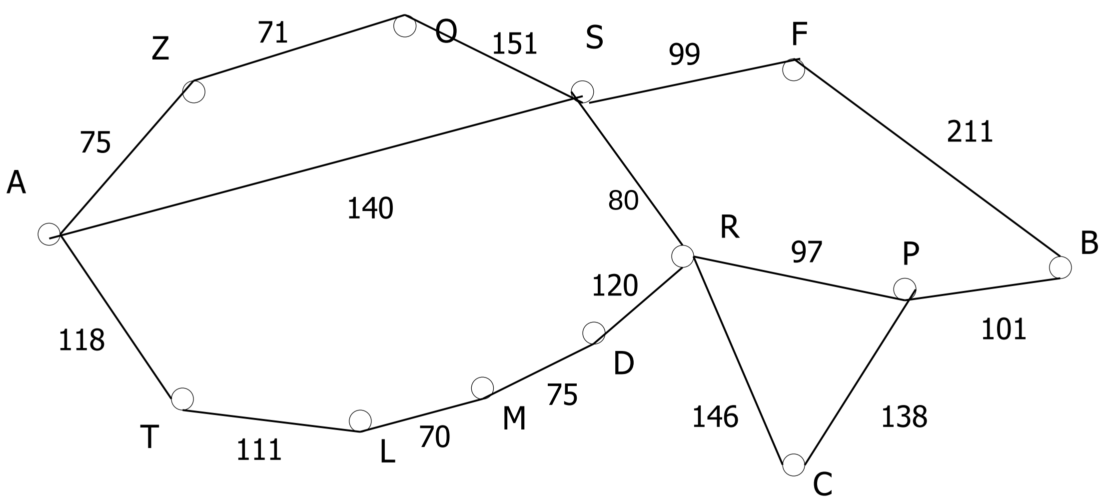
Depth=2: A → Z_A, S_A, T_A → O_{AZ}, S_A, T_A → O_{AZ} : cutoff → F_{AS}, R_{AS}, T_A → F_{AS} : cutoff → R_{AS} : cutoff → L_{AT} → L_{AT} : cutoff

Depth=3: A → Z_A, S_A, T_A → O_{AZ}, S_A, T_A → S_{AZO}, S_A, T_A → S_{AZO}: cutoff → F_{AS}, R_{AS}, T_A → B_{ASF}, R_{AS}, T_A → B_{ASF}

Stop: B=goal, path: A→ S → F → B, path-cost = 450

Uniform Cost Search (UCS)

- BFS & IDS find path with fewest steps
- If steps ≠ cost, this is not relevant (to optimal solution)
- How can we find the shortest path (measured by sum of distances along path)?



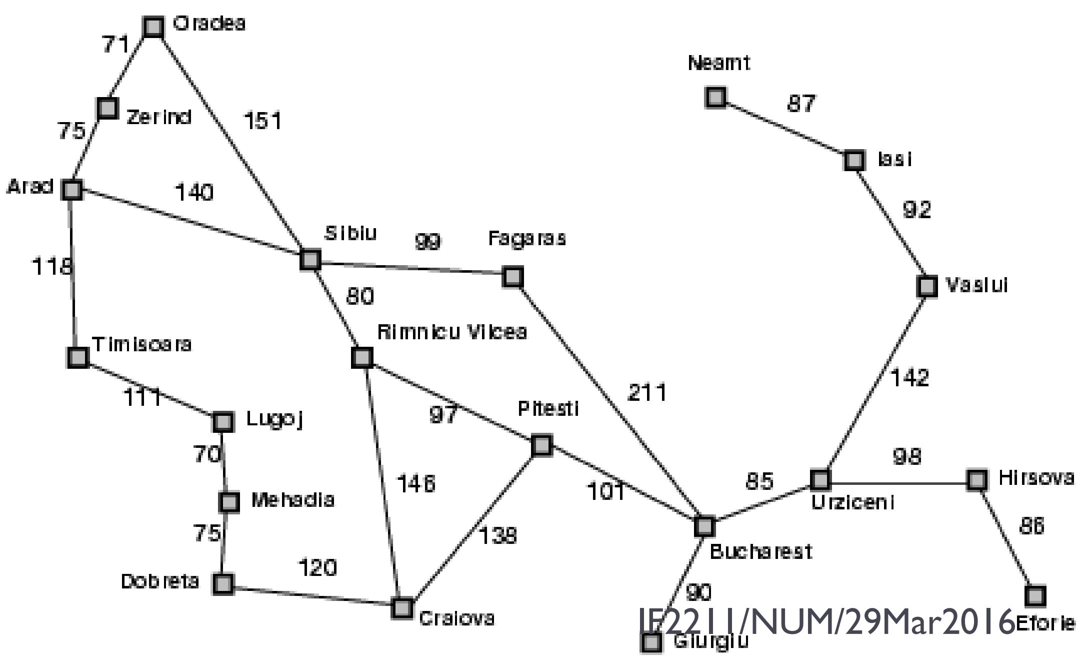
Path: A→ S → R → P → B
Path-cost = 418

Simpul-E	Simpul Hidup
A	Z_{A-75}, T_{A-118}, S_{A-140}
Z_{A-75}	T_{A-118}, S_{A-140}, O_{AZ-146}
T_{A-118}	S_{A-140}, O_{AZ-146}, L_{AT-229}
S_{A-140}	O_{AZ-146}, R_{AS-220}, L_{AT-229}, F_{AS-239}, O_{AS-291}
O_{AZ-146}	R_{AS-220}, L_{AT-229}, F_{AS-239}, O_{AS-291}
R_{AS-220}	L_{AT-229}, F_{AS-239}, O_{AS-291}, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$
L_{AT-229}	F_{AS-239}, O_{AS-291}, $M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$
F_{AS-239}	O_{AS-291}, $M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$, $B_{ASF-450}$
O_{AS-291}	$M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$, $B_{ASF-450}$
$M_{ATL-299}$	$P_{ASR-317}$, $D_{ASR-340}$, $D_{ATLM-364}$, $C_{ASR-366}$, $B_{ASF-450}$
$P_{ASR-317}$	$D_{ASR-340}$, $D_{ATLM-364}$, $C_{ASR-366}$, $B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$
$D_{ASR-340}$	$D_{ATLM-364}$, $C_{ASR-366}$, $B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$
$D_{ATLM-364}$	$C_{ASR-366}$, $B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$
$C_{ASR-366}$	$B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$

Informed Search

Greedy Best-First Search

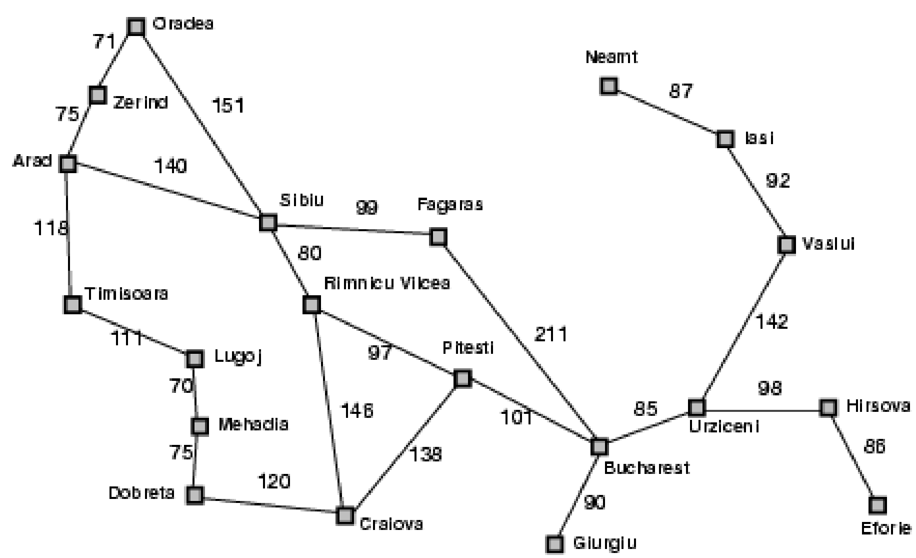
- Idea: use an evaluation function *f(n)* for each node
  - *f(n) = h(n) → estimates of cost from n to goal*
  - e.g., $h_{SLD}(n)$ = straight-line distance from *n* to Bucharest
- Greedy best-first search expands the node that appears to be closest to goal
- Romania with step costs in km



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

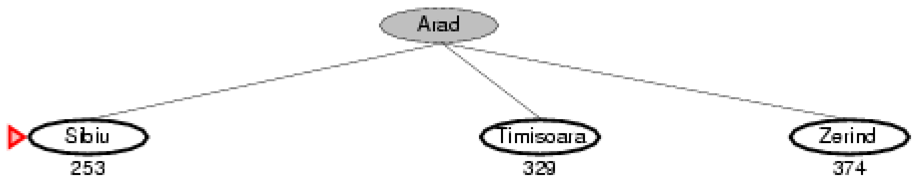
Greedy best-first search example

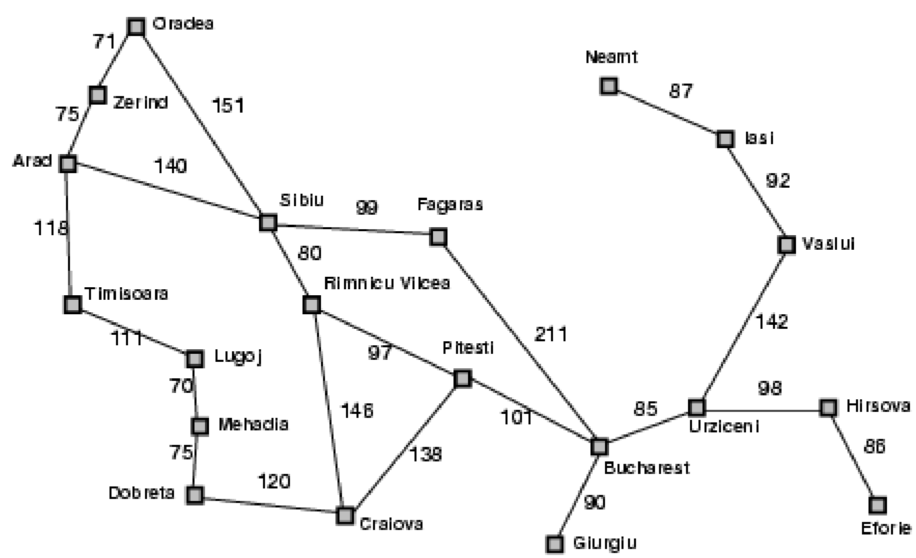


Straight–line distance to Bucharest

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

Greedy best-first search example

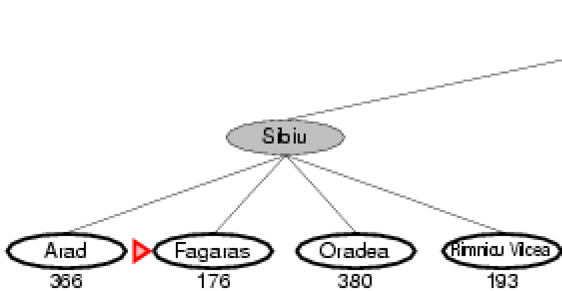


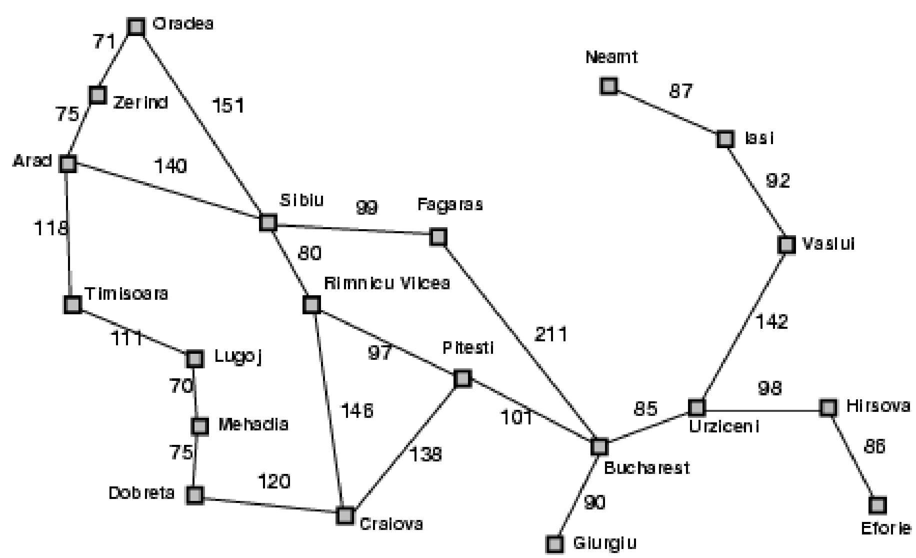


Straight–line distance
to Bucharest

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

Greedy best-first search example

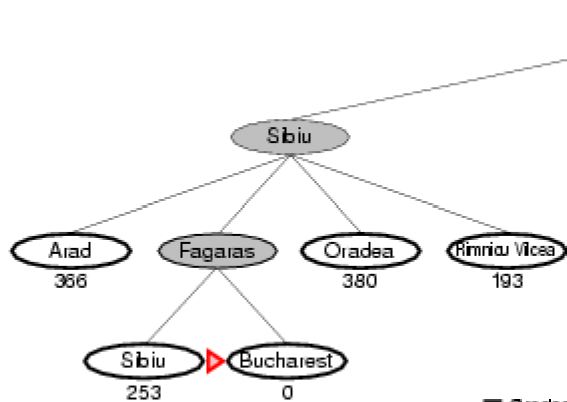


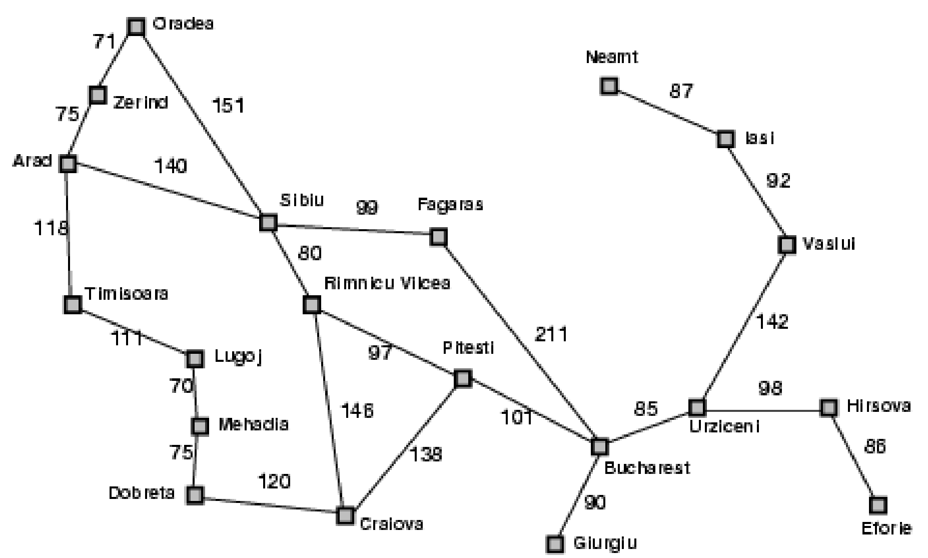


Straight–line distance to Bucharest

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

Greedy best-first search example





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

Problems with Greedy Best First Search

- Not complete
- Get Stuck with Local Minima/ Plateu
- Irrevocable (not able to be reversed/ changed)
- Can we incorporate heuristics in systematic search?

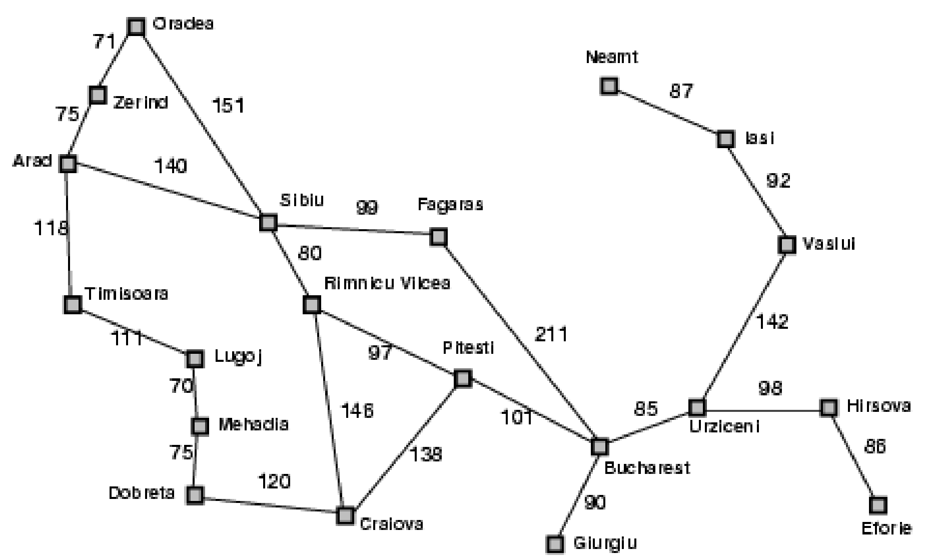
Heuristic Search

- Heuristic estimates value of a node
  - promise of a node
  - difficulty of solving the subproblem
  - quality of solution represented by node
  - the amount of information gained
- f(n)- heuristic evaluation function.
  - depends on n, goal, search so far, domain

A* Search

- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal

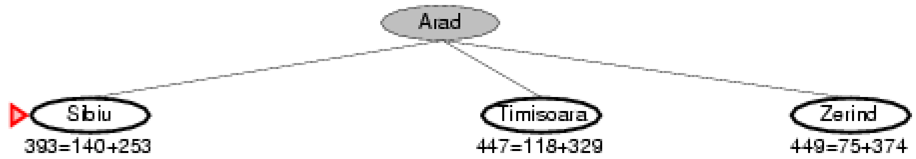
A* search example

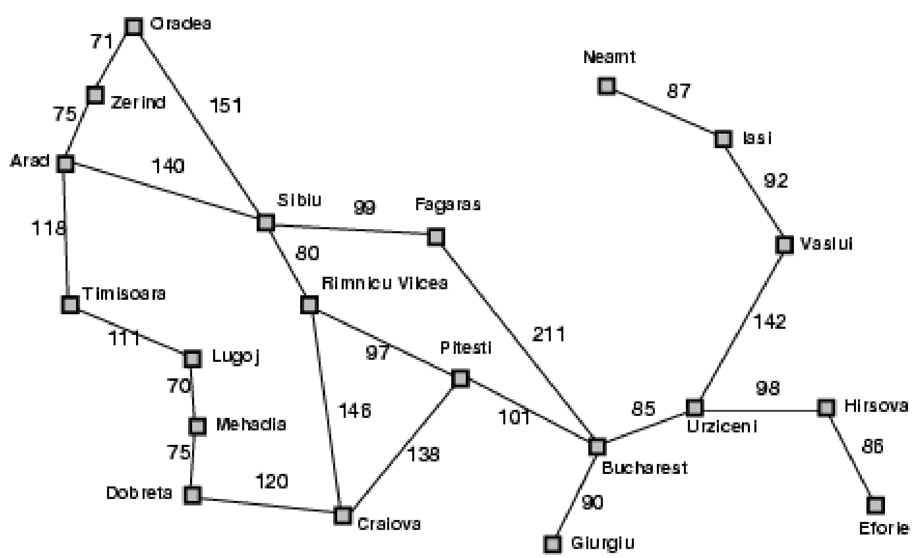


Straight–line distance to Bucharest

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

A* search example

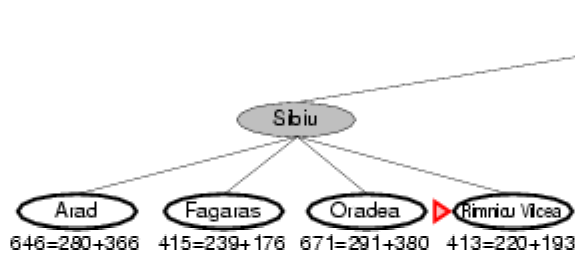


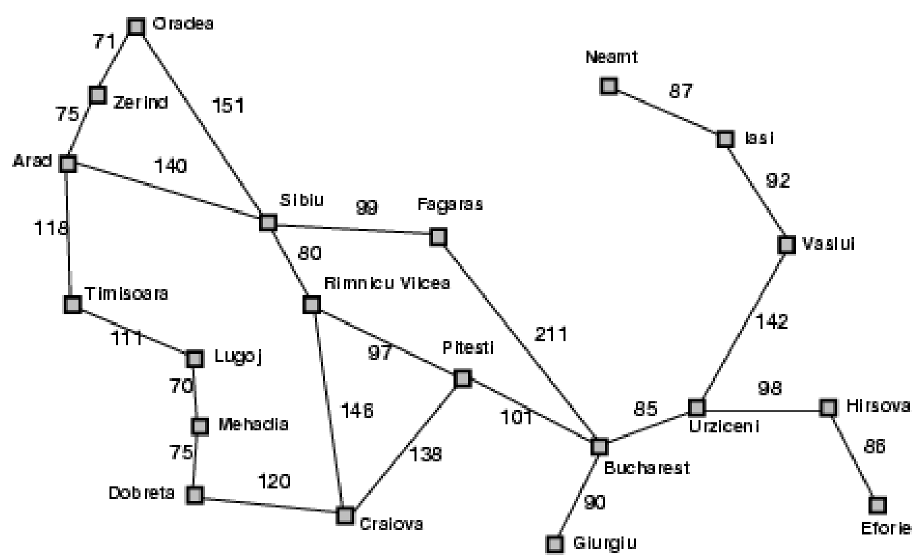


Straight–line distance
to Bucharest

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

A* search example

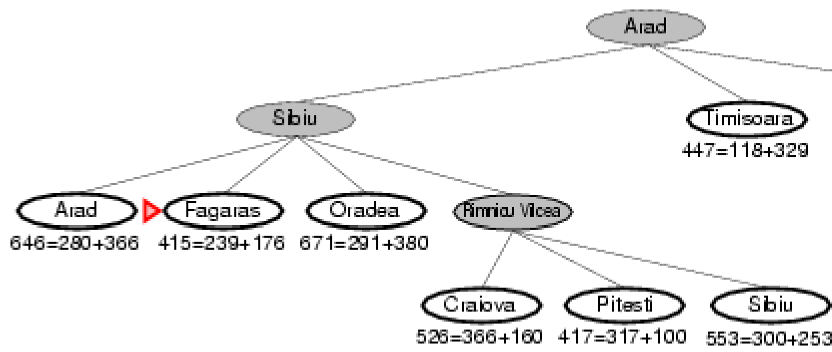


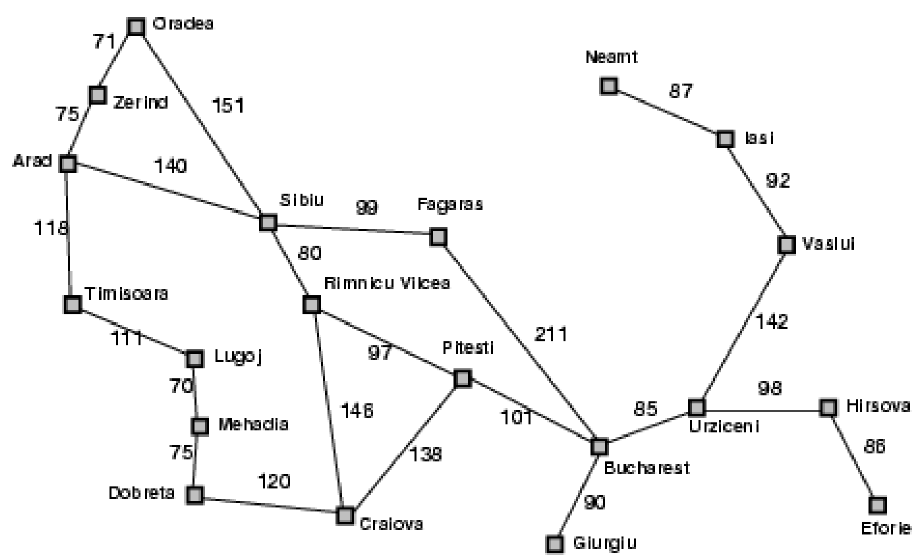


Straight–line distance to Bucharest

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

A* search example

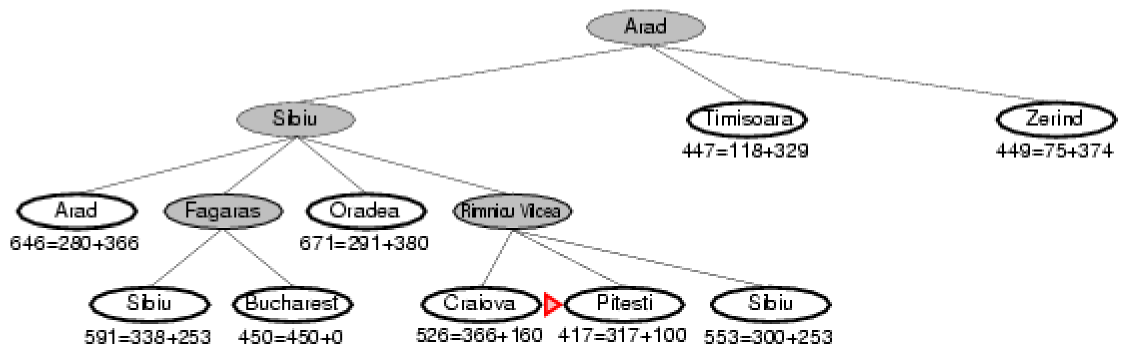


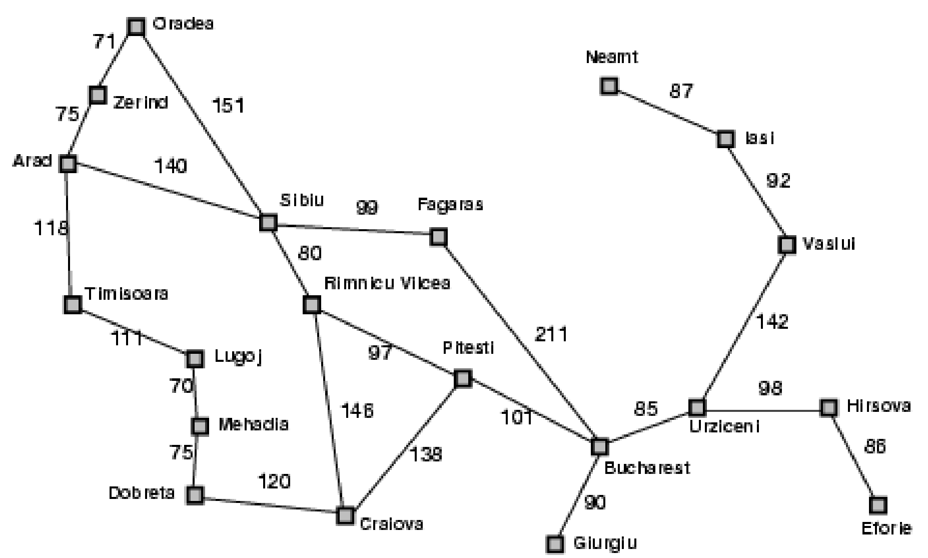


Straight–line distance
to Bucharest

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

A* search example

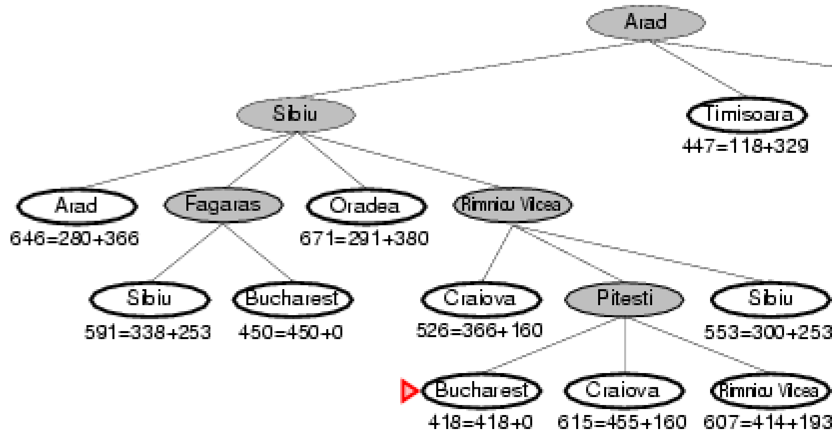


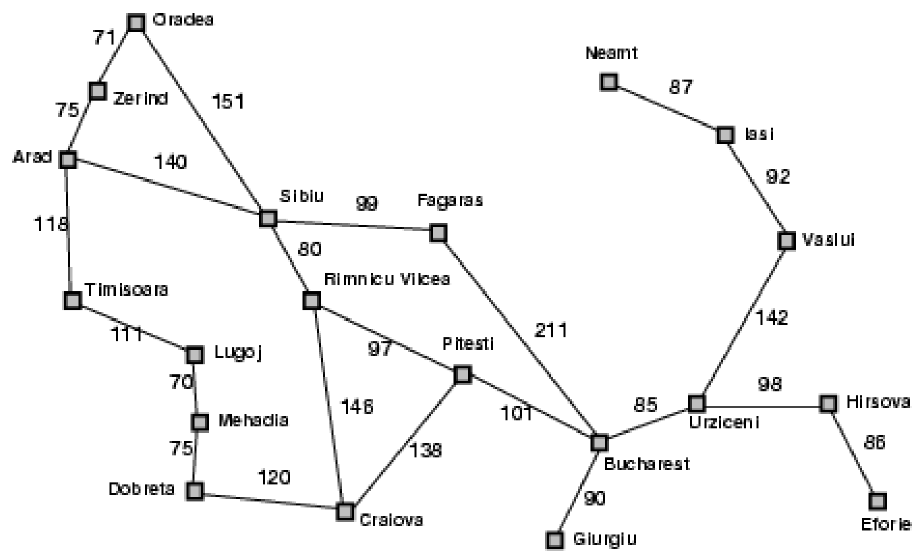


Straight–line distance to Bucharest

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

A* search example





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

A* Special

- Goal: find a minimum sum-cost path
- Notation:
  - c(n,n') - cost of arc (n,n')
  - g(n) = cost of current path from start to node n in the search tree.
  - h(n) = estimate of the cheapest cost of a path from n to a goal.
  - Special evaluation function: f = g+h
- f(n) estimates the cheapest cost solution path that goes through n.
  - h*(n) is the true cheapest cost from n to a goal.
  - g*(n) is the true shortest path from the start s, to n.
- If the heuristic function, h always underestimate the true cost (h(n) is smaller than h*(n)), then A* is guaranteed to find an optimal solution → admissible; and also has to be consistent

Properties of A*

- Complete? Yes, unless there are infinitely many nodes with $f \leq f(G)$
- Time? Exponential: $O(b^m)$
- Space? Keep all the nodes in memory: $O(b^m)$
- Optimal? Yes

Branch-and-Bound vs A*

- As in A*, look for a ***bound*** which is guaranteed lower than the true cost
- Search the branching tree in any way you like
  - e.g. depth first (no guarantee), best first
- Cut off search if cost + bound > best solution found
- If heuristic is cost + bound, search = best first
  - then BnB = A*
- Bounds often much more sophisticated
  - e.g. using mathematical programming optimisations

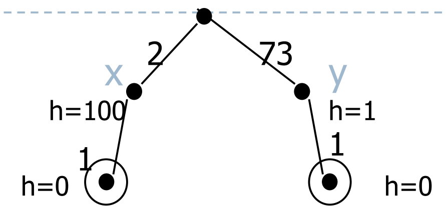
Admissible heuristics

- A heuristic *h(n)* is admissible if for every node *n*, $h(n) \le h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from *n*.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

- Theorem: If *h(n)* is admissible, A* using `TREE-SEARCH` is optimal

Admissibility

- What must be true about h for A* to find optimal path?
- A* finds optimal path if h is admissable; h is admissible when it never overestimates.
- In this example, h is not admissible.
- In route finding problems, straight-line distance to goal is admissible heuristic.



g(X)+h(X)=2+100=102

G(Y)+h(Y)=73+1=74

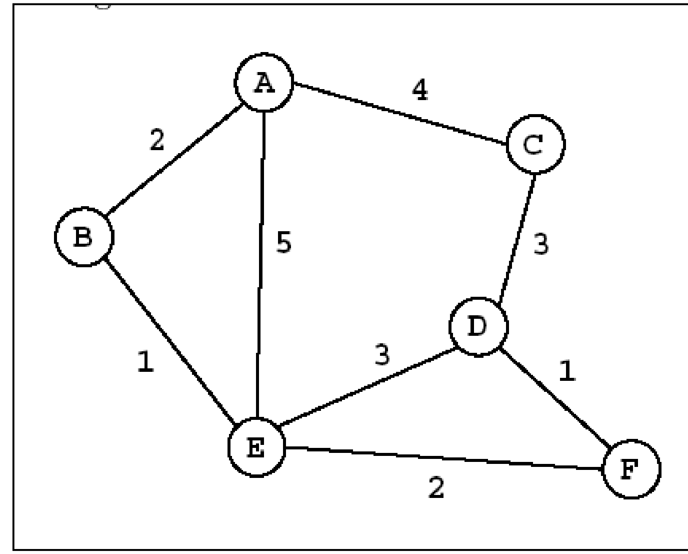
Optimal path is not found!

Because we choose Y, rather than X which is in the optimal path.

Contoh Soal UAS Sem 2 2014/2015

Dalam permainan *video game*, adakalanya entitas bergerak dalam *video game* perlu berpindah dari satu posisi ke posisi lain. Seringkali proses perpindahan perlu mengutamakan jalur terdekat atau biaya minimal karena berhubungan dengan poin yang diperoleh. Gambar di bawah ini menunjukkan contoh jalur yang mungkin dilewati oleh entitas bergerak dalam suatu *video game*. Suatu entitas akan berpindah dari posisi titik A menuju ke posisi titik F. Jika diperlukan informasi heuristik, nilai heuristik dari suatu simpul adalah **banyaknya busur minimal** yang menghubungkan titik tersebut ke titik tujuan.

Contoh Soal UAS Sem 2 2014/2015 (2)



- Pencarian solusi dengan:

a. UCS

b. Greedy Best First

c. A Star

Untuk masing-masing pendekatan tuliskan:

- Formula
- Iterasi
  - Simpul ekspan
  - Simpul hidup & nilai f(n)

Urut abjad, simpul ekspan tidak mengulang, tidak membentuk sirkuit, berhenti saat satu solusi ditemukan

Solusi:

Jawaban:

<table>
<tr><th rowspan="3">Iterasi</th><th colspan="2">UCS</th><th colspan="2">Greedy Best First Search</th><th colspan="2">A Star</th></tr>
<tr><th colspan="2">Formula: f(n) = g(n)</th><th colspan="2">Formula: f(n) = h(n)</th><th colspan="2">Formula: f(n) = g(n) + h(n)</th></tr>
<tr><th>Simpul - Ekspan</th><th>Simpu lHidup</th><th>Simpul–Ekspan</th><th>Simpul Hidup</th><th>Simpul - Ekspan</th><th>Simpul Hidup</th></tr>
<tr><td rowspan="3">1</td><td rowspan="3">A</td><td>Ba
f(B) = 2</td><td rowspan="3">A</td><td>Ea
f(Ea) = 1</td><td rowspan="3">A</td><td>Ba
f(Ba) = 2+2 = 4</td></tr>
<tr><td>Ca
f(C) = 4</td><td>Ba
f(Ba) = 2</td><td>Ca
f(Ca) = 4 + 2 = 6</td></tr>
<tr><td>Ea
f(E) = 5</td><td>Ca
f(Ca) = 2</td><td>Ea
f(Ea) = 5 + 1 = 6</td></tr>
<tr><td rowspan="5">2</td><td rowspan="3">Ba</td><td>Eba
f(Eba) = 3</td><td rowspan="5">Ea</td><td>De
f(De) = 1</td><td rowspan="3">Ba</td><td>Eba
f(Eba) = 3 + 1 = 4</td></tr>
<tr><td>Ca
f(C) = 4</td><td>Fea
f(Fea) = 0</td><td>Ca
f(Ca) = 4 + 2 = 6</td></tr>
<tr><td>Ea
f(E) = 5</td><td>Bea
f(Bea) = 2</td><td>Ea
f(Ea) = 5 + 1 = 6</td></tr>
<tr><td></td><td></td><td>Ba
f(Ba) = 2</td><td></td><td></td></tr>
<tr><td></td><td></td><td>Ca
f(Ca) = 2</td><td></td><td></td></tr>
</table>

3	Eba	Ca f(C) = 4	Fea	Sudah sampai solusi	Eba	Feba f(Feba) = 5 + 0 = 5
		Ea f(E) = 5				Ca f(Ca) = 4 + 2 = 6
		Feba f(Feba) = 5				Ea f(Ea) = 5 + 1 = 6
		Deba f(Deba) = 6				Deba f(Deba) = 6+1 = 7
4	Ca	Ea f(E) = 5			Feba	Sudah sampai solusi
		Feba f(Feba) = 5				
		Deba f(Deba) = 6				
		Dca f(Dca) = 7				
5	Ea	Feba f(Feba) = 5				
		Deba f(Deba) = 6				
		Dca f(Dca) = 7				
		Fea f(Fea) = 7				
		Dea f(Dea) = 8				

6	Feba	Solusi sudah ditemukan				
	Jalur: A-B-E-F		Jalur: A-E-F		Jalur: A-B-E-F	
	Jarak: 5		Jarak: 7		Jarak: 5	
Hasil	Banyaknya iterasi: 6		Banyaknya iterasi: 3		Banyaknya iterasi: 4	

Selamat Belajar