Artificial Intelligence

By Dr. Naglaa fathy

Lecturer, Computer Science Department

Faculty of Computers and Artificial Intelligence

Benha University

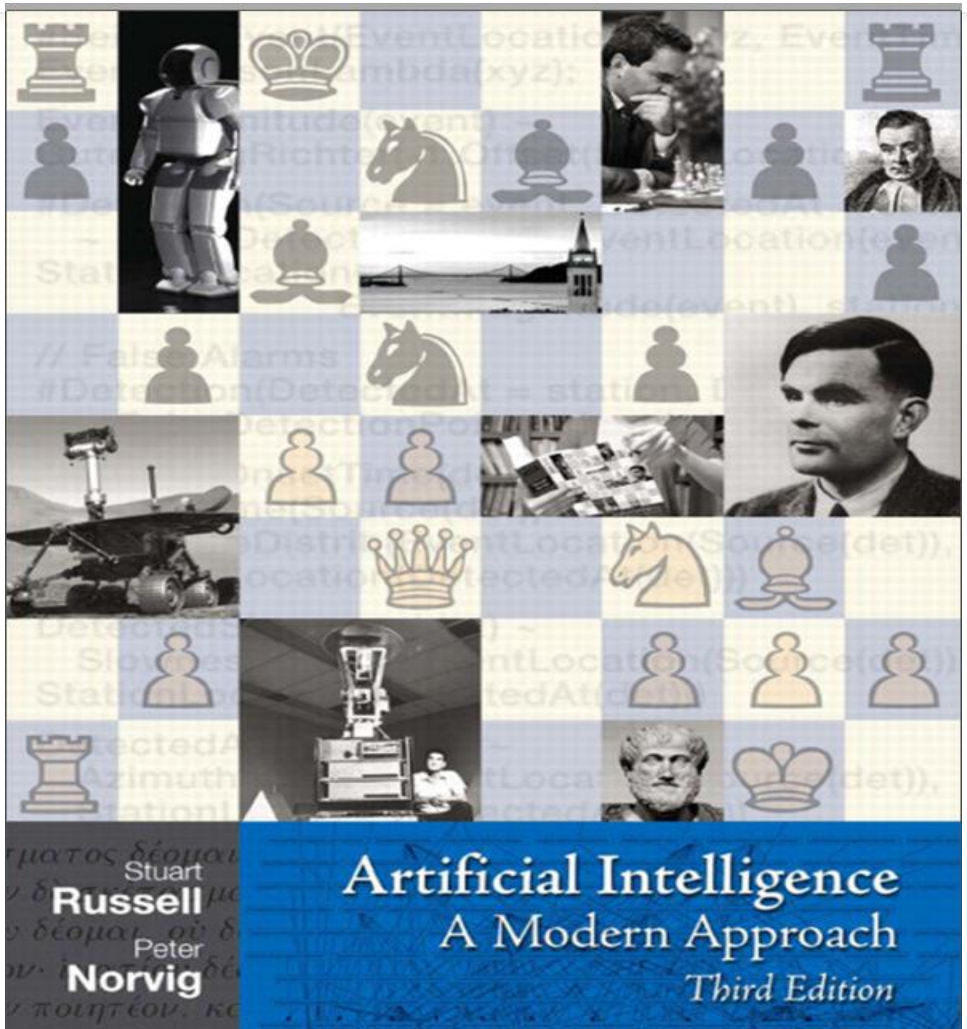
Lecture 5
Informed search algorithms

Lectures References

Artificial Intelligence
A Modern Approach

Third Edition

Stuart J. Russell and Peter Norvig



Agenda

- Best-first search
- Greedy best-first search
- A^* search

Informed search algorithms

- An informed search strategy—one that uses problem-specific knowledge beyond the definition of the problem itself—can find solutions more efficiently than an uninformed strategy.
- **Informed search algorithm** is also called **heuristic** search or directed search.
- In contrast to uninformed search algorithms, informed search algorithms require details such as
  - distance to reach the goal,
  - steps to reach the goal,
  - cost of the paths which makes this algorithm more efficient.
- Here, the goal state can be achieved by using the heuristic function.
-

Heuristic function

- The heuristic function is used to achieve the goal state with the lowest cost possible.
- Heuristic functions are the most common form in which additional knowledge of the problem is imparted to the search algorithm.
- This function estimates how close a state is to the goal.
- h(n) = estimated cost of the cheapest path from the state at node n to a goal state
- Specifically, h(n) = estimated cost (or distance) of minimal cost path from n to a goal state.
- h(n) takes a node as input, but, unlike g(n), it depends only on the state at that node.)
- If n is a goal node, then h(n) = 0.

Best-first search

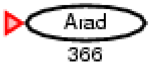
- **Best-first search** is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an evaluation function, f(n).
- **The evaluation function** is construed as a cost estimate, so the node with the lowest evaluation is expanded first.
- **Implementation:**

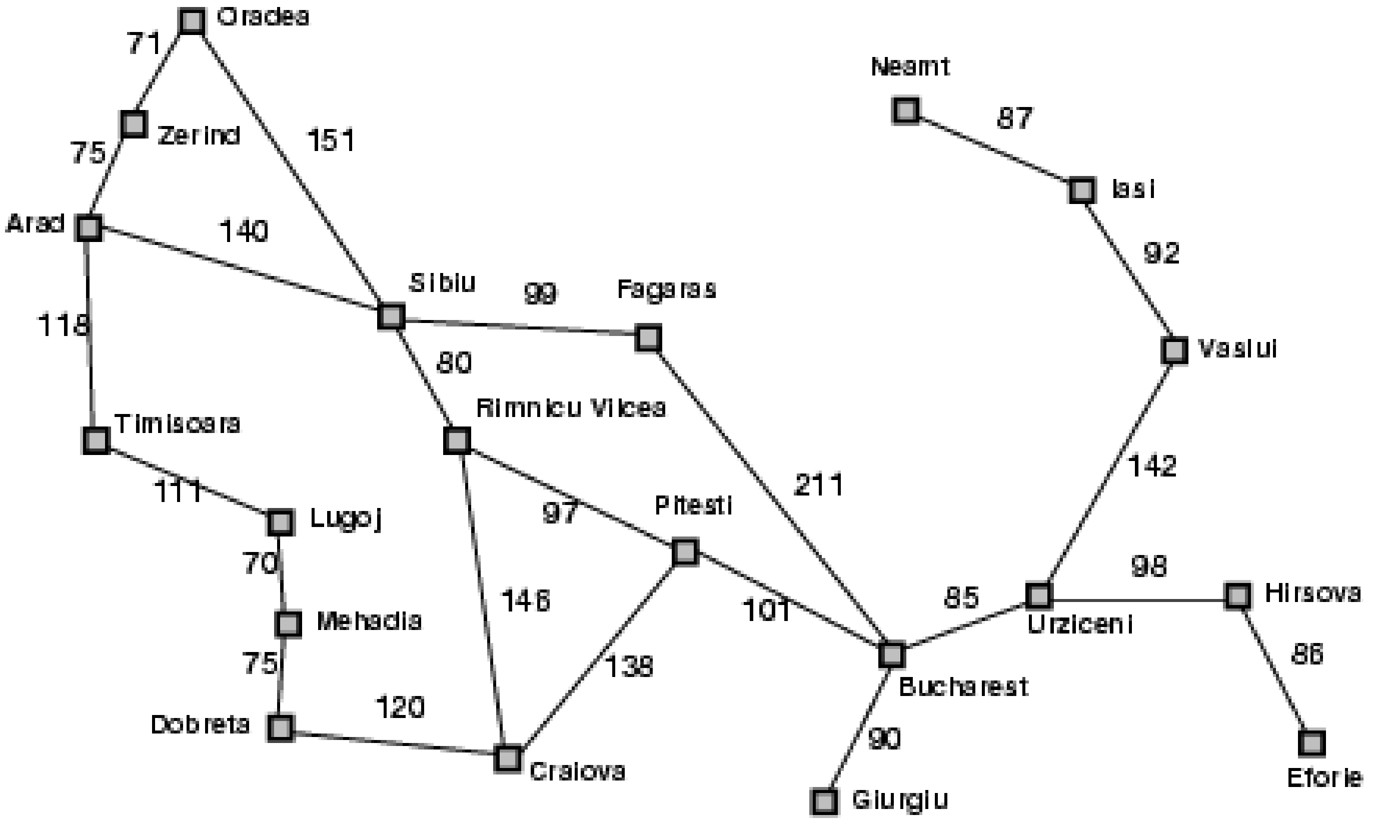
  Order the nodes in fringe increasing the order of cost.
- **Special cases:**
  - greedy best-first search
  - A* search
- If f(n)=g(n) uniform cost search
- If f(n)= h(n) greedy best search
- If f(n)=g(n)+h(n) A*

Greedy best-first search

- **Greedy best-first** search tries to expand the node that is closest to the goal, because this is likely to lead to a solution quickly.
- Greedy best-first search expands the node that appears to be closest to goal
- Thus, it evaluates nodes by using just the heuristic function; that is, f(n) = h(n).

Greedy best-first search example

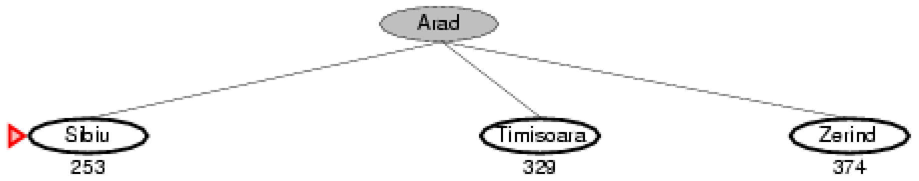


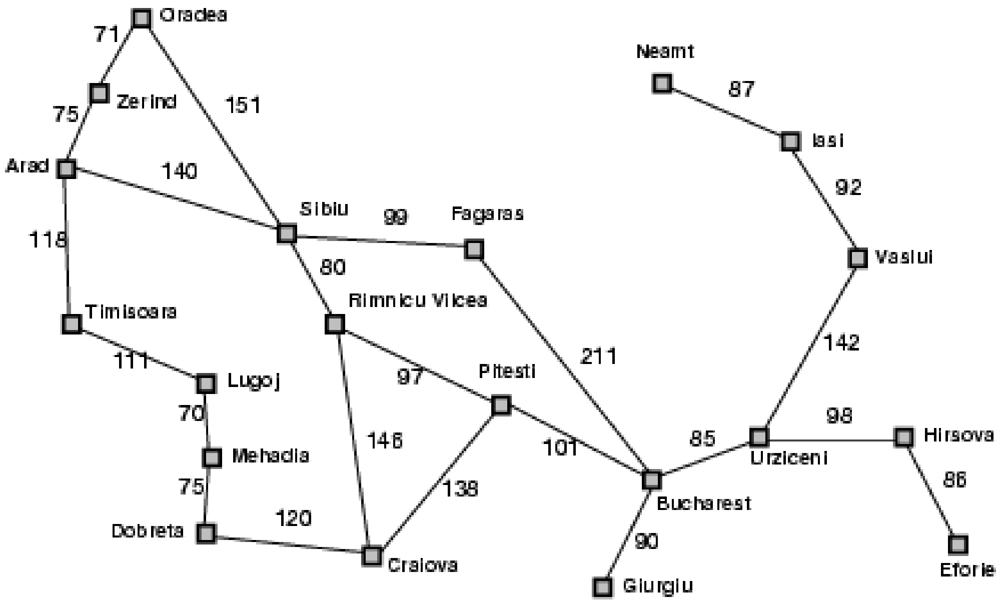


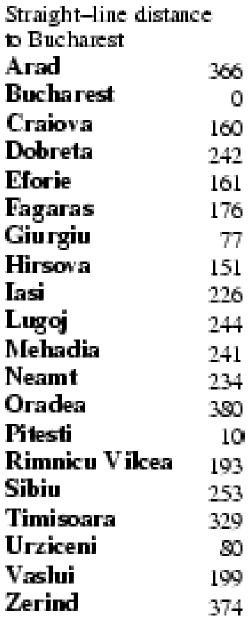
Straight–line distance
to Bucharest

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

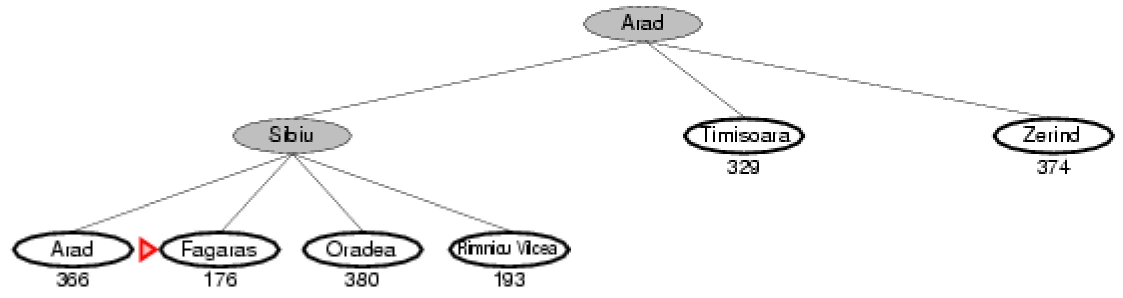


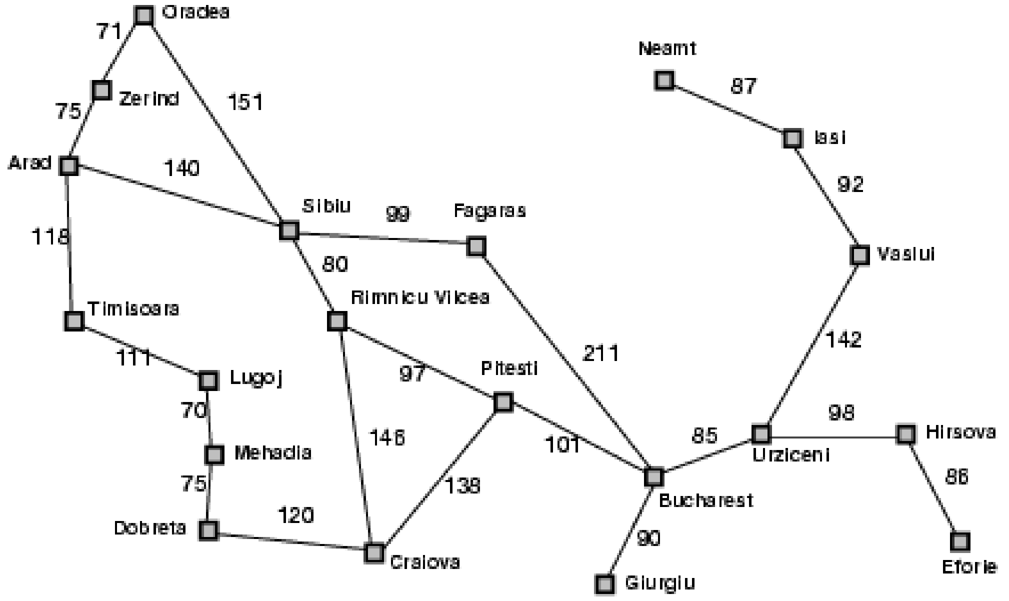




Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

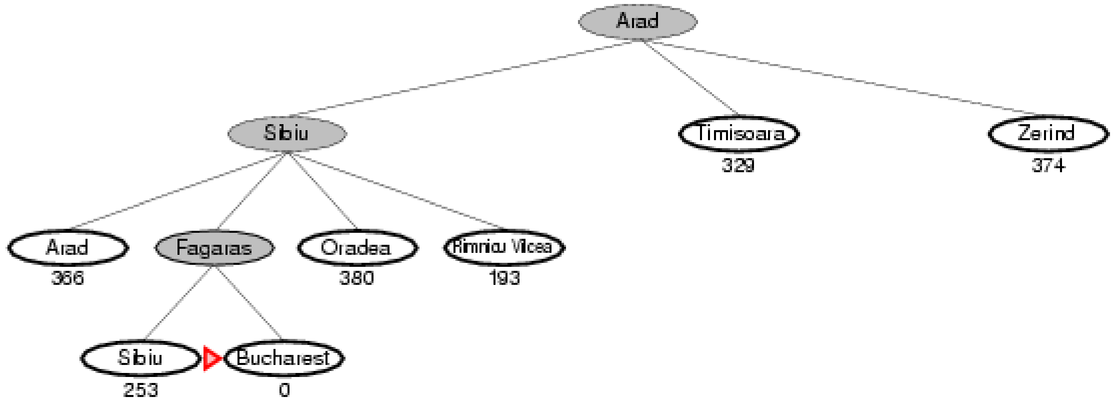
Greedy best-first search example

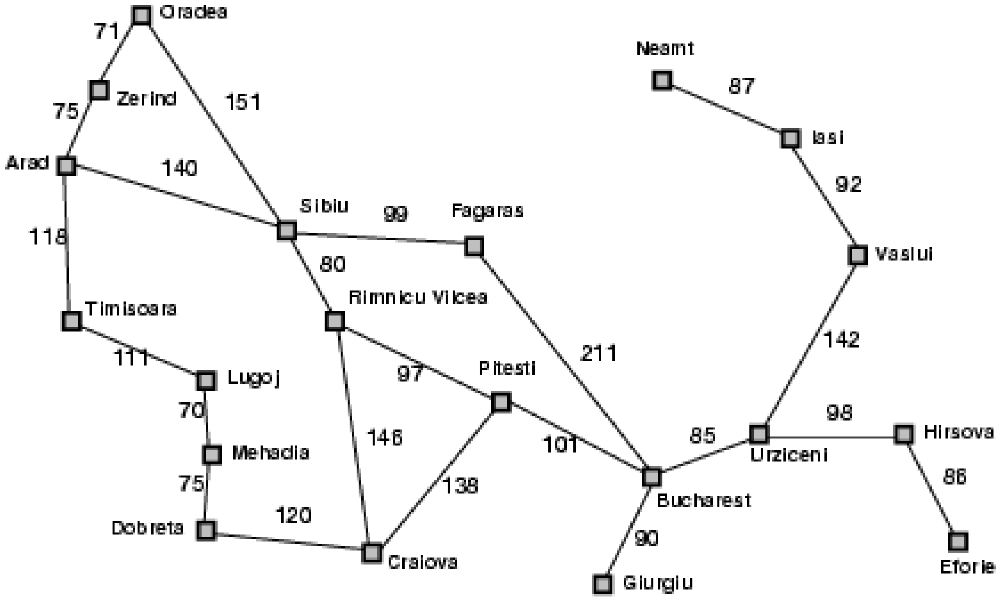




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search example

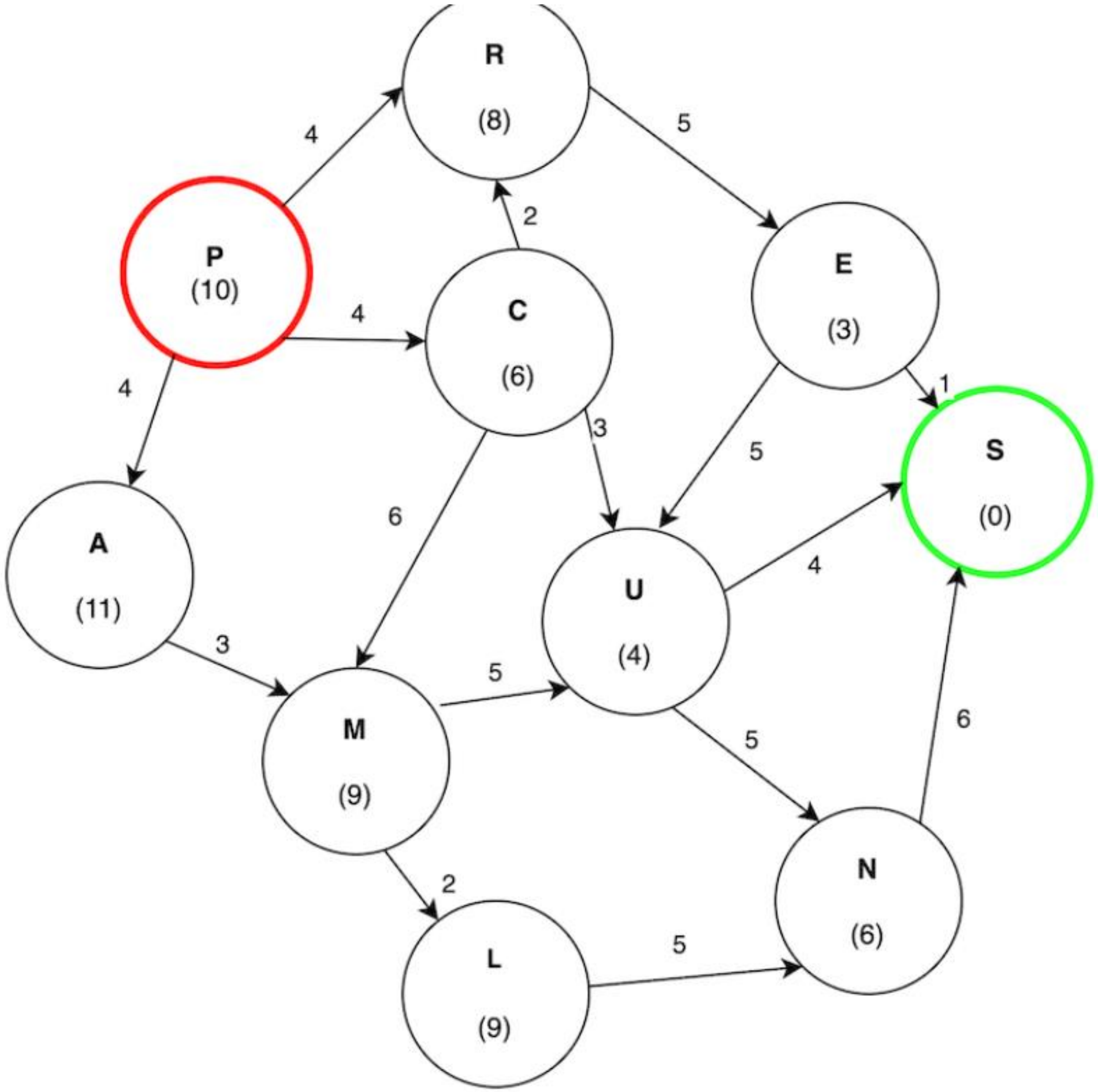




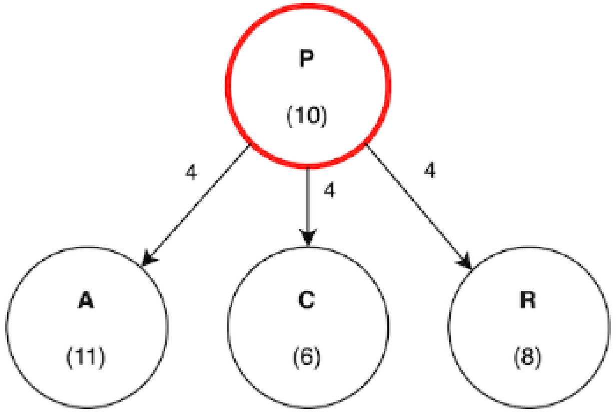
Straight–line distance
to Bucharest

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

Example



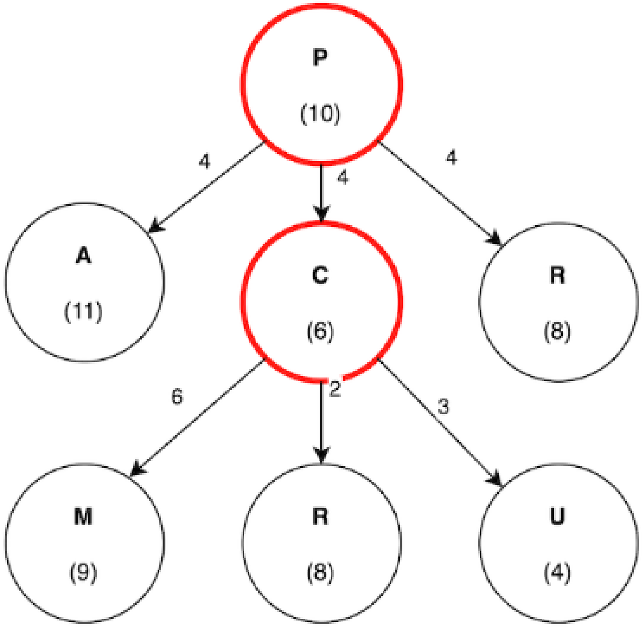
Example



Node[cost]
A[11]
C[6]
R[8]

Closed List
P

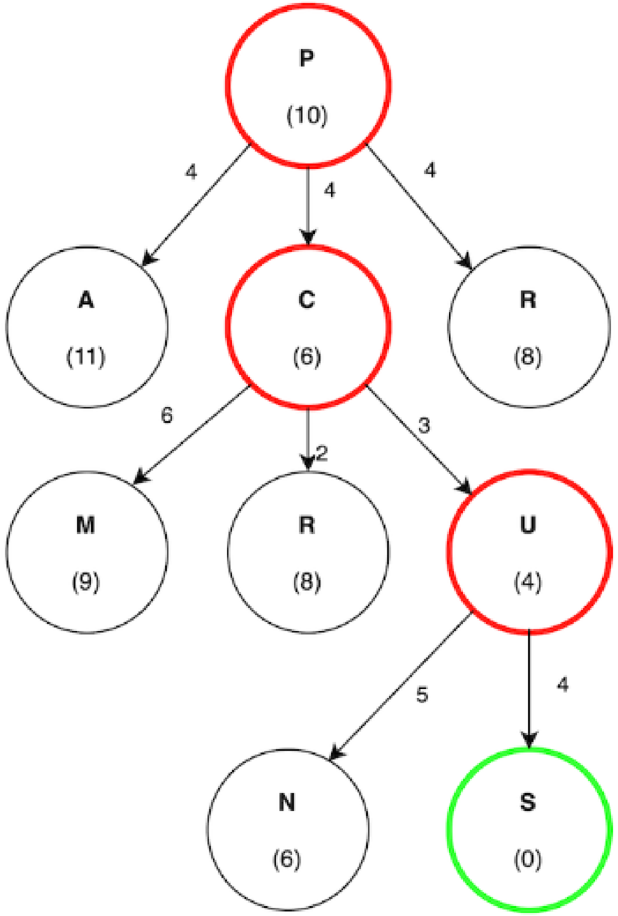
Example



Node[cost]
M[9]
R[8]
U[4]

Closed List
P
C

Example



Node[cost]
N[6]
S[0]

Closed List
P
C
U

Properties of greedy best-first search

Completeness: No – can get stuck in loops.

Optimality: no

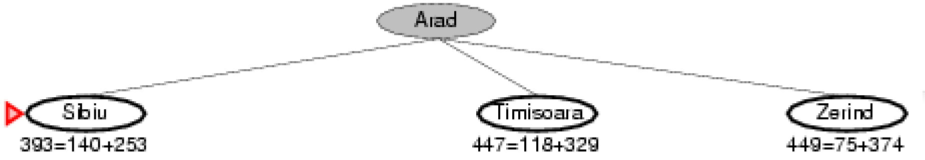
Time complexity: *exponential*

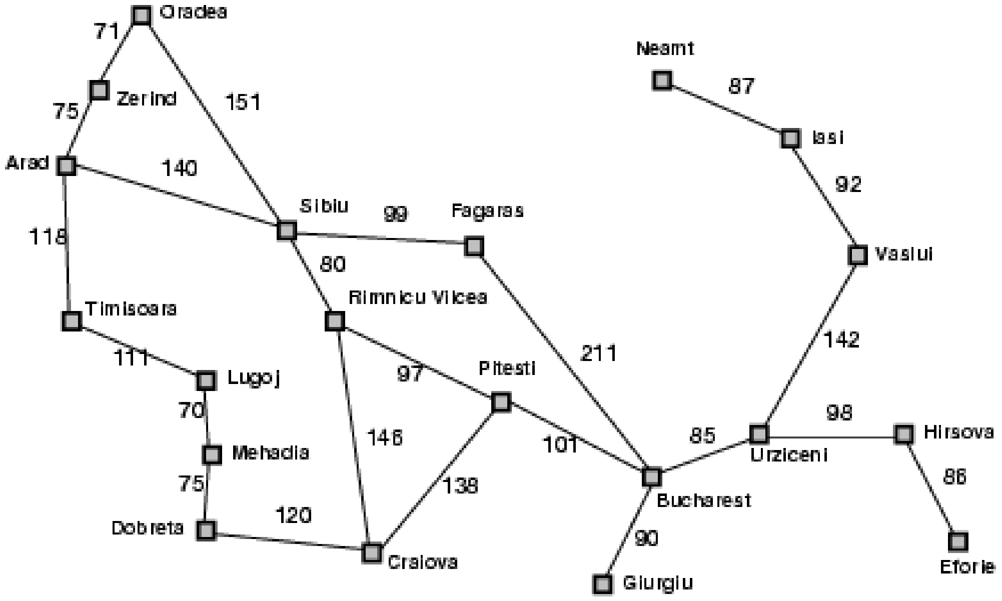
Space complexity: keeps all nodes in memory

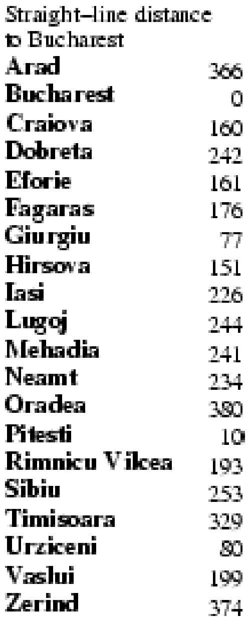
A* search

- 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

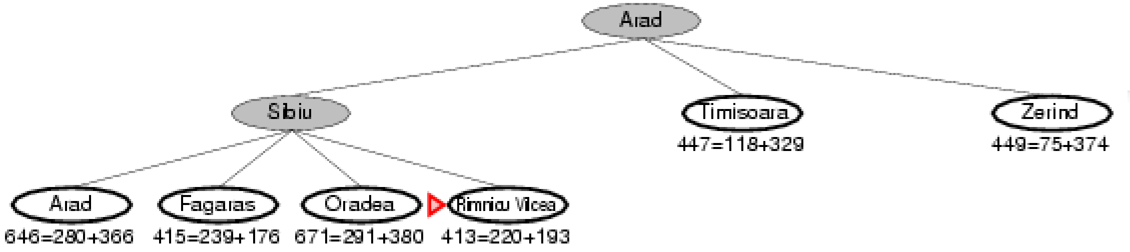


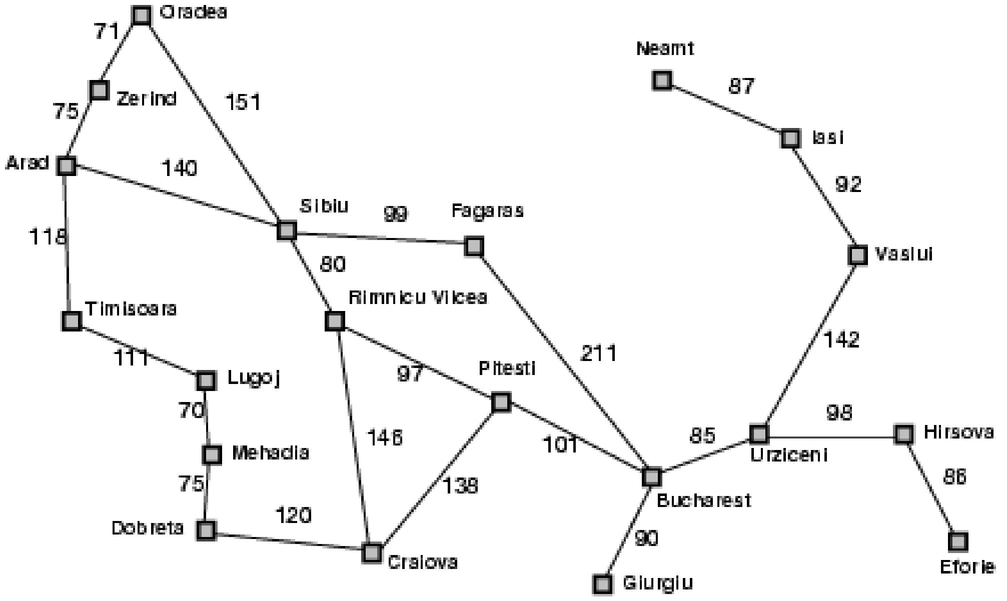


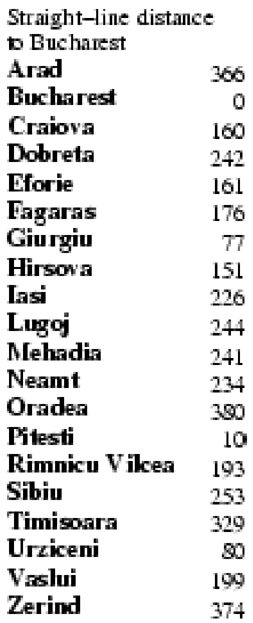


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example



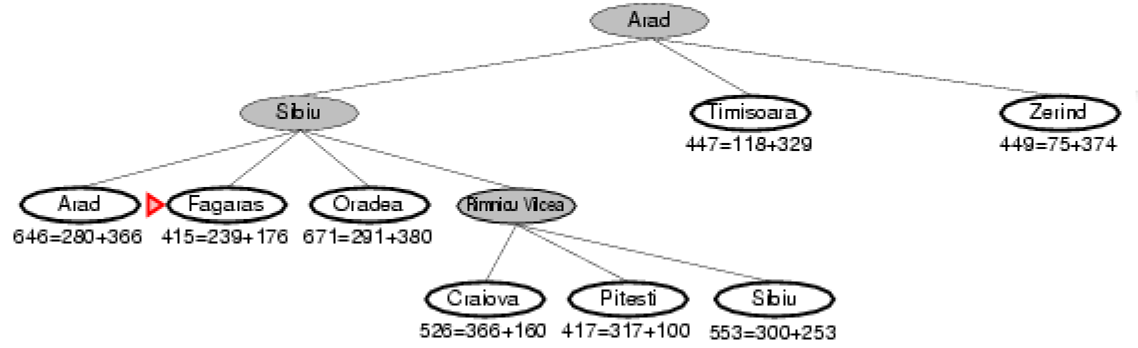


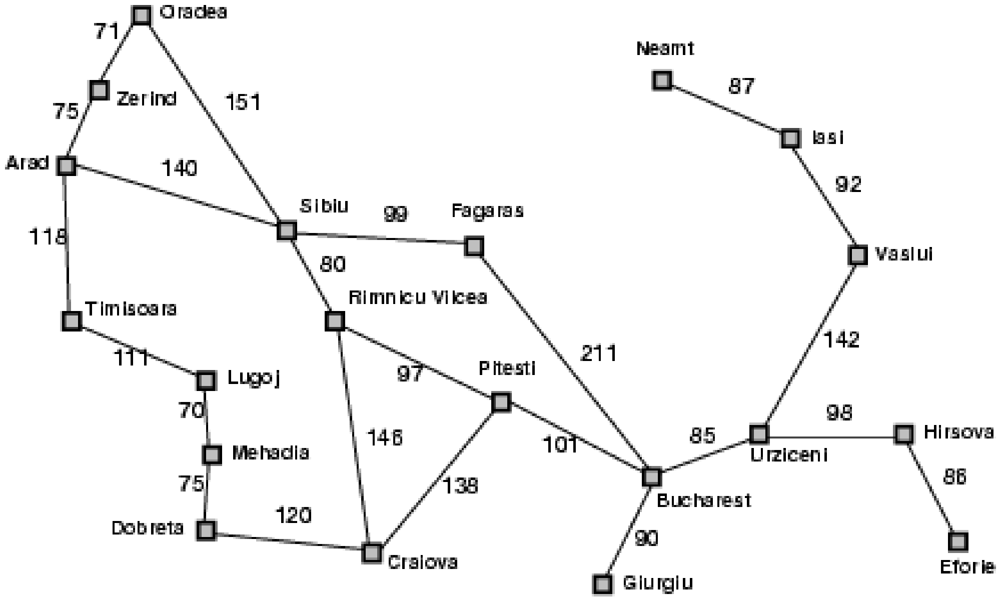


Straight–line distance to Bucharest

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

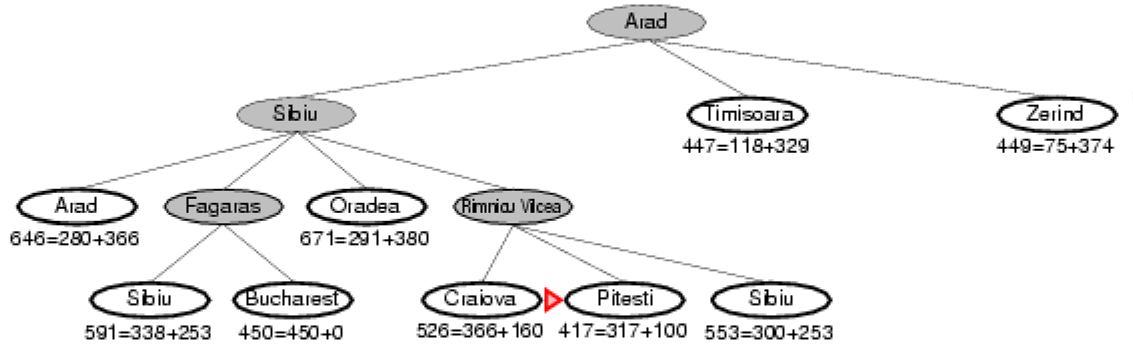
A* search example

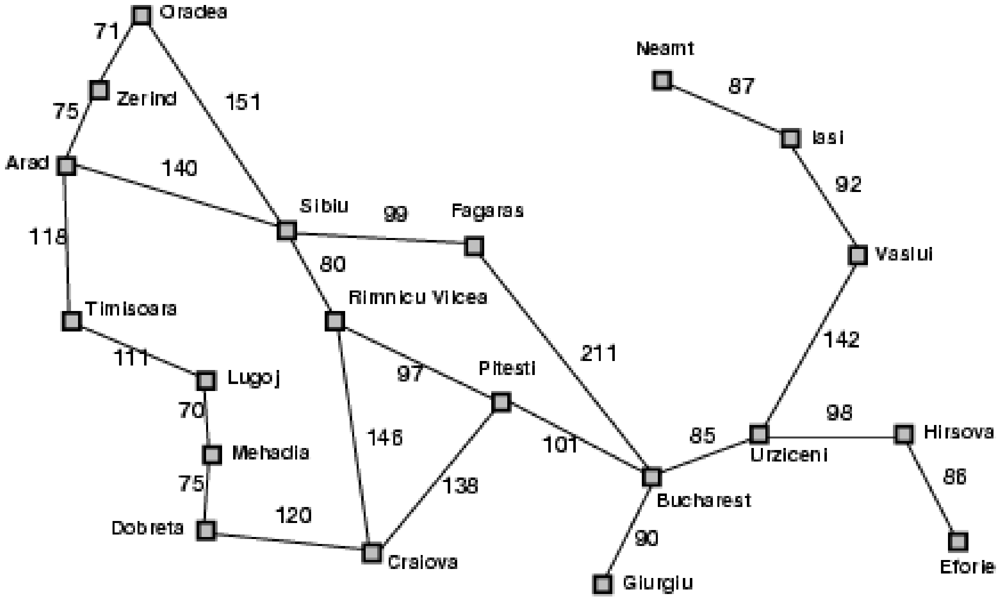


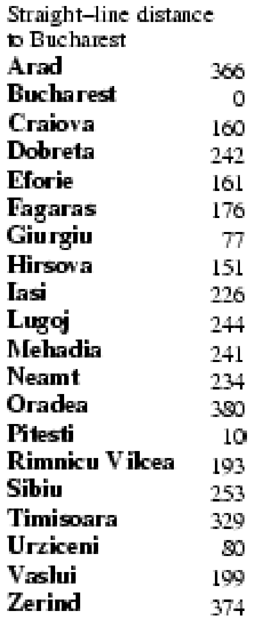


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

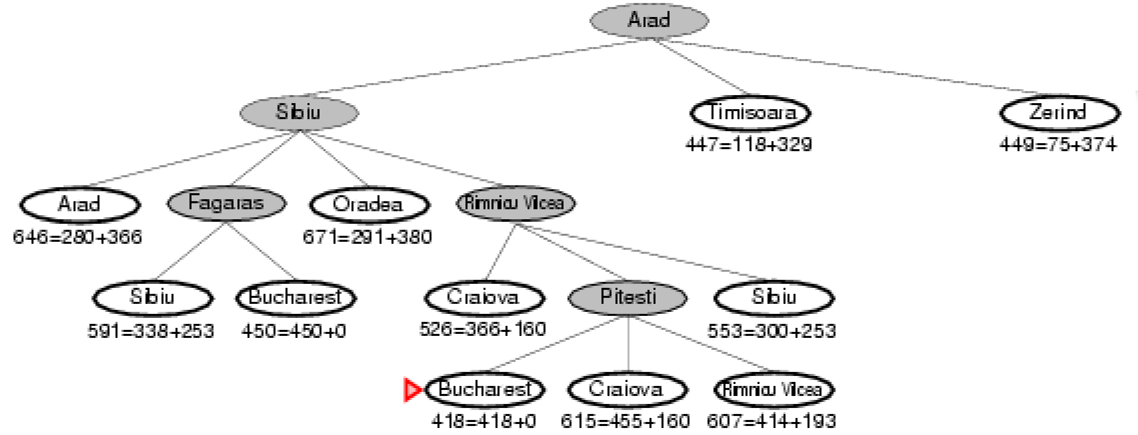


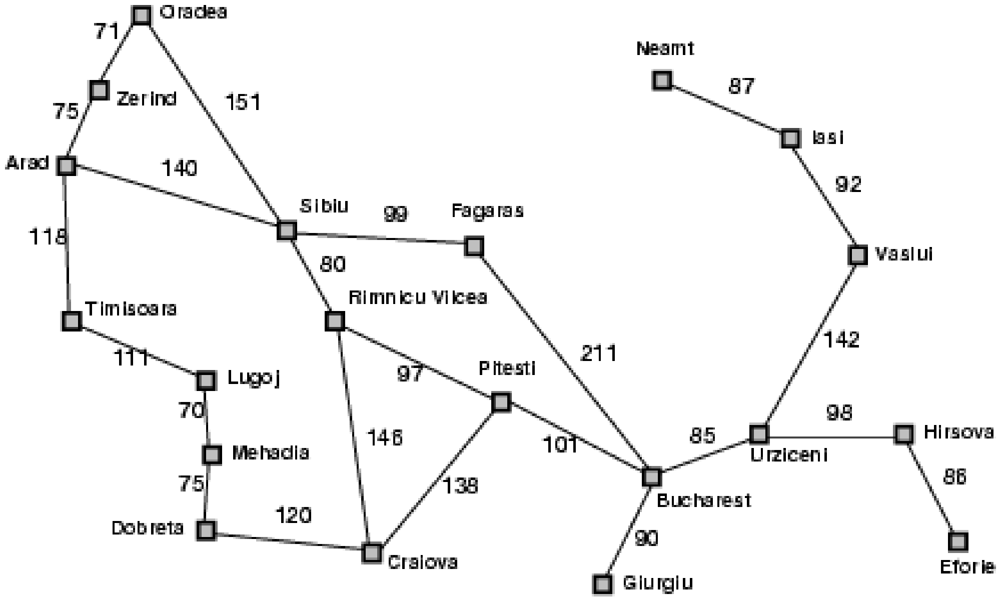




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

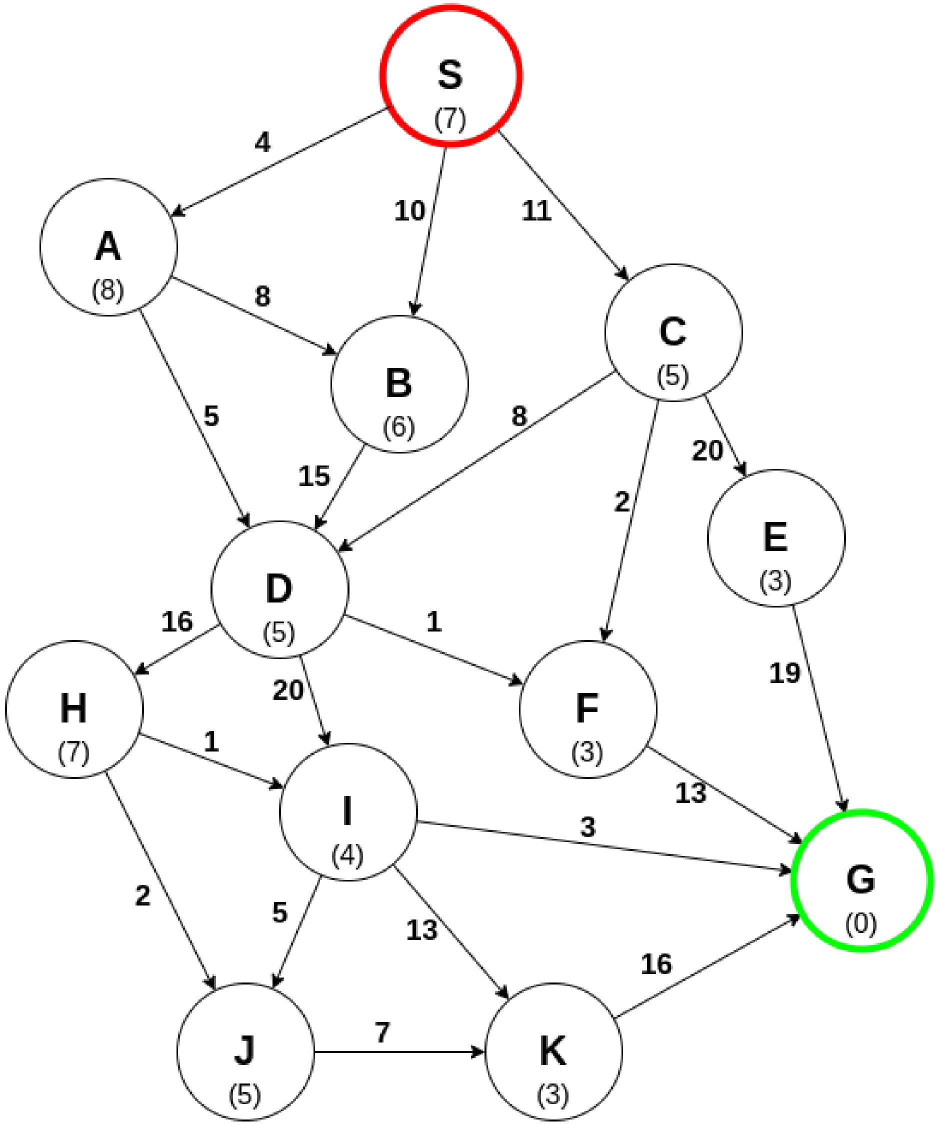




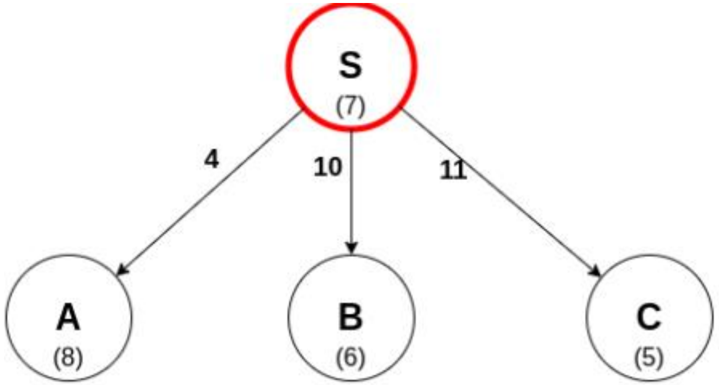
Straight–line distance
to Bucharest

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

Example


S
(7)
4
10
11
A
(8)
8
B
(6)
C
(5)
5
15
8
20
2
E
(3)
D
(5)
16
1
19
H
(7)
20
F
(3)
1
13
I
(4)
3
G
(0)
2
5
13
16
J
(5)
7
K
(3)

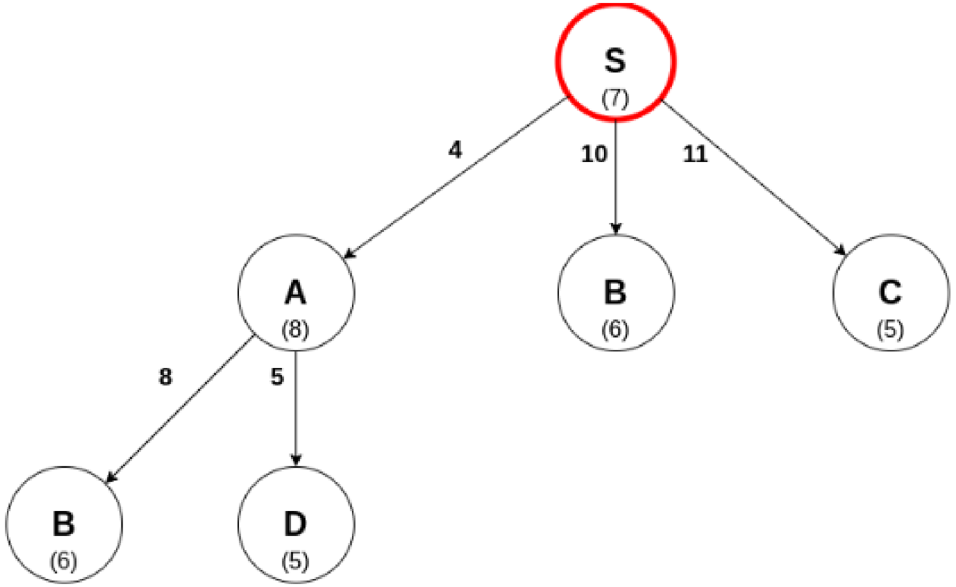
Example



Node[cost]
A[12]
B[16]
C[16]

Closed List
S

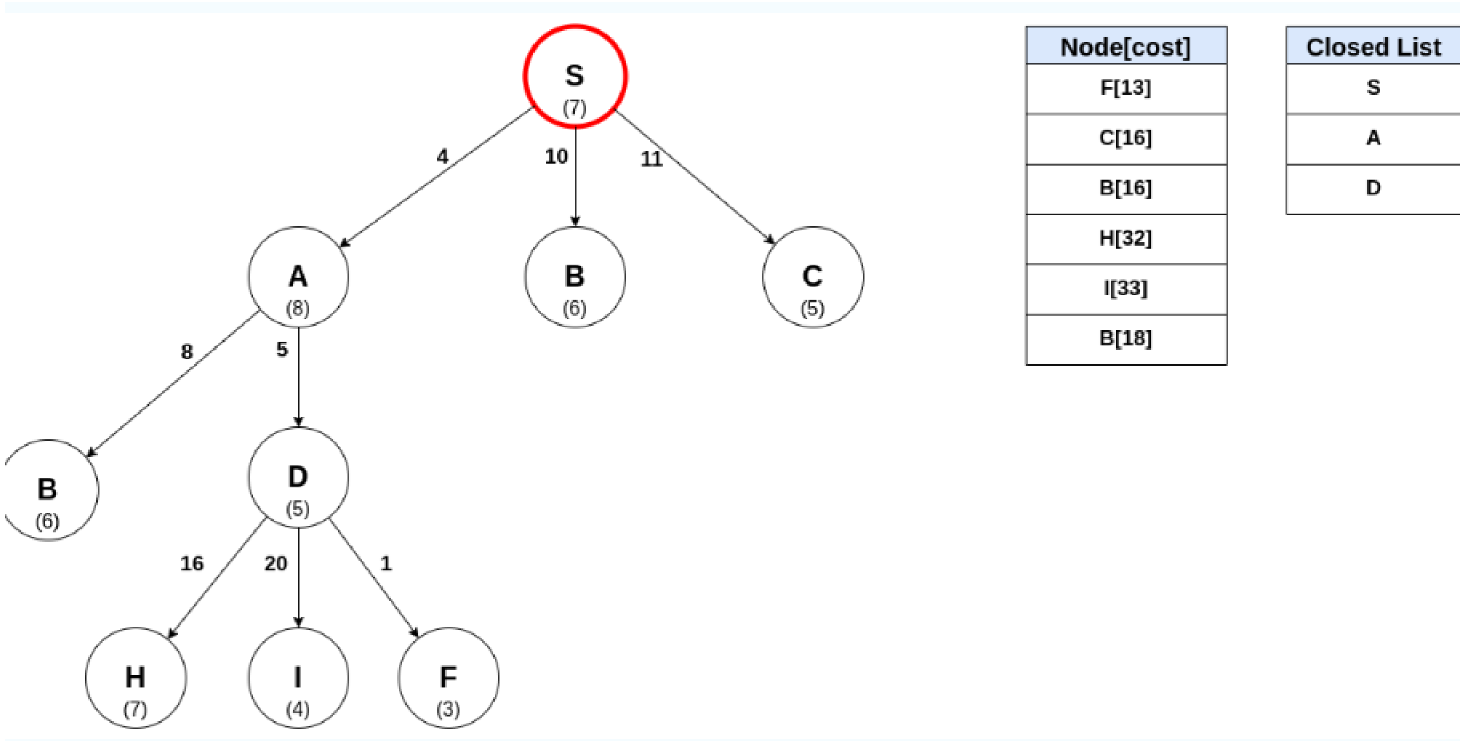
Example



Node[cost]
D[14]
C[16]
B[16]
B[18]

Closed List
S
A

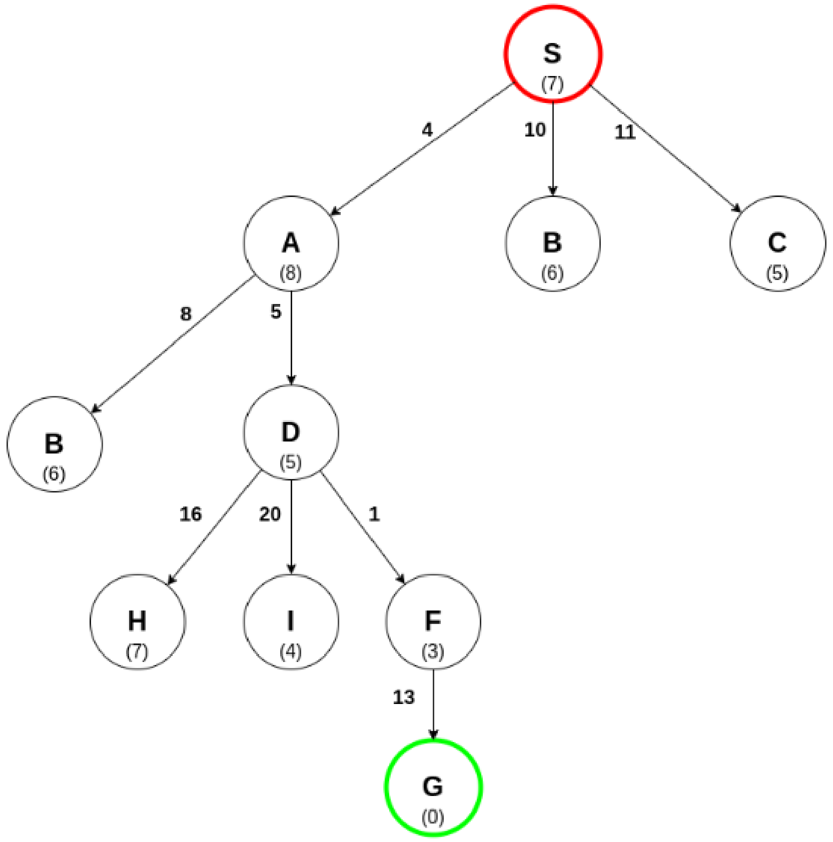
Example



Node[cost]
F[13]
C[16]
B[16]
H[32]
I[33]
B[18]

Closed List
S
A
D

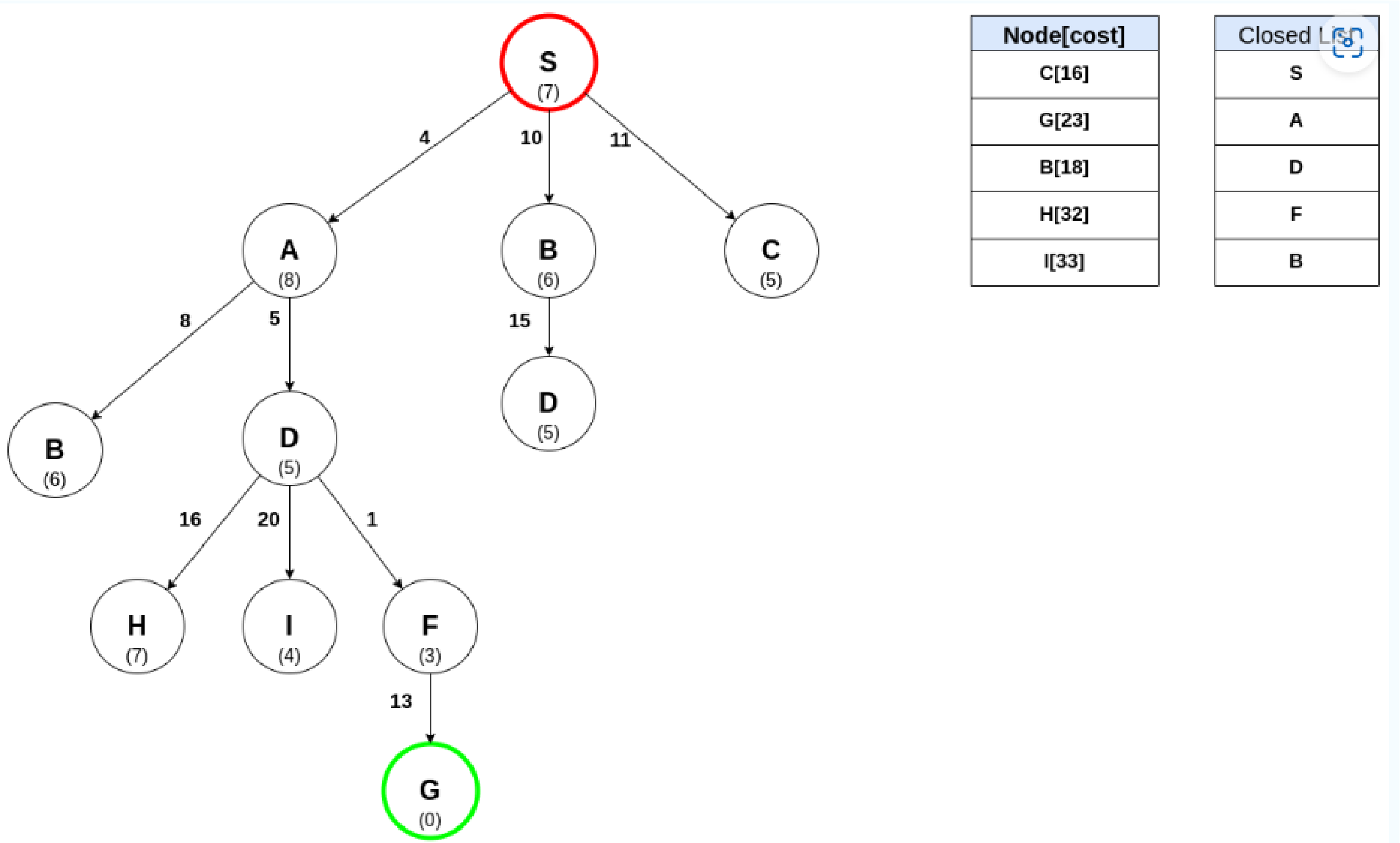
Example



Node[cost]
B[16]
C[16]
B[18]
H[32]
I[33]
G[23]

Closed List
S
A
D
F

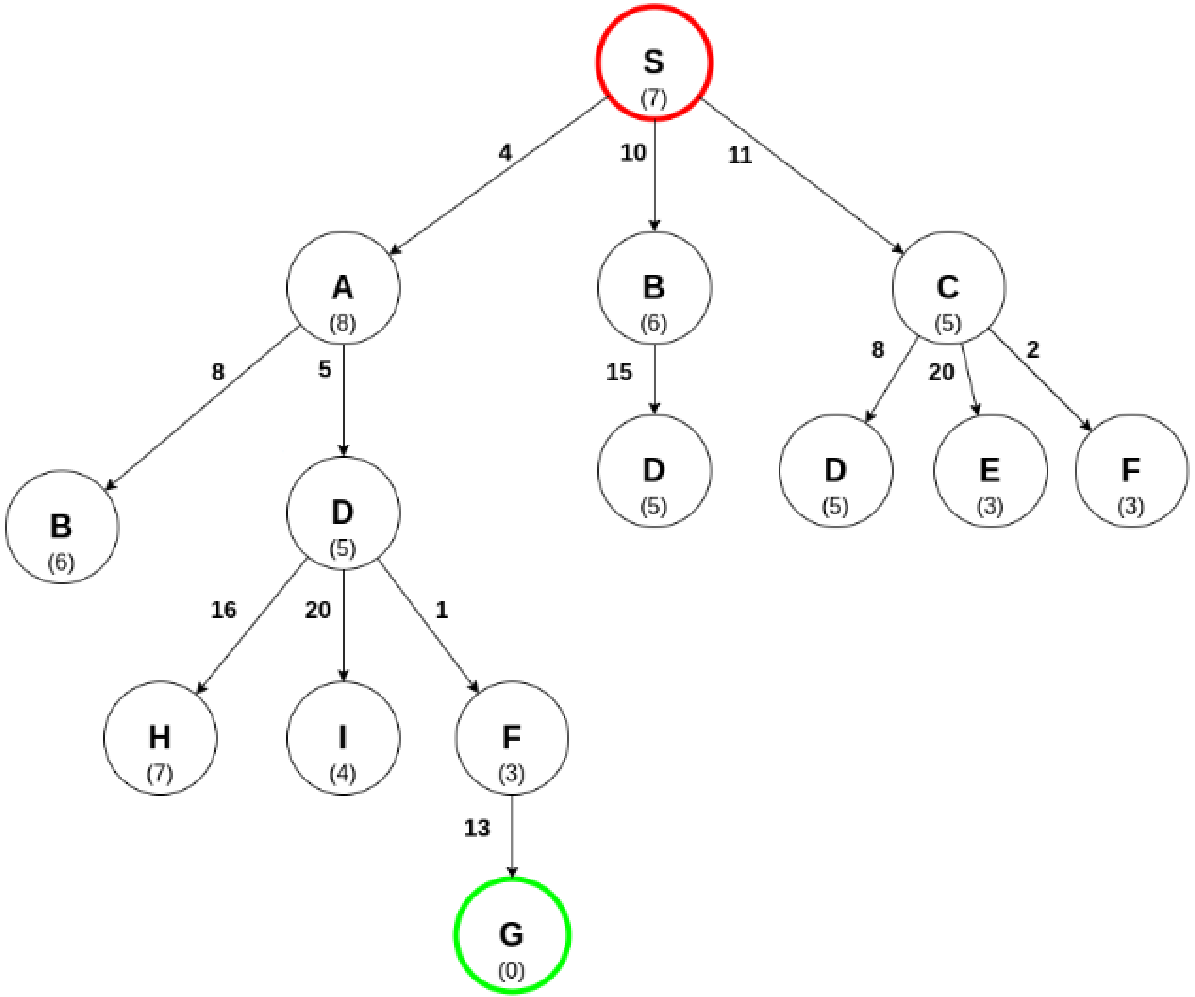
Example



Node[cost]
C[16]
G[23]
B[18]
H[32]
I[33]

Closed List
S
A
D
F
B

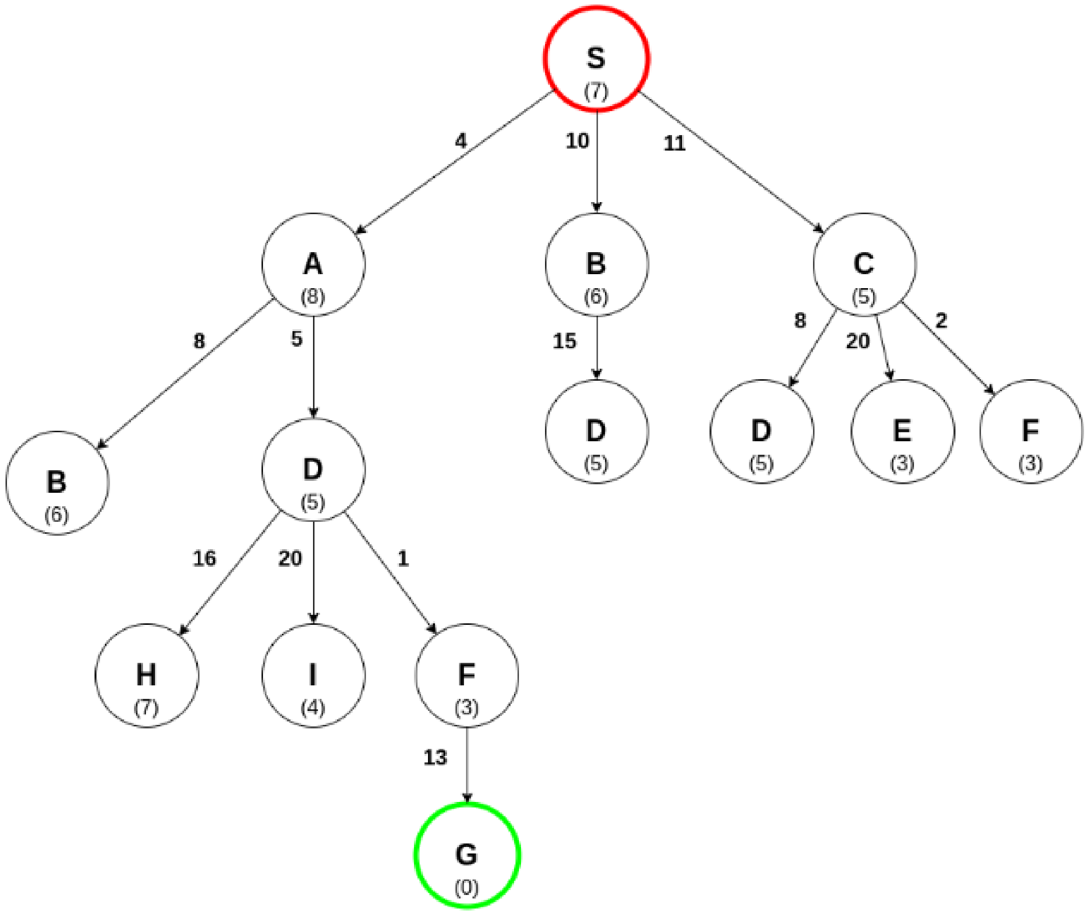
Example



Node[cost]
B[18]
G[23]
I[33]
H[32]
E[36]

Closed List
S
A
D
F
B
C

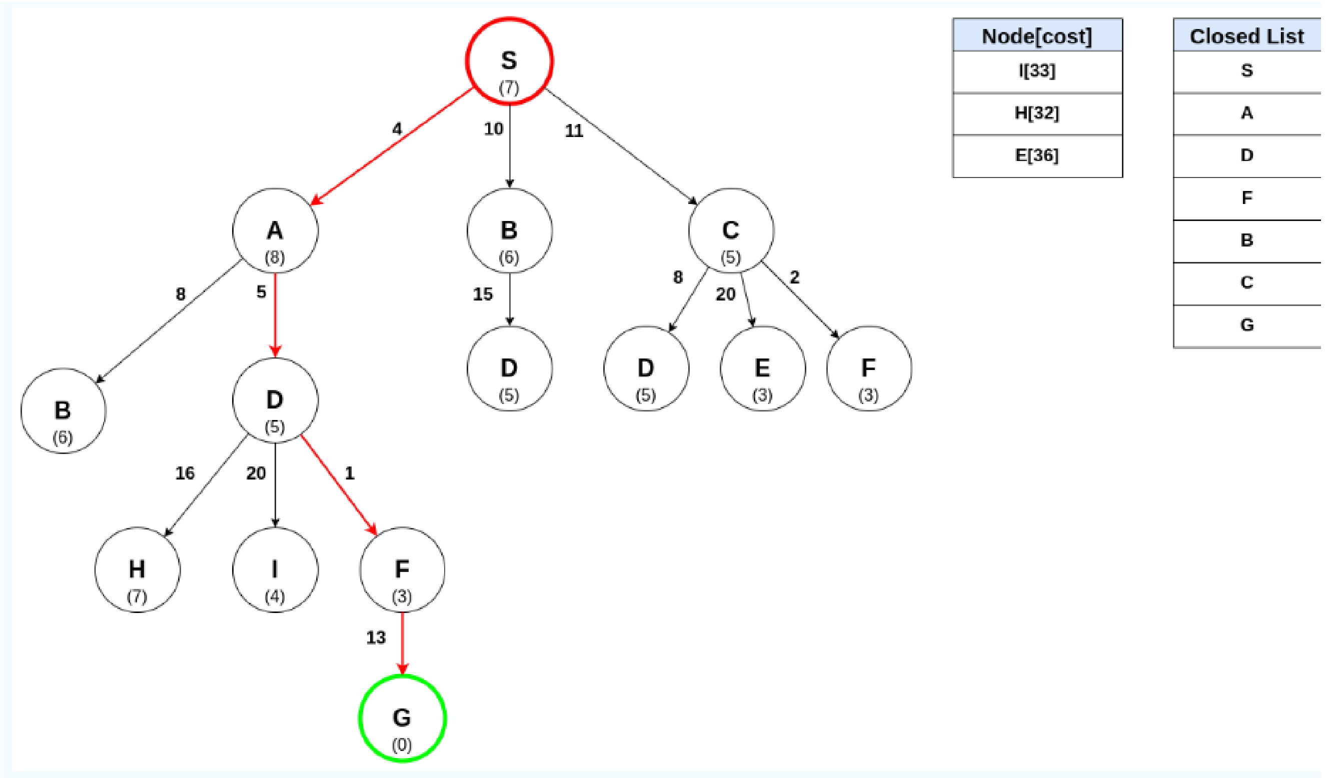
Example



Node[cost]
G[23]
I[33]
H[32]
E[36]

Closed List
S
A
D
F
B
C

Example



Node[cost]
I[33]
H[32]
E[36]

Closed List
S
A
D
F
B
C
G

Properties of A*search

Completeness: yes.

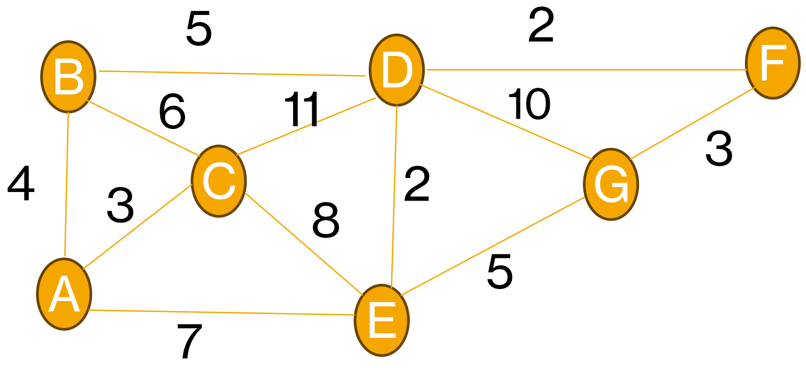
Optimality: yes

Time complexity: exponential

Space complexity: keep all nodes in memory

Assignment

- Let h(n)=node_level(n) *2
- node_level(n)=sum of all connected edges to n
- Find the shortest path between A and F using
  - greedy best search
  - A*
  - Uniform cost search



THANKYOU