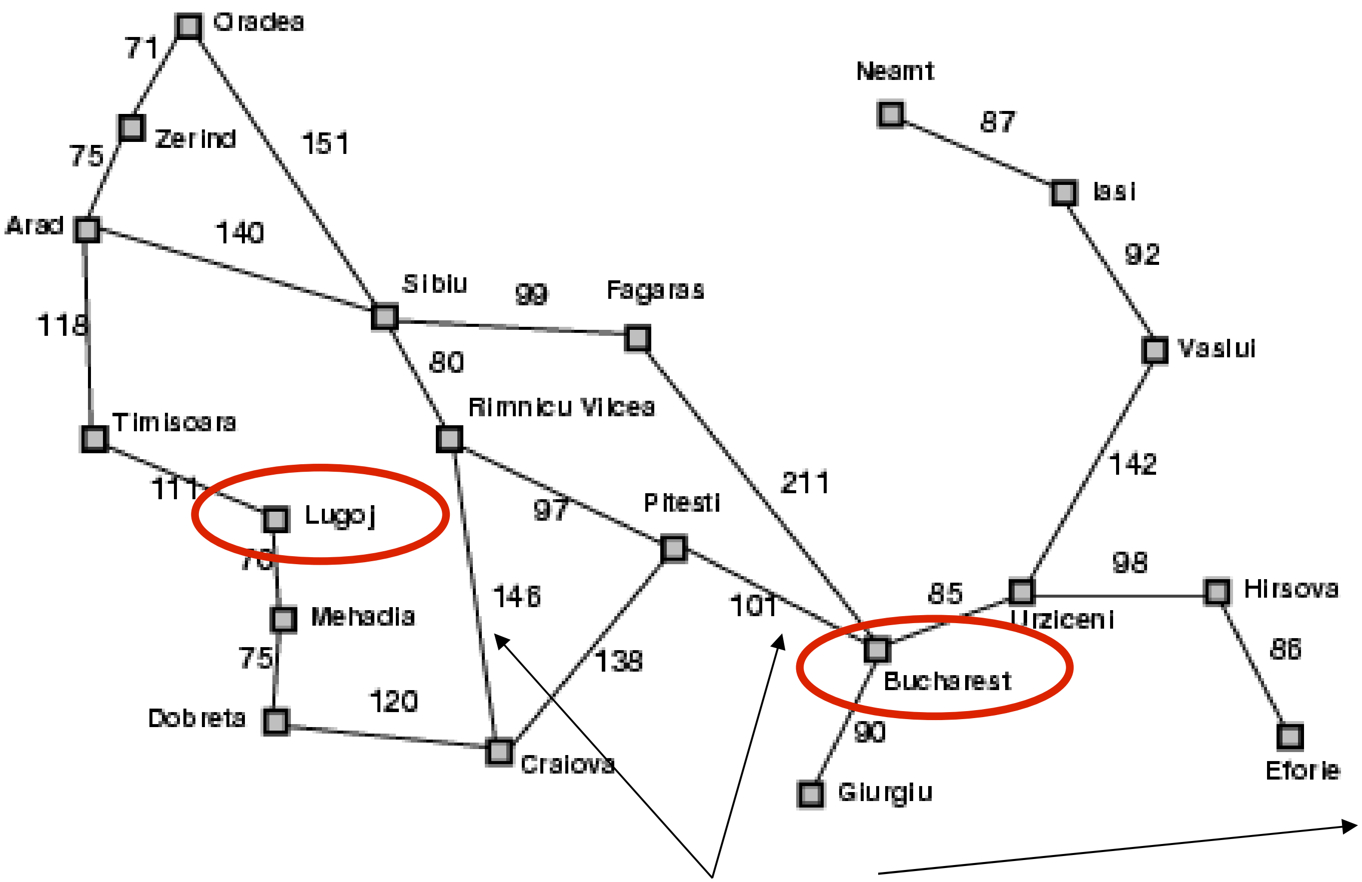
CompSci 171: Intro AI

Homework 3

Informed search

4.1 A* search: From Lugoj to Bucharest

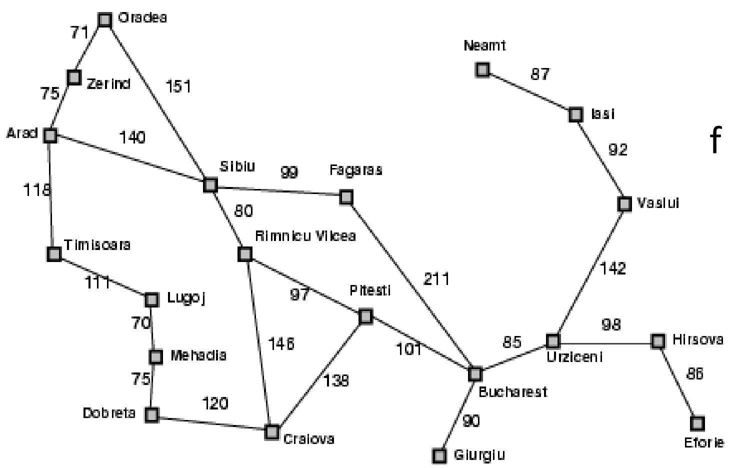


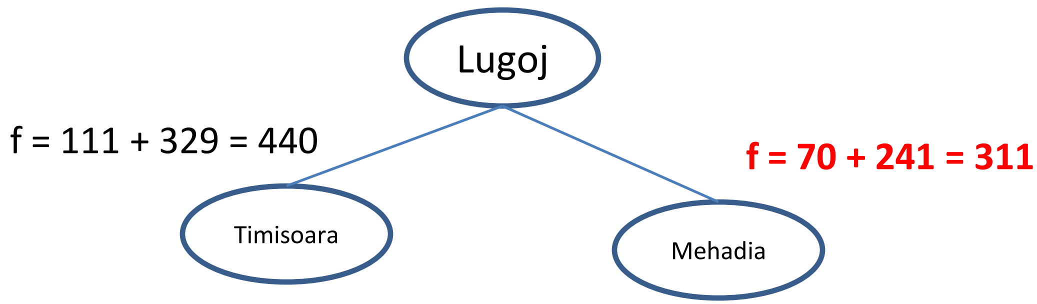
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

f(n) = g(n) + h(n)

A* search is guided by evaluation function f(n)

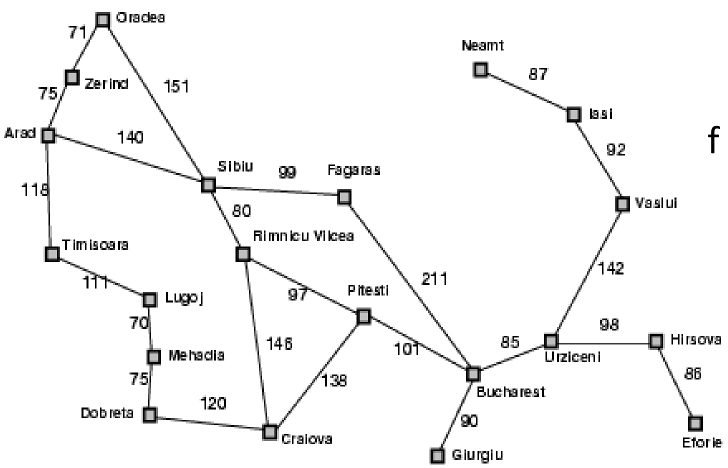
A* search: From Lugoj to Bucharest

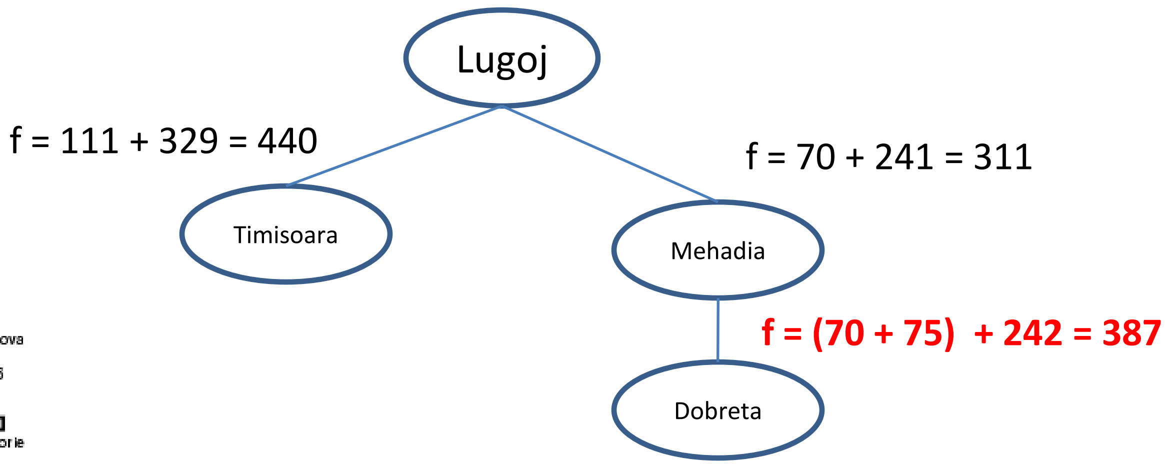




Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

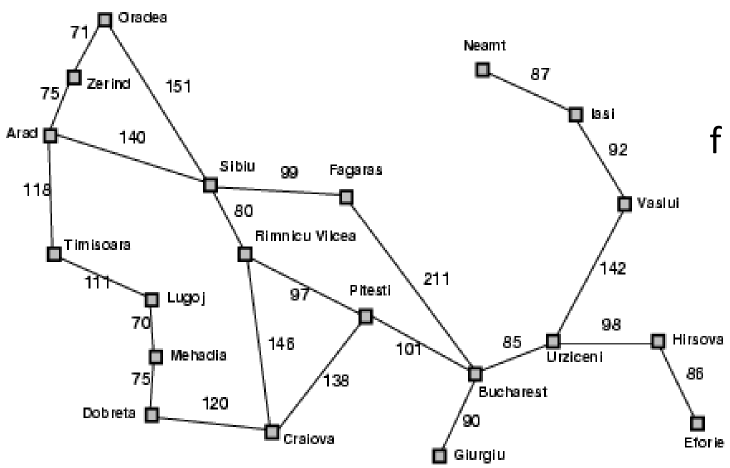
A* search: From Lugoj to Bucharest

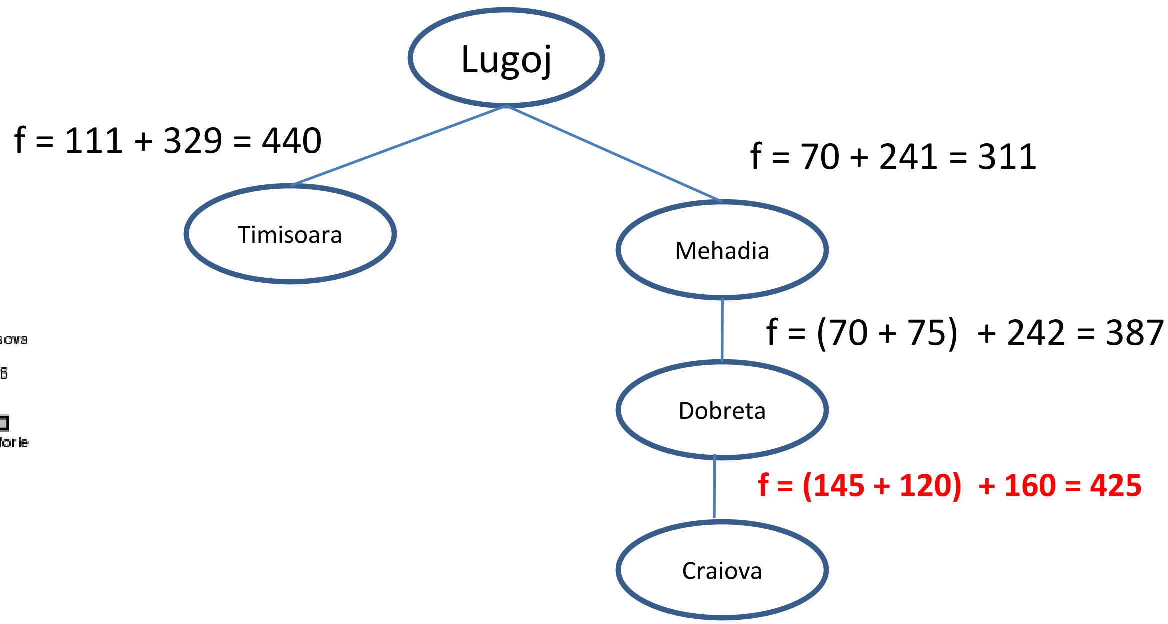




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search: From Lugoj to Bucharest

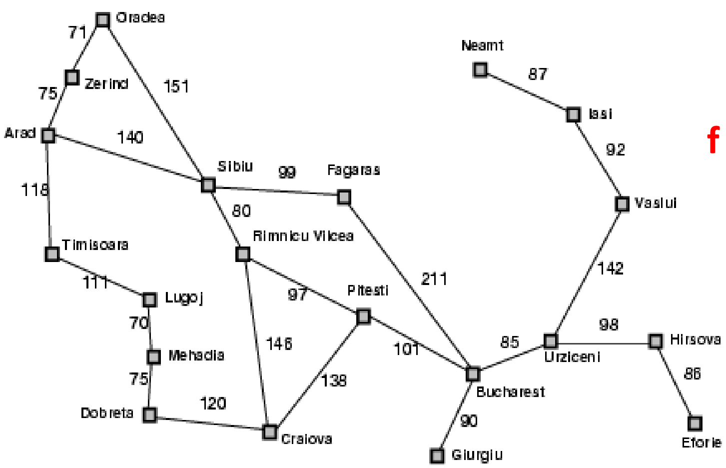


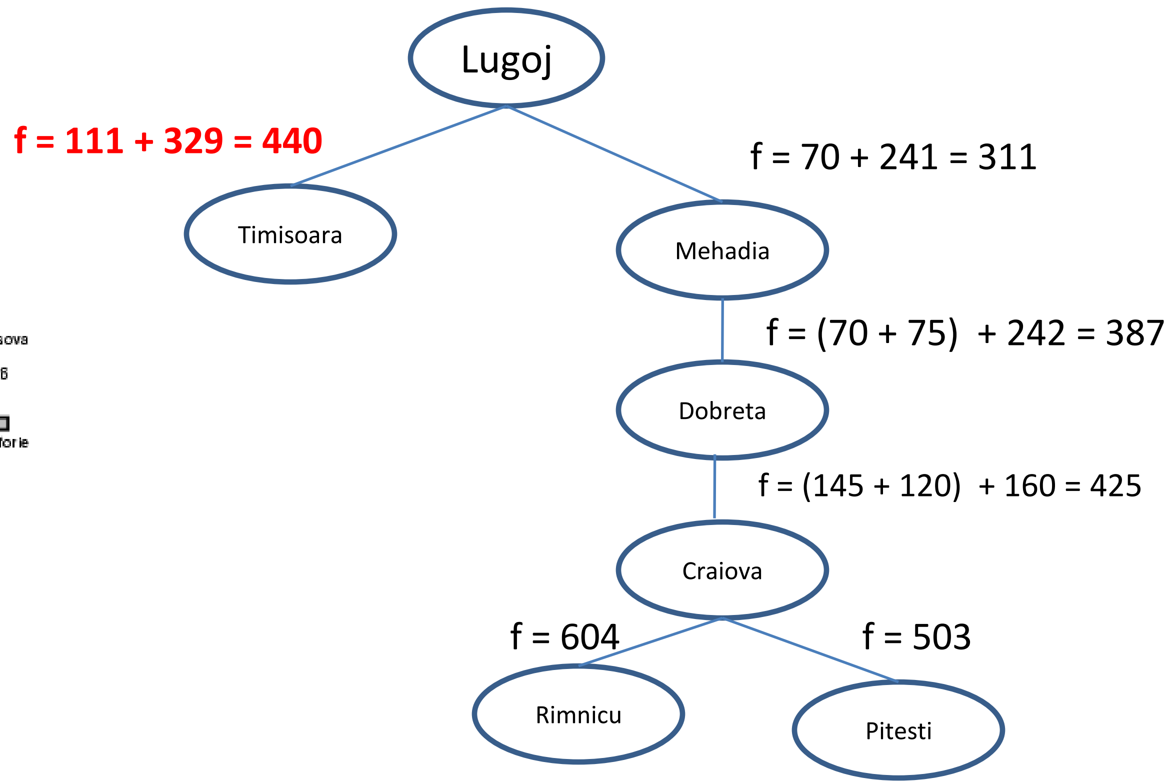


Straight–line distance to Bucharest

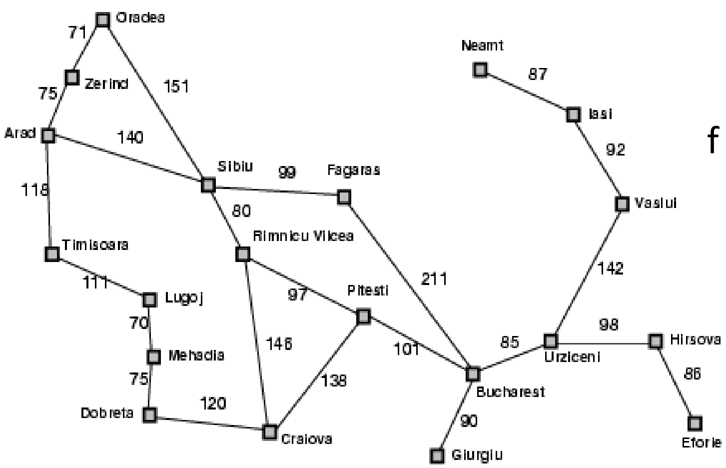
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

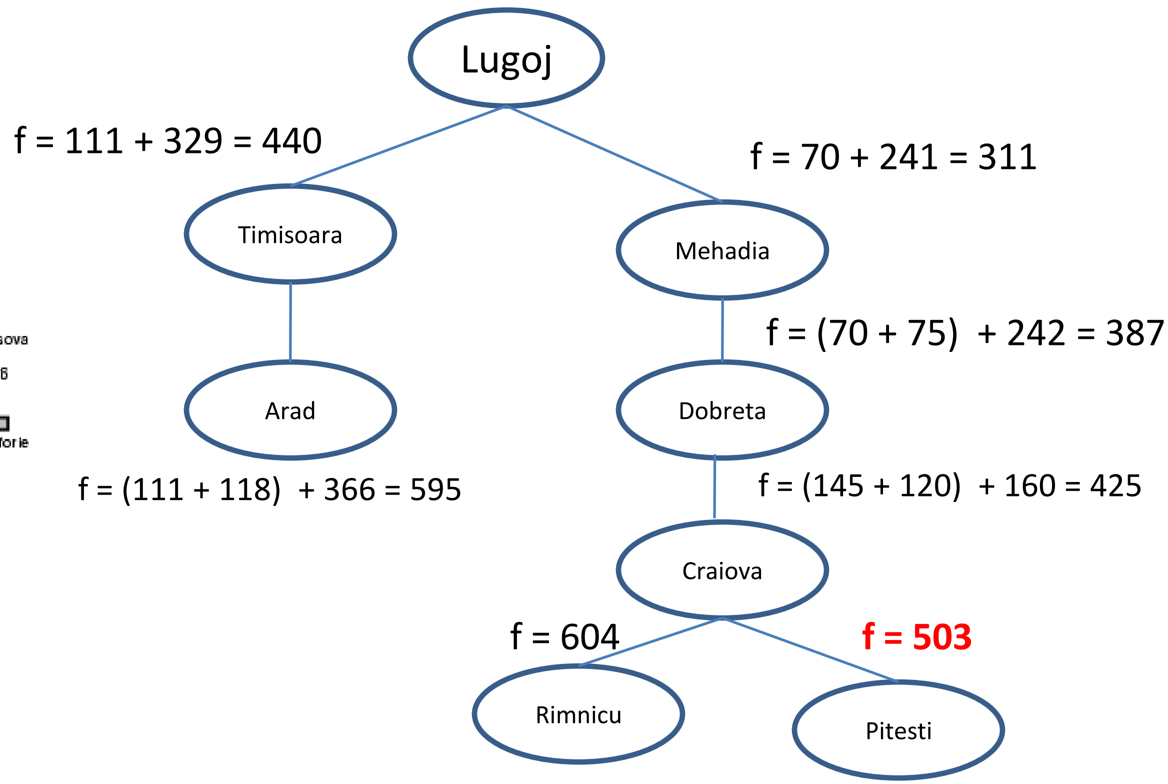
A* search: From Lugoj to Bucharest



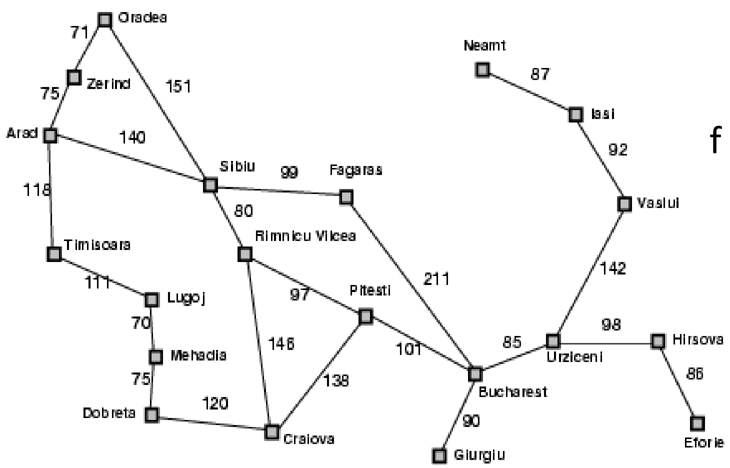


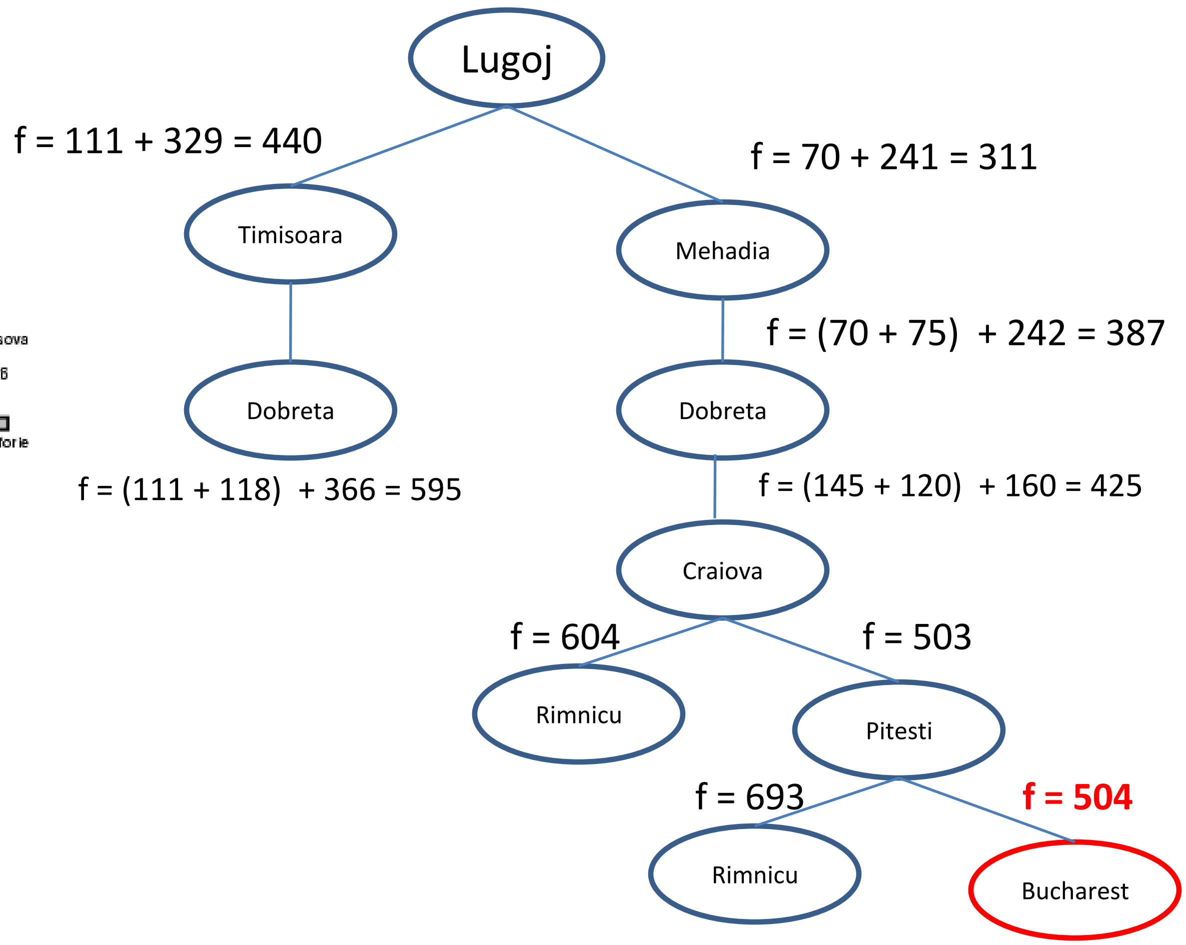
A* search: From Lugoj to Bucharest





A* search: From Lugoj to Bucharest





f = 70+75+120+138+101 = 504

4.2 Heuristic path algorithm

$f(n) = (2 - w)g(n) + wh(n)$

For what value of w is this algorithm guaranteed to be optimal?

g(n): a path cost to n from a start state

h(n): a heuristic estimate of cost from n to a goal state

4.2 Heuristic path algorithm

If h(n) is admissible, the algorithm is guaranteed to be optimal

$$f(n)=(2-w)[g(n)+\frac{w}{2-w}h(n)]$$

which behaves exactly like A* search with a heuristic

$$f(n)=g(n)+\frac{w}{2-w}h(n)$$

To be optimal, we require $\frac{w}{2-w}\leqslant 1$

$$w\leqslant 1$$

4.2 Heuristic path algorithm

For w = 0: $f(n) = 2g(n)$ -> Uniform-cost search

For w = 1: $f(n) = g(n) + h(n)$ -> A* search

For w = 2: $f(n) = 2h(n)$ -> Greedy best search

4.3

(a) Breadth-first search is a special case of uniform-cost search

When all step costs are equal (and let's assume equal to 1), g(n) is just a multiple of depth *n*. Thus, breadth-first search and uniform-cost search would behave the same in this case

$$f(n) = g(n) = 1*(\text{depth of } n)$$

4.3

(b) BFS, DFS and uniform-cost search are special cases of best-first search

- BFS: $f(n) = depth(n)$

- DFS: $f(n) = -depth(n)$

- UCS: $f(n) = g(n)$

4.3

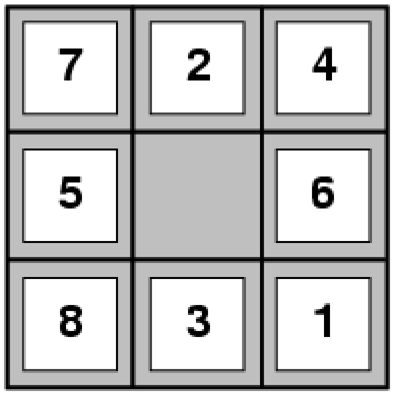
(c) Uniform-cost search is special case of A* search

A* search: $f(n) = g(n) + h(n)$

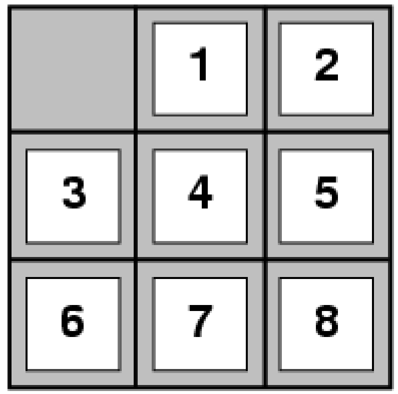
Uniform-cost search: $f(n) = g(n)$

Thus, for $h(n) = 0$, uniform cost search will produce the same result as A* search

4. Prove that the Manhattan Distance heuristic for 8-puzzle is admissible



Start State



Goal State

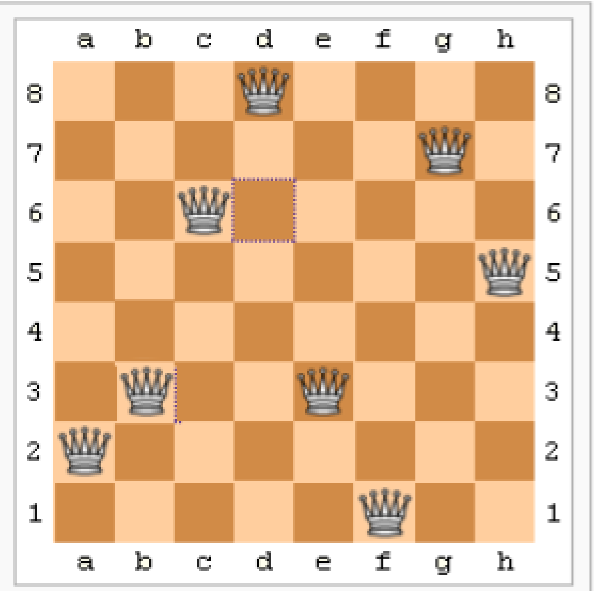
Manhattan Distance for points $P_1(x_1,y_1)$, $P_2(x_2,y_2)$ is defined by:

$$d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$$

Heuristic: $$h = \sum_{n=1}^{8} d(n)$$

- Tiles cannot move along diagonals, so each tile has to move at least *d(n)* steps to its goal
- Any move can only move one tile at a time

5. Eight Queens problem

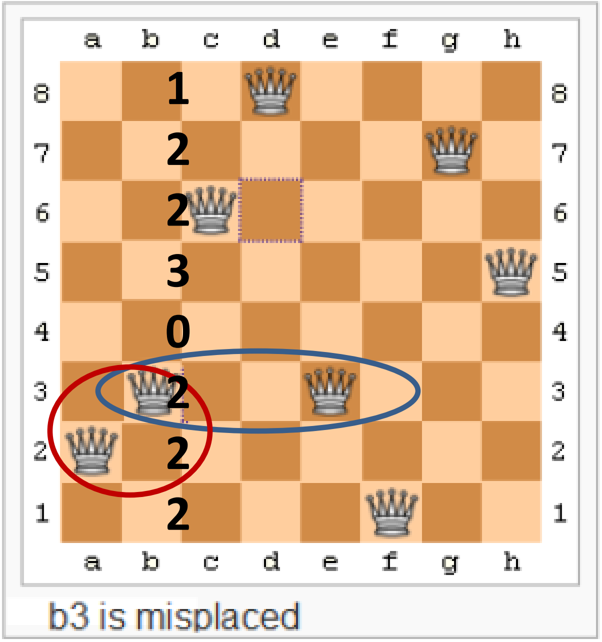


h – number of pairs of queens that are attacking each other

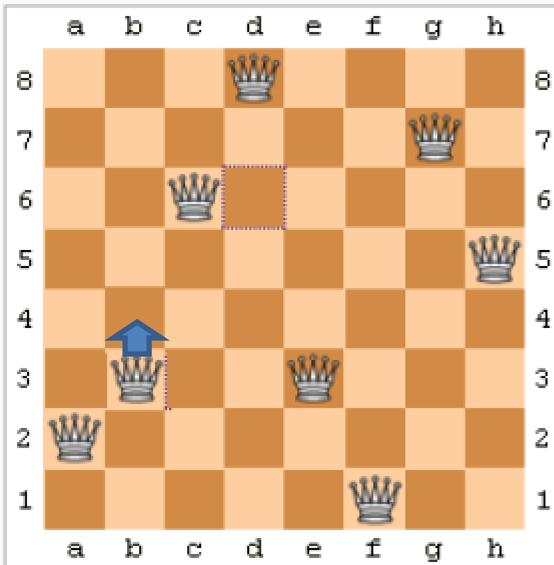
5. Eight Queens problem

h = 2

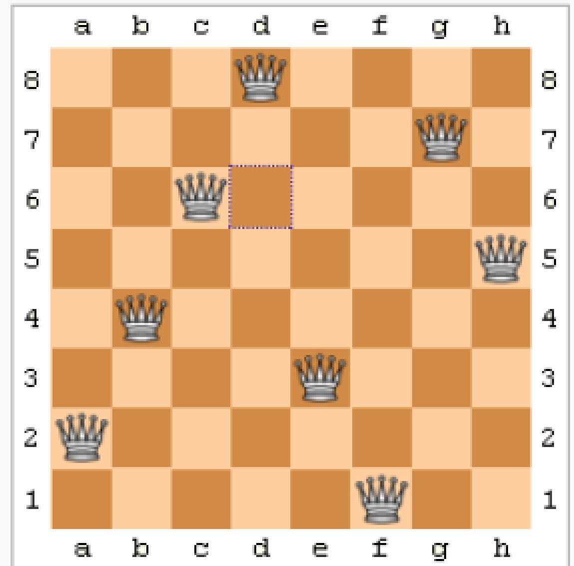
However, number of conflicts for Queen at “b3” are:



5. Eight Queens problem



b3 is misplaced

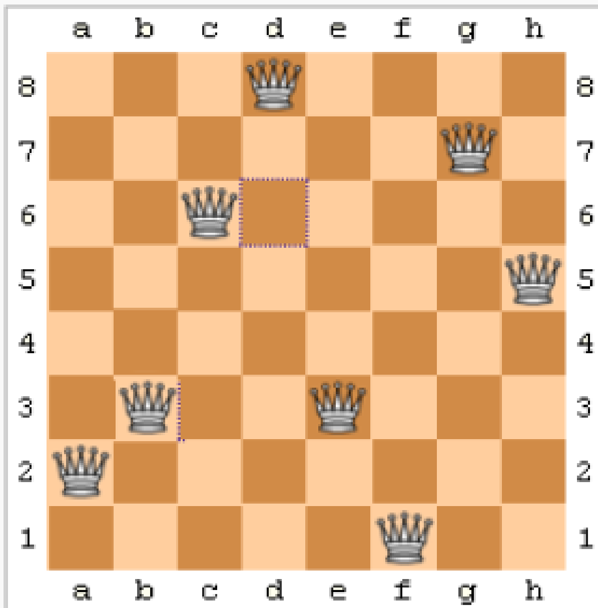


One solution.

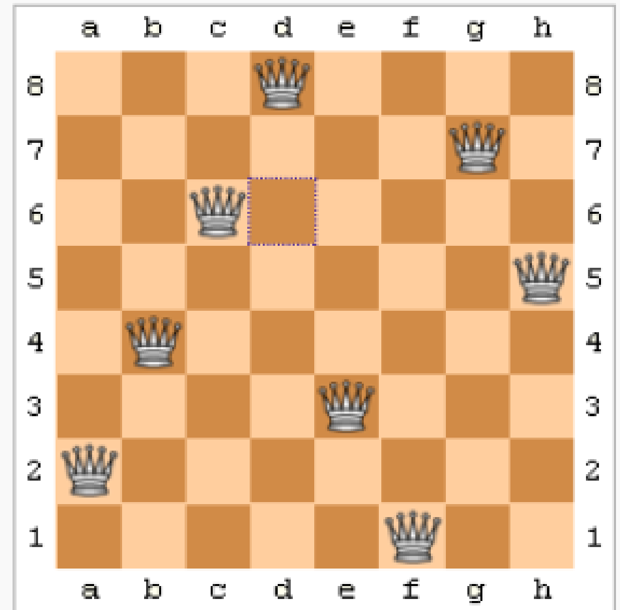
Therefore, the true cost to reach the goal state h* is 1

Thus, h>h* → heuristic is not admissible

5. Eight Queens problem



b3 is misplaced



One solution.

We can propose another heuristic.
For example, we can propose heuristic derived form a relaxed (and trivial) version of 8-Queens problem, that the eight queens must be placed in the board so that no two queens are on the same row.

Thus, h = Σ (# queens that are on the same row – 1) for all conflicting rows

4.11

(a) Local beam search with k=1

- We would randomly generate 1 start state
- At each step we would generate all the successors, and retain the 1 best state

- Equivalent to HILL-CLIMBING

4.11

(b) Local beam search with $k=\infty$

- 1 initial state and no limit of the number of states retained
- We start at initial state and generate all successor states (no limit how many)
- If one of those is a goal, we stop
- Otherwise, we generate all successors of those states (2 steps from the initial state), and continue

- Equivalent to BREADTH-FIRST SEARCH

4.11

(c) Simulated annealing with T = 0 at all times

- If T is very small, the probability of accepting an arbitrary neighbor with lower value is approximately 0
- This means that we choose a successor state randomly and move to that state if it is better than the current state

- Equivalent to FIRST-CHOICE HILL CLIMBING

4.11

(d) Genetic algorithm with population size N = 1

- If selection step necessarily chooses the single population member twice, so the crossover steo does nothing.
- Moreover, if we think of the mutation step as selecting a successor at random, there is no guarantee that the successor is an improvement over the parent

- Equivalent to RANDOM WALK

4-Queens problem

Min-conflict algorithm:

1. Randomly choose a variable from set of problematic variables
2. Reassign its value to the one that results in the fewest conflicts overall
3. Continue until there are no conflicts

Q1			
	Q2		
		Q3	
			Q4

4-Queens problem

	A	B	C	D
4	3 Q1	1	2	1
3	1	3 Q2	1	2
2	2	1	3 Q3	1
1	1	2	1	3 Q4

Number of conflicts

All queens are attacked.

Pick Q2 randomly

We can move Q2 to B2 or B4

Randomly, move Q2 to B4

4-Queens problem

	A	B	C	D
4	3 Q1	1 Q2	2	2
3	1	3	1	1
2	1	1	2 Q3	2
1	1	2	1	2 Q4

All queens are attacked.

Pick Q1 randomly

We can move Q1 to A1 ~ A3

Randomly, move Q1 to A2

4-Queens problem

	A	B	C	D
4	3	0 Q2	2	1
3	1	3	1	1
2	1 Q1	2	2 Q3	3
1	1	3	1	1 Q4

Q1, Q3 and Q4 are attacked.

Pick Q3 randomly

We can move Q3 to C1 or C3

Randomly, we select C1

4-Queens problem

	A	B	C	D
4	2	0 Q2	2	1
3	2	2	1	0
2	0 Q1	2	2	3
1	2	3	1 Q3	1 Q4

Q3 and Q4 are attacked.

Pick Q4 randomly

We can move Q4 D3

4-Queens problem

	A	B	C	D
4	2	0 Q2	2	1
3	2	2	1	0 Q4
2	0 Q1	2	2	3
1	2	3	1 Q3	1

No conflicts!

8. Compute the following gradients

$$f(x, y, z, t) = (x-1)(2-y)z + (t^3 - 1)xyz$$

$$g(x, y) = \frac{1}{1 + \exp(-(ax + by + c))}$$

$$h(x, y, z) = (x-1)^2 \exp(x) + (y-2)^3 z^3$$

$$c(x, y, z) = (x - z - 2y^{-2})^b$$

$$g(x, y) = 2(x-1)^2 + 2(y-2)^2 - 2(x-1)(y-2)$$

a, b, c are some arbitrary constants

8. Compute the following gradients

$$f(x, y, z, t) = (x-1)(2-y)z + (t^3 - 1)xyz$$

$$g(x, y) = \frac{1}{1 + \exp(-(ax + by + c))}$$

$$h(x, y, z) = (x-1)^2 \exp(x) + (y-2)^3 z^3$$

$$c(x, y, z) = (x - z - 2y^{-2})^b$$

$$g(x, y) = 2(x-1)^2 + 2(y-2)^2 - 2(x-1)(y-2)$$

$$\nabla f = (\frac{\partial f}{\partial X_1}, \frac{\partial f}{\partial X_2}, \frac{\partial f}{\partial X_3}, \dots, \frac{\partial f}{\partial X_n})$$

8. Compute the following gradients

$$f(x, y, z, t) = (x-1)(2-y)z + (t^3-1)xyz$$
$$\nabla f = ((2-y)z + (t^3-1)yz, -(x-1)z + (t^3-1)xz, (x-1)(2-y) + (t^3-1)xy, 3t^2 xyz)$$

$$g(x, y) = \frac{1}{1+\exp(-(ax+by+c))}$$
$$\nabla g = (\frac{a\exp(-(ax+by+c))}{(1+\exp(-(ax+by+c)))^2}, \frac{b\exp(-(ax+by+c))}{(1+\exp(-(ax+by+c)))^2})$$

$$h(x, y, z) = (x-1)^2\exp(x) + (y-2)^3 z^3$$
$$\nabla h = ((x^2-1)\exp(x), 3(y-2)^2 z^3, 3(y-2)^3 z^2)$$

$$c(x, y, z) = (x-z-2\mathrm{y}^{-2})^b$$
$$\nabla c = (b(x-z-2\mathrm{y}^{-2})^{b-1}, 4\mathrm{b}(x-z-2\mathrm{y}^{-2})^{b-1} y^{-3}, -b(x-z-2\mathrm{y}^{-2})^{b-1})$$

$$g(x, y) = 2(x-1)^2 + 2(y-2)^2 - 2(x-1)(y-2)$$
$$\nabla g = (4\mathrm{x}-2\mathrm{y}, -2\mathrm{x}+4\mathrm{y}-6)$$

Pseudo code for gradient descent algorithm that minimize g(x, y)

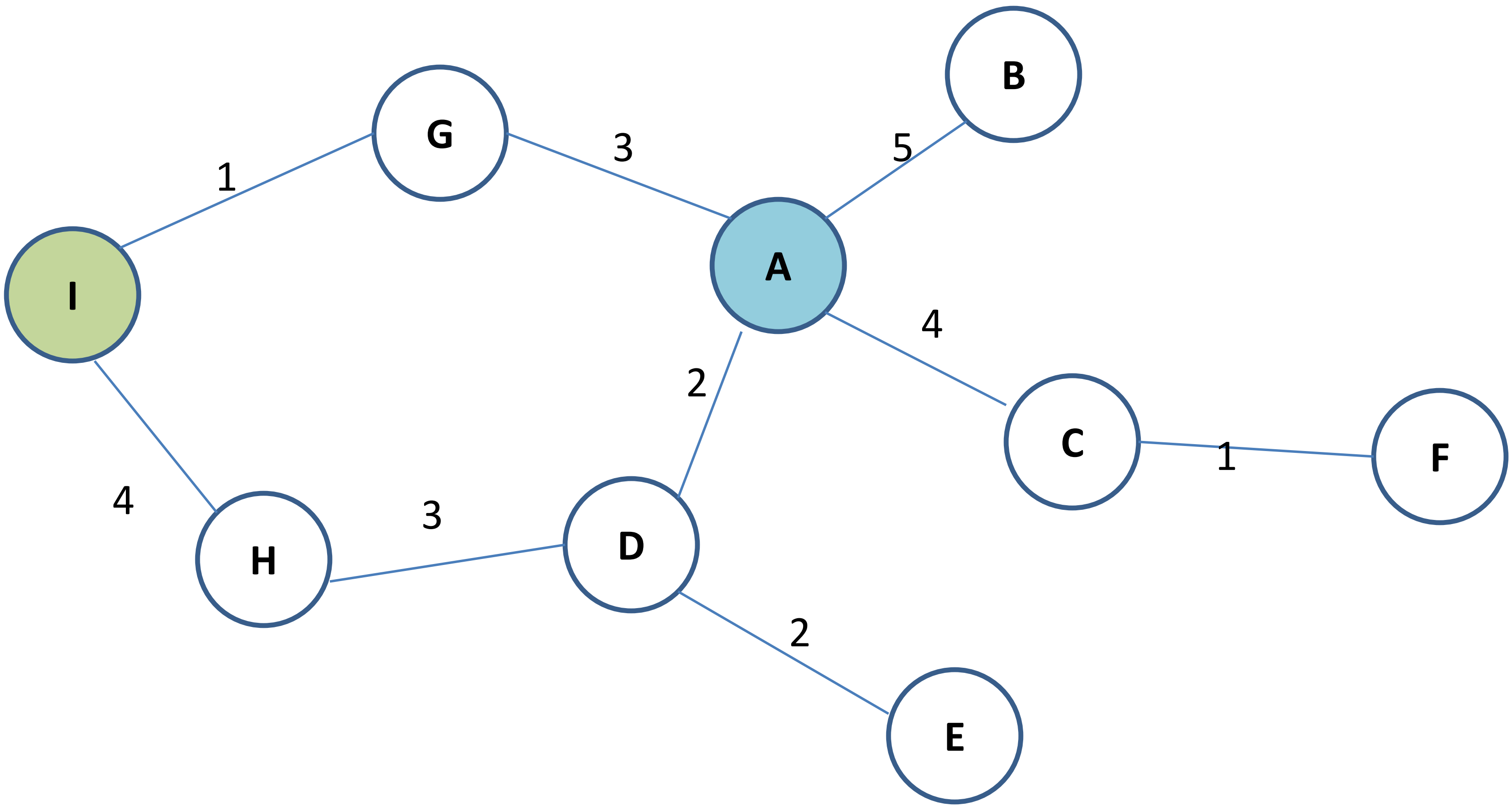
```
pCur = (0,0)                    # current point
pNxt = (5,5)                    # next point
eps = 10e-2                     # step size

precision = 10e-5;

while (|pCur - pNxt|) > precision):
        pCur = pNxt;
        pNxt = pNxt – eps *  ∇g(pNxt) ;

print "Local minimum occurs at ", pCur
```

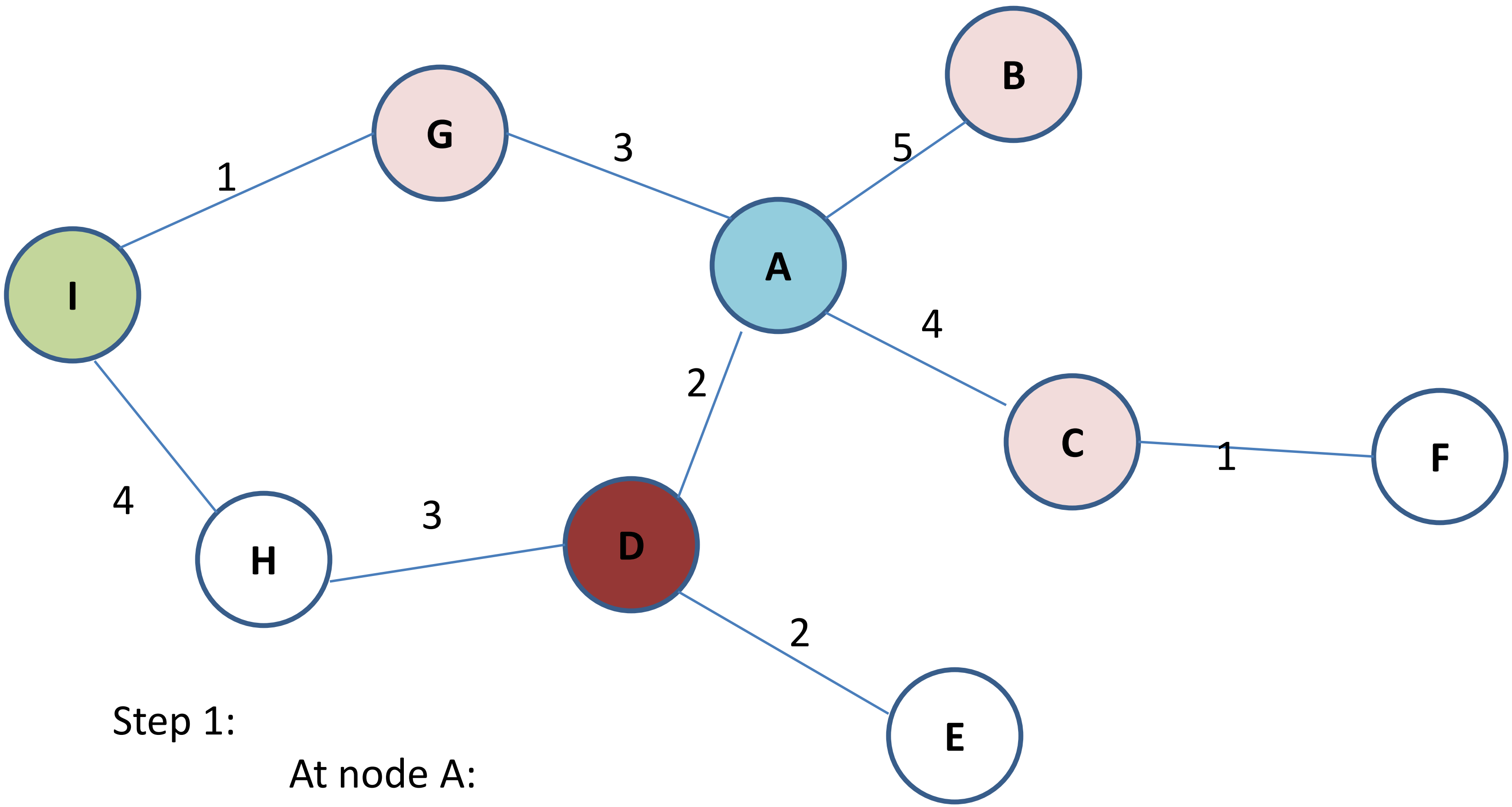
Uniform Cost Search



Goal: path AI

Queue: A (root)

Uniform Cost Search



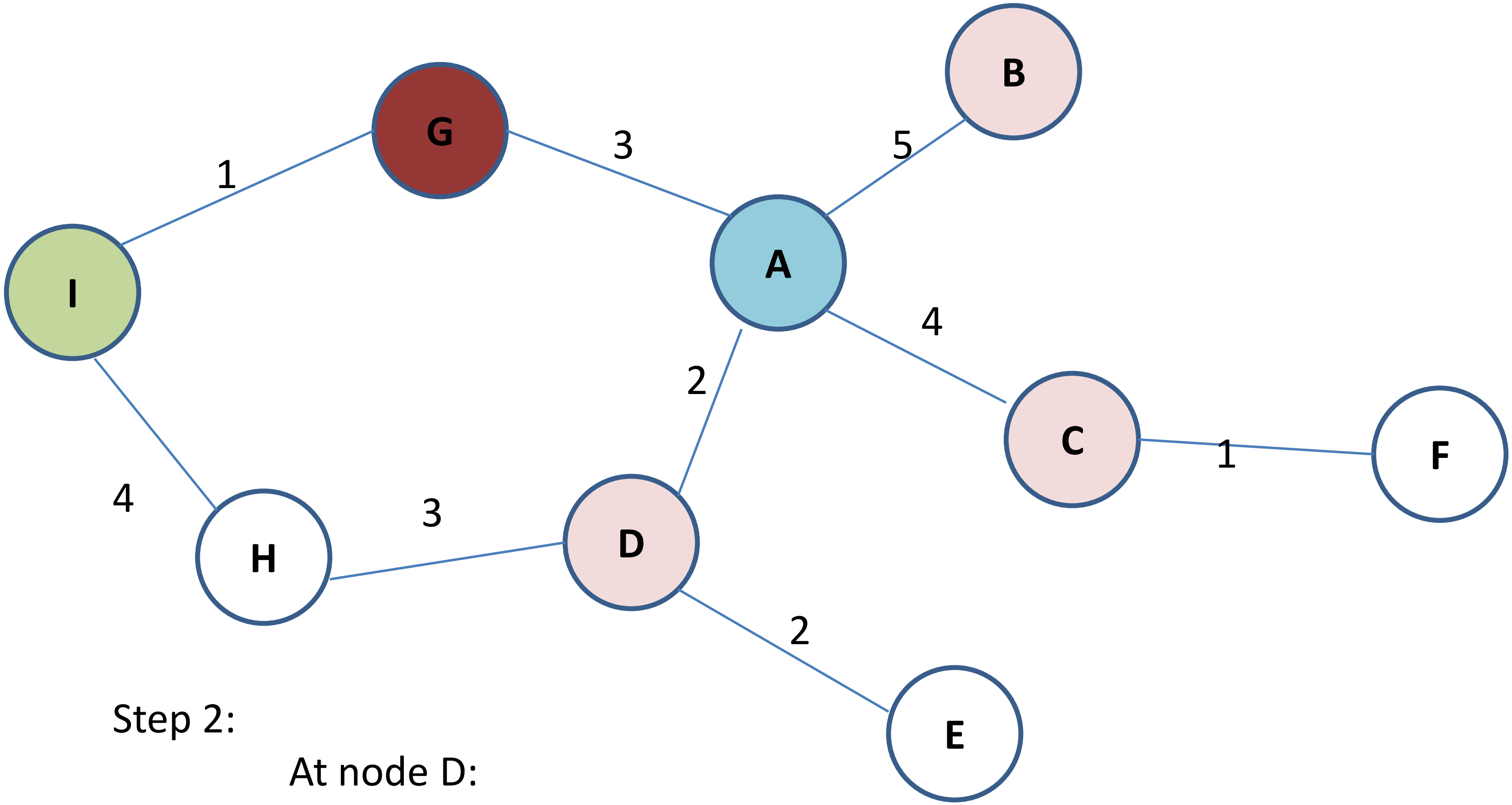
Step 1:

At node A:

Queue: D=2, G=3, C=4, B=5

Note: nodes in the queue are sorted by distance from the root

Uniform Cost Search



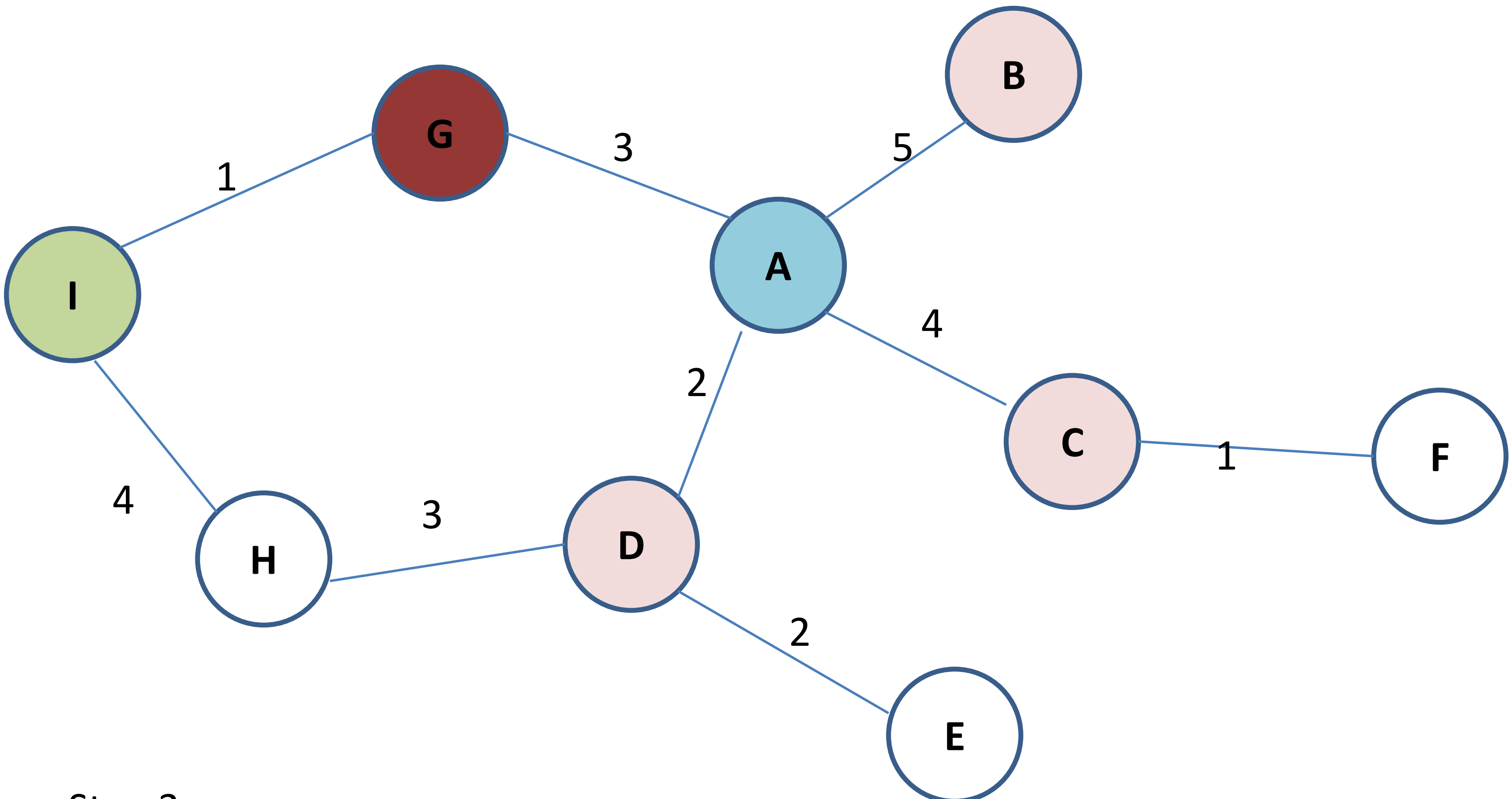
Step 2:

At node D:

Queue: G=3, C=4, E=4, B=5, H=5

Note: nodes in the queue are sorted by distance from the root

Uniform Cost Search



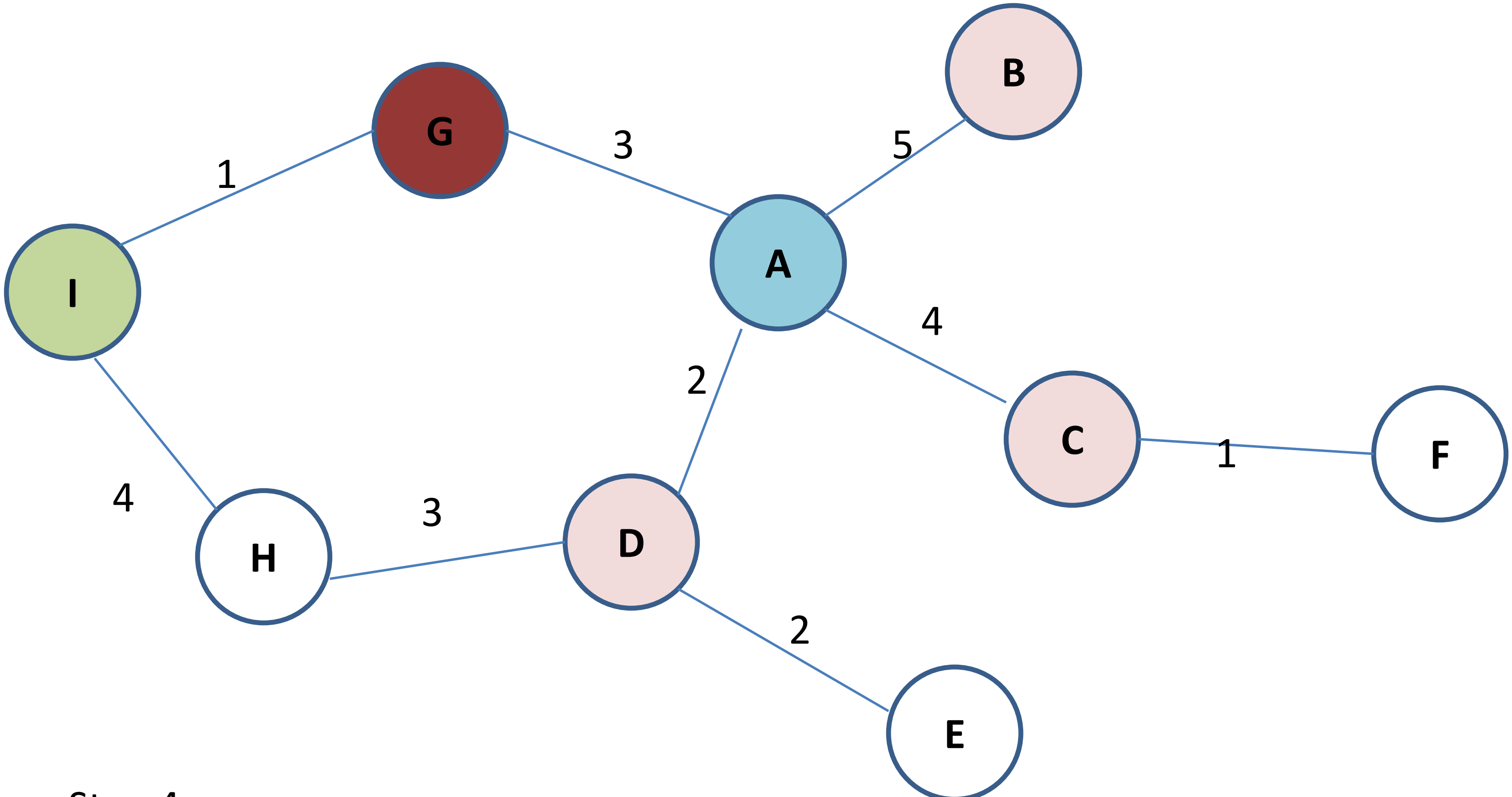
Step 3:

At node G:

Queue: I=4, C=4, E=4, B=5, H=5

Note: nodes in the queue are sorted by distance from the root

Uniform Cost Search

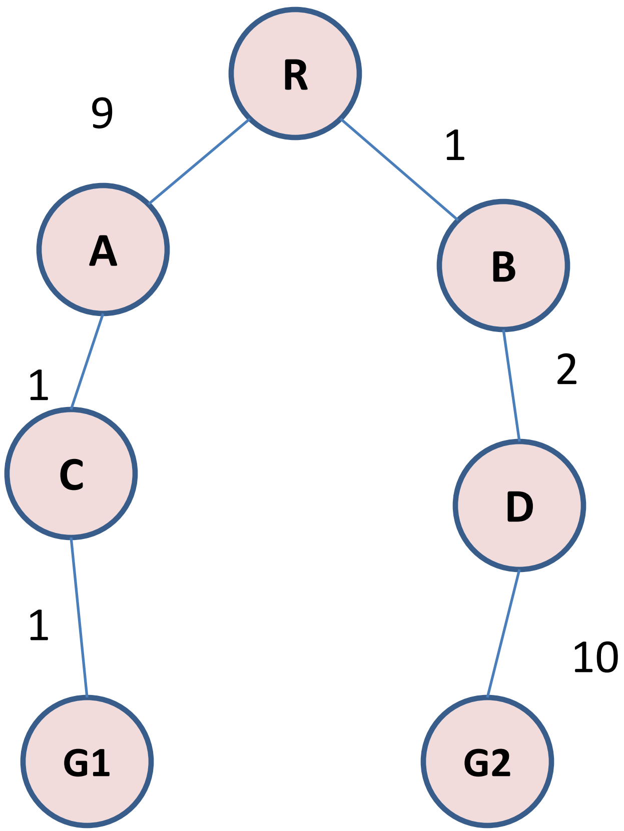


Step 4:

At node I:

GOAL NODE FOUND!!!!!

A* Search



f(n) = g(n) + h(n)

Where:

f(n) – estimated total cost of path through n to goal

g(n) – cost so far to reach n

h(n) – estimated cost from n to goal

Heuristic Estimates:

h(B -> G2) = 9

h(D -> G2) = 10

h(A -> G1) = 2

h(C -> G1) = 1

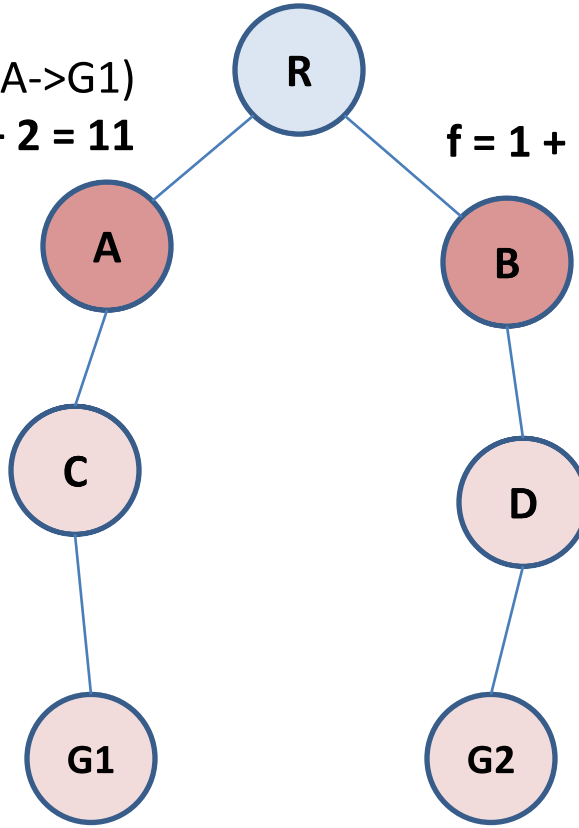
A* Search

f(n) = g(n) + h(n)

h(B -> G2) = 9
h(D -> G2) = 10
h(A -> G1) = 2
h(C -> G1) = 1

f(A) = g(A) + h(A->G1)
f = 9 + 2 = 11

f = 1 + 9 = 10



f (A) = 11 > f(B) = 10

A* Search

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

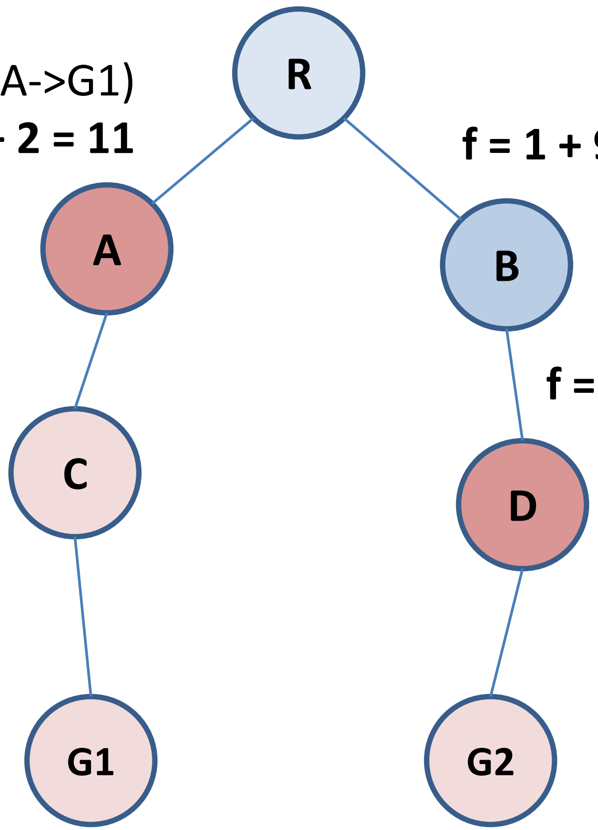
$h(B \rightarrow G2) = 9$
$h(D \rightarrow G2) = 10$
$h(A \rightarrow G1) = 2$
$h(C \rightarrow G1) = 1$

f(A) = g(A) + h(A->G1)
f = 9 + 2 = 11

f = 1 + 9 = 10

f = (1 + 2) + 10 = 13

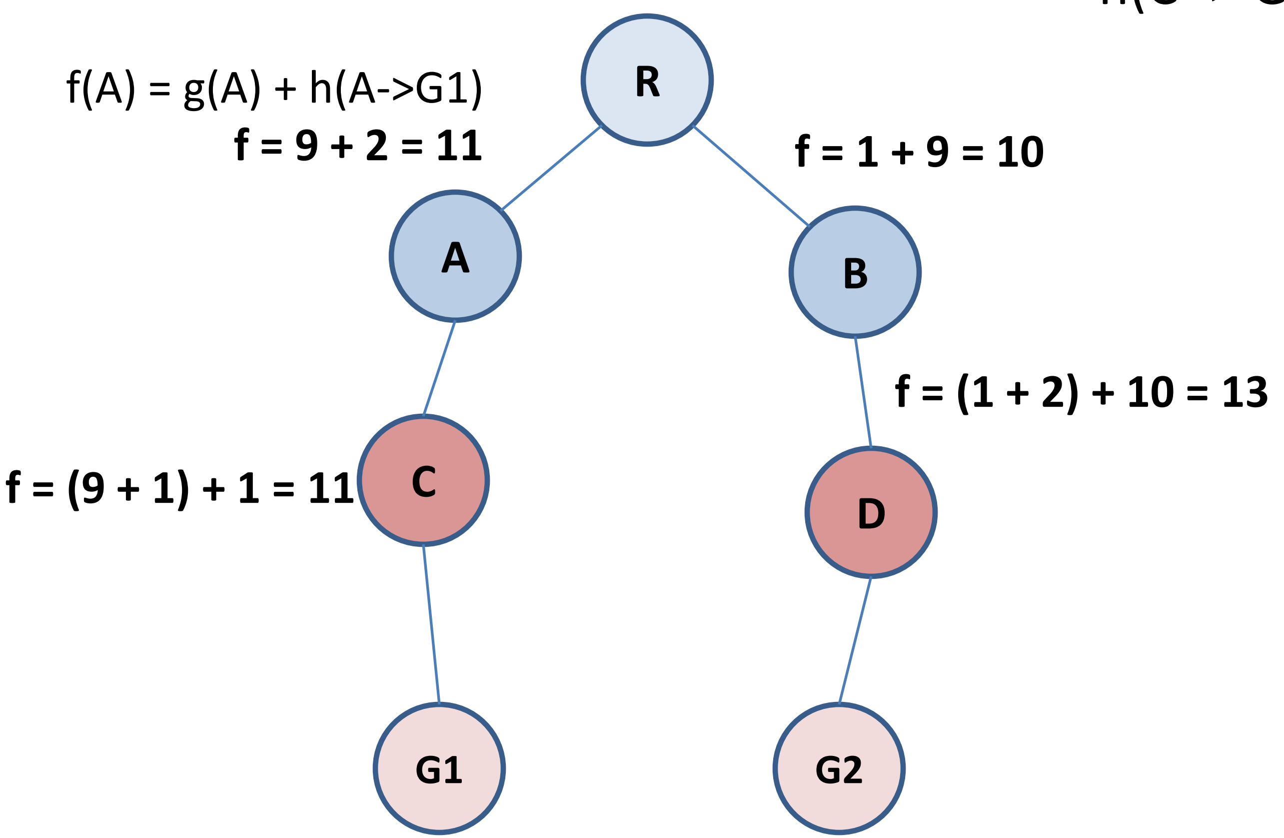
f (A) = 11 < f(D) = 13



A* Search

f(n) = g(n) + h(n)

h(B -> G2) = 9
h(D -> G2) = 10
h(A -> G1) = 2
h(C -> G1) = 1



f (C) = 11 < f(D) = 13

A* Search

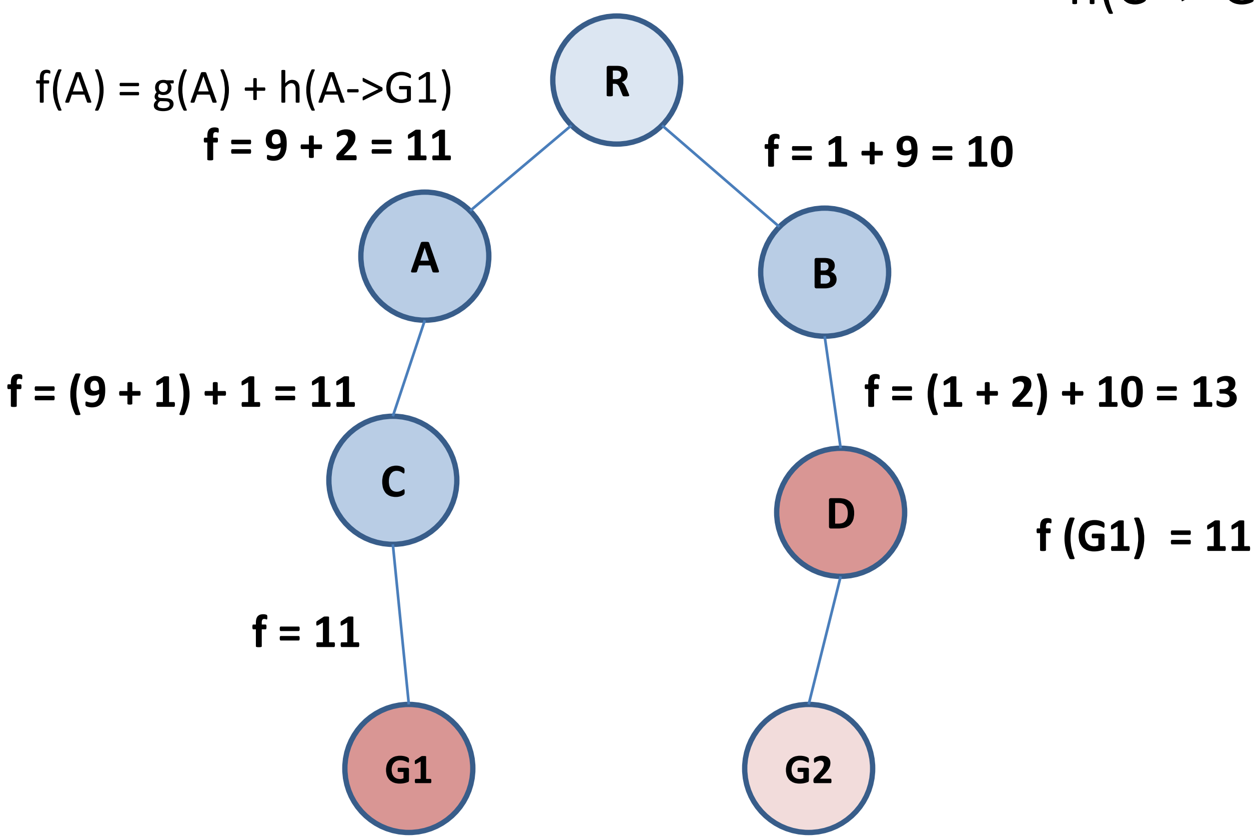
f(n) = g(n) + h(n)

h(B -> G2) = 9

h(D -> G2) = 10

h(A -> G1) = 2

h(C -> G1) = 1



f (G1) = 11 < f(D) = 13

A* Search

Order:
R – B – A – C – G1

There is no need to check the unfinished path (cost of G2), because it already costs more than the current path does.