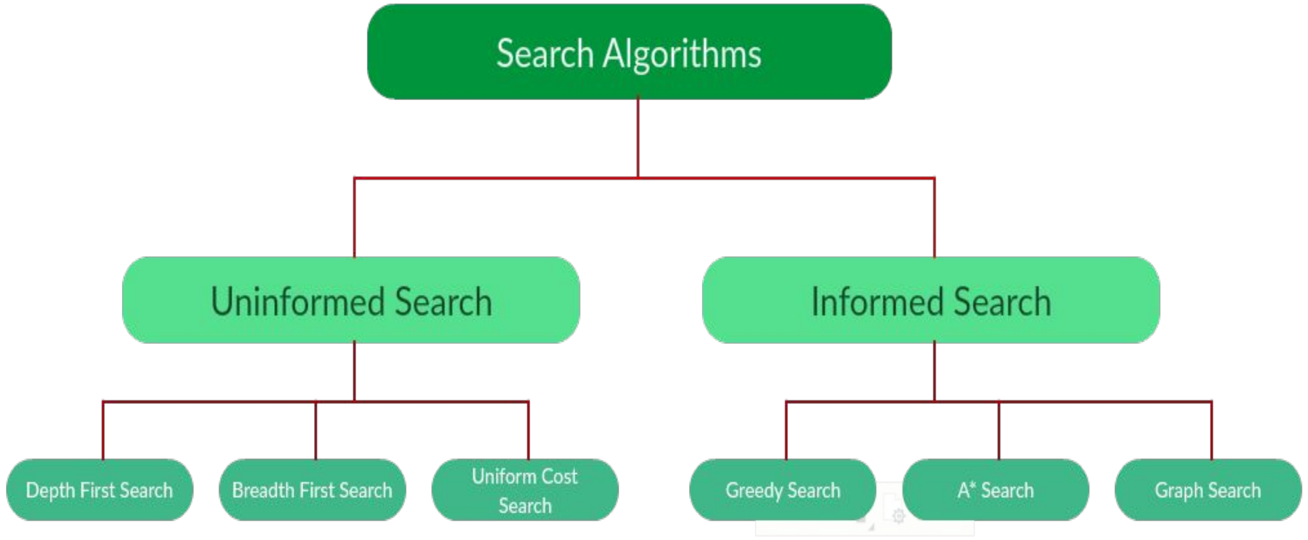
Artificial Intelligence



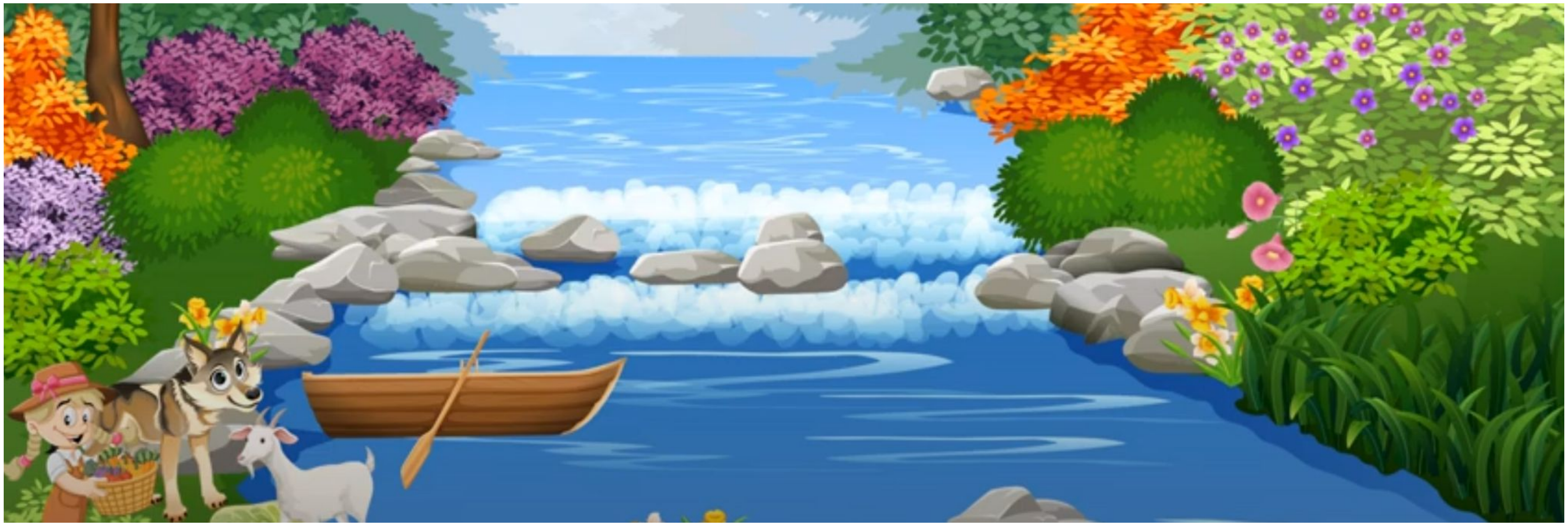
Lecture: Informed Search

Farmer, Goat, Wolf and Cabbage

There is a farmer who wishes to cross a river but he is not alone. He also has a goat, a wolf, and a cabbage along with him. There is only one boat available which can support the farmer and either of the goat, wolf or the cabbage. So at a time, the boat can have only two objects (farmer and one other).

But the problem is, if the goat and wolf are left alone (either in the boat or onshore), the wolf will eat the goat. Similarly, if the Goat and cabbage are left alone, then goat will eat the cabbage. The farmer wants to cross the river with all three of his belongings: goat, wolf, and cabbage.

What strategy he should use to do so?



Search Strategies

- Search Strategies are classified into
    - **uninformed search** (or) **Blind search**
    - **informed search** (or) **Heuristic search**

Uninformed Vs Informed

- **Uninformed search methods** that systematically explore the state space and find the goal. They are inefficient in most cases.

- **Informed search methods** use problem specific knowledge, and may be more efficient. At the heart of such algorithms there is the concept of a heuristic function.

Heuristics

- Heuristic means "**rule of thumb**".

- To quote J. Pearl, "*Heuristics are criteria, methods or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal*".

- In heuristic search or informed search, heuristics are used to identify the **<u>most promising search path</u>**.

- **Informed (or heuristic) search** uses problem-specific heuristics to improve efficiency
    - **Best-first**
    - **A***
    - **RBFS**
    - **SMA***
    - **Techniques for generating heuristics**

Example of Heuristic Function

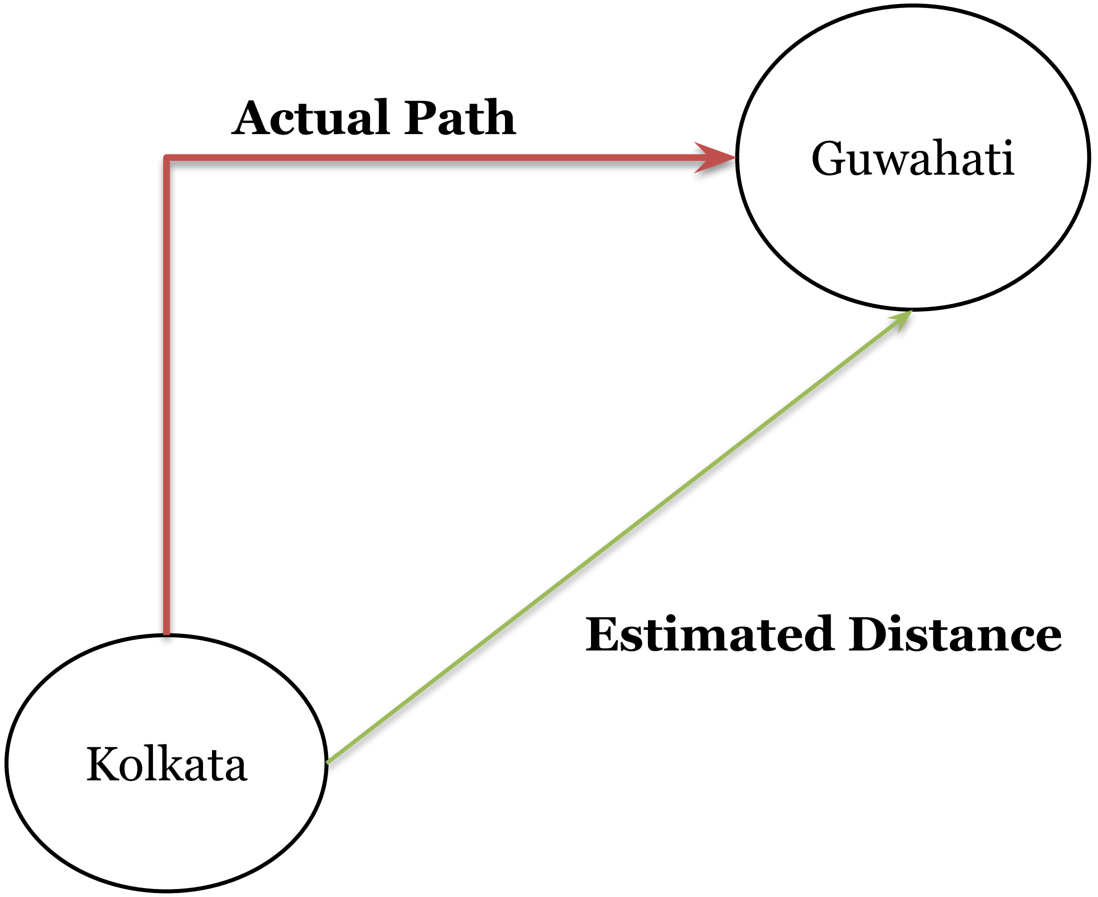
- A heuristic function at a **node n** is an estimate of the optimum cost from the current node to **a goal**. It is denoted by *h(n)*.

  h(n) **= estimated cost of the cheapest path from node n to a goal node**

- Example 1: We want a path from Kolkata to Guwahati

h(n)

- Metric on states. Estimate of shortest distance to some goal.
- h : state → estimate of distance to goal
- h (goal) = 0 for all goal nodes



Heuristic for Guwahati may be straight-line distance between Kolkata and Guwahati

h(Kolkata) = euclideanDistance(Kolkata, Guwahati)

Example 2: 8-puzzle: Misplaced Tiles Heuristics is the number of tiles out of place.

2	8	3
1	6	4
	7	5

Initial State

1	2	3
8		4
7	6	5

Goal state

2	8	3
1	6	4
	7	5

Initial State

1	2	3
8		4
7	6	5

Goal state

Tiles: 2, 8, 1, 6 and 7 => Not in Correct location

The first picture shows the current state *n*, and the second picture the goal state.
h(n) = 5

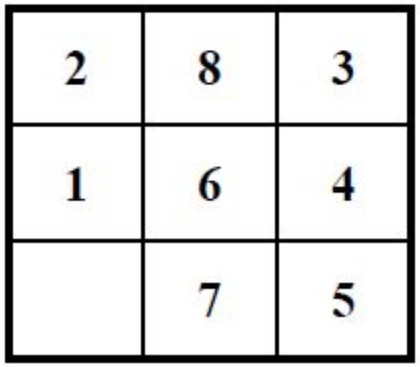
5 tiles are not their correct location

We must make at least 5 moves to move them to their correct location

So, h(n) is ***the underestimate of actual number of step required to move to their goal state***

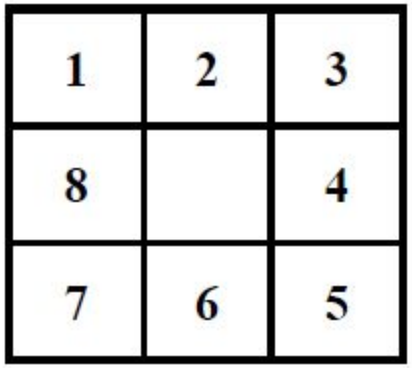
Heuristic Example

8-Puzzle: Manhattan Distance (distance tile is out of place)



2	8	3
1	6	4
	7	5

Initial State



1	2	3
8		4
7	6	5

Goal state

This heuristic sums the distance that the tiles are out of place. The distance of a tile is measured by the sum of the differences in the x-positions and the y-positions.

- For the above example, using the Manhattan distance heuristic,

$$h(n) = 1 + 1 + 0 + 0 + 0 + 1 + 1 + 2 = 6$$

8 puzzle problem using heuristic

1	2	3
	4	6
7	**5**	**8**

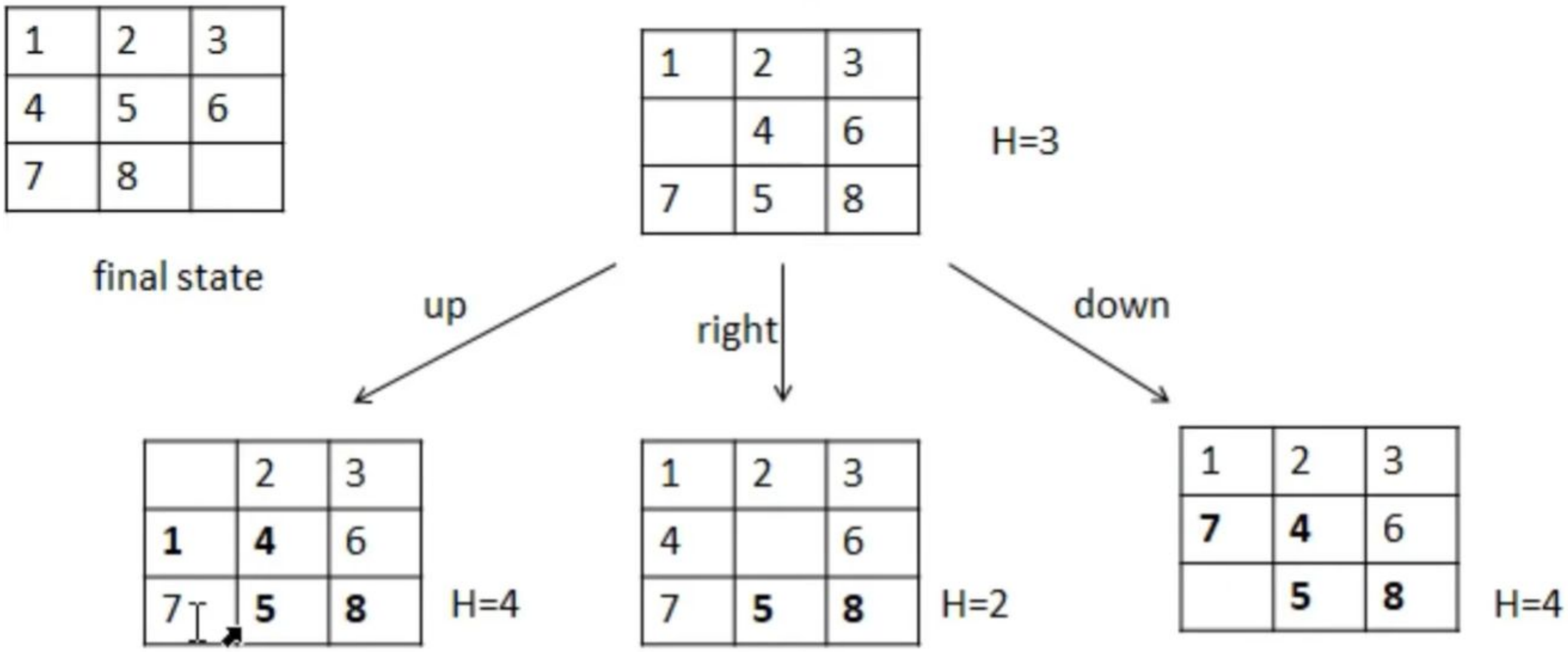
Initial state

1	2	3
4	5	6
7	8	

final state

Heuristic value=1+1+1=3

Set of action={up, down, left, right}



final state

1	2	3
4	5	6
7	8	

H=3

1	2	3
	4	6
7	5	8

up, H=4

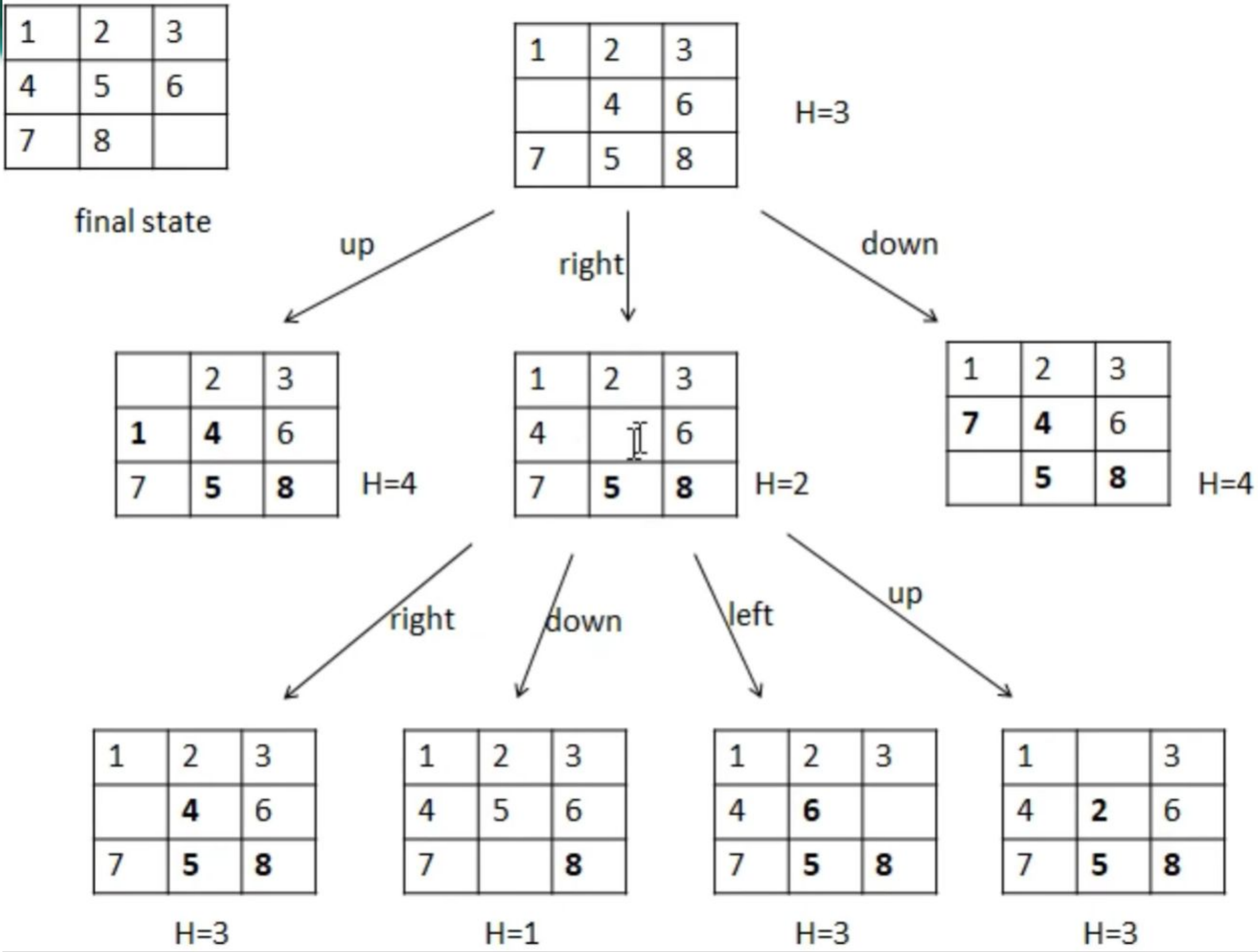
	2	3
1	**4**	6
7	**5**	**8**

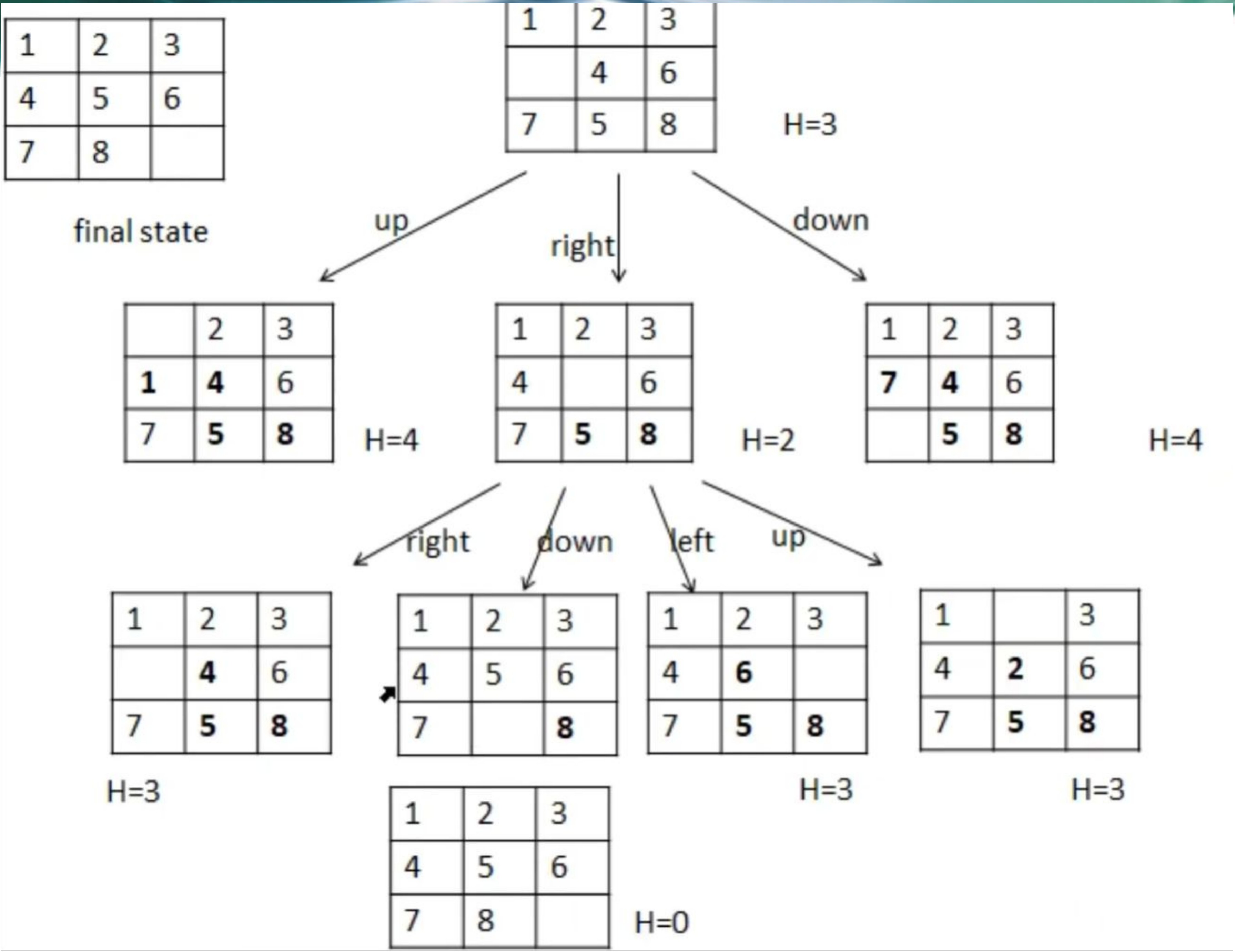
right, H=2

1	2	3
4		6
7	**5**	**8**

down, H=4

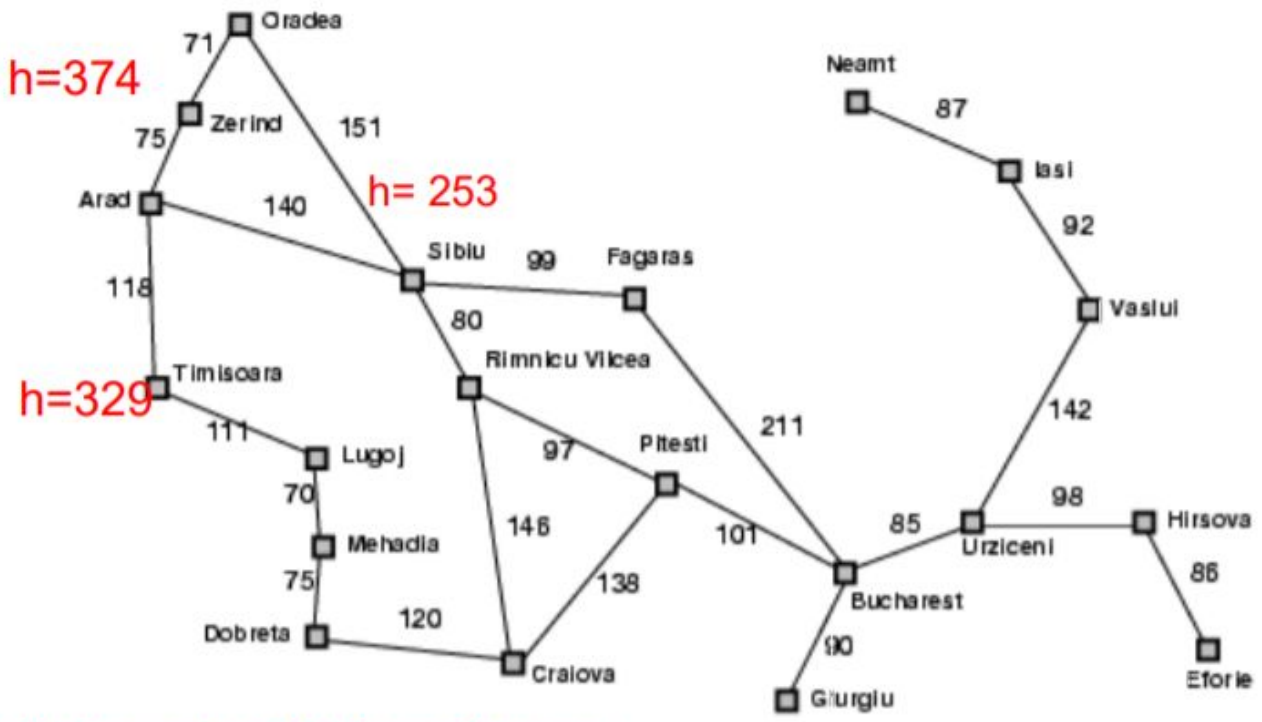
1	2	3
7	**4**	6
	5	**8**


1 2 3 / 4 5 6 / 7 8
final state
1 2 3 / 4 6 / 7 5 8
H=3
up
right
down
2 3 / 1 4 6 / 7 5 8
H=4
1 2 3 / 4 6 / 7 5 8
H=2
1 2 3 / 7 4 6 / 5 8
H=4
right
down
left
up
1 2 3 / 4 6 / 7 5 8
H=3
1 2 3 / 4 5 6 / 7 8
H=1
1 2 3 / 4 6 / 7 5 8
H=3
1 3 / 4 2 6 / 7 5 8
H=3


final state
H=3
up
right
down
H=4
H=2
H=4
right
down
left
up
H=3
H=3
H=3
H=0

Heuristic Search

- State-Space Search: every problem is like search of a map
- A problem solving agent finds a path in a state-space graph from start state to goal state, using heuristics



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

Heuristic = straight-line distance

Searching Strategies

•**Blind search** → traversing the search space until the goal nodes is found (might be doing exhaustive search).

•*Techniques* : **Breadth First Uniform Cost ,Depth first, Interactive Deepening search**.

•**Heuristic search** → search process takes place by traversing search space with applied rules (information).

•*Techniques*: **Greedy Best First Search, A* Algorithm**

Best-First Search

- Idea: use an evaluation function *f(n)* for each node
  - f(n) provides an estimate for the total cost.
  - Expand the node n with smallest f(n).

- Implementation:
  Order the nodes in fringe increasing order of cost.

- Special cases:
  - greedy best-first search
  - A^* search

Best-first Search

- Open list of nodes reached but not yet expanded
- Closed list of nodes that have been expanded
- Choose lowest cost node on Open list
- Add it to Closed, add its successors to Open
- Stop when Goal is **first** removed from Open

Dijkstra: cost, $f(N) = g(N)$ = distance from start

A*: cost, $f(N) = g(N) + h(N)$

Greedy Search

- **Idea**: Expand the node with the **smallest estimated cost** to reach the goal.
- We use a heuristic function

  $f(n) = h(n)$

  $h(n)$ estimates the distance remaining to a goal.
- Greedy algorithms often perform very well.
- **Disadvantage**:
  - They tend to find good solutions quickly, although not always optimal ones.
  - The algorithm is also incomplete, and it may fail to find a solution even if one exists.

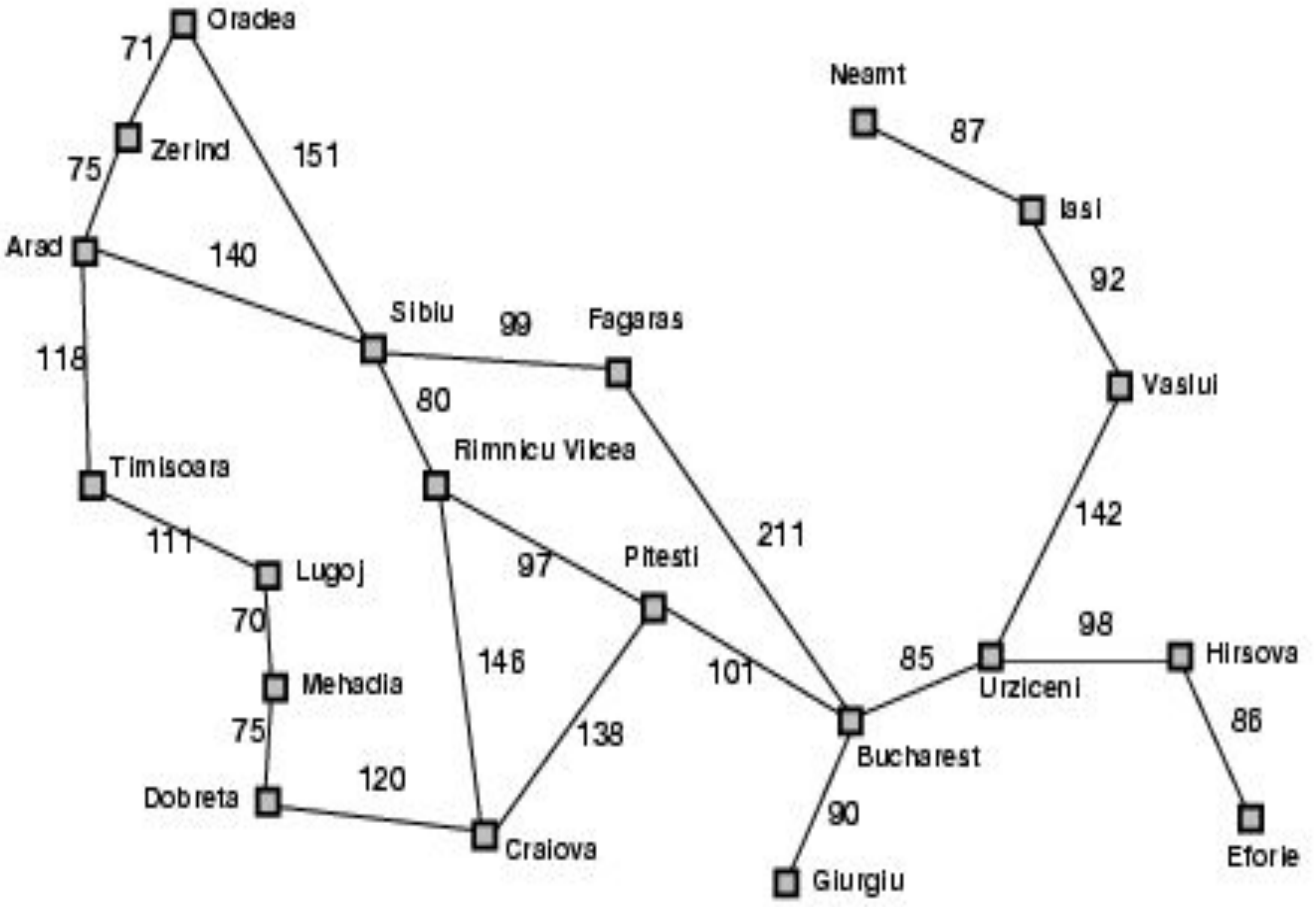
•In general:

–$h(n) \geq 0$ for all nodes n

–$h(n) = 0$ implies that n is a goal node

–$h(n) = \infty$ implies that n is a dead-end that can never lead to a goal

Romania with straight-line dist.



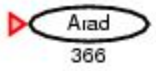
Straight–line distance to Bucharest

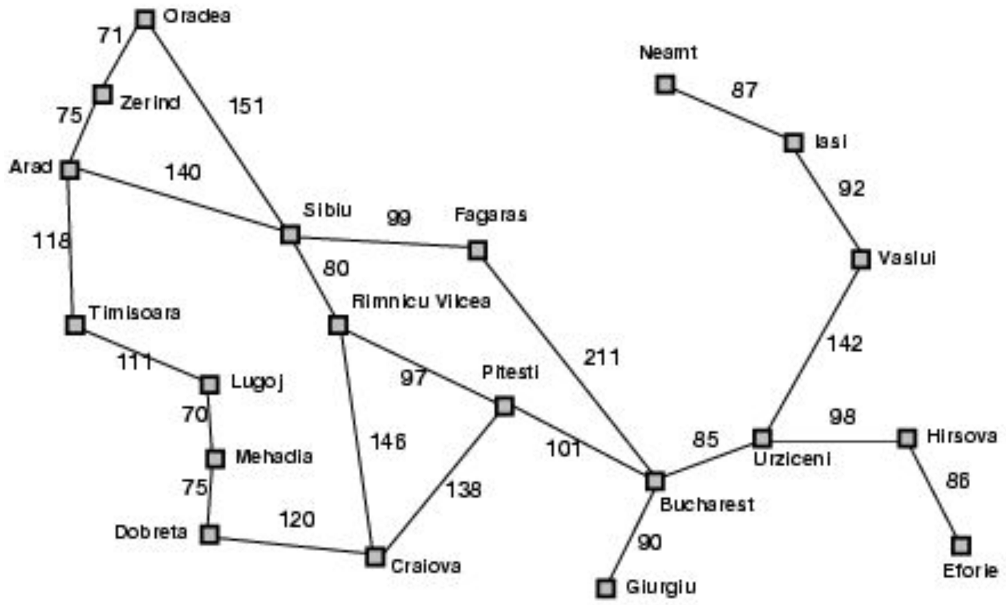
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

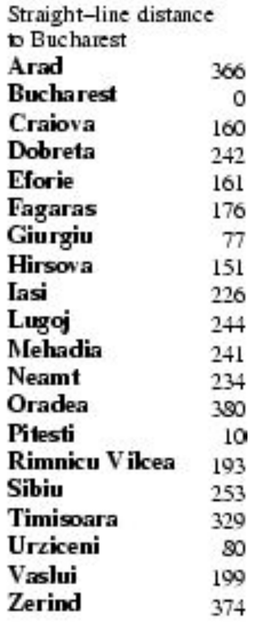
Greedy best-first search

- f(n) = estimate of cost from *n* to *goal*
- e.g., *f(n)* = straight-line distance from *n* to Bucharest
- Greedy best-first search expands the node that appears to be closest to goal.

Greedy best-first search example

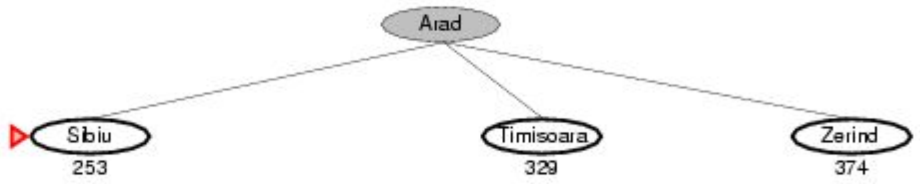


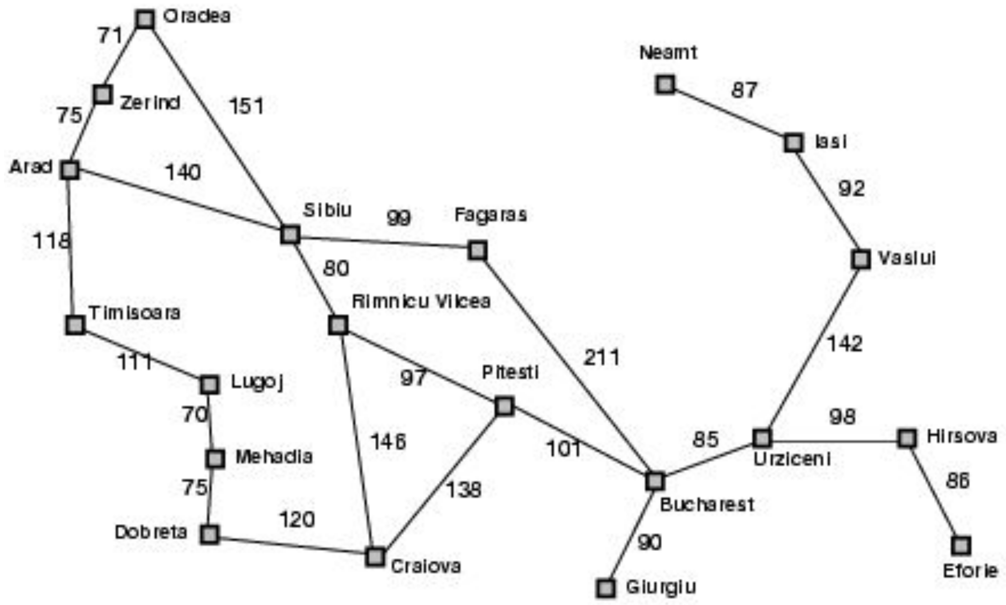



Straight–line distance to Bucharest

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

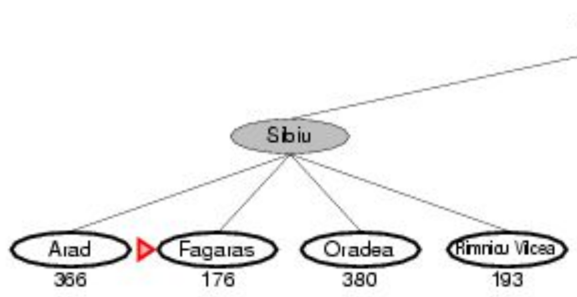


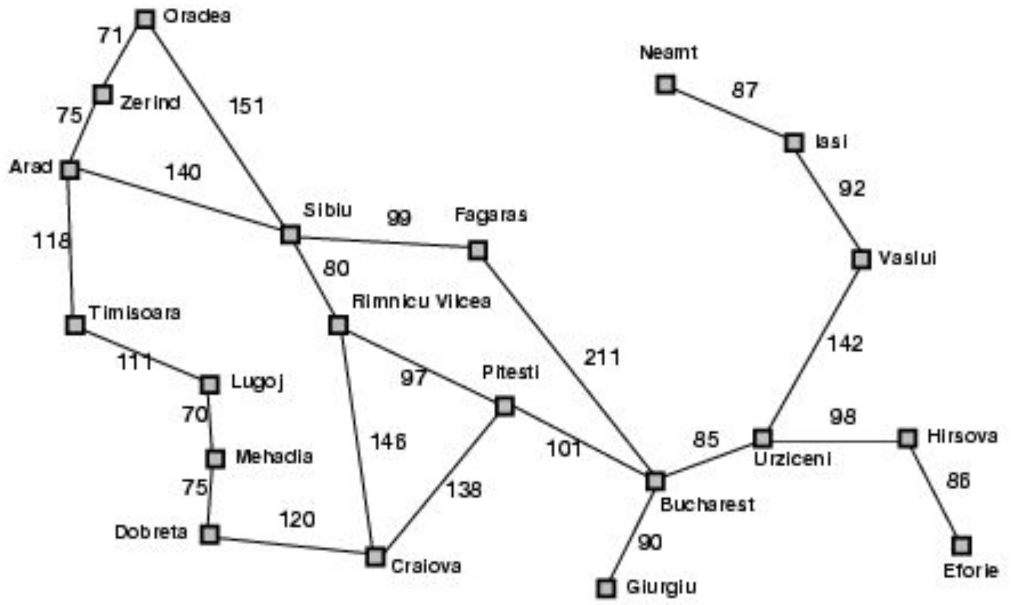


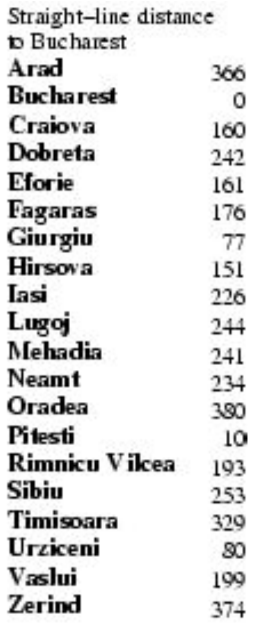
Straight–line distance to Bucharest

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

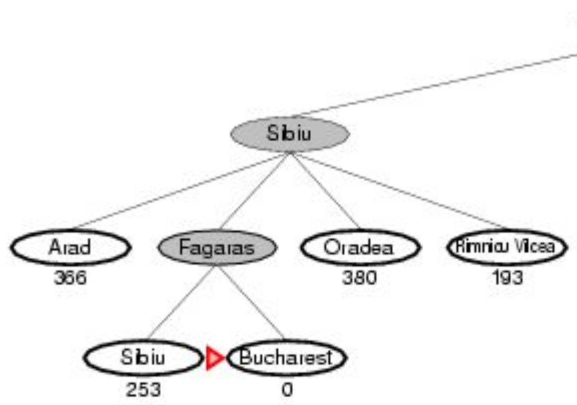


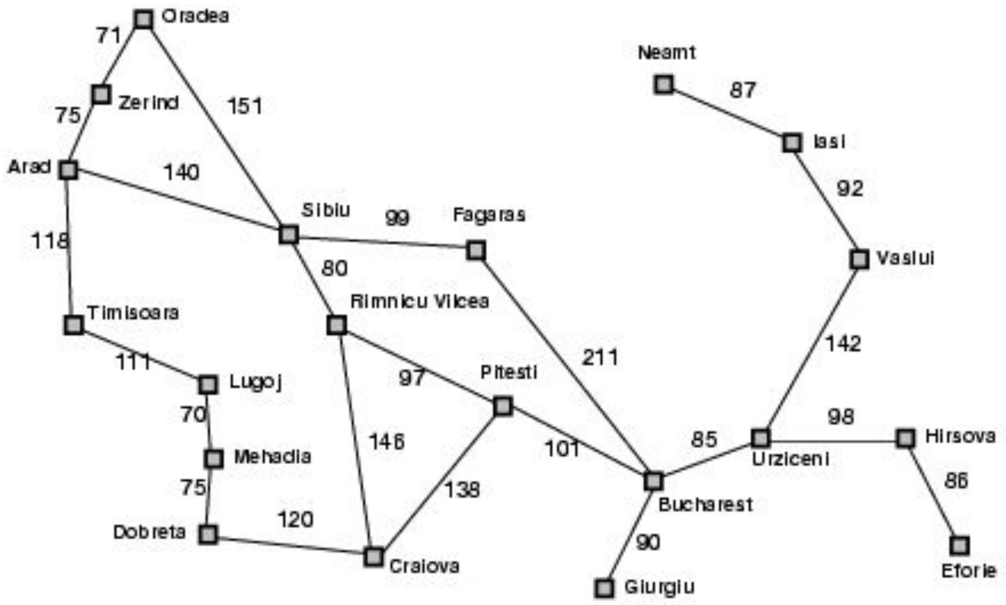



Straight–line distance to Bucharest

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

Properties of greedy best-first search

- Complete? No – can get stuck in loops.
- Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- Space? $O(b^m)$ - keeps all nodes in memory
- Optimal? No

e.g. Arad☐Sibiu☐Rimnicu Virea☐Pitesti☐Bucharest is shorter!

A^* search

- Idea: avoid expanding paths that are already expensive

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

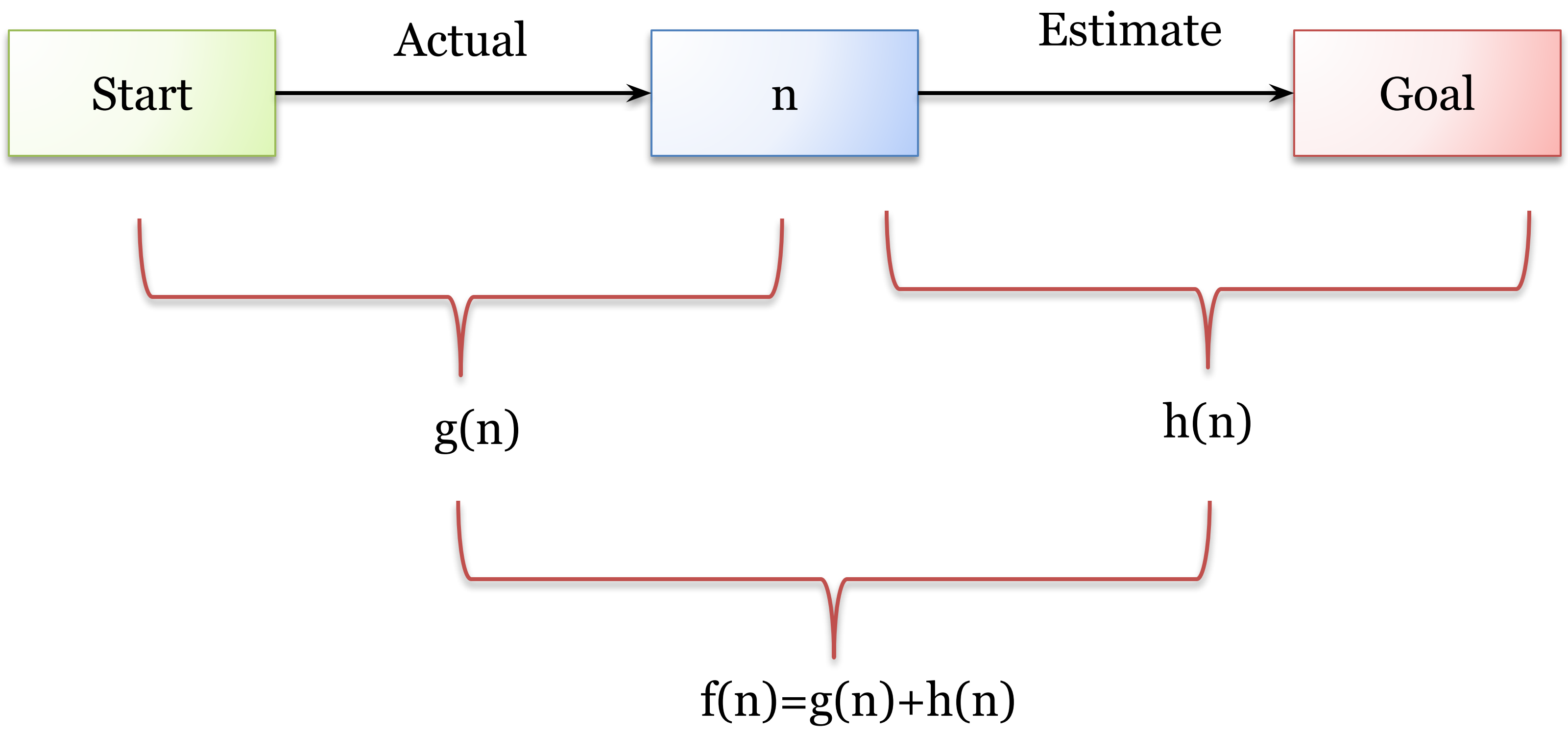
$g(n)$ = cost so far to reach n

$h(n)$ = estimated cost from n to goal

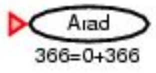
$f(n)$ = estimated total cost of path through n to goal

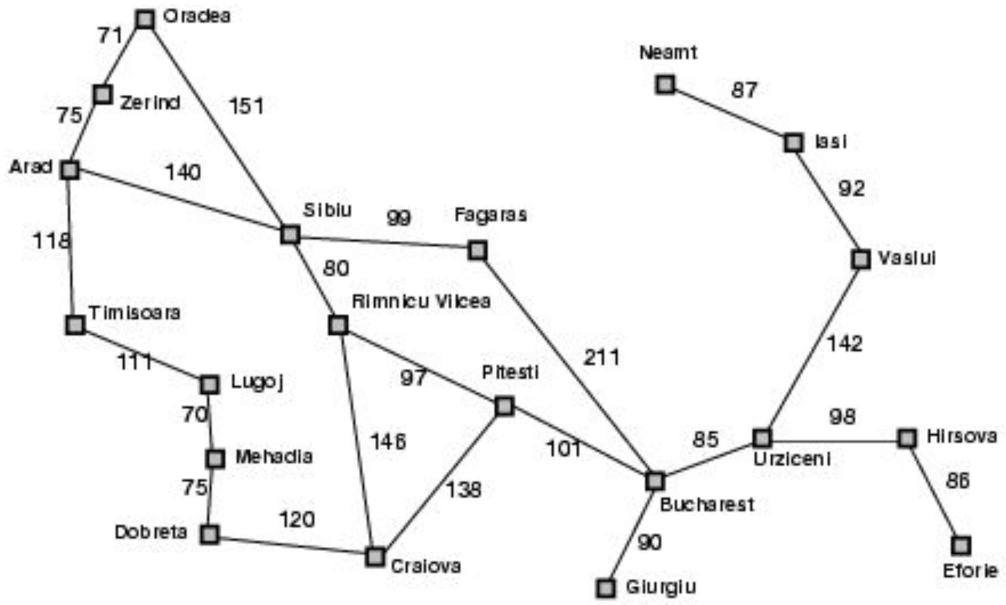
A* Search

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



A* search example

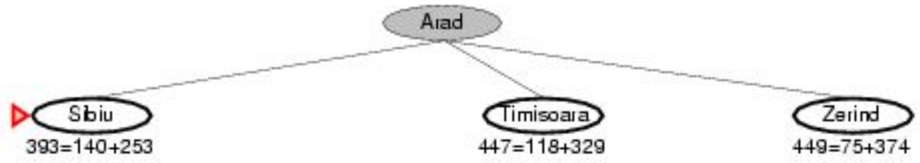


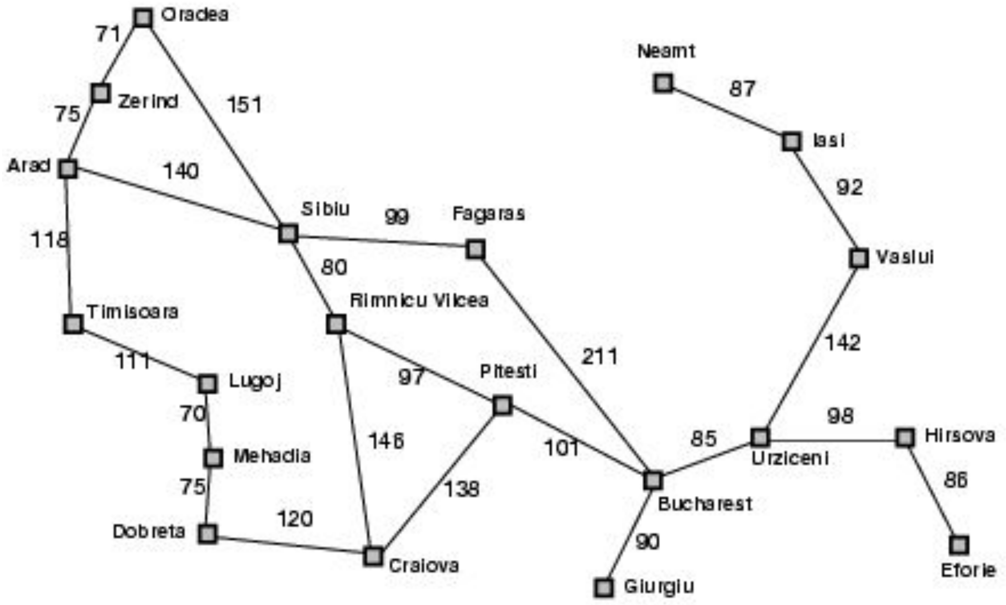


Straight–line distance to Bucharest

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

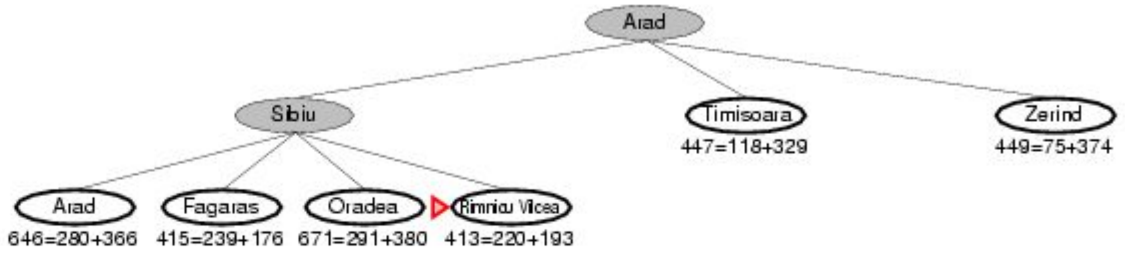
A* search example

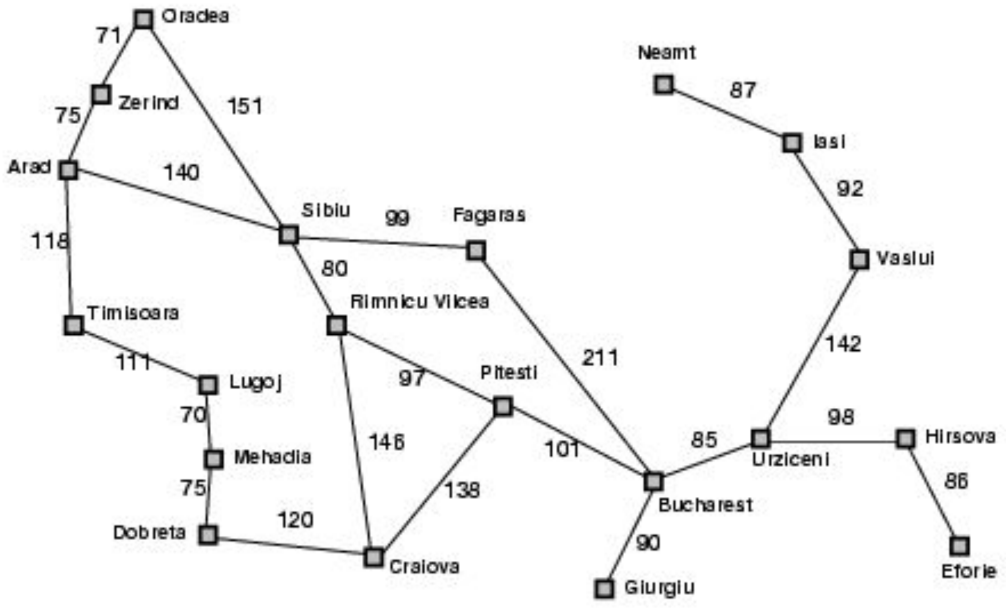




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

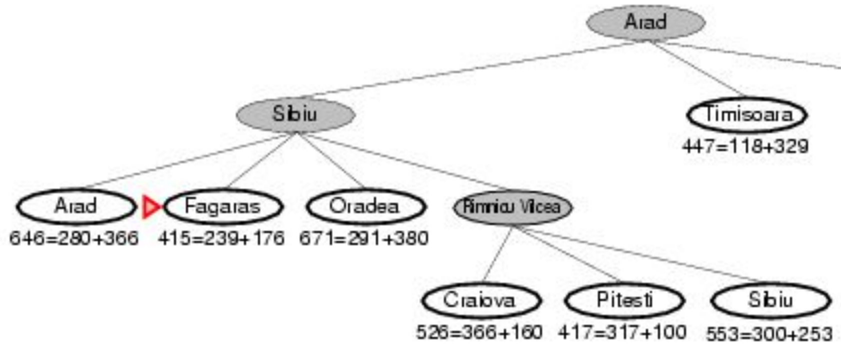


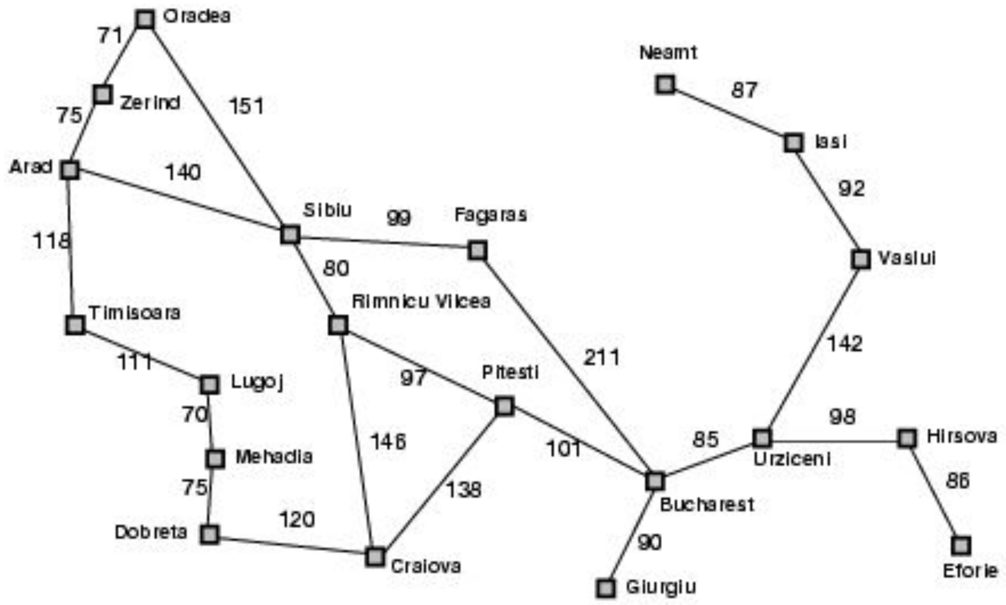


Straight–line distance to Bucharest

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

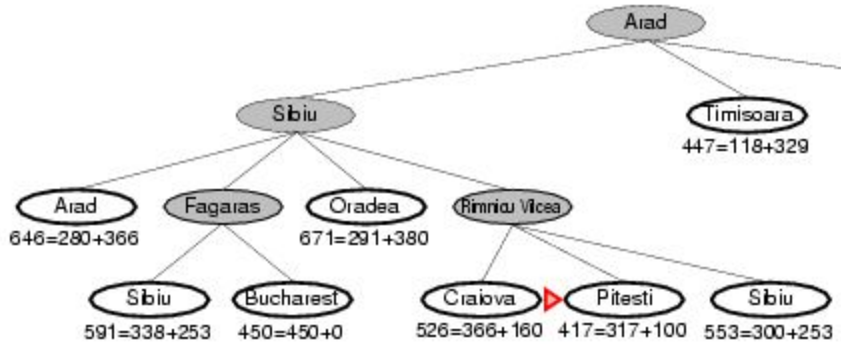


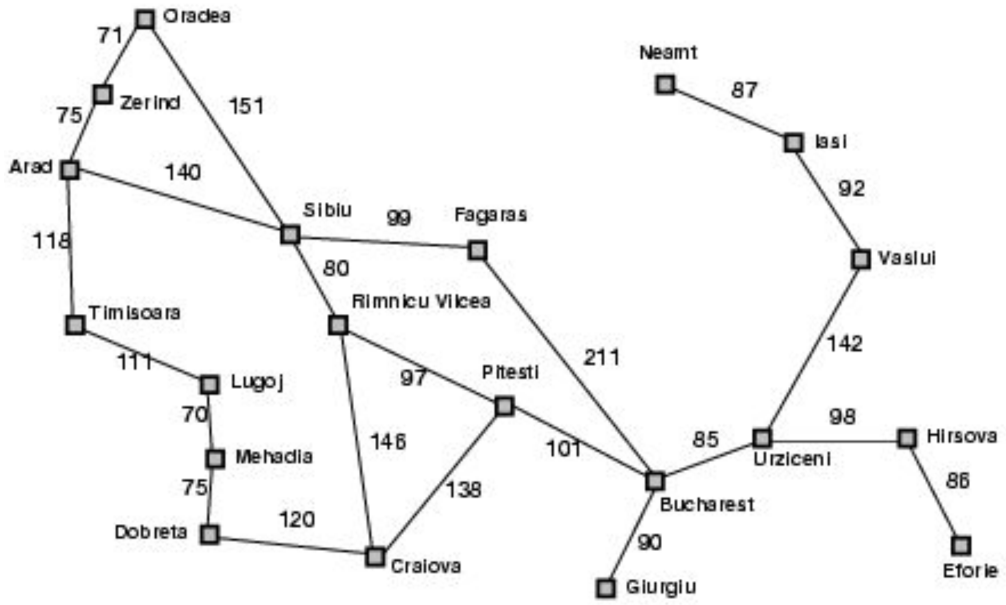


Straight–line distance to Bucharest

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

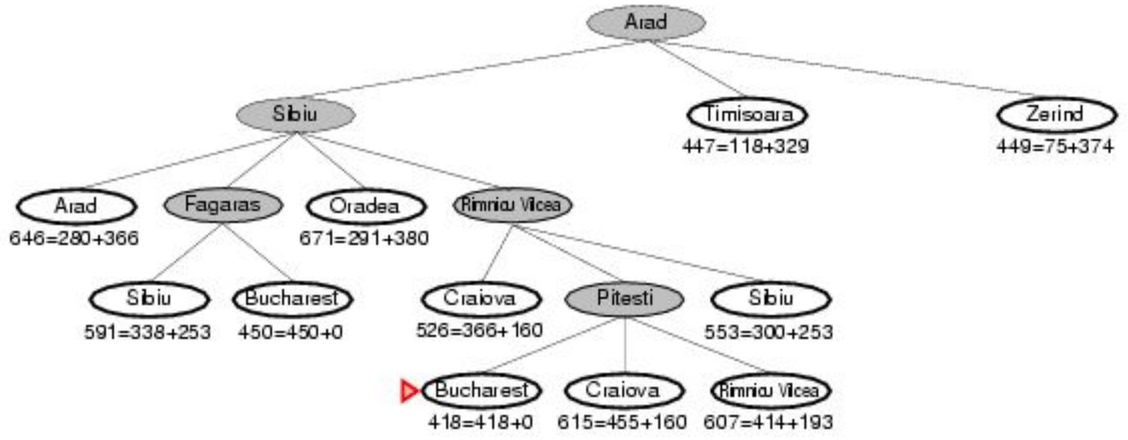


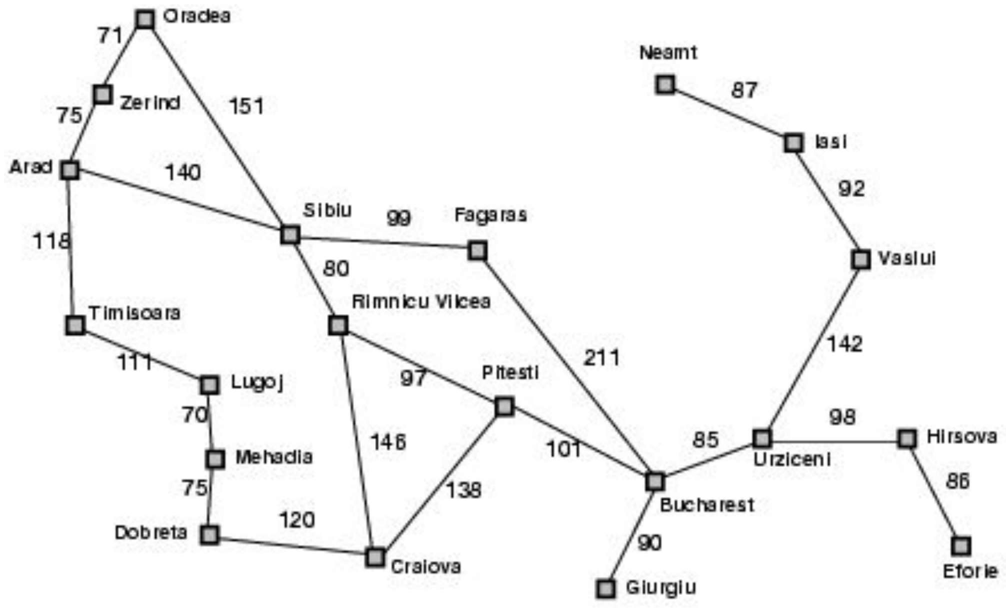


Straight–line distance to Bucharest

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

Admissible heuristics

- A heuristic $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$, 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

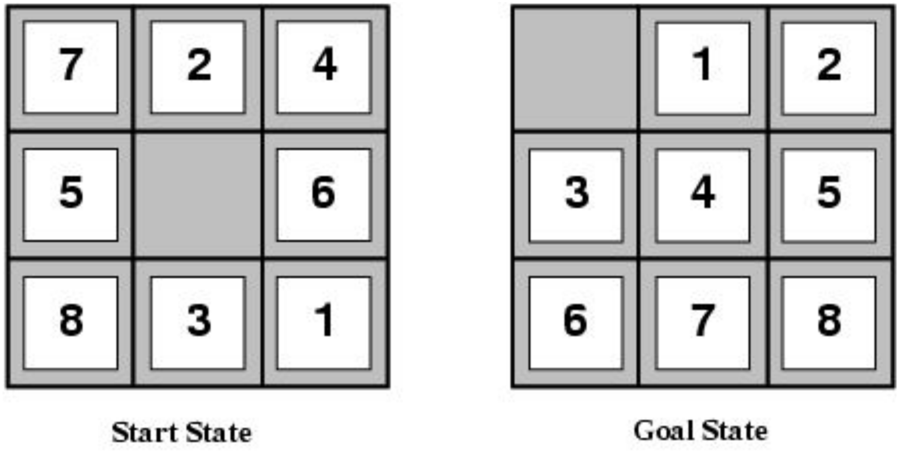
Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)

- $h_1(S) = ?$
- $h_2(S) = ?$

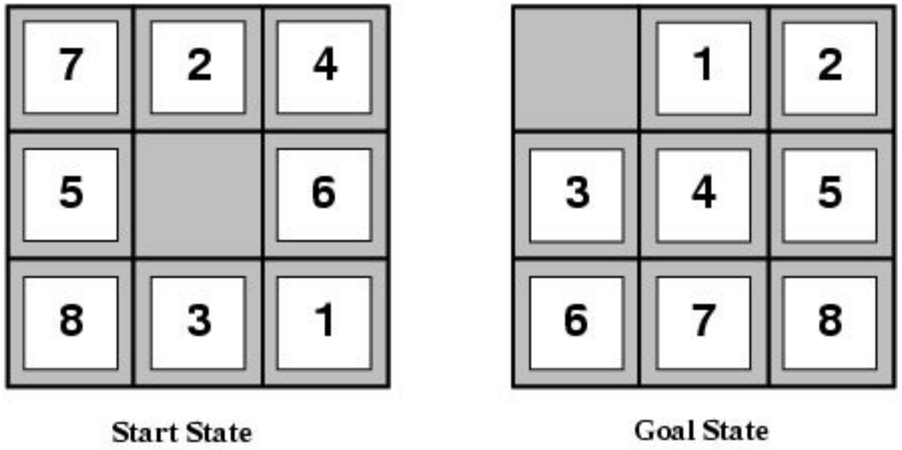


Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



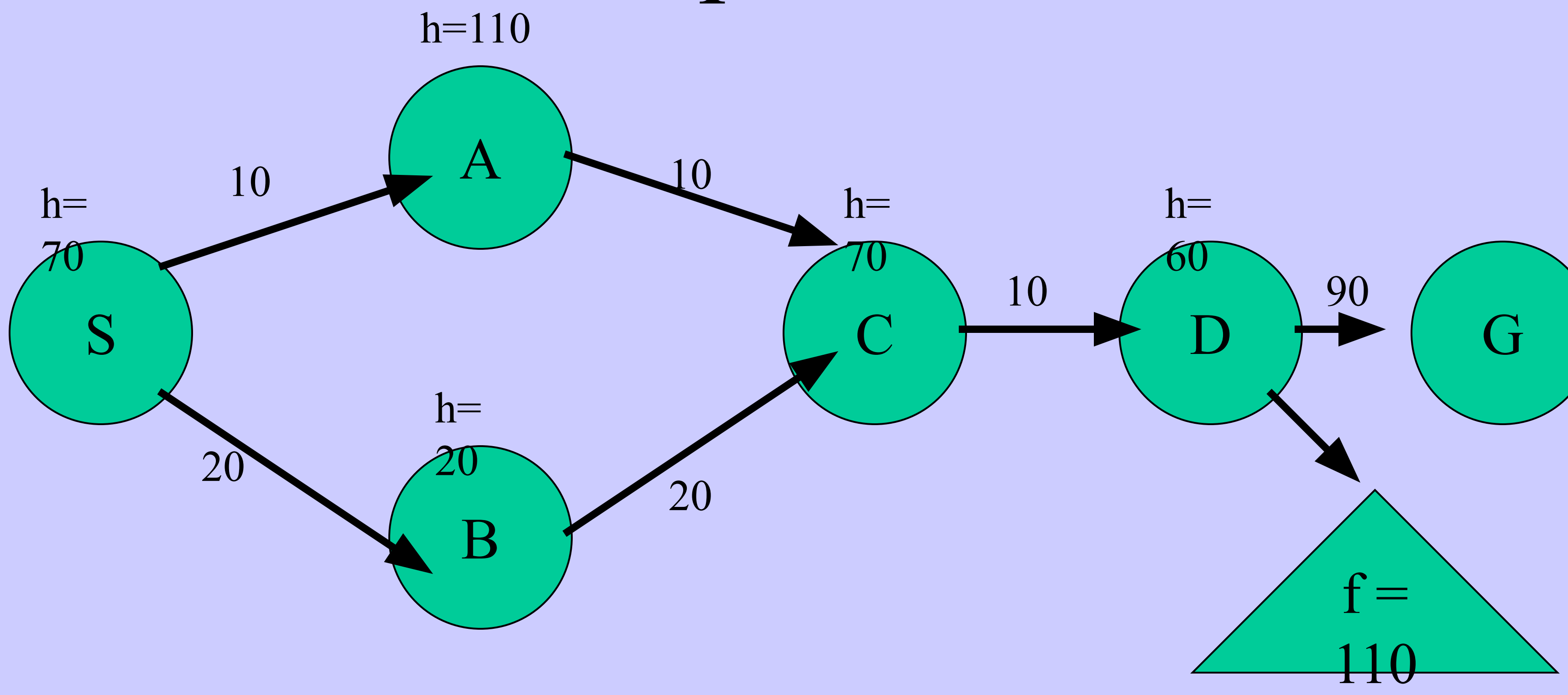
- $h_1(S) = ?$ 8
- $h_2(S) = ?$ 3+1+2+2+2+3+3+2 = 18

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible) then h_2 dominates h_1
- h_2 is better for search: it is guaranteed to expand less or equal no. of nodes.

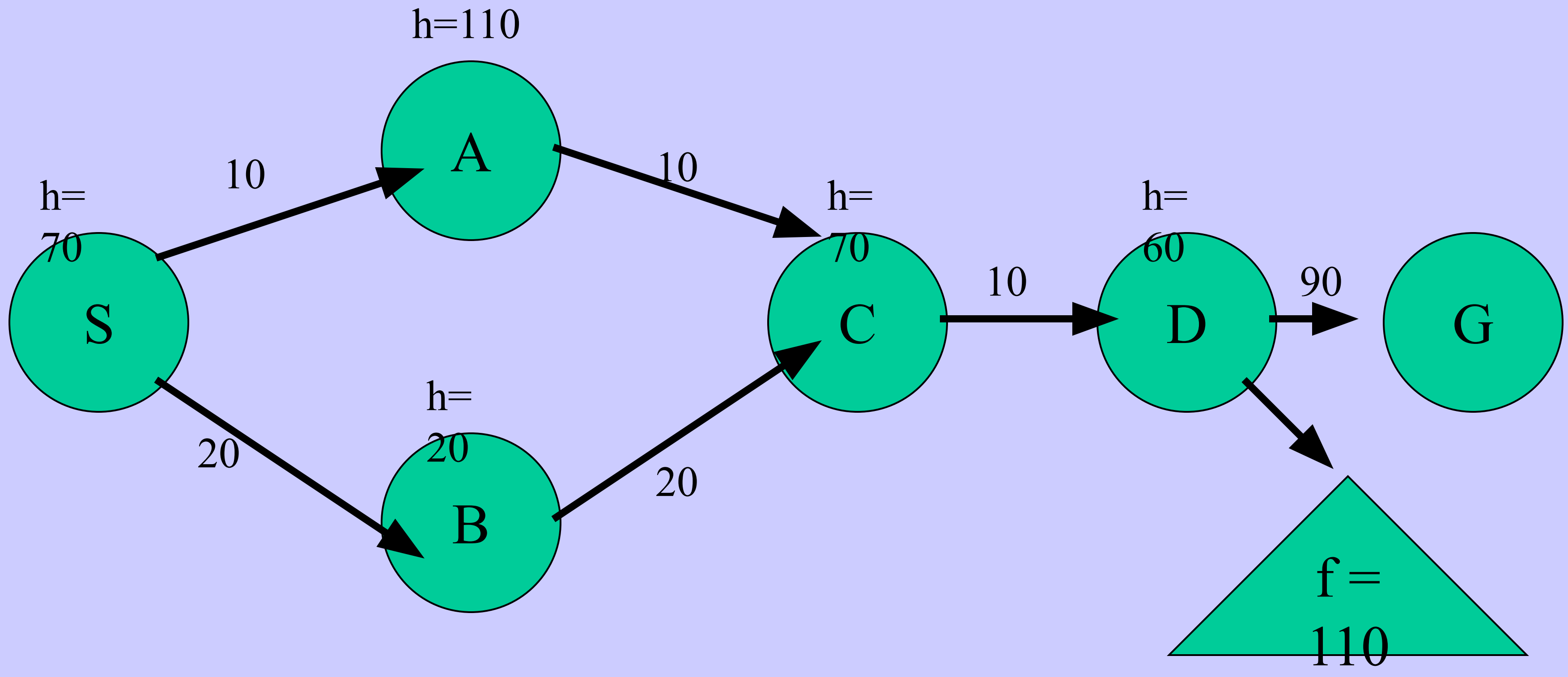
Example: A*

A* must re-open closed nodes



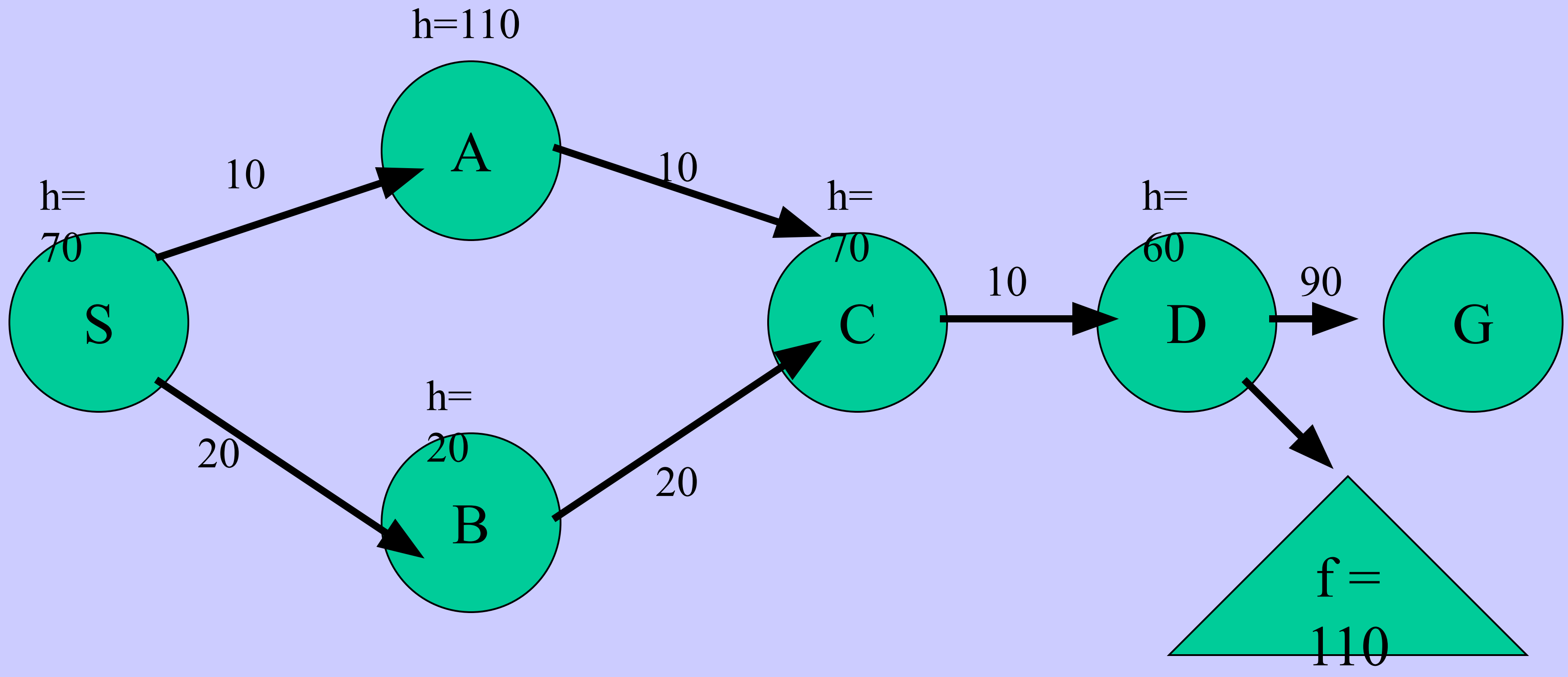
OPEN: (S,70)
CLOSED:

A* must re-open closed nodes



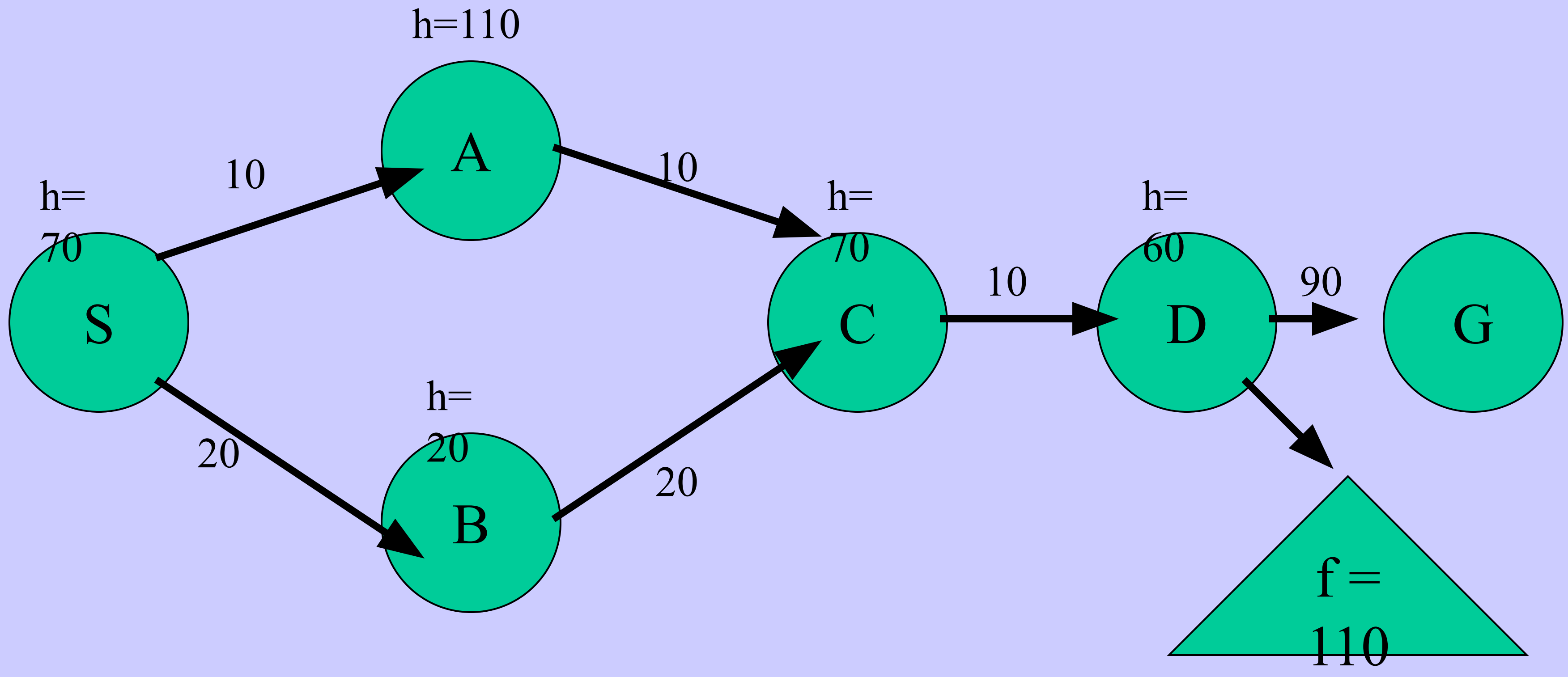
OPEN: (A,120), (B,40)
CLOSED: (S,70)

A* must re-open closed nodes



OPEN: (A,120), (C,110)
CLOSED: (S,70), (B,40)

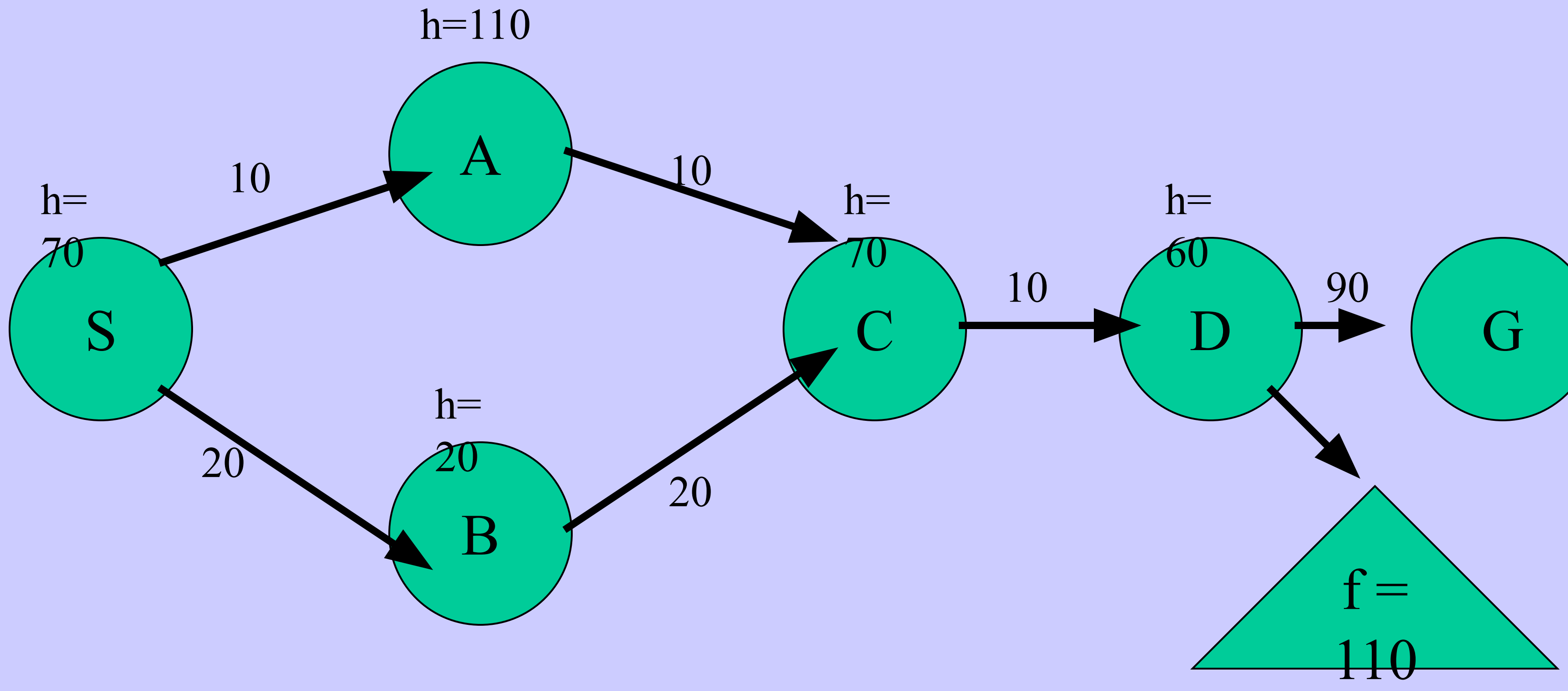
A* must re-open closed nodes



OPEN: (A,120), (D,110)

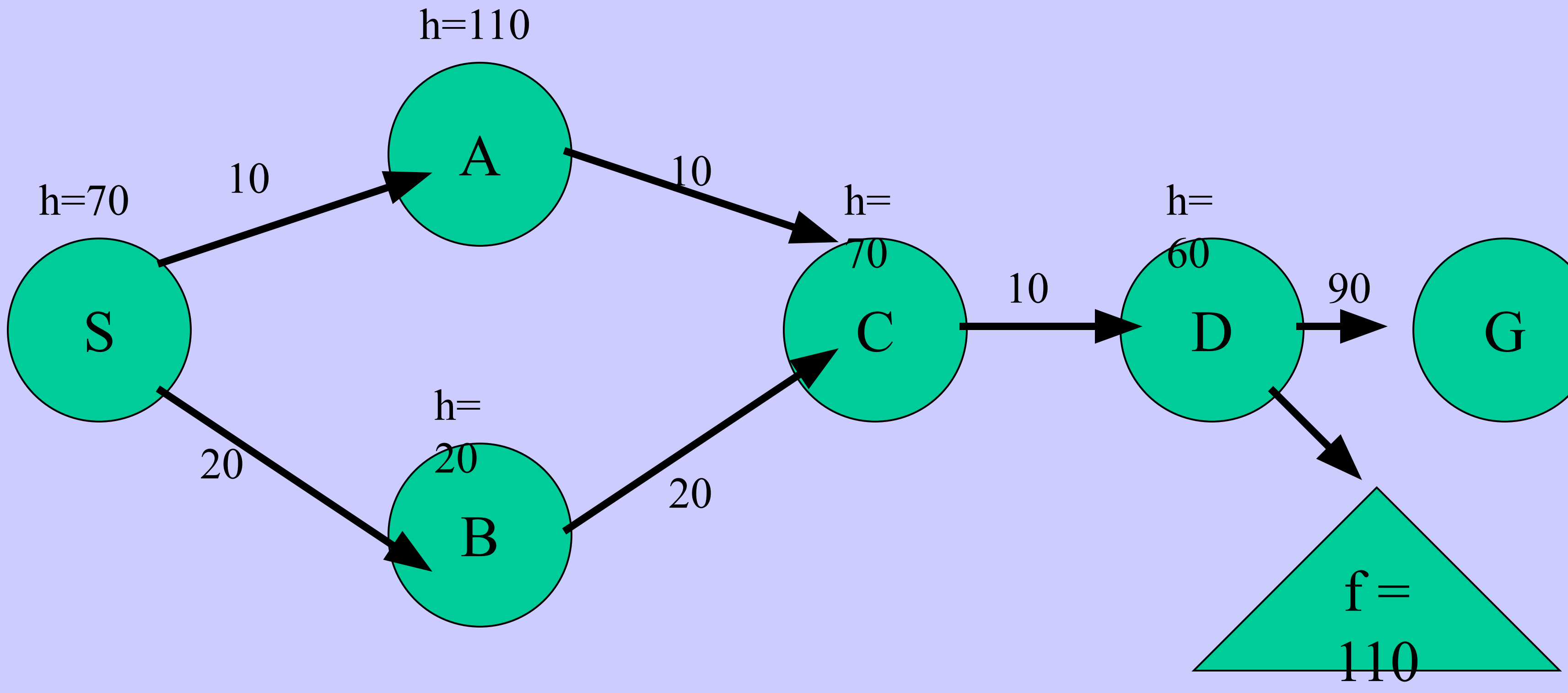
CLOSED: (S,70), (B,40), (C,110)

A* must re-open closed nodes



OPEN: (A,120), (G,140), (subtree with f=110)
CLOSED: (S,70), (B,40), (C,110), (D,110)

A* must re-open closed nodes



OPEN: (A,120), (G,140)

CLOSED: (S,70), (B,40), (C,110), (D,110), …

Thank You!

Any Questions?