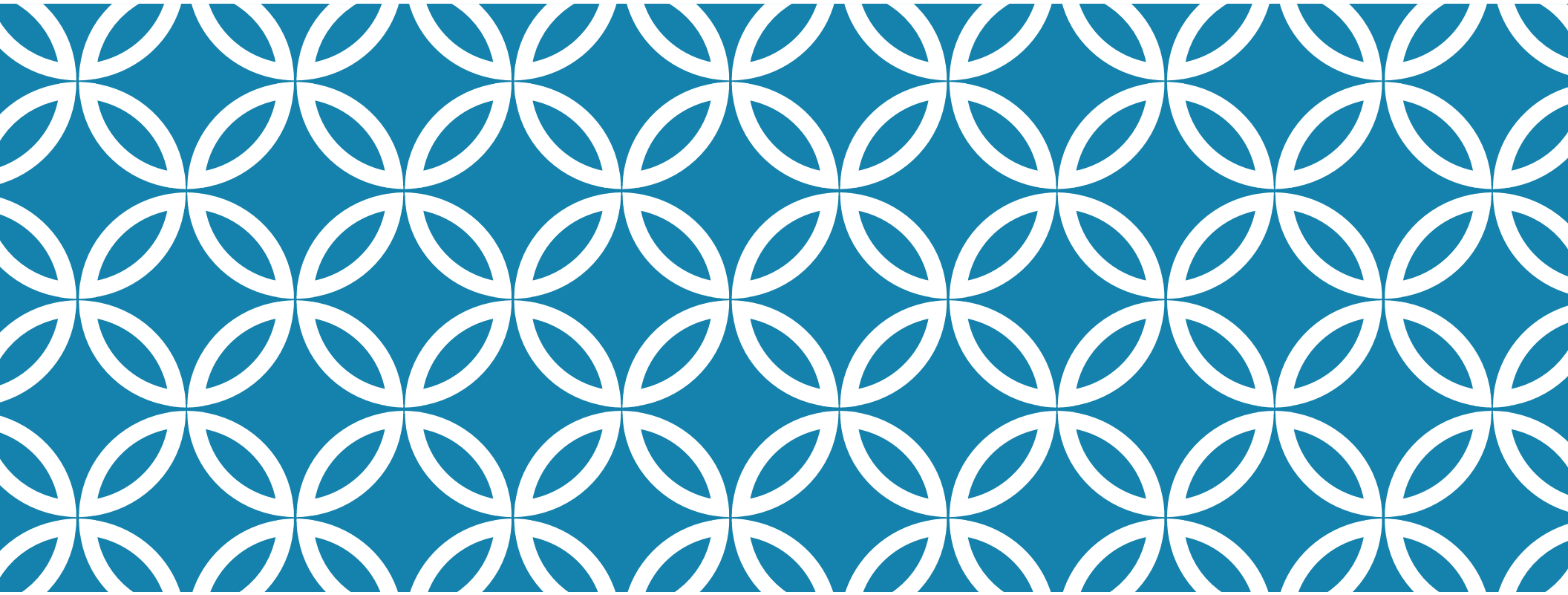
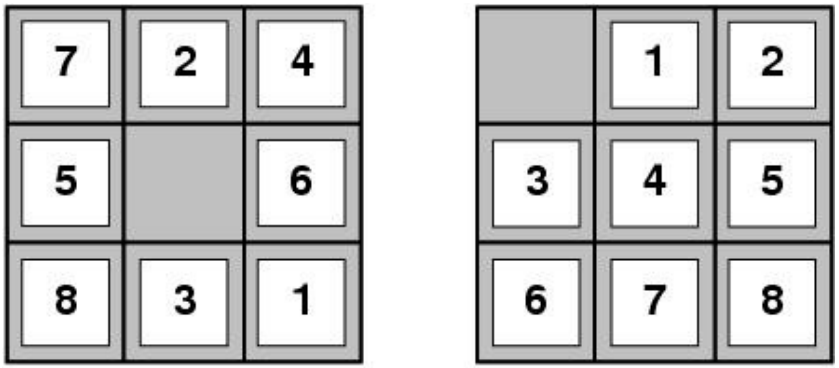


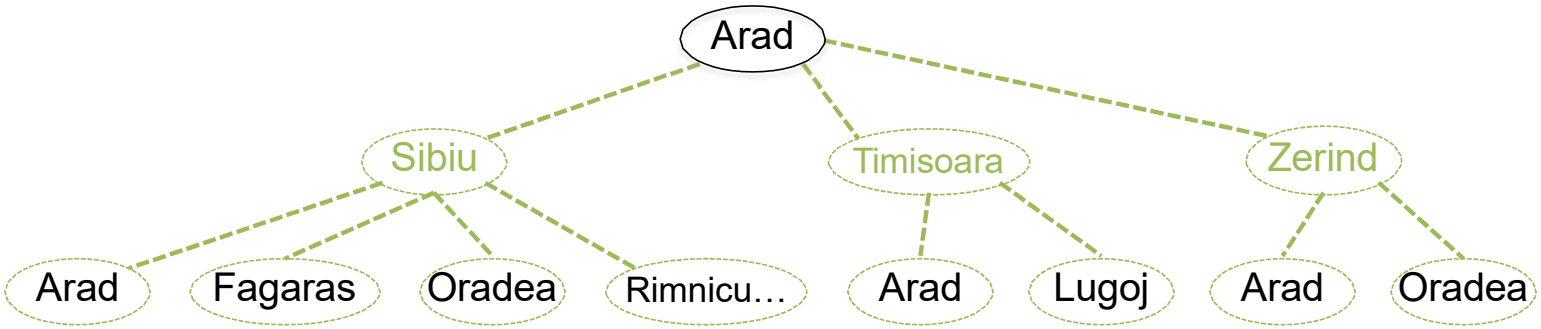
ARTIFICIAL INTELLIGENCE

Limitations of uninformed search



- Search space size makes search tedious
  - Combinatorial explosion
- Ex: 8-Puzzle
  - Average solution cost is ~ 22 steps
  - Branching factor ~ 3
  - Exhaustive search to depth 22: 3.1×10^{10} states
  - 24-Puzzle: 10^{24} states (much worse!)

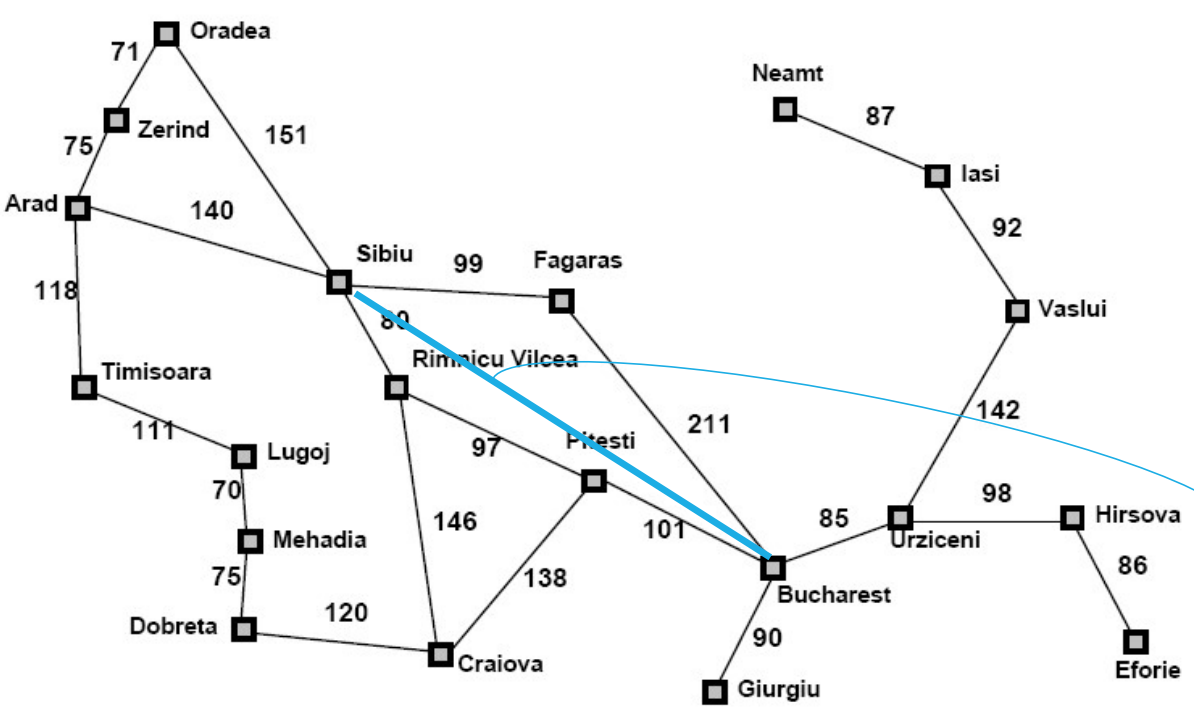
Recall: tree search



TREE-SEARCH (*problem*, *strategy*) : returns a solution
    initialize the search tree using the initial state of *problem*
    while (true):
        if no candidates for expansion: return failure
        choose a leaf node for expansion according to *strategy*
        if the node contains a goal state: return the corresponding solution
        else: expand the node and add the resulting nodes to the search tree

INFORMED (HEURISTIC) SEARCH

Informed Search - one that uses problem-specific knowledge beyond the definition of the problem itself.



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

HEURISTIC FUNCTION

- Heuristic
  - Definition: a commonsense rule or rules intended to increase the probability of solving some problem
  - Using rules of thumb to find answers

- Heuristic function h(n)
  - Estimate of (optimal) remaining cost from n to goal
  - Defined using only the *state* of node n
  - h(n) = 0 if n is a goal node
  - Example: straight line distance from n to Bucharest
    - Not true state space distance, just estimate! Actual distance can be higher

- Provides problem-specific knowledge to the search algorithm

HEURISTIC FUNCTION

- Idea: use a heuristic function h(n) for each node
  - g(n) = known path cost so far to node n
  - h(n) = *estimate* of (optimal) cost to goal from node n
  - f(n) = g(n)+h(n) = *estimate* of total cost to goal through n
  - f(n) provides an estimate for the total cost

- "Best first" search implementation
  - Order the nodes in frontier by an evaluation function
  - Greedy Best-First: order by h(n)
  - A* search: order by f(n)

- Search efficiency depends on heuristic quality!
  - The better your heuristic, the faster your search!

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 implementation of best-first graph search is identical to that for uniform-cost search except for the use of f instead of g to order the priority queue.

The choice of f determines the search strategy.

RELATIONSHIP OF SEARCH ALGORITHMS

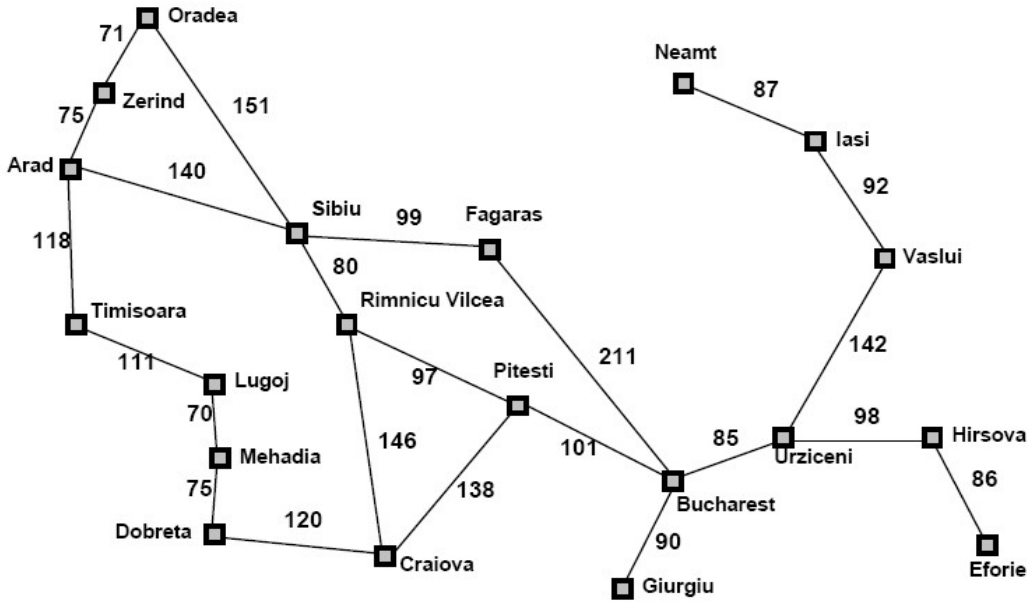
- Notation
  - g(n) = known cost so far to reach n
  - h(n) = estimated (optimal) cost from n to goal
  - f(n) = g(n)+h(n) = estimated (optimal) total cost through n
- Uniform cost search: sort frontier by g(n)
- Greedy best-first search: sort frontier by h(n)
- A* search: sort frontier by f(n)
  - Optimal for admissible / consistent heuristics
  - Generally the preferred heuristic search framework

STRATEGY 1: **GREEDY BEST-FIRST SEARCH**

f(n) = h(n)

h(n) -> straight line distance of node n from goal (Bucharest) as mentioned in table

g(n) -> path cost from starting node (Arad) until node n (implicitly determined from the problem itself)

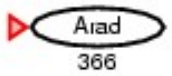


g(n)

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

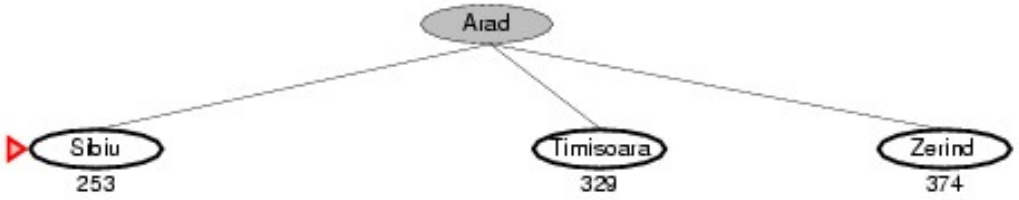
h(n)

GREEDY BEST-FIRST SEARCH EXAMPLE



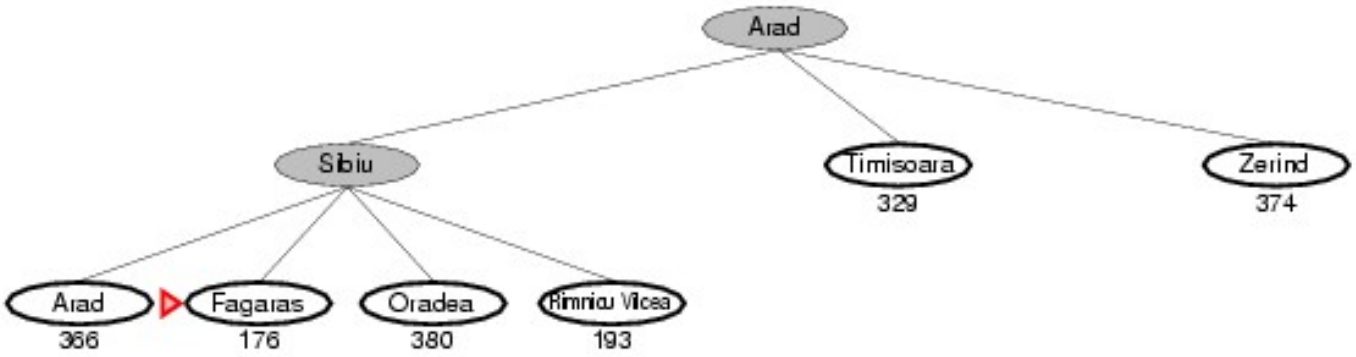
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

GREEDY BEST-FIRST SEARCH EXAMPLE



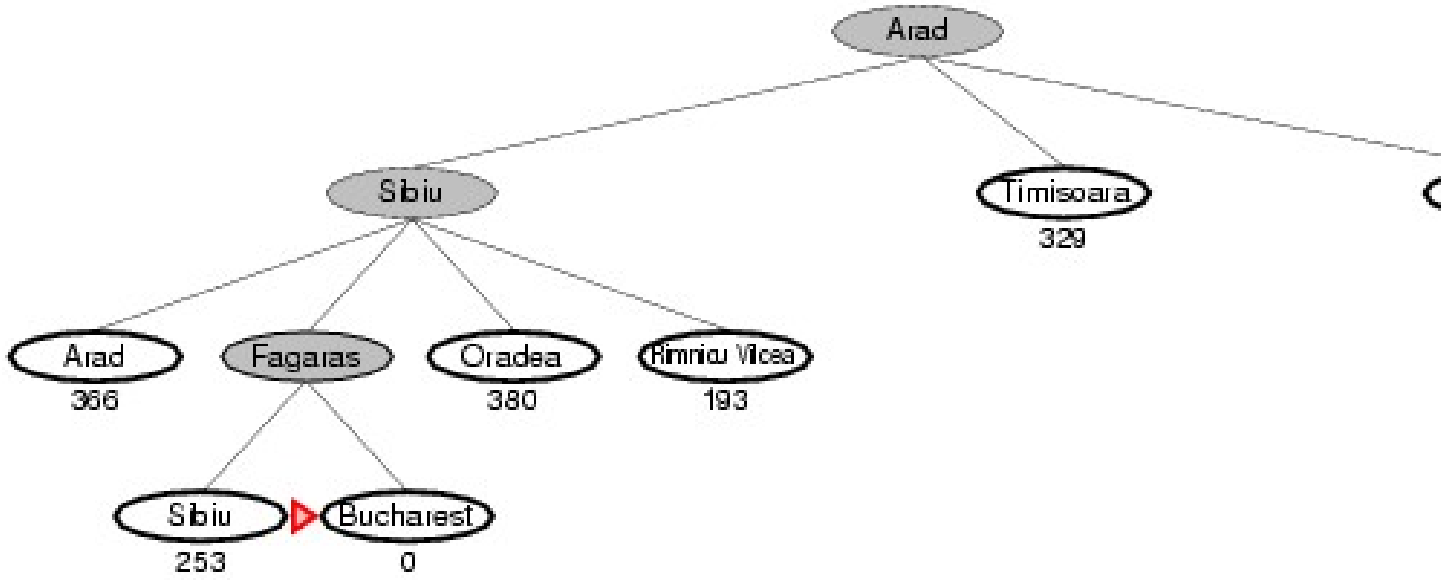
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

GREEDY BEST-FIRST SEARCH EXAMPLE

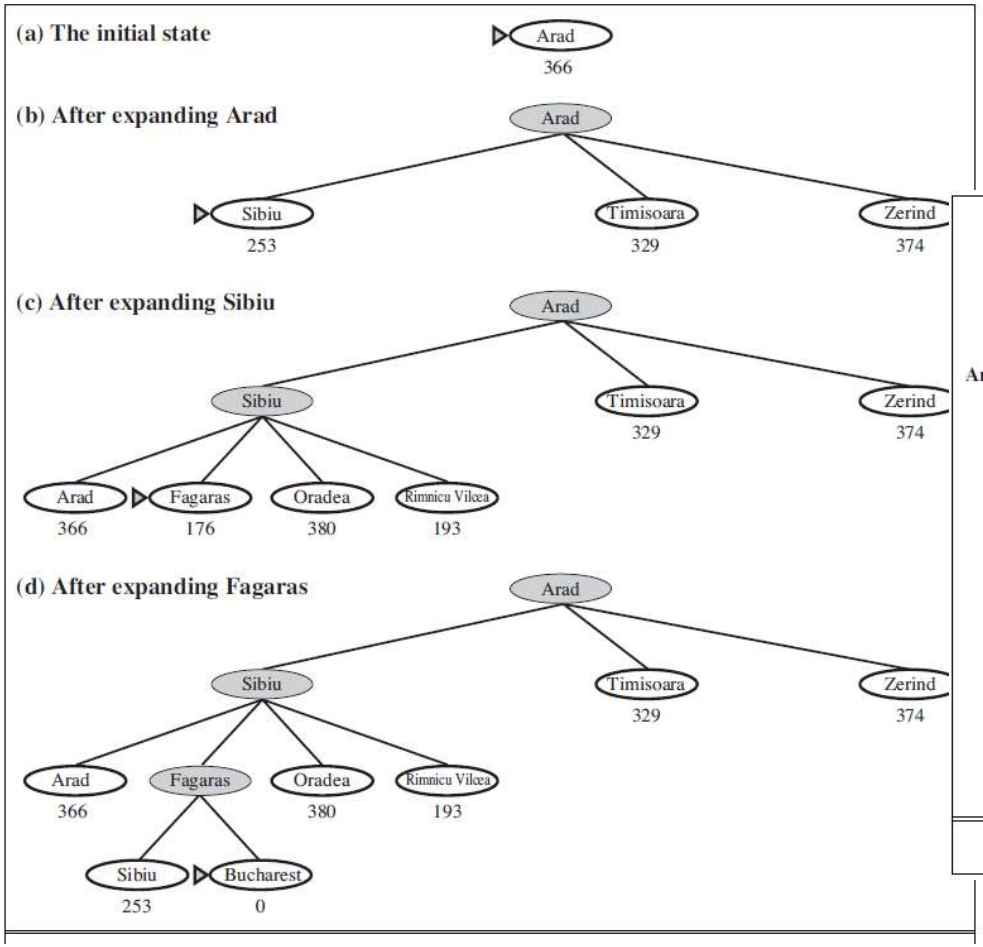


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

GREEDY BEST-FIRST SEARCH EXAMPLE



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



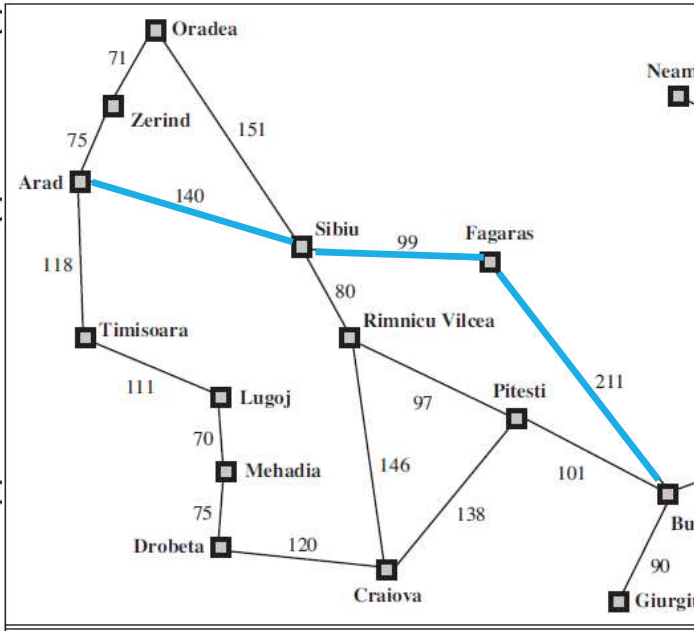


Figure 3.2 A simplified road map of part of Romania.

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

A* SEARCH:

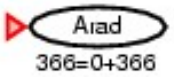
Minimizes the total estimated cost solution

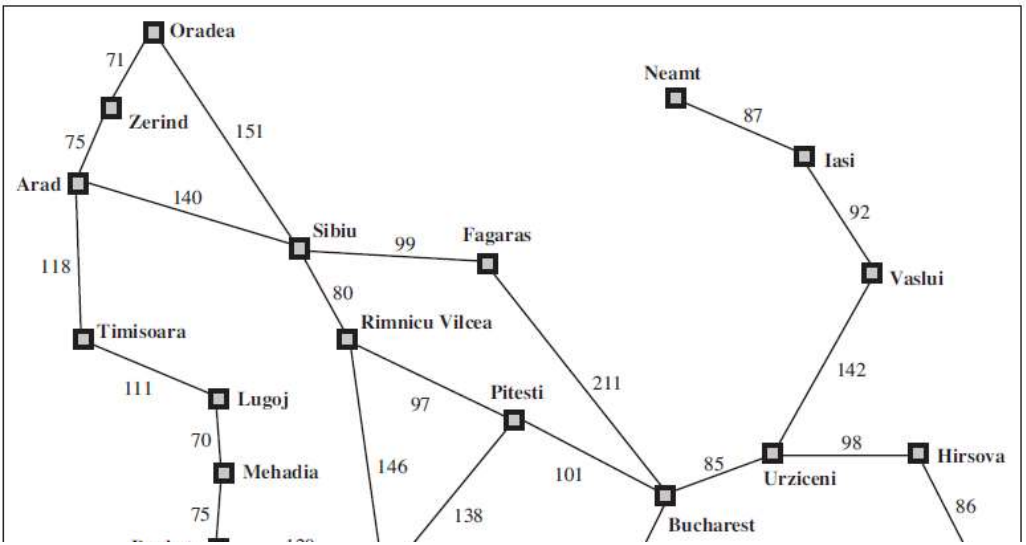
$f(n) = g(n) + h(n)$

Since g(n) gives the path cost from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal, we have

f(n) = estimated cost of the cheapest solution through n

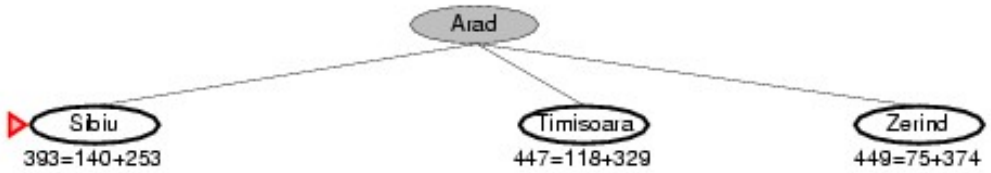
A* SEARCH EXAMPLE

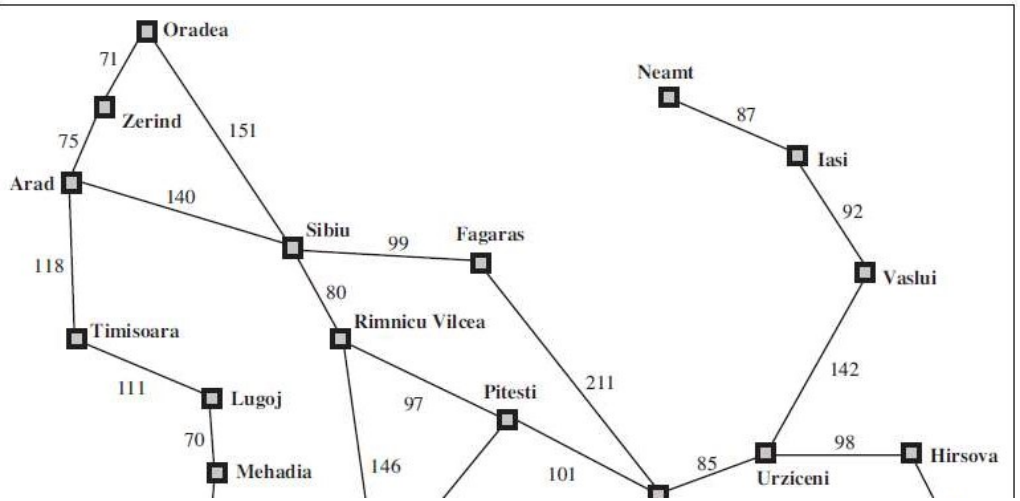




Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

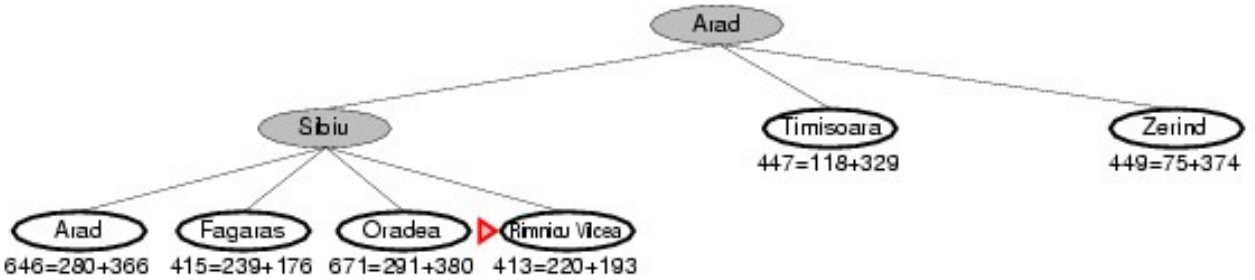
A* SEARCH EXAMPLE

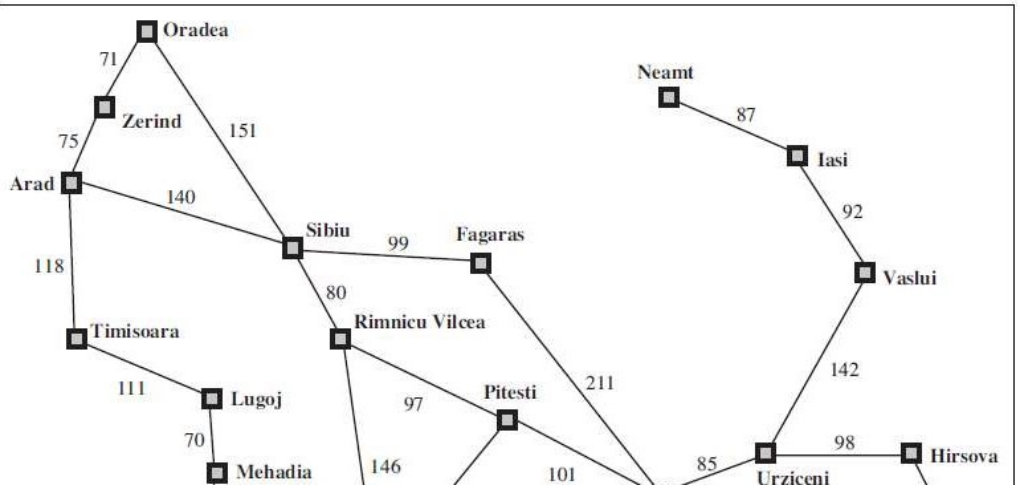




Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

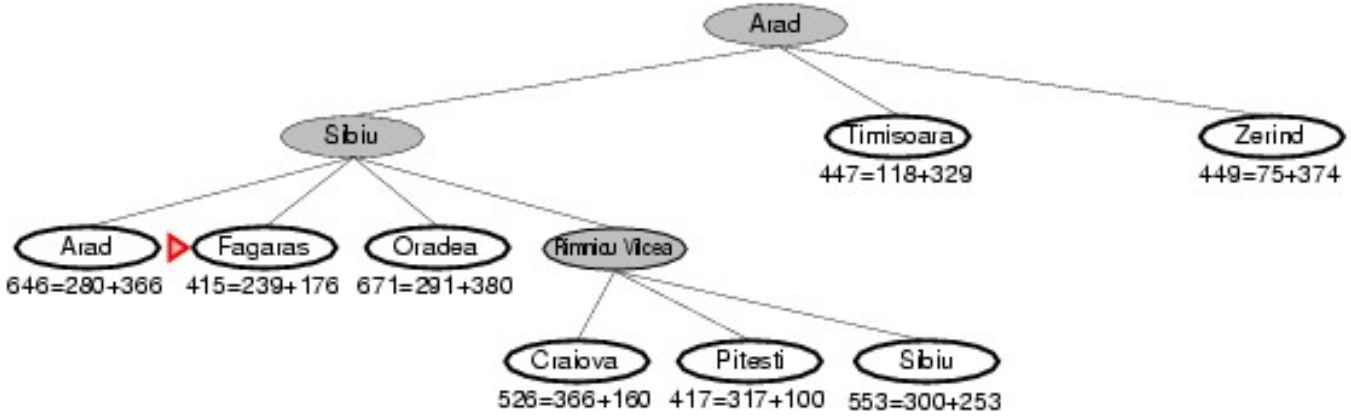
A* SEARCH EXAMPLE

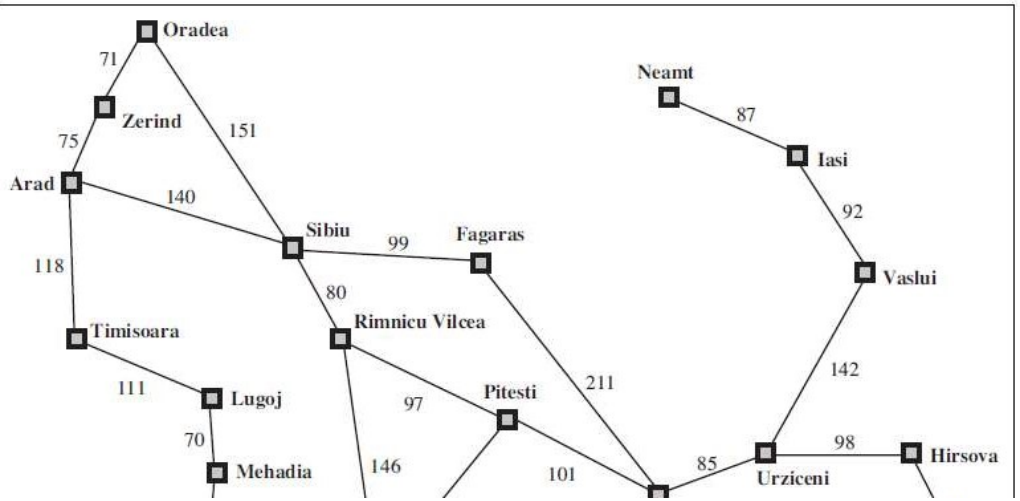




Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

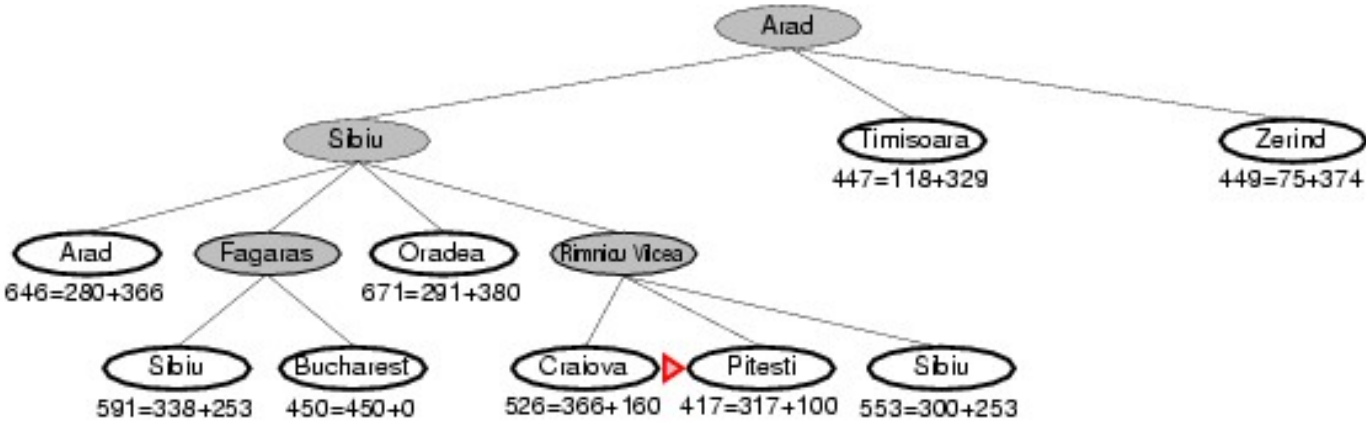
A* SEARCH EXAMPLE

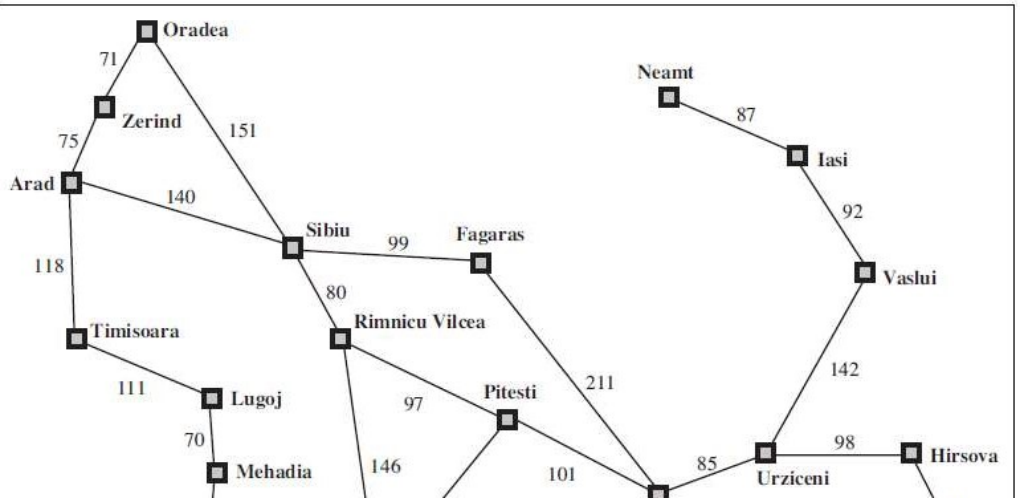




Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

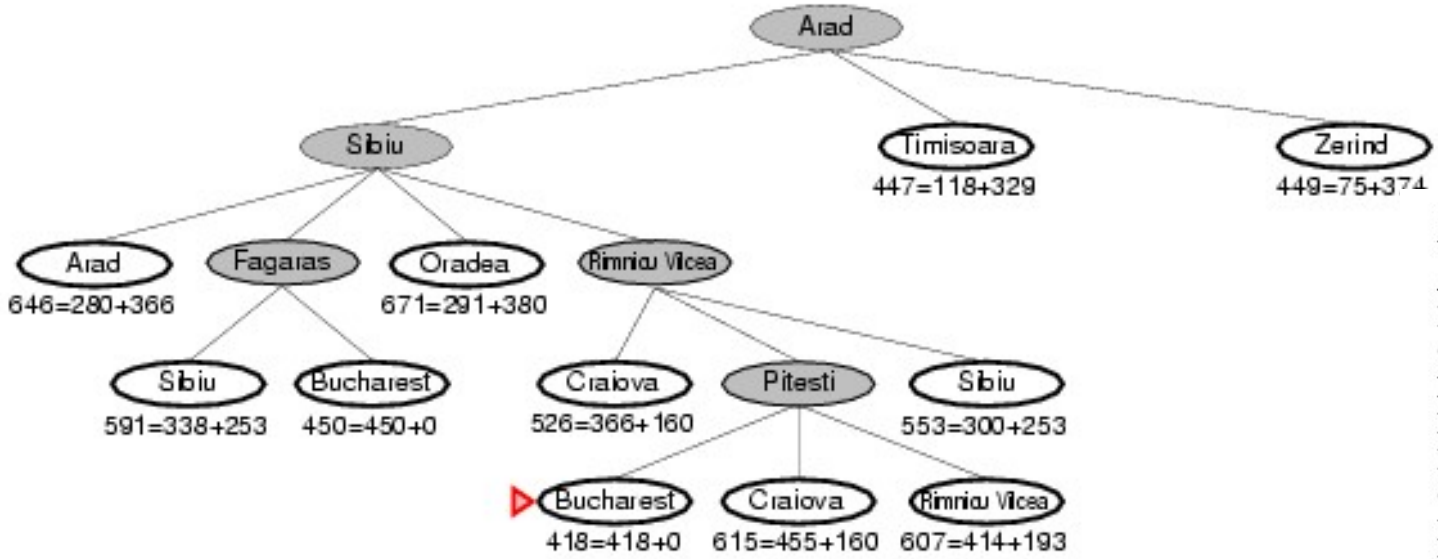
A* SEARCH EXAMPLE

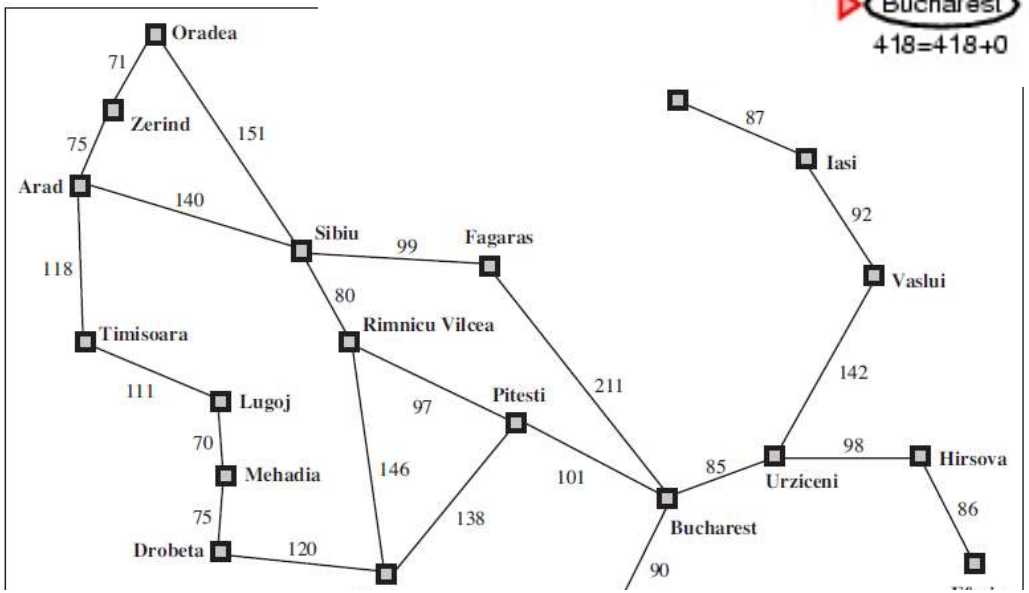




Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

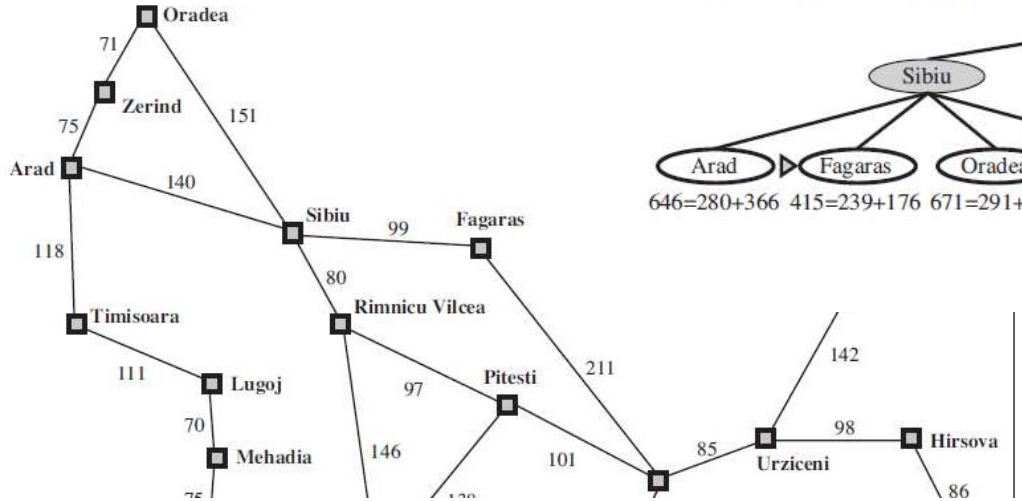
A* S



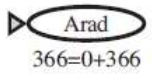


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

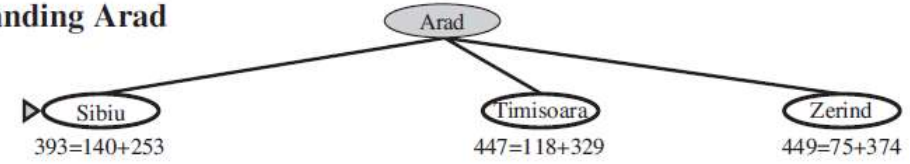
A* SEARCH:



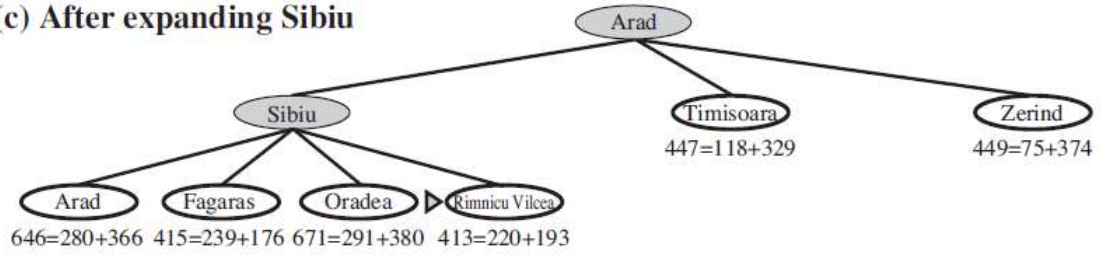
(a) The initial state



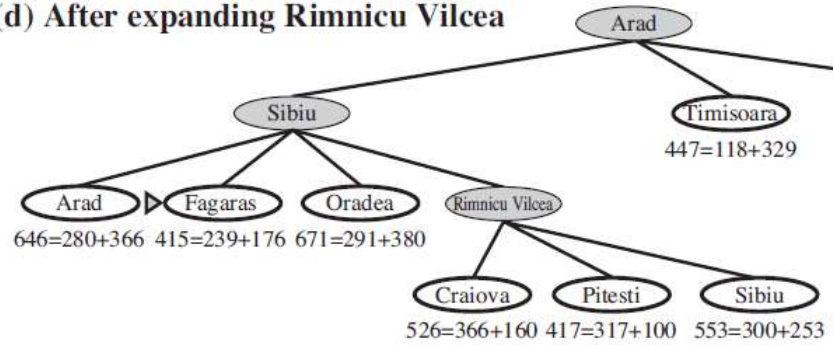
(b) After expanding Arad



(c) After expanding Sibiu

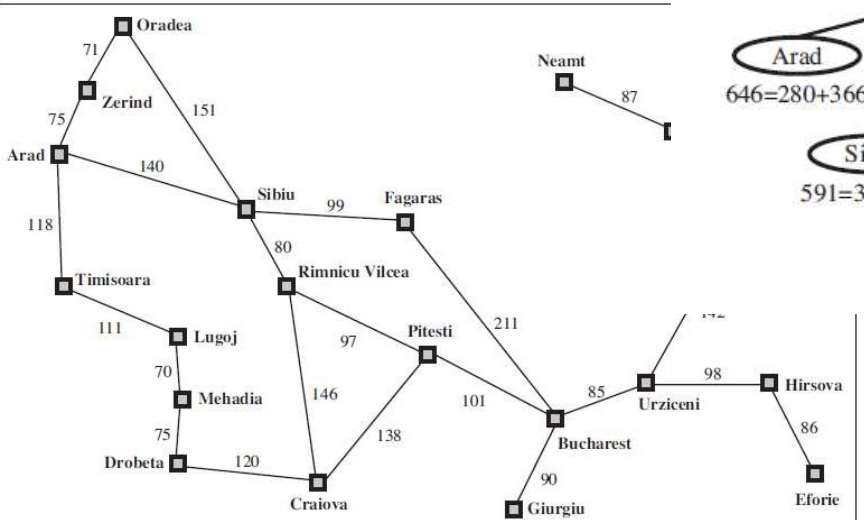


(d) After expanding Rimnicu Vilcea

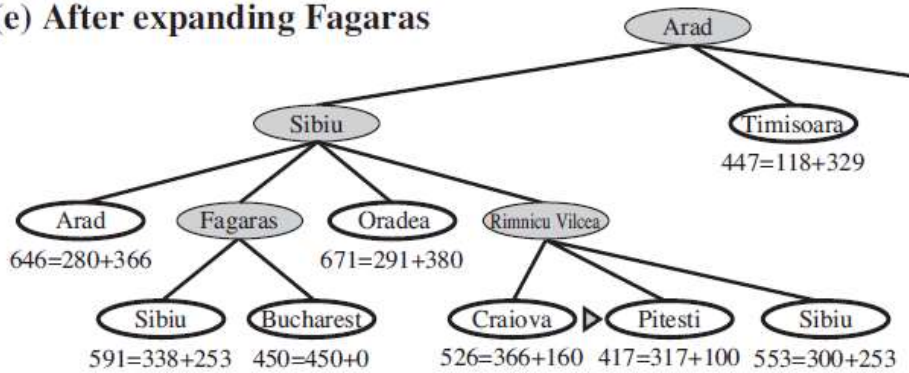


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

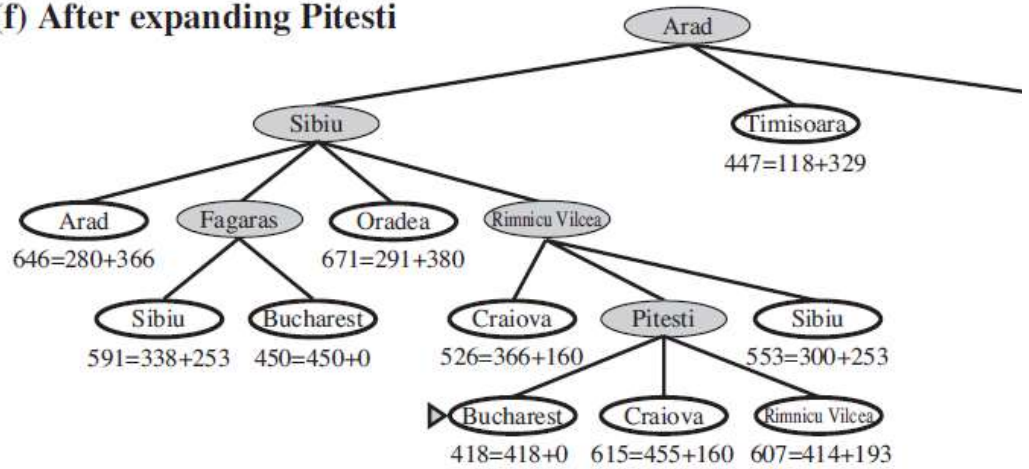
A* SEARCH:



(e) After expanding Fagaras



(f) After expanding Pitesti



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

ADMISSIBLE HEURISTICS – A* OPTIMILATY

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

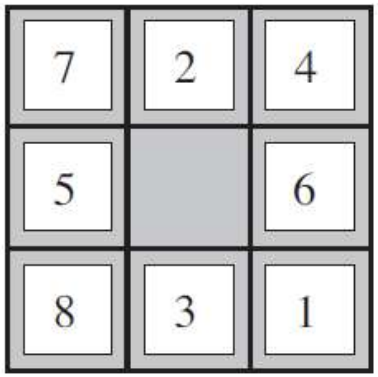
  $h^*(n)$ = the true cost to reach the goal state from n

- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is never pessimistic
  - Ex: straight-line distance never overestimates road distance

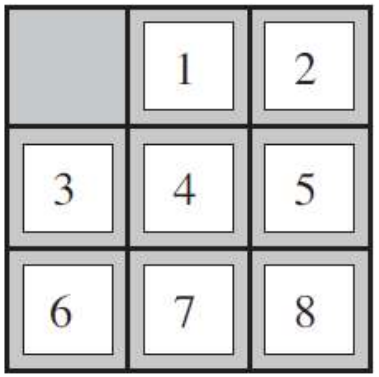
- **Theorem:**

  if *h(n)* is admissible, A* using Tree-Search is optimal

SAMPLE HEURISTICS



Start State



Goal State

h1 = the number of misplaced tiles

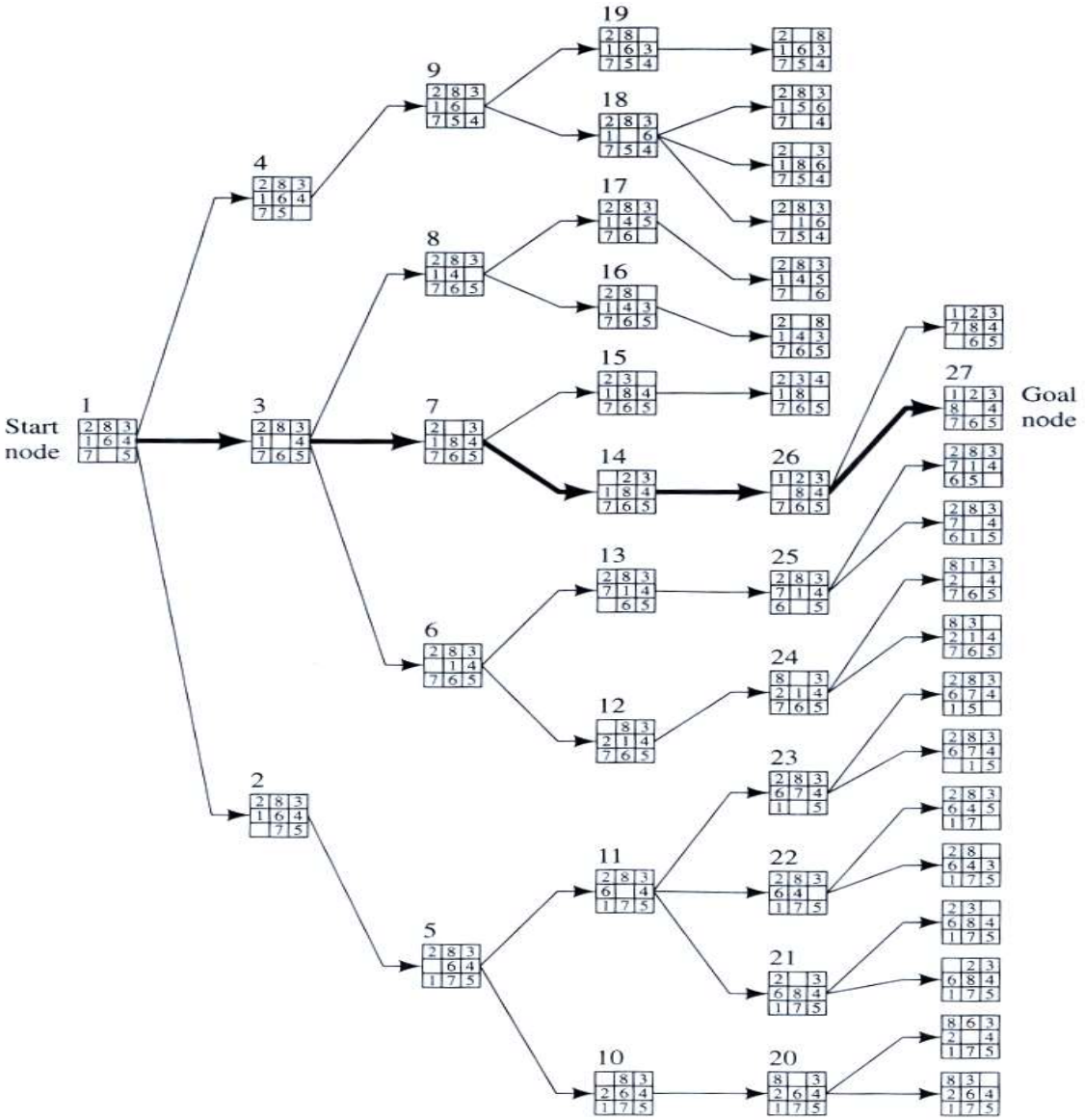
h1(starting state) = 8

h2 = the sum of the distances of the tiles from their goal positions.

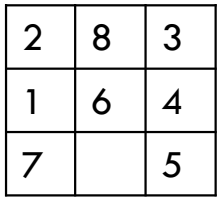
Because tiles cannot move along diagonals, the distance we will count is the sum of the horizontal and vertical distances. This is sometimes called the **city block distance** or **Manhattan distance**.

h2(starting state) = 3 (tile1) +1 (tile2) + 2 (tile3) + 2 (tile4) + 2 (tile5) + 3 (tile6) + 3 (tile7) + 2 (tile8) = 18

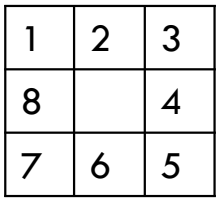
BFS SEARCH:


Start node
Goal node
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

A* EXAMPLE:

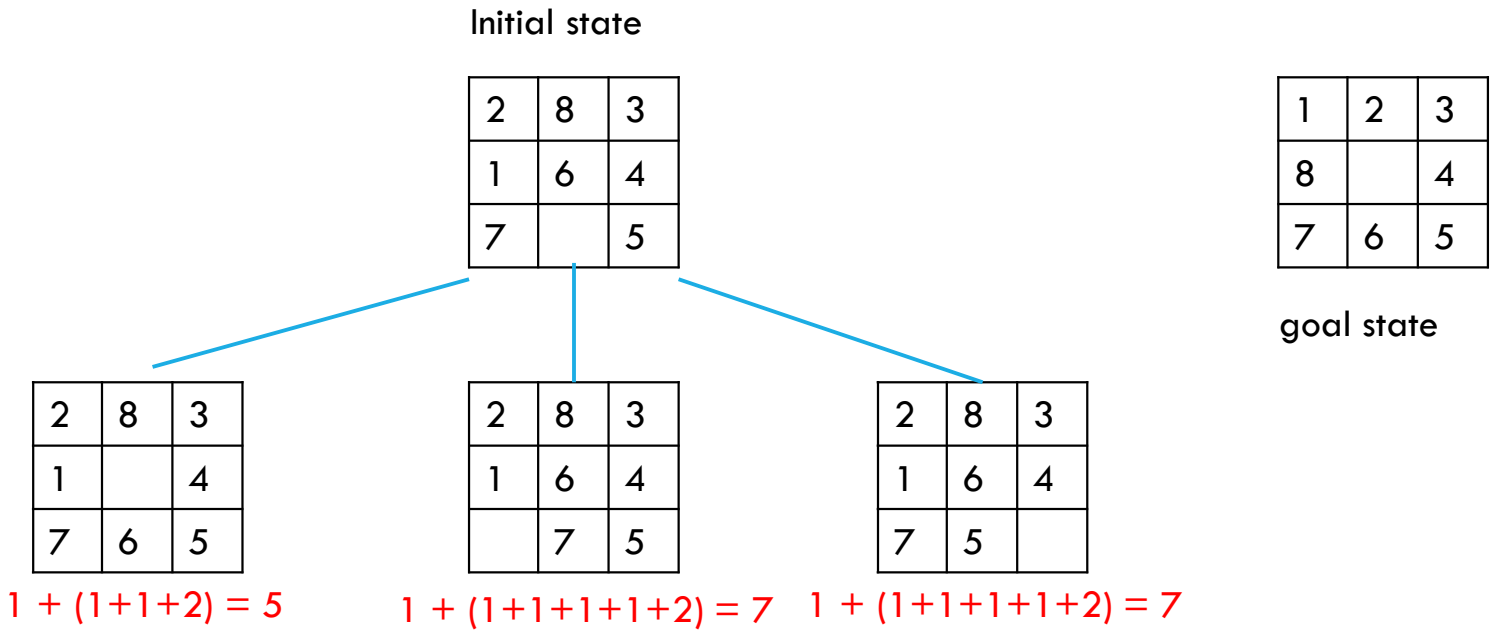


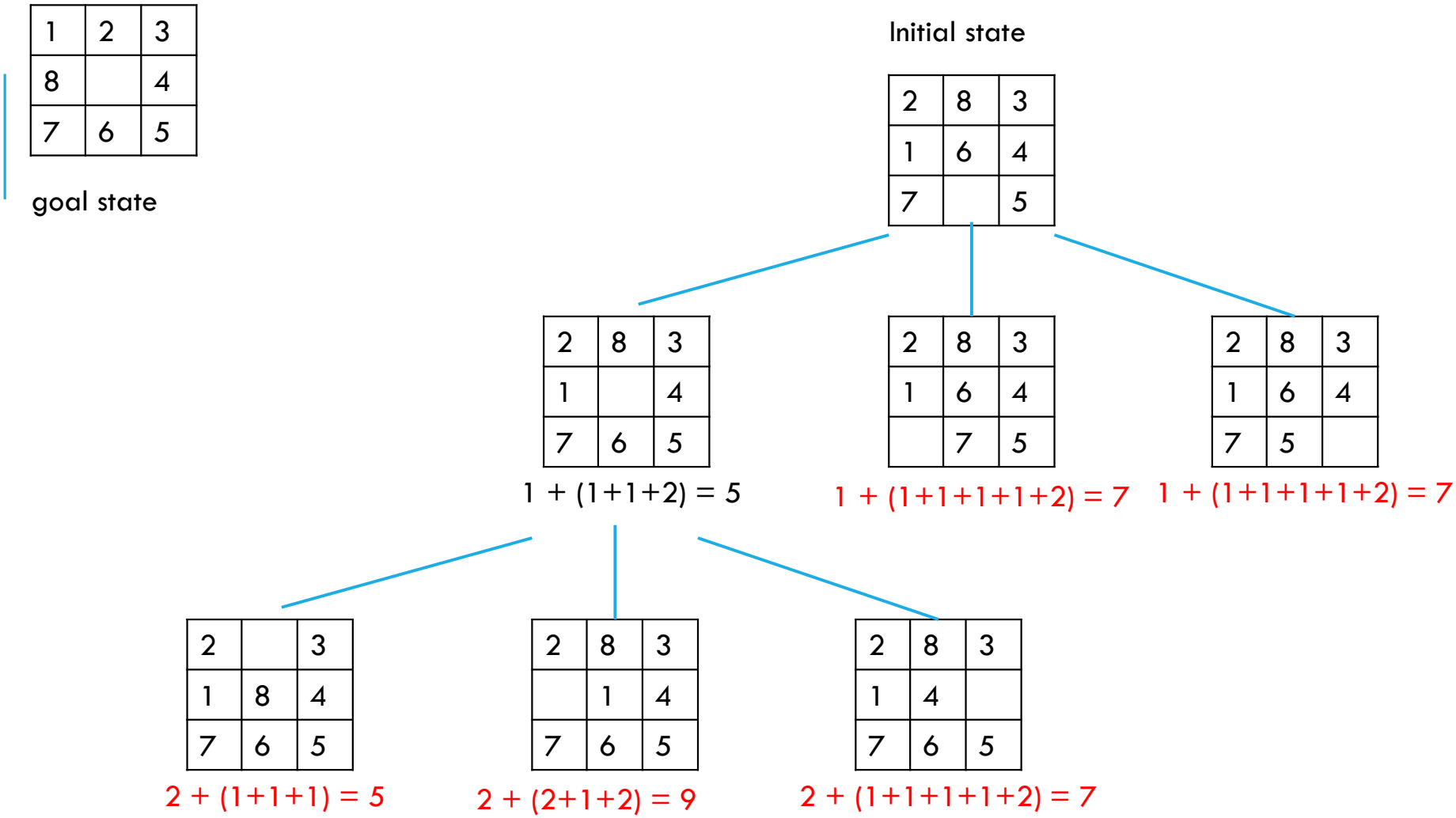
Initial state

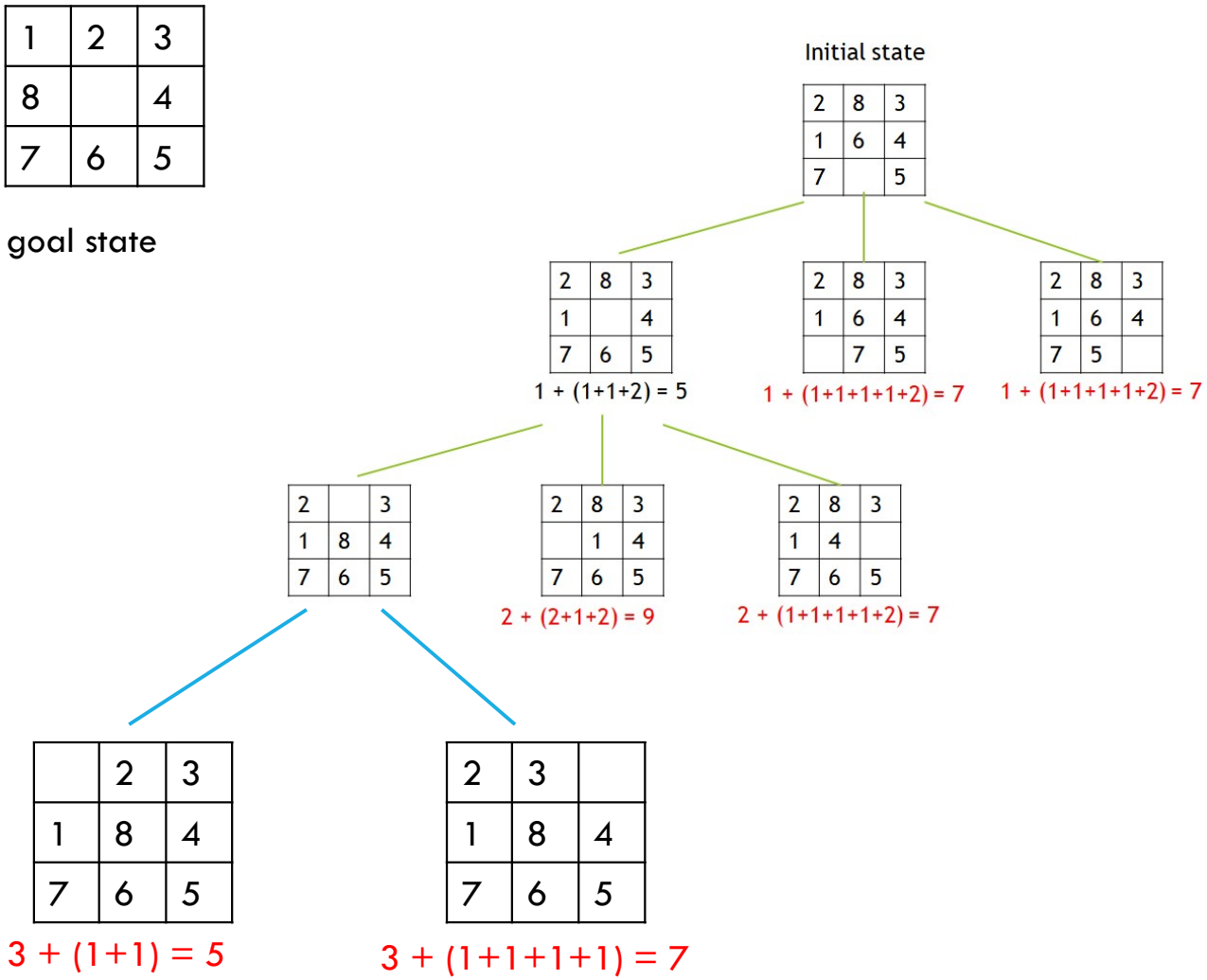


goal state

- Path cost, g = 1 for every move
- h(n)= given a node n, the sum of the distances of the tiles from their goal positions.

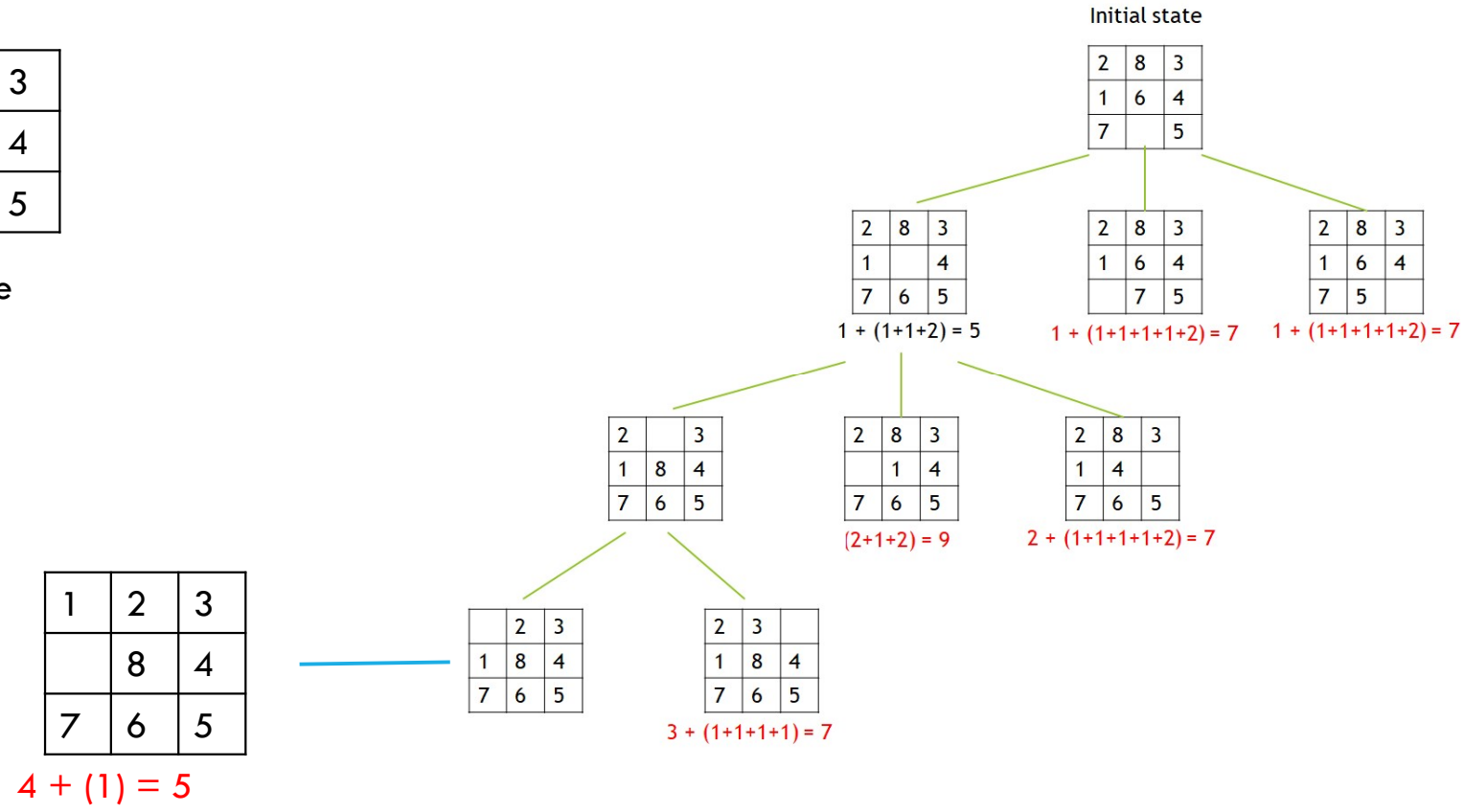

Initial state
2 8 3
1 6 4
7 5
1 2 3
8 4
7 6 5
goal state
2 8 3
1 4
7 6 5
1 + (1+1+2) = 5
2 8 3
1 6 4
7 5
1 + (1+1+1+1+2) = 7
2 8 3
1 6 4
7 5
1 + (1+1+1+1+2) = 7


1 2 3
8 4
7 6 5
goal state
Initial state
2 8 3
1 6 4
7 5
2 8 3
1 4
7 6 5
1 + (1+1+2) = 5
2 8 3
1 6 4
7 5
1 + (1+1+1+1+2) = 7
2 8 3
1 6 4
7 5
1 + (1+1+1+1+2) = 7
2 3
1 8 4
7 6 5
2 + (1+1+1) = 5
2 8 3
1 4
7 6 5
2 + (2+1+2) = 9
2 8 3
1 4
7 6 5
2 + (1+1+1+1+2) = 7


1 2 3
8 4
7 6 5
goal state
Initial state
2 8 3
1 6 4
7 5
2 8 3
1 4
7 6 5
1 + (1+1+2) = 5
2 8 3
1 6 4
7 5
1 + (1+1+1+1+2) = 7
2 8 3
1 6 4
7 5
1 + (1+1+1+1+2) = 7
2 3
1 8 4
7 6 5
2 + (2+1+2) = 9
2 8 3
1 4
7 6 5
2 + (1+1+1+1+2) = 7
2 3
1 8 4
7 6 5
3 + (1+1) = 5
2 3
1 8 4
7 6 5
3 + (1+1+1+1) = 7

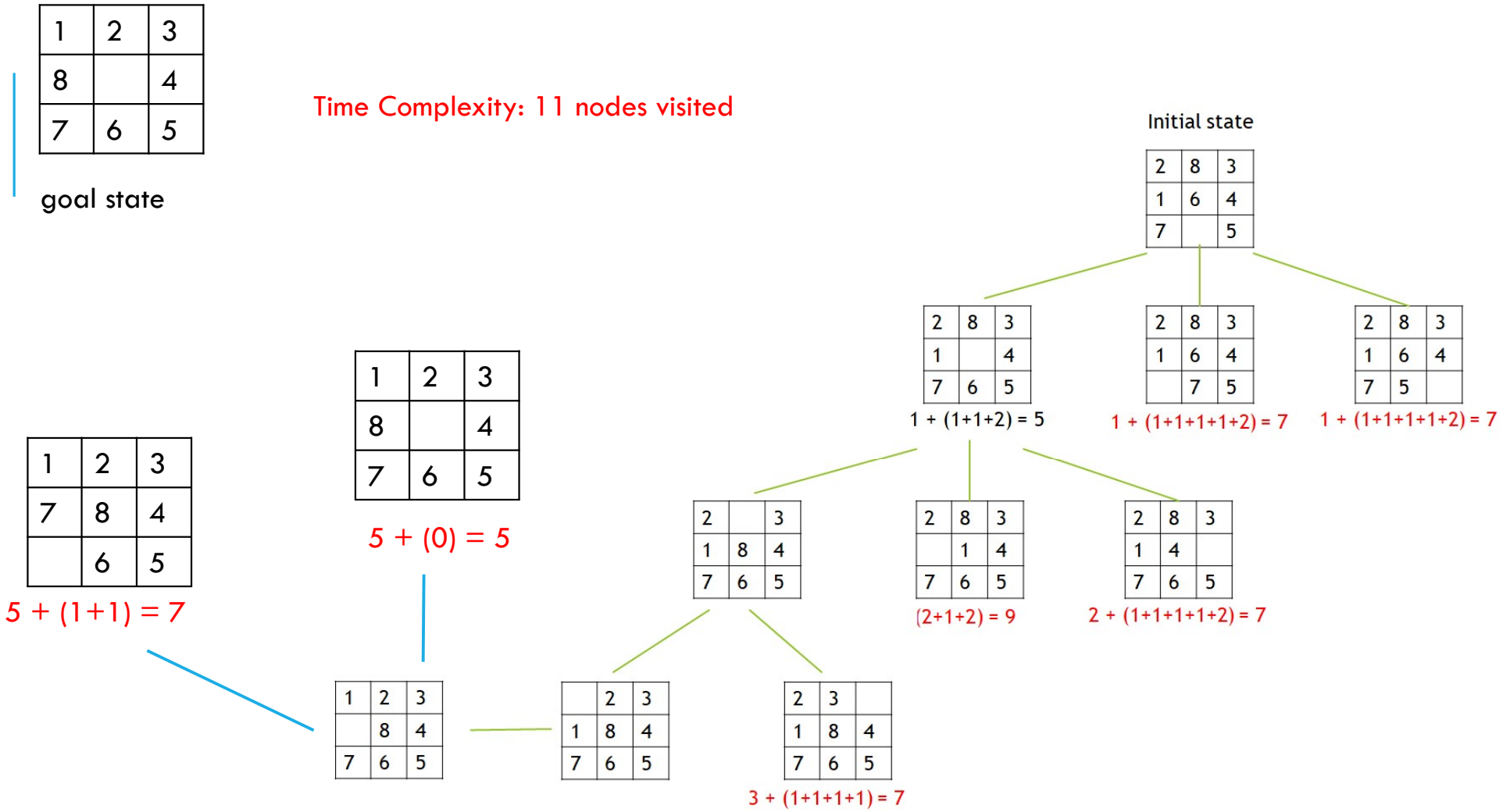
1	2	3
8		4
7	6	5

goal state



1	2	3
	8	4
7	6	5

4 + (1) = 5


goal state
Time Complexity: 11 nodes visited
Initial state
1 + (1+1+2) = 5
1 + (1+1+1+1+2) = 7
1 + (1+1+1+1+2) = 7
5 + (0) = 5
5 + (1+1) = 7
(2+1+2) = 9
2 + (1+1+1+1+2) = 7
3 + (1+1+1+1) = 7

BFS Time Complexity: 26 nodes visited

How many to visit for DFS?

How about UCS?