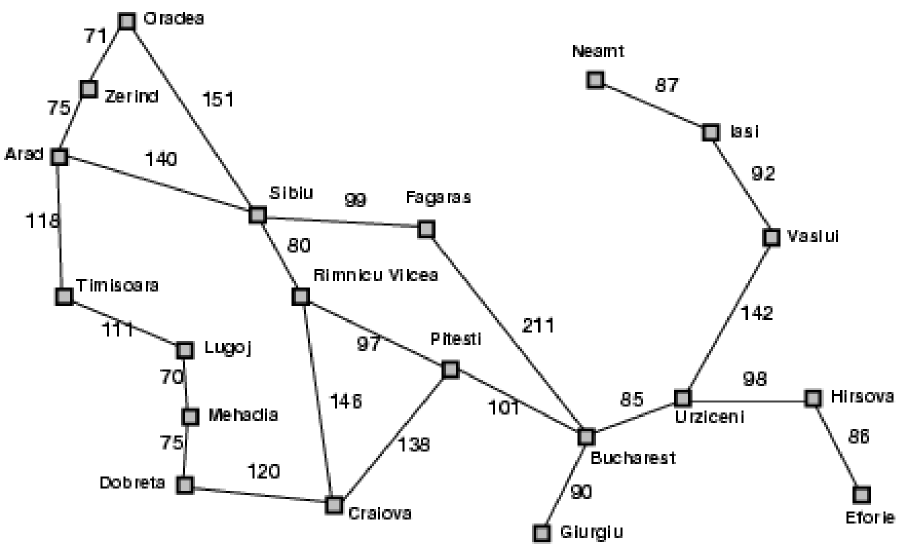
CS3243 Foundations of Artificial Intelligence (2005/2006 Semester 2)
Tutorial 3

1. Suppose we define a new heuristic function h_3 which is the average of h_1 and h_2, and another heuristic function h_4 which is the sum of h_1 and h_2. That is,

$$h_3 = \frac{h_1 + h_2}{2}$$
$$h_4 = h_1 + h_2$$

where h_1 and h_2 are defined as “the number of misplaced tiles”, and “the sum of the distances of the tiles from their goal positions”, respectively. Are h_3 and h_4 admissible? If admissible, compare their dominance with respect to h_1 and h_2.

2. Refer to the Figure below. Apply the best-first search algorithm to find a path from *Fagaras* to *Craiova*, using the following evaluation function *f(n)*:



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

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

where

$$h(n) = | h_{SLD}(Craiova) - h_{SLD}(n) |$$

and $h_{SLD}(n)$ is the straight-line distance from any city n to Bucharest given in Figure 4.1. Trace the best-first search algorithm by showing the series of search trees as each node is expanded, based on the TREE-SEARCH algorithm below. Prove that h(n) is an admissible heuristic.

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
   fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
   loop do
      if fringe is empty then return failure
      node ← REMOVE-FRONT(fringe)
      if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
      fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

3. (a) Prove that if a heuristic function is consistent, then it must be admissible.
(b) Give an example of an admissible heuristic function that is not consistent.

4. Assume that we have the following initial state and goal state for the 8-puzzle game. We will use h_1 defined as "the number of misplaced tiles" to evaluate each state.
(a) Apply the hill-climbing search algorithm in Figure 4.11 (reproduced here). Can the algorithm reach the goal state?

```
function HILL-CLIMBING( problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node
   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
      if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
```

(b) Identify a sequence of actions leading from the initial state to the goal state. Is it possible for simulated annealing to find such a solution?

1	2	8
	4	3
7	6	5

initial state

1	2	3
8		4
7	6	5

goal state