National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 3: Informed Search

Issued: February 25, 2020 Due: Week 5 in the tutorial class

Important Instructions:

- *Your solutions for this tutorial must be TYPE-WRITTEN.*
- *Make TWO copies of your solutions: one for you and one to be SUBMITTED TO THE TUTOR IN CLASS. Your submission in your respective tutorial class will be used to indicate your CLASS ATTENDANCE. Late submission will NOT be entertained.*
- *YOUR SOLUTION TO QUESTION* 5 *WILL BE GRADED for this tutorial.*
- *You may discuss the content of the questions with your classmates. But everyone should work out and write up ALL the solutions by yourself.*

1. Consider the 8-puzzle that we discussed in class. 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.

Solution: Since $h_1(n) \leq h_2(n)$ for all n,

$$h_3(n) = \frac{h_1(n) + h_2(n)}{2} \leq \frac{h_2(n) + h_2(n)}{2} = h_2(n) \leq h^*(n)$$

where the last inequality holds since h_2 is admissible. Hence, h_3 is admissible. Since for all n,

$$h_1(n) = \frac{h_1(n) + h_1(n)}{2} \leq \frac{h_1(n) + h_2(n)}{2} = h_3(n),$$

we have $h_1(n) \leq h_3(n) \leq h_2(n)$ for all n. That is, h_2 dominates h_3, and h_3 dominates h_1.

On the other hand, h_4 is not admissible. Consider a board n in which moving one tile will reach the goal. In this case, $h_1(n) = h_2(n) = h^*(n) = 1$, and

$$h_4(n) = h_1(n) + h_2(n) = 1 + 1 > h^*(n) .$$

2. Refer to the Figure 1 below. Apply the best-first search algorithm to find a path from Fagaras to Craiova, using the following evaluation function $f(n)$:

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

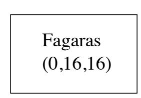
where $h(n) = |h_{SLD}(\text{Craiova}) - h_{SLD}(n)|$ and $h_{SLD}(n)$ is the straight-line distance from any city n to Bucharest given in Figure 3.22 of AIMA 3rd edition (reproduced in Fig. 1).

   (a) 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 (Fig. 2).

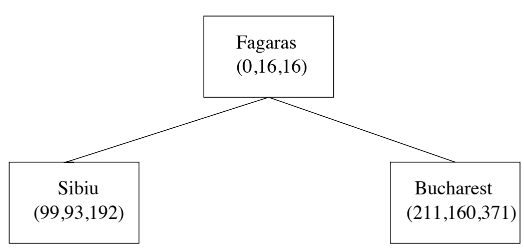
   (b) Prove that $h(n)$ is an admissible heuristic.

Solution:

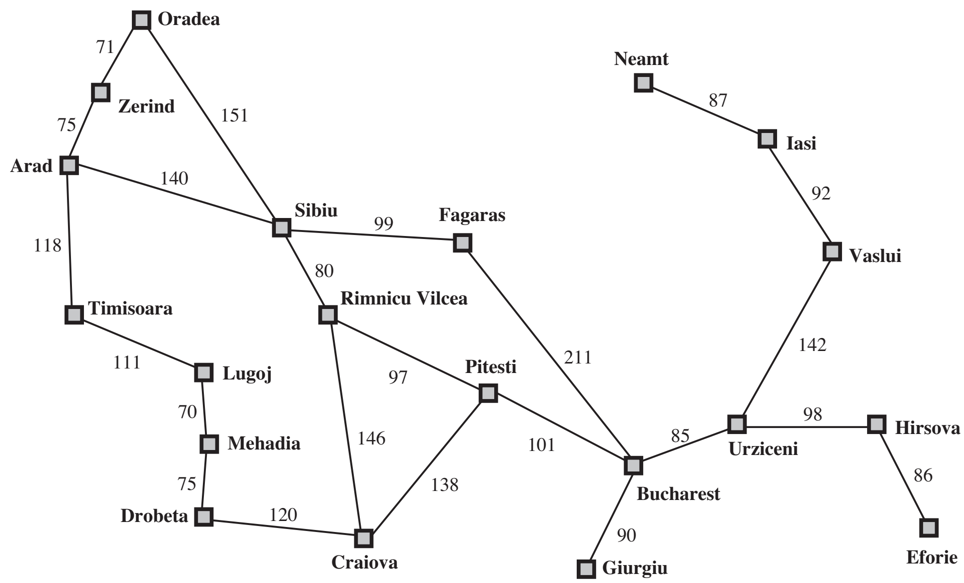
(a) The series of search trees, where the 3-tuple in each node denotes (g, h, f):



(I)



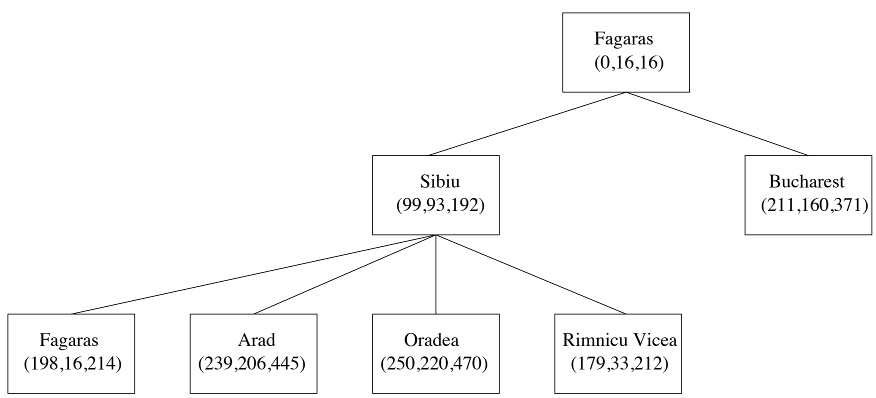
(II)



Values of h_{SLD} - straight-line distances to Bucharest

Arad	366
Bucharest	0
Craiova	160
Drobeta	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

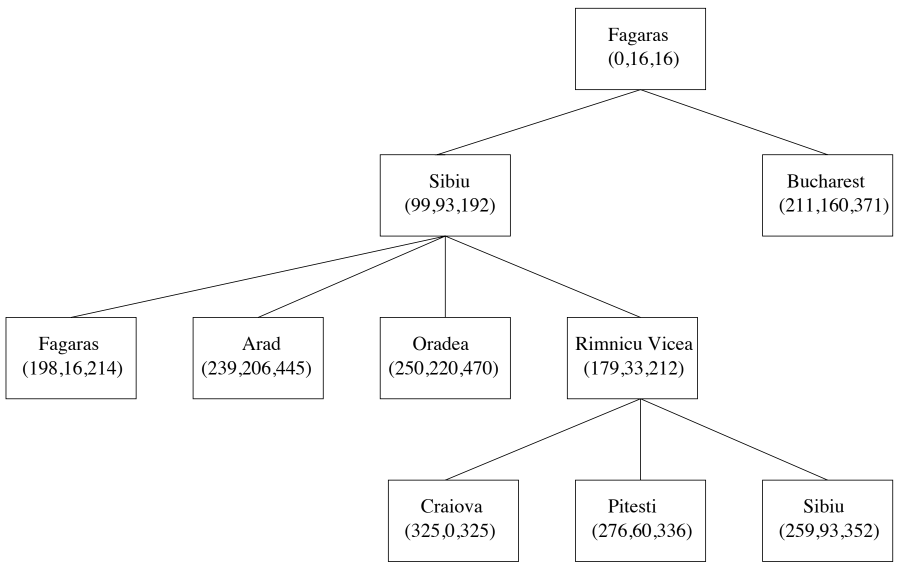
Figure 1: Graph of Romania.



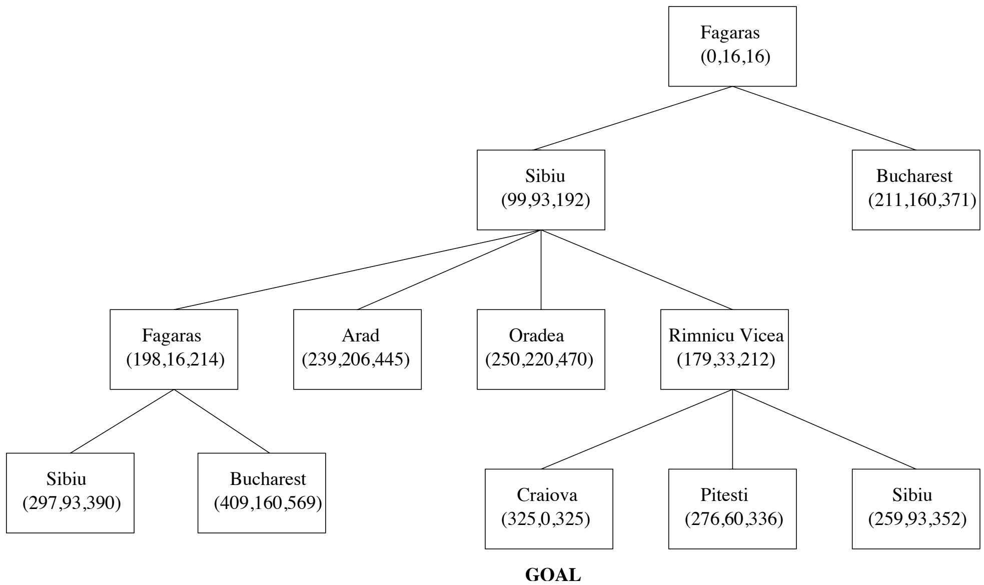
(III)

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

Figure 2: Tree search algorithm.



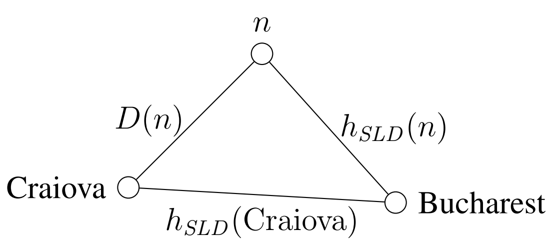
(IV)



(V)

Note that, in step IV, we need to expand Faragas even though we have reached the destination because getting to the destination is not sufficient. A^* finds the optimal solution.

(b) Now consider the following triangle:



Let $D(n)$ be the straight-line distance between n and Craiova. Using the Triangle Inequality we have that

$$D(n) + h_{SLD}(n) \geq h_{SLD}(\text{Craiova}) \Rightarrow D(n) \geq h_{SLD}(\text{Craiova}) - h_{SLD}(n)$$

and

$$D(n) + h_{SLD}(\text{Craiova}) \geq h_{SLD}(n) \Rightarrow D(n) \geq h_{SLD}(n) - h_{SLD}(\text{Craiova}).$$

We note that for any two numbers $a, b \in \mathbb{R}$

$$(c \geq a - b) \wedge (c \geq b - a) \Rightarrow c \geq |a - b|.$$

so

$$D(n) \geq |h_{SLD}(n) - h_{SLD}(\text{Craiova})| = h(n).$$

Now, let $h^*(n)$ be the true cost of reaching *Craiova* (**not Bucharest**, note that Craiova is the goal node now). We have $h^*(n) \geq D(n) \geq h(n)$, so $h(n)$ is admissible.

3. (a) Given that a heuristic h is such that $h(G) = 0$, where G is any goal state, prove that if h is consistent, then it must be admissible.

Solution:

The proof is by induction on $k(n)$, which denotes the number of actions required to reach the goal from a node n to the goal node G.

Base case ($k = 1$, i.e., the node n is one step from G):
Since the heuristic function h is consistent,

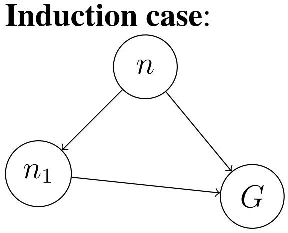
$$h(n) \leq c(n, a, G) + h(G)$$

Since $h(G) = 0$,

$$h(n) \leq c(n, a, G) = h^*(n)$$

Therefore, h is admissible.

Induction case:



Suppose that our assumption holds for every node that is $k - 1$ actions away from G, and let us observe a node n that is k actions away from G; that is, the least-actions optimal path from n to G has $k > 1$ steps. We write the optimal path from n to G as

$$n \to n_1 \to n_2 \to \cdots \to n_{k-1} \to G.$$

Since h is consistent, we have

$$h(n) \leq c(n, a, n_1) + h(n_1).$$

Now, note that since n_1 is on a least-cost path to G from n, we must have that the path $n_1 \to n_2 \to \cdots \to n_{k-1} \to G$ is a minimal-cost path from n_1 to G as well. By our induction hypothesis we have

$$h(n_1) \leq h^*(n_1)$$

Combining the two inequalities we have that

$$h(n) \leq c(n, a, n_1) + h^*(n_1)$$

Note that $h^*(n_1)$ is the cost of the optimal path from n_1 to G; by our previous observation (that $n_1 \to n_2 \to \dots n_{k-1} \to G$ is an optimal cost path from n_1 to G), we have that the cost of the optimal path from n to G — i.e. $h^*(n)$ — is exactly $c(n, a, n_1) + h^*(n_1)$, which concludes the proof.

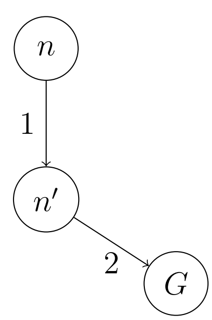
(b) Give an example of an admissible heuristic function that is not consistent.
Solution: An example of an admissible heuristic function that is not consistent is as follows:

$$\begin{aligned} h(n) =& 3 \\ h(n') =& 1 \\ h(G) =& 0 \end{aligned}$$

h is admissible since

$$\begin{aligned} h(n) \leq& h^*(n) = 1 + 2 = 3 \\ h(n') \leq& h^*(n') = 2 \end{aligned}$$

However, h is not consistent since $3 = h(n) > c(n, a, n') + h(n') = 1 + 1 = 2$.



(c) Is it possible for a heuristic to be consistent and yet not admissible? If not, prove it. If it is, define one such heuristic.
Solution: This is only possible if we allow heuristic functions for which $h(G) > 0$ (note that the solution to part (a) implies that if we have h consistent + $h(G) = 0$ it must be that h is admissible). Taking any consistent and admissible heuristic h, let us define $h'(n) = h(n) + 1$. It is easy to check that h' is still consistent, but it is not admissible since $h'(G) = 1$.

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.

1	2	8
	4	3
7	6	5

initial state

1	2	3
8		4
7	6	5

goal state

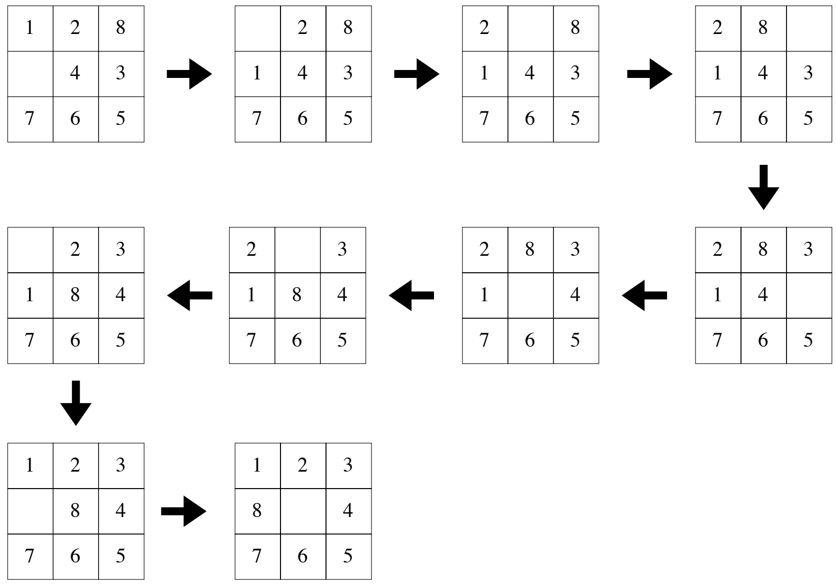
(a) Apply the hill-climbing search algorithm in Figure 4.2 of AIMA 3rd edition (reproduced below). Can the algorithm reach the goal state?

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

Figure 4.2 The hill-climbing search algorithm, which is the most basic local search technique. At each step the current node is replaced by the best neighbor; in this version, that means the neighbor with the highest VALUE, but if a heuristic cost estimate h is used, we would find the neighbor with the lowest h.

Solution: According to Figure 4.2 of AIMA 3rd edition, the hill-climbing search algorithm will move to a neighboring state only if the neighboring state is better. Since no successor of the initial state has a better h_1 value, the algorithm is stuck in a local optimum and will not reach the goal state.

(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?
Solution: The following sequence of actions leads from the initial state to the goal state:



Simulated annealing allows an action that leads to a worse value to be taken with some probability. Thus it is possible for simulated annealing to find the above solution.

5. You have learned before that A* using graph search is optimal if $h(n)$ is consistent. Does this optimality still hold if $h(n)$ is admissible but inconsistent? Using the graph in Figure 3, let us now show that A* using graph search returns the non-optimal solution path (S, B, G) from start node S to goal node G with an admissible but inconsistent $h(n)$. We assume that $h(G) = 0$.

   Give nonnegative integer values for $h(A)$ and $h(B)$ such that A* using graph search returns the non-optimal solution path (S, B, G) from S to G with an admissible but inconsistent $h(n)$, and tie-breaking is not needed in A*.

   Solution: First let us recall that A^* maintains a priority queue containing elements of

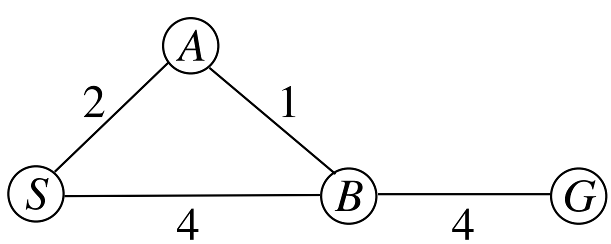


Figure 3: Graph.

the form $\langle n, f(n)\rangle$ in increasing order of $f(n)$ (ties broken arbitrarily). The key difference between tree search and graph search is that under graph search, once a node n gets explored, it will never be added to the priority queue again. There are several possible solutions one might consider. First, let us set

$$\begin{aligned} h(S) &= 7 \\ h(B) &= 0 \\ h(A) &= 3, 4 \text{ or } 5 \\ h(G) &= 0 \end{aligned}$$

Prove to yourself that this h is indeed admissible! Let's assume that $h(A) = 3$; what will A^* do in this case? It starts with a queue holding S with $f(S) = g(S) + h(S) = 0 + 7$: $[\langle S, 7\rangle]$, and it explores S. We have that

$$\begin{aligned} f(A) &= g(A) + h(A) = 2 + 3 = 5 \\ f(B) &= g(B) + h(B) = 4 + 0 = 4 \end{aligned}$$

Our queue looks like this now

$$[\langle B, 4\rangle, \langle A, 5\rangle]$$

so A^* selects B; it adds G to the queue next, with $f(G) = 8 + 0 = 8$, so our queue looks like this

$$[\langle A, 5\rangle, \langle G, 8\rangle].$$

It next explores A, *but does not add any of its neighbors again since this is graph search!* Thus, nothing gets added to the queue, G gets processed and we end up with a suboptimal solution.

What would tree search do on this heuristic? In order to understand more clearly what goes on, we add to our data structure the path that is taken to reach the node as well. Tree search would still select B first and then A, but having explored A it would add *all of its neighbors to the queue in order of* f, so after A gets explored we end up with a priority queue that looks like this:

$$[\langle B, 3, (S, A, B)\rangle, \langle G, 8, (S, B, G)\rangle, \langle S, 11, (S, A, S)\rangle].$$

Note that the entry $\langle S, 11, (S, A, S)\rangle$ comes from the cost of reaching S via A ($g(S) = 2 + 2 = 4$) plus $h(S) = 7$. We would next process B and again add its neighbors to the queue for

$$[\langle A, 7, (S, A, B, A)\rangle, \langle G, 7, (S, A, B, G)\rangle, \langle G, 8, (S, B, G)\rangle, \langle S, 11, (S, A, S)\rangle].$$

Note that $f(A) = g(A) + h(A) = (1 + 2 + 1) + 3 = 7$ and $f(G) = g(G) + h(G) = (3 + 4) + 0 = 7$. After we process A we (again) add its neighbors to have

$$[\langle B, 5, (S, A, B, A, B)\rangle, \langle G, 7, (S, A, B, G)\rangle, \langle G, 8, (S, B, G)\rangle,$$
$$\langle S, 11, (S, A, S)\rangle, \langle S, 14, (S, A, B, A, S)\rangle].$$

We process B again and obtain

$$[\langle G, 7, (S, A, B, G)\rangle, \langle G, 8, (S, B, G)\rangle, \langle A, 9, (S, A, B, A, B, A)\rangle,$$
$$\langle G, 9, (S, A, B, A, B, G)\rangle, \langle S, 11, (S, A, S)\rangle, \langle S, 14, (S, A, B, A, S)\rangle].$$

At this point G gets (finally) processed and we are done. We mention that in this example we broke ties in path costs arbitrarily (the result remains the same for other tie-breaking schemes). Other admissible inconsistent heuristics that work here include setting

$$h(B) = 1, h(A) = 4 \text{ or } 5$$
$$h(B) = 2, h(A) = 5.$$

All we need to ensure is that $h(B) < h(A) - 2$ to hold for h to be admissible but not consistent.