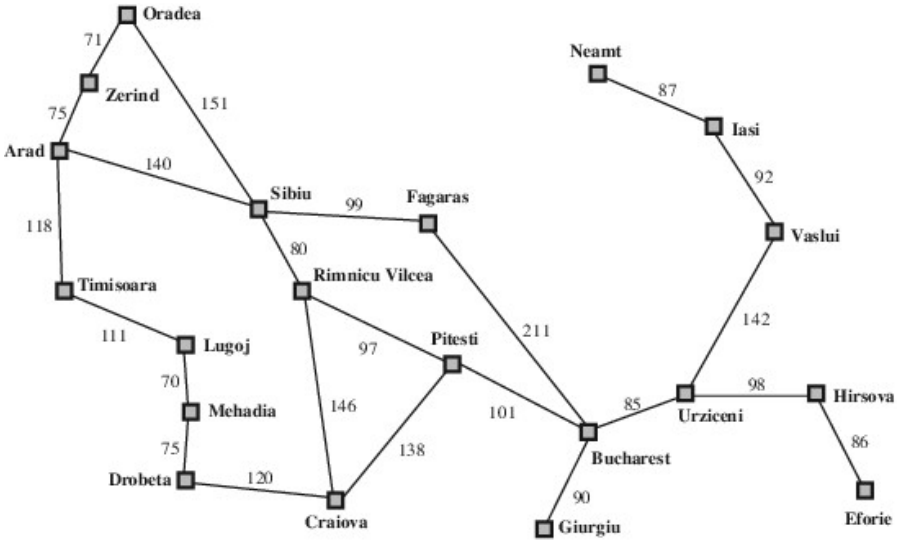
COSC1125/1127 Artificial Intelligence

School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 3
Heuristic and Adversarial Search

1. Consider the problem of getting from Arad to Bucharest in Romania and assume the straight line distance (SLD) heuristic will be used.

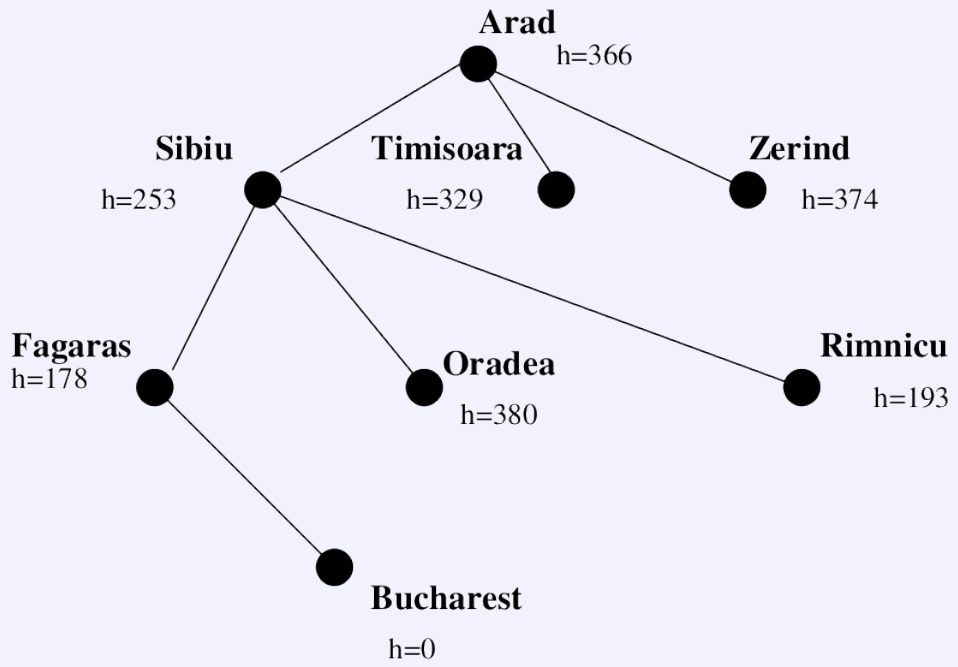


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

(a) Give the part of the search space that is realized in memory and the order of node expansion for:

i. Greedy search assuming that a list of states already visited is maintained.

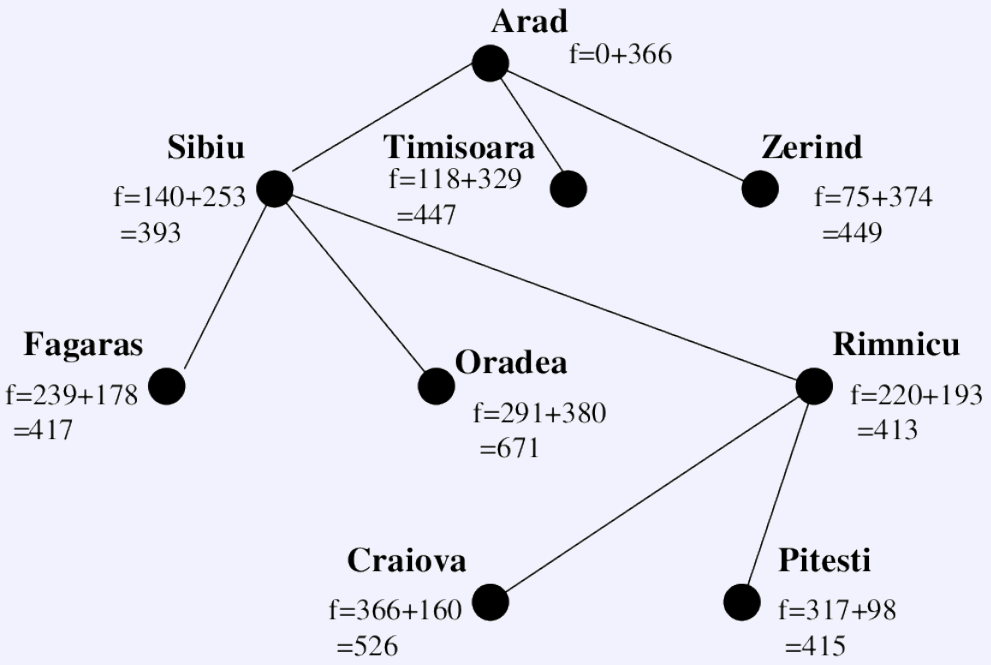
Answer



Greedy search for Bucharest, using the straight-line distance to Bucharest as the heuristic function h_{SLD}. Nodes are labelled with their h-values.
The order of nodes chosen for expansion: Arad, Sibiu, Fagaras, Bucharest.

ii. A* search assuming that a list of states already visited is maintained.

Answer

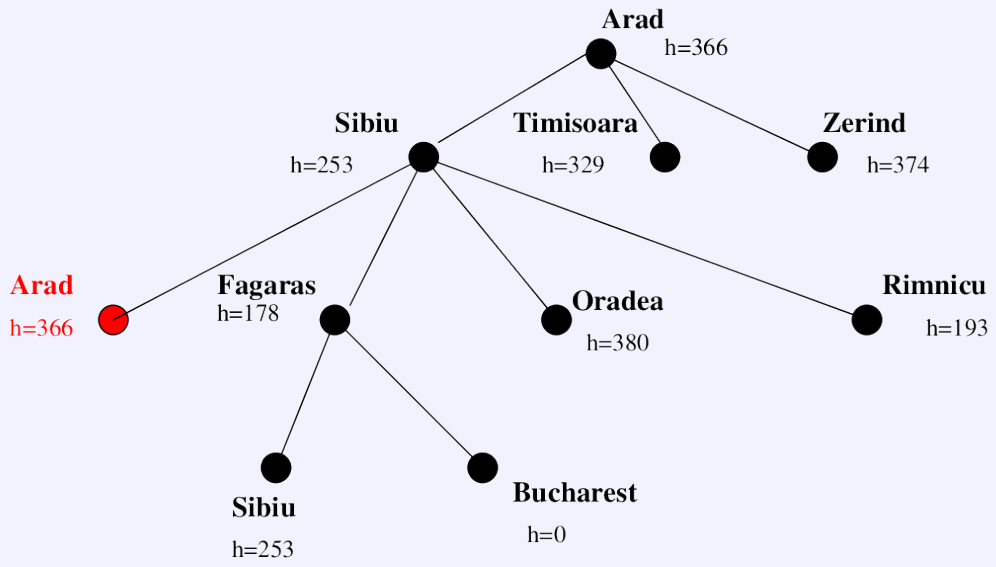


A* search for Bucharest. Nodes are labelled with $f = g + h$. the h values are the straight-line distances to Bucharest taken from the right column of Q1 Figure. The order of nodes chosen for expansion: Arad, Sibiu, Rimnicu, Pitesti.

(b) How would the above searches differ if the list of states already visited is NOT maintained?

Answer

If the list of states already visited is NOT maintained, then there should be an extra node "Arad" added to both the above figures. For example, for greedy search:



(c) How do the above searches perform for planing a trip from Iasi to Fagaras? You do not

need to do the detailed search (you are not given the heuristic function to Fagaras), but use the graphical map to extract the straight distance.

Answer

If the list of states visited is not maintained when appying the greedy search, the search will get stuck at "Neamt", because it gives the lowest h value (straight-line distance). If the visited states are checked and not allowed to be revisited, then both the greedy and A^* searches should be able to find Fagaras eventually, but they might perform differently (similar to the above two figures).

2. Suppose we run a greedy search algorithm with $h(n) = -g(n)$. What sort of search will result?

Answer

This heuristic tries to get as far away from the initial state as possible, but it has no concern with where the goal state is. This of course assumes that the algorithm tries to minimizes $f(n)$.

3. Suppose we run an A^* algorithm with $h(n) = 0$. What sort of search will result?

Answer

A^* with $h(n) = 0$ is the same as uniform cost search, which always expands the node with the smallest cost (or distance in the above example) to the initial state.

4. Explain why the set of states examined by A^* is a subset of those examined by breadth-first search when the cost of every step is always 1.

Answer

Breadth-first search can be seen as an A^* search with $g(n)$ being the depth of the search node n and $h(n) = 0$. So, in BFS, the decision for considering a state is based solely on its distance from the start state, so it will always be more "conservative" than any A*. Using any heuristic that is better than just zero, will cause the search to explore less nodes.

Refer to "heuristic domination" in the book (Section 4 in edition 2). Also, in section 4.2.3 of *Luger, G.F. Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Addison-Wesley* (edition 2002) one can find a *proof* that shows that better heuristics makes A* explore less nodes.

5. Is there a heuristic that would be useful for the missionaries and cannibals problem? The generalized missionaries and cannibals problem (n missionaries and n cannibals)?

Answer

One obvious heuristic measuring the goodness of a state is "number of people on the starting bank" - initially 6, goal 0 (or counting the number of people on the opposite riverbank). In fact, we can simplify the representation used in the M&C example used in the lecture slides:

(# cannibals on the left, # missionaries on the left, (1 if boat is at left, else 0))

Moves are shown as #cannibals + #missionaries in boat - F = forward, B = back:

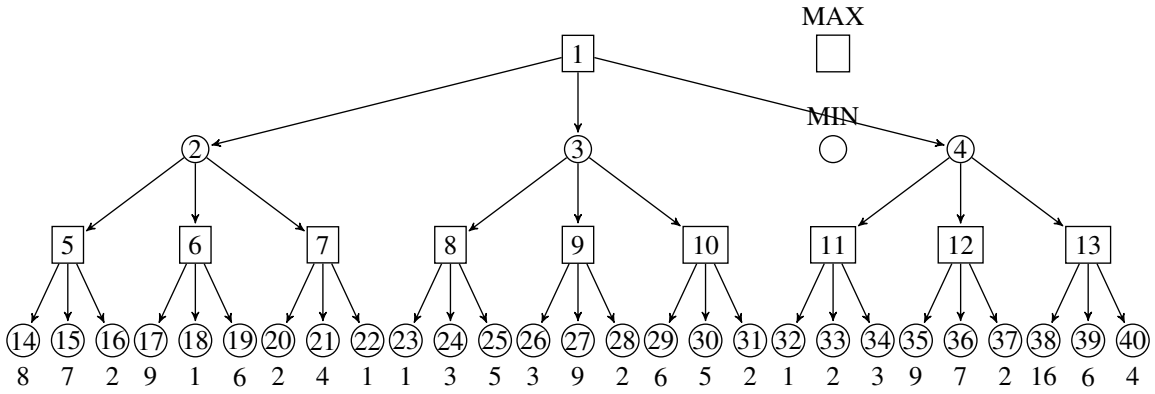
$(3,3,1) \underset{1+1F}{\rightarrow} (2,2,0) \underset{0+1B}{\rightarrow} (2,3,1) \underset{2+0F}{\rightarrow}$

$(0,3,0) \underset{1+0B}{\rightarrow} (1,3,1) \underset{0+2F}{\rightarrow} (1,1,0) \underset{1+1B}{\rightarrow}$

$(2,2,1) \underset{0+2F}{\rightarrow} (2,0,0) \underset{1+0B}{\rightarrow} (3,0,1) \underset{2+0F}{\rightarrow}$

$(1,0,0) \underset{1+0B}{\rightarrow} (2,0,1) \underset{2+0F}{\rightarrow} (0,0,0)$

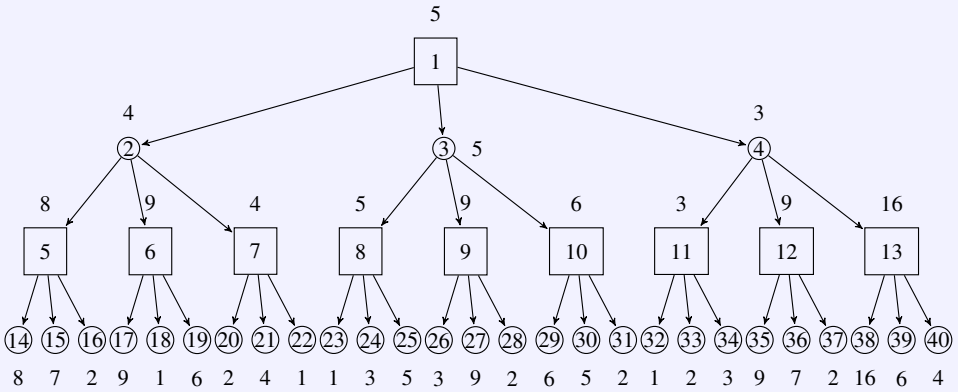
6. Consider the following game tree. Player MAX plays first and is represented with rectangles; MIN player is represented with circles. Numbers in each node are names used for convenience to refer to them (starting node is node 1). Finally, utility of leaf nodes are shown below them (e.g., the utility of node 21 is 4).



(a) Use mini-max to determine the best move for MAX.

Answer

MAX moves to the node 3 with value of 5:

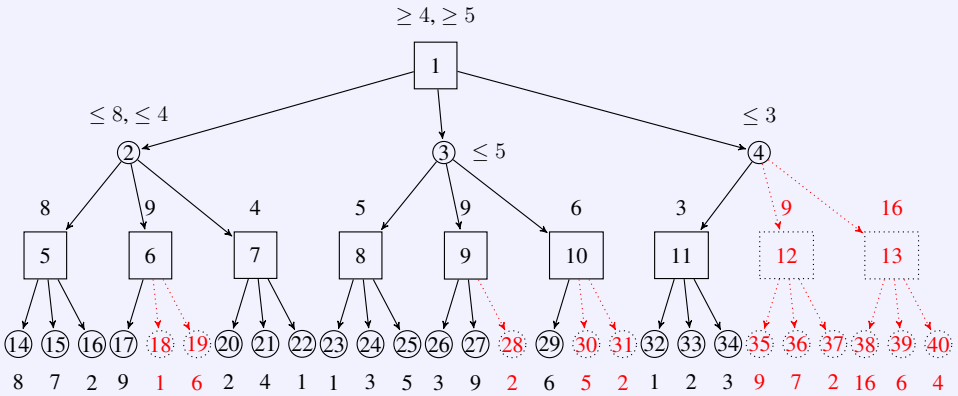


Try it in in http://kra.lc/projects/gamevisual/launch.php

(b) Which nodes will not be examined if the alpha-beta procedure is used?

Answer

The Figure below shows the nodes pruned by the alpha-beta procedure in red and dotted edges:



Try it in in http://kra.lc/projects/gamevisual/launch.php using $b = 3$, $d = 4$, and the following list of values: 8,7,2,9,1,6,2,4,1,1,3,5,3,9,2,6,5,2,1,2,3,9,7,2,16,6,4

(c) In which order will the nodes be examined by the alpha-beta procedure (assuming its depth-first implementation)?

Answer

14, 15, 16, 5, 17, 6, 20, 21, 22, 7, 2
23, 24, 25, 8, 26, 27, 9, 29, 10, 3,
32, 33, 34, 11, 4

(d) Did the alpha-beta procedure give the same best move as mini-max?

Answer

Yes.

7. Does A*'s search time always grow at least *exponentially* with the length of the optimal solution?

Answer

No. It will be exponential in the worst case but there are examples where it will not. For example, if $h(n) = h^*(n)$, that is, the heuristic is perfect, then it will be linear on the depth of the optimal solution as the search will go directly towards it. The better the heuristic, the more efficient the search will be.

8. If $h(\cdot)$ is admissible and s is the start node, how is $h(s)$ related to the cost of the solution found by A* search?

Answer

$h(s)$ will be less or equal than the solution found by A*. Because it is admissible, $h(s) \leq h^*(s)$, where $h^*(s)$ is the (real) cost of the optimal solution from s. Now, because $h(s)$ is admissible, we know that A* will find an optimal solution, that is, will find a solution with cost $h^(s)$.

9. Prove that if $h(n) = h^*(n)$ for all nodes n, then whenever A* expands a node x, x must lie on an optimal path to a goal.

Answer

There are many ways to prove this one. Here is one:

Suppose x is node that has been expanded but it is not in the optimal path to a goal. Suppose that $h(s_0) = h^*(s_0) = c^*$, that is, the cost of the optimal path from the initial state is c^*. First, if x_o is a node on an optimal path to the goal, we know that $f(x_o) = g(x_o) + h(x_o) = g(x_o) + h^*(x_o) = c^*$ So, if x was at any point expanded, it means that $f(x) < f(x_o)$ for some node x_o lying on an optimal path to the goal, that is, at some point of the search x was preferred over nodes lying on the optimal path. This means that $g(x) + h(x) < c^*$ and because $h(x) = h^*(x)$ we get that $g(x) + h^*(x) < c^*$. But this means that going thought node x is actually cheaper than c^* and hence $f(s_0) = h(s_0) = h^*(s_0) = c^* < c^*$, a contradiction. Thus, that node x expanded but not lying on an optimal path to the goal cannot exist.

10. Prove that if a heuristic is consistent, then it is also admissible. What about the converse?

Answer

(This is not the proof but a strong hint how to do it)

By induction on the length of the optimal path. The converse is not true (a heuristic can be admissible but not consistent): find a counter-example (hint: consider a graph that is a path from the initial state to the goal state).

11. Prove that A* without remembering nodes (i.e., without a closed list) is optimal when using an admissible heuristic.

Answer

The main idea of the proof is that at any point in the tree search ((i.e., not remembering nodes)) if a non-optimal goal node G is in the open list and a node N in the path to an optimal goal node G^* is in the open list, then N will be expanded first. This goes all the way towards G^* which ultimately will also be expanded before non-optimal node G.

See a sketchy but quite detailed proof that I did here. You can now make it step by step and reconstruct the reasoning!

It is important that we are relying on TREE SEARCH (i.e., not remembering nodes), so that if we get to a previously visited state x in a cheaper way, we keep and process it, as this could be the way to the goal!. (If we use a CLOSED list, i.e., GRAPH SEARCH, then x would be ignored and hence optimality lost (unless heuristic is monotonic/consistent)