

Artificial Intelligence

Take home Quiz 01

NAME:

DATE:

This homework quiz is meant to help you prepare for the mid-term exams. The mid-term will cover concepts from Agents, Environments, Search (both informed and uninformed), Adversarial Search and Constraint Satisfaction Problems..

1. Prove each of the following statements, or give a counterexample (6 points)

a. Breadth-first search is a special case of uniform-cost search.

b. Depth-first search is a special case of best-first tree search.

c. Uniform-cost search is a special case of A* search.

2. Decide whether the following statements are true or false. If true, explain why. If false, give a contradicting example. Recall that B is the average branching factor and L is the length of the shortest path from start to goal. (8 points)

a) Bi-directional BFS is always faster than BFS when $B \geq 2$ and $L \geq 4$.

b) A* search always expands fewer nodes than DFS does.

c) For any search space, there is always an admissible and consistent A* heuristic.

d) IDA* does not need a priority queue as in A*, but can use the program stack in a recursive implementation as in DFS.

3. Tree search algorithm

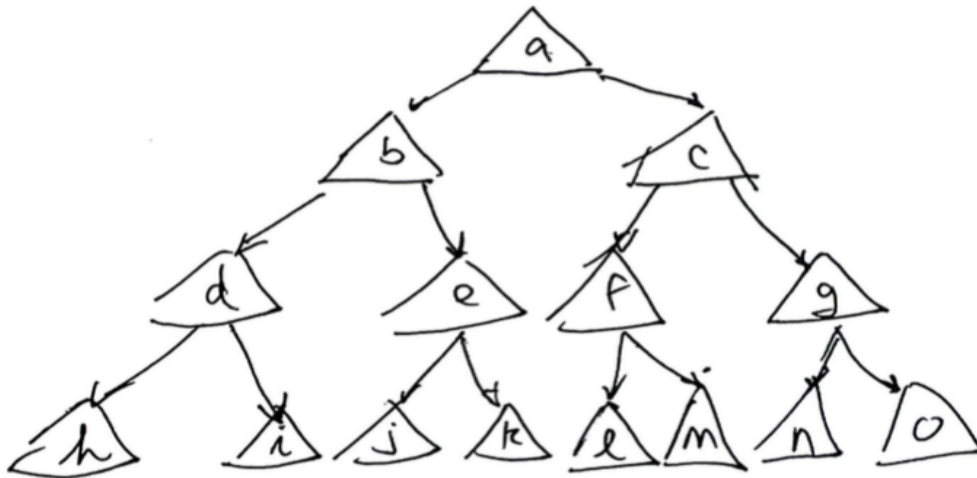
Tree search algorithm is applicable to searches where there is no worry about re-visiting prior states. It is simpler than graph-search because there is no need to maintain the closed set (of previously visited states). Use tree search to solve the following problems

NOTES

“The intuitive idea behind generic search algorithm, given a graph, a set of start nodes, and a set of goal nodes, is to incrementally explore paths from the start nodes. This is done by maintaining a **frontier** (or **fringe**) of paths from the start node that have been explored. The frontier contains all of the paths that could form initial segments of paths from a start node to a goal node. Initially, the frontier contains trivial paths containing no arcs from the start nodes. As the search proceeds, the frontier expands into the unexplored nodes until a goal node is encountered. To expand the frontier, the searcher selects and removes a path from the frontier, extends the path with each arc leaving the last node, and adds these new paths to the frontier. A search strategy defines which element of the frontier is selected at each step.”

Refer to the reference books (Artificial Intelligence: Foundations of Computation Agents - Chapter 3 (It was in our reading materials))

A. Breadth-first search using FIFO queue



Frontier (fill from L-R):

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Using the tree search algorithm, perform the search for **goal state o** starting from initial state *a*.

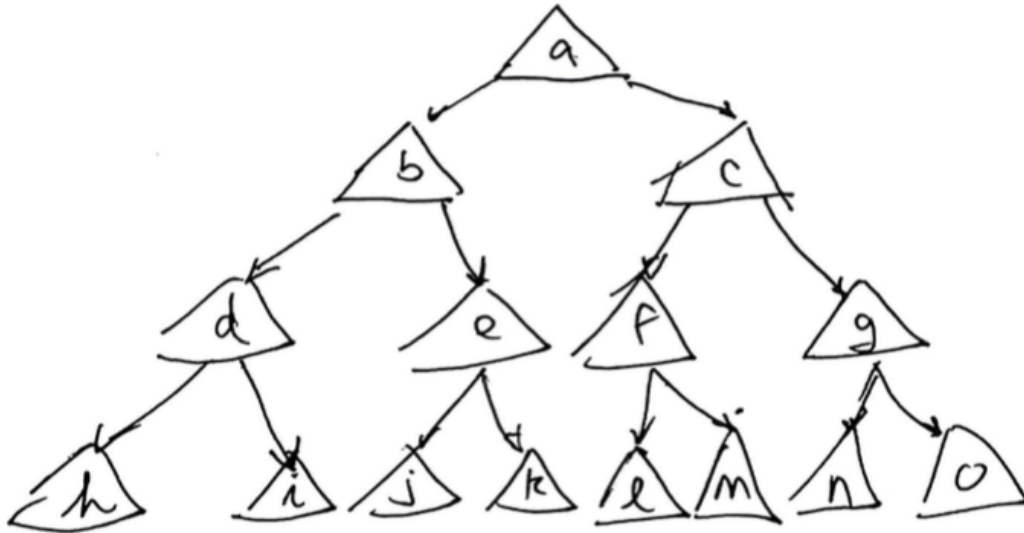
Expand new state/action pairs from left to right. Draw nodes in the Fringe as they are expanded. When they get removed for goal-testing, cross them out. (4 points)

Draw nodes as a circle with a letter inside. This is to visually distinguish them from states (triangles).

Treat the Fringe as a FIFO queue. This should produce breadth-first search.

- What goal node is returned by the algorithm? (2 points)
- How many states were expanded in total? (2 points)
- At most, how many nodes were actively in the fringe at one time during the algorithm? (2 points)

B) Depth-first search using LIFO stack



Frontier (fill from L-R):

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Using the tree search algorithm, perform the search for **goal state h** starting from initial state *a*.

Expand new state/action pairs from left to right. Draw nodes in the Fringe as they are expanded. When they get removed for goal-testing, cross them out. (4 points)

Draw nodes as a circle with a letter inside. This is to visually distinguish them from states (triangles).

Treat the Fringe as a LIFO stack. This should produce depth-first search.

When done:

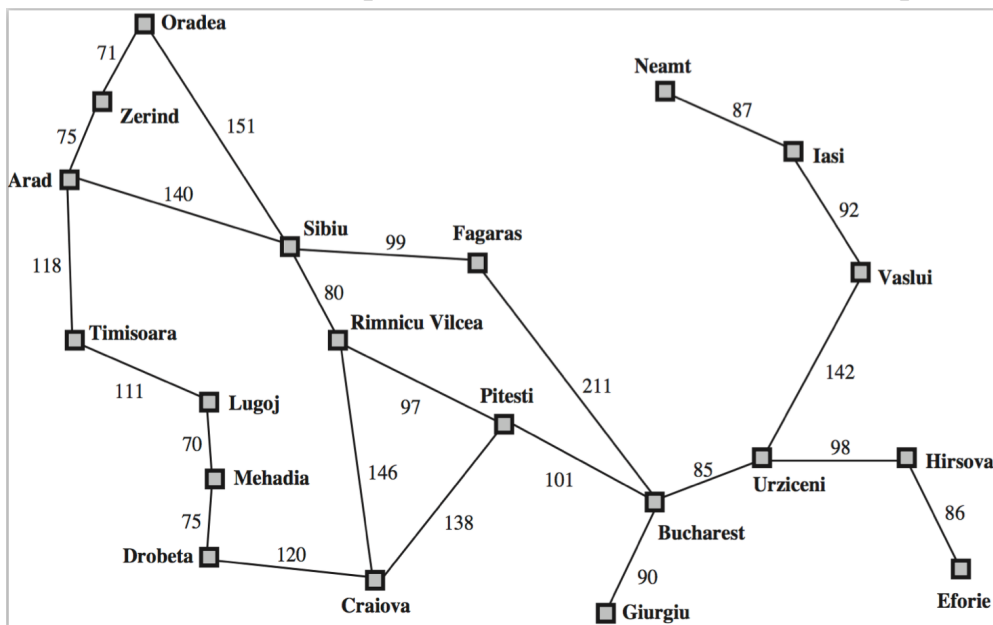
a. What goal node is returned by the algorithm? (2 points)

b. How many states were expanded in total? (2 points)

c. At most, how many nodes were actively in the fringe at one time during the algorithm? (2 points)

4. Uniform cost search

Uniform cost search (UCS) is a breadth-first search where the node to be expanded is the one with the lowest accumulated path cost in the frontier. Let's use it to search for the best path from Arad to Bucharest. "Best path" is defined

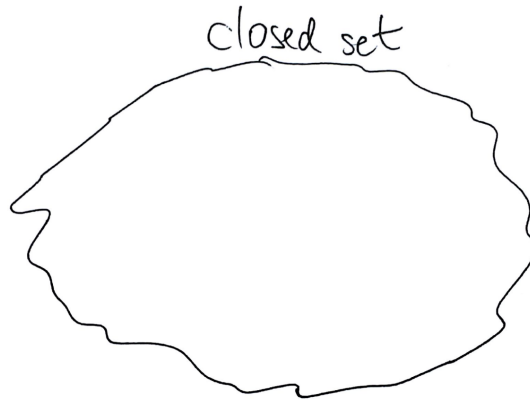


as shortest distance.

We will use **Graph Search**

```
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
  closed ← an empty set
  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 node
    if STATE[node] is not in closed then
      add STATE[node] to closed
      for child-node in EXPAND(STATE[node], problem) do
        fringe ← INSERT(child-node, fringe)
      end
    end
  end
```

Uniform Cost Search using priority queue



Frontier (fill from L-R):

--	--	--	--	--	--	--	--	--	--	--	--

Start state is Arad.

Goal state is Bucharest.

Draw nodes in the Fringe as they are expanded. When they get removed for goal-testing, cross them out and put them in the closed set as states.

Draw nodes as a circle with a letter inside, and include the total path cost. (4 points)

Treat the Fringe as a priority queue. When performing "Remove-Front," take the node with the smallest accumulated cost.

a. What path is returned by the algorithm? (2 points)

b. What is its path cost (total distance)? (2 points)

c. What did you observe that was interesting about the algorithm? (2 points)

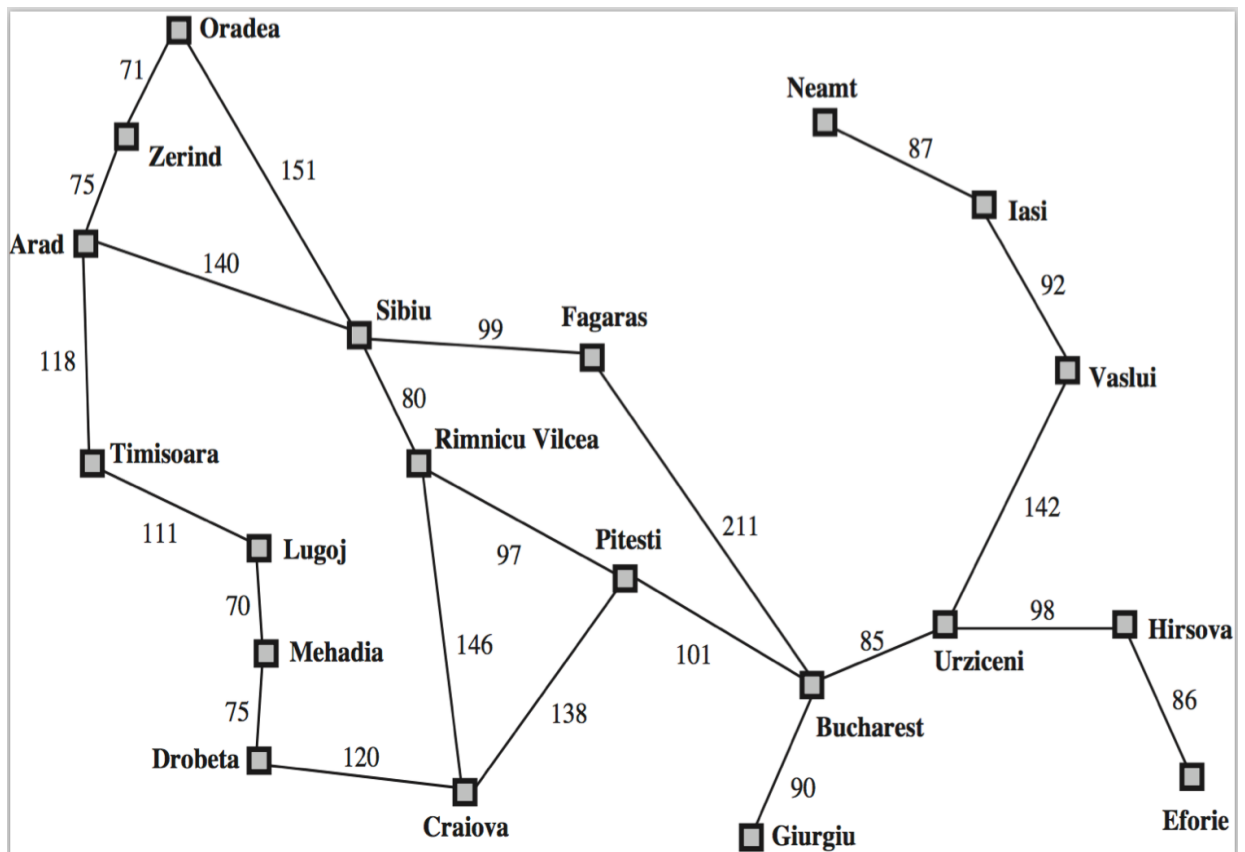
5. A* Search

A* search (pronounced “a star”) is a breath-first search where the node to be expanded is the one with the best (lowest) value of (accumulated path cost in the frontier) + (heuristic distance to the goal). In other words, the priority function f for a node n is:

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

The heuristic function we will use is straight line distance (*hsl*d), which is guaranteed to be admissible for Euclidean spaces.

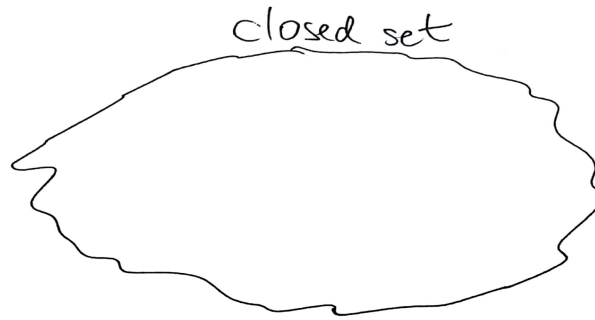
The actual path cost may be equal to or greater than this heuristic value, but it cannot be less than it.



Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Perform A* search using the priority function $f(n) = g(h) + h(n)$. Start state is Arad. Goal state is Bucharest. Draw nodes in the Fringe as they are expanded. Include their f value at the point of insertion. (4 points)

When nodes get removed for goal-testing, cross them out and put them in the closed set as states.



Treat the Fringe as a priority queue. When performing “Remove-Front,” take the node with the smallest $f=g+h$ cost.

Frontier (fill from L–R):

--	--	--	--	--	--	--	--	--	--	--	--

a. What path is returned by the algorithm? (2 points)

- b. What is its path cost (total distance)? (2 points)
- c. What did you observe that was interesting about the algorithm, compared to UCS? (2 points)

6. More A*

- a) Suppose we have two admissible heuristic functions, h_1 and h_2 , where for all search states s , $h_1(s) < h_2(s)$. In other words:

$$\forall(s): h_1(s) < h_2(s)$$

What will be the impact on the search process? Specifically:

- i) Optimality: Will the search return the best solution when using h_1 ? What about when using h_2 ? (3 points)
- ii) Efficiency: Will more nodes be expanded when using h_1 or h_2 ? The same? (3 points)

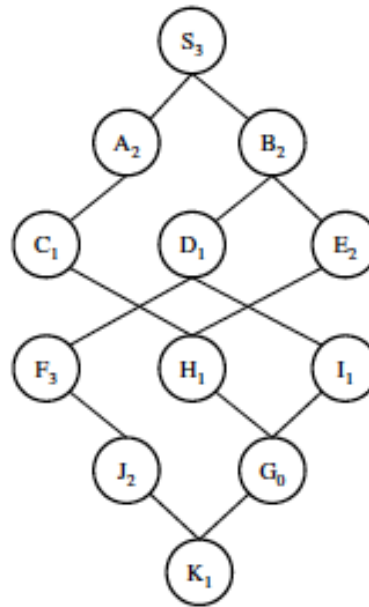
b) Suppose we use a heuristic h_3 that is *not* admissible (i.e., it produces values that are larger than the actual best path to the goal). What will be the implications of this if only one solution exists, and there is only one path to it. Consider this in terms of:

i) Optimality: Will h_3 find the solution? Why or why not? (2 points)

ii) Efficiency: Will h_3 cause more, fewer, or the same number of nodes to be expanded vs. an admissible heuristic? Why? (3 points)

7. Consider the problem of moving a knight on a 3x4 board, with start and goal states labeled as S and G in figure 3. The search space can be translated into the following graph. The letter in each node is its name and the subscript digit is the heuristic value. All transitions have cost 1.

S ₃	H ₁	D ₁	K ₁
I ₁	J ₂	A ₂	E ₂
C ₁	B ₂	G ₀	F ₃



Make the following assumptions:

- All the algorithms do not generate paths with loops.
- Nodes are selected in alphabetical order when the algorithm finds a tie.
- A node is visited when that node is at the front of the search queue.

Write the sequence of nodes in the order visited by the specified methods.

Note: You may find it useful to draw the search tree corresponding to the graph above.

(a) Depth-first search. (4 points)

(b) Breadth-first search. (4 points)

(c) A* search. In addition to the sequence of nodes visited, you are also to attach the estimated total cost $f(s)=g(s)+h(s)$ for each node visited and return the final path found and its length. (4points)