DAT405 Introduction to Data Science and AI, 2019 –2020,

Assignment 8:

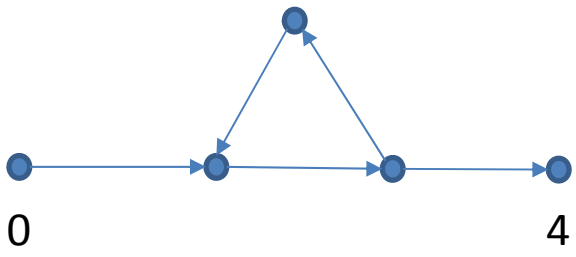
Answer to each question with enough discussion.

1) The branching factor *d* of a directed graph is the maximum number of children (outer degree) of a node in the graph. Suppose that the shortest path between the initial state and a goal is of length *r.*

a) What is the maximum number of BFS iterations required to reach the solution in terms of *d* and *r*?

b) Suppose that storing each node requires one unit of memory. Hence, storing a path with *k* nodes requires *k* units of memory. What is the maximum amount of memory required for BFS in terms of *d* and *r*?

2) Take the following graph where 0 and 4 are respectively the initial and the goal states. The other nodes are to be labeled by 1,2 and 3.



Suppose that in case of a tie, the DFS method takes the path with the smallest label of the last node. Show that there exists a labeling of these three nodes, where DFS will never reach to the goal! What can be added to DFS to avoid this situation?

3) This question investigates using graph searching to design video presentations. Suppose there exists a database of video segments, together with their length in seconds and the topics covered, set up as follows:

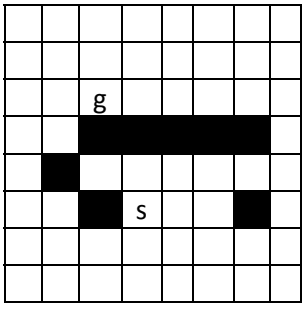
Segment	Length	Topics covered
seg0	10	[welcome]
seg1	30	[skiing, views]
seg2	50	[welcome, artificial_intelligence, robots]
seg3	40	[graphics, dragons]
seg4	50	[skiing, robots]

We define a node as a pair ⟨*To_Cover, Segs*⟩

- where *Segs* is a list of segments that must be in the presentation and *To_Cover* is a list of topics that also must be covered, but is not covered yet by *Segs*. Hence for a valid node, none of the segments in *Segs* cover any of the topics in *To_Cover*.
- The children of a node are obtained by first selecting a topic from *To_Cover*, adding a segment to *Segs* which covers this topic and finally deleting any topic from *To_Cover covered by this segment*. For example, the children of the node *⟨[welcome, robots],[]⟩* , assuming that *welcome* was selected, are *⟨[],[seg2]⟩* and *⟨[robotos],[seg0]⟩* .
- Thus, each arc adds exactly one segment but can cover one or more topics. Suppose that the cost of the arc is equal to the time of the segment added.
- The goal is to design a presentation that covers all of the topics in a list named *MustCover*. The starting node is *⟨MustCover,[]⟩* , and the goal nodes are of the form *⟨[], Presentation⟩* for some list *Presentation*. The cost of the path from a start node to a goal node is the time of the entire presentation. Thus, an optimal presentation is a shortest presentation that covers all the topics in *MustCover*.
    - (a) Suppose that the goal is to cover the topics [*welcome,skiing,robots*] and the algorithm always selects the leftmost topic to find the neighbors for each node. Draw (by hand) the search space as a tree expanded for a lowest-cost-first search until the first solution is found. This should show all nodes expanded, which node is a goal node, and the frontier when the goal was found.
    - (b) Give a non-trivial heuristic function h that is admissible. [$h(n)=0$ for all n is the trivial heuristic function.]

4) Consider the problem of finding a path in the grid shown below from the position s to the position g. A piece can move on the grid horizontally or vertically, one square at a time. No step may be made into a forbidden shaded area. Consider the Manhattan distance as the heuristic.

a) Write the paths stored and selected in the first five iterations of the A* algorithm, assuming that in the case of tie the algorithm takes the shortest path with the smallest lexicographical index.



b) Try to solve this problem by the software in http://qiao.github.io/PathFinding.js/visual/ Use Manhattan distance, no diagonal step and compare A*, BFS and Best-First-Search. Write a short description about your observation. How does each of these methods reach the solution? Why? Which one is faster?