Homework 1

CSCI 4511W Spring 2018

January 31, 2018

Instructor: Dr.Maria Gini

Joowon Kim(kimx4342)

---

1. A robot has to deliver three identical packages to locations A,B and C in an office environment. The robot starts in location S holding the packages. The environment is a grid of squares, some of which are free (so the robot can move into them) and some of which are occupied (by walls, doors, etc.) The robot can move in the horizontal or vertical direction into neighboring squares, one square at a time, and can pick up and drop packages if they are in the same square as robot.

   (a) Specify the state space representation of the problem by specifying the states, the initial state, the goal test, the actions, and the cost of the actions. Be precise and consistent

   The state is determined by location and the fact whether there are packages that needed to be delivered or not. There are not any exact mentions about size of the states The initial state is S where the robot holds the package at start point. The goal test is that whether the packages are delivered in all of 3 locations A, B and C. The action is moving left, right, up, down and pick up and drop. Each basic step costs is 1 and it could be differ depending on how many packages the robot is holding. The cost for robot holding 3 packages at once and holding 1 package at a time has different cost.

2. This question is on using the proper search algorithm for each problem. The algorithms to compare are:

   1. depth-first
   2. breadth-first
   3. depth-limited depth first

4. iterative deepening depth first

5. bidirectional

For each of those algorithms describe briefly an example of a problem for which the algorithm is well suited and one for which it is not. Be specific and precise. For example, uniform cost is appropriate to find a path in a graph, where each node represents a location and the cost of each arc is the distance between the two locations connected by the arc. Uniform cost is not useful for problems where all the actions have the same cost, and for problems that have a complete state formulation.

(a) depth-first

Depth-first search is suited for solving problems that have only one solution because if the goal node is in the deep level in a searching tree, the solution can be obtained quickly. But DFS is not suitable for problems such as finding the shortest path that has many paths to the goal.

(b) breadth-first

Breadth-first search is suited for finding the shortest path between two nodes that have a path length, which is counted by the number of edges. Breadth-first searches all the connected paths at a time. But BFS is not suitable for problems such as a finite graph that has no solutions. It will search all of the graphs but it will end up unsuccessfully.

(c) depth-limited depth search

Depth-limited depth first search is suitable for finding the answer that has infinite loop. It can stop the problem before it gets to the unexpected iterative loop. But DLS is not suitable for problems such as finding an answer that has the goal node in deep levels. If DLS is trying to solve this kind of problems, it might end up stop searching before it gets to the goal node and eventually fails to find the answer.

(d) iterative deepening depth first

Iterative deepening depth first is suitable for finding the solution for the problems that has big amount of memory. If DFS is used for such a problem, nontermination happens as the length of path is infinite. But IDS has O(db), d for depth and b for branching factor, that it has less space complexity and it can complete the search. However, IDS is not suitable for a problem such as finding a path to a goal node. Because when the node is at very deep level, IDS has to expand all nodes and it slows down the searching time that it is less efficient than DFS in this case.

(e) bidirectional

Bidirectional is suitable when there is only one goal state so that backward and forward search is similar. 8-puzzle and finding a route might be good examples. However, bidirectional search is not suitable when the goal is an abstract description like n-queens problems mentioned in the textbook.

3. This question is on properties of search algorithms. You need to explain the reasons for your answers, not simply write the name of one algorithm

   (a) If you want to limit the memory requirements, which algorithm(s) would you choose? why?

   Iterative deepening depth first search would be a good option. Because the complexity of IDS is O(db) that is relatively less and also would consider depth-limited search for limited memory requirements. It's possible to limit the depth level as much as memory required.

   (b) If you want to limit the time required to find a solution, which algorithm(s) would you choose? why?

   Bidirectional search would be a good option. Because the time complexity of bidirectional search is $O(b^{d/2})$ and this is takes the least time than any other search algorithms.

   (c) If you want to find the minimum cost solution, which algorithm(s) would you choose? why?

   Uniform cost search, breadth-first search and iterative deepening search would be good options. In case of uniform cost search, nodes are expanded by the cost from the root and minimum cost to some nodes can be found. If costs for each steps are same, breadth-first search and iterative-deepening search are the best option since both search have the shallowest path.

   (d) If you want to solution with the minimum number of steps, but you do not want to use a lot of memory, which algorithm(s) would you choose? why?

   Iterative deepening depth first search would be a good option. The space complexity for IDS is O(db) and the search is similar to BFS that for finding the minimum steps. And IDS can limit the memory requirements so that it is useful when don't want to use lots of amount of memory.

   (e) If you want to parallelize your algorithm, which algorithm(s) would you prefer? why?

   Depth first search might be good option. By dividing the search space into subtrees explored in parallel, DFS can be used. Processors make the same frontier for searching and save the references of the nodes in their memories.

4. (a) If you are given a problem described using a state space representation and add 10 to the cost of each action, will the optimal solution be the same as the one with the original costs? Explain.

   The optimal solution would be different. For example, breadth-first search and iterative deepening search assumes that they are optimal when the step costs are all same. As the increasing cost of each action would affect this assumption and the costs of actions increases polynomially, the optimal solution would be different.

(b) If in the previous problem instead of adding 10 to the costs of each actions you double its cost, will the optimal solution be the same as the one with the original costs? Explain.

The costs of actions grow exponentially so the problem would reach its optimal solution more dynamically and faster than a problem with polynomially increasing costs of actions. So the optimal solution would be different to the one with the original costs.

5. Does a finite state space always lead to a finite search tree? What if the finite state space is a tree? Can you be more precise about what types of state spaces always lead to finite search trees? (Be short but precise. Explain answers. You can support your answer with example or counterexamples, or use more formal arguments.)

   A finite search space can lead to an infinite search tree if repeated states happen. For example, in the 8-puzzle, it is possible to go back to the state of a node's parent by sliding back the empty tile. An infinite number of search nodes can be generated if the above process is repeated. Directed Acyclic Graphs type of state space can lead to finite search trees. Also duplicate checked finite state space is able to lead to finite search trees.

6. If you do not use CLOSED (this is what the textbook calls explored set in Fig 3.7) in an algorithm that uses it, what happens? be specific on what happens in each of algorithms that we have studied.

   When CLOSED list is not used, algorithms will keep explore redundant and duplicate paths repeatedly and it will cost a lot of resources. For depth-first search and depth-limited search, when the closed list is not used, it cannot keep track of nodes that were already searched. Even though there is a finite graphs, the search would be not complete. For breadth-first search, if the close list is not used, it cannot record the state of each nodes and it will keep search the nodes in frontier. For iterative-deepening search, it uses depth-limited search to seek for the goal node, IDS has same problems that come from not using the closed list. For the Uniform cost search, if closed list is not used, it cannot store the state of each nodes and cannot be uniformly stay at the frontier's child node.