Oya İlayda Yalçın
60692
COMP 341
Homework #1

Q1: What are some differences between DFS and BFS in terms of path cost and number of expanded nodes? When and why would you prefer BFS over DFS? When and why would you prefer DFS over BFS?

Due to the nature of the algorithms the number of expanded nodes is larger at BFS compared to DFS. In addition, in terms of path cost, since BFS returns an efficient solution the path cost is smaller compared to BFS. The reason behind these differences is the processing structure of two algorithms. BFS explores the nodes level by level in a search tree in order to find the goal node while dfs explores the deepest branch. Thus the number of explored nodes in BFS is larger than DFS. At the same time, since DFS is focused on finding the deepest goal branch, it does not return the most efficient path while BFS returns the path with least cost among the explored nodes. Thus, the cost of a path in DFS is larger than BFS.
One can prefer to use BFS over DFS if space complexity is not an issue and the most efficient cost is the priority result. On the other hand, one can prefer to use DFS over BFS if an acceptable path is needed in a short time or there are space complexity limitations in a problem.

Q2: What are some differences between UCS and A* in terms of path cost and number of expanded nodes? When and why would you use UCS over A*? When and why would you prefer A* over UCS?

UCS work with the principle of following the least cost path via expanding the branches in terms of cost while A* both utilizes heuristic values and costs of nodes in order to return a path. Due to this working principle, UCS returns a path with lower cost as cost is its only variable during search. On the other hand, A* utilizing heuristics expands fewer nodes compared to UCS since heuristic function contributes to the expanding process. If one does not have information about

the problem UCS is a good solution but if an informed search can be conducted, A* is able to reach the solution faster with better understanding.

Q3: Comment on your choice of state in the finding all the corners problem. Why does it allow you to solve the problem?

To solve the problem, I have implemented getting start state as a function which returns both the starting position and the corners to be explored. Storing this information at the “state” then I found the successor by checking whether in the next step we have encountered a corner or not. If a corner is encountered I remove the found corner from the corners list stored in “state” and if a corner is not found I move on to the next step. The stopping mechanism works as if the length of the corners list stored in “state” is zero. This means we have visited all of the corners so we have reached our goal. This algorithm is working because it is generated by node vise. The mechanism is taking successors as the base so it is able to check itself and make decisions each time a successor is created.

Q4: Comment on your choice of heuristic in the finding all the corners problem.. Why did you settle on that heuristic? Why is it admissible and consistent?

For a heuristic to be admissible it must be lower bound on the actual shortest path cost to nearest goal and for it to be consistent, it must additionally hold that if an action has cost c, then taking that action can only cause a drop in heuristic at most c. For my heuristic design to hold these prerequisites, I created a heuristic in which manhattan distance is used in order to evaluate the path cost. The heuristic principle goes as, for every position Pac-man takes the algorithm looks at manhattan distance to every corner in order to get their path costs. Then, in order to choose the heuristic, (which is initialized as 0 for calculation reasons) it takes the maximum path cost among all corners. In this way we are assured that, heuristic value will not be larger than cost to go to the corner at the same time if an action has cost c, then taking that action can only cause a drop in heuristic at most c since heuristic is determined by max c’s.

Q5: Comment on your choice of heuristic in the eating all the dots problem. Why did you settle on that heuristic? Why is it admissible and consistent?

I was not able to expand nodes less than 1200 in this question even though I tried various methods. The problem, arised since I was not able to implement a heuristic which consists of both the distance factor and the number of food left factors. Since both are important factors for heuristic's performance, I was not able to find the most efficient heuristic algorithm. I thought of improving the heuristic by summing it with "number of food on the grid -1" but it did not seem logical to me since I will not be able to know the number of food left at the specific state. In addition, I was not able to find a function that returns the number of food left in that specific state. So this part is incomplete. For the rest of the answer I used the Manhattan distance as I did in the corners problem. Going through every coordinate of the food grid, I checked whether there is food and if there is food what is its Manhattan distance to my current position. In the end I choose the max cost to food in order to make the heuristic admissible and consistent with the same logic explained in the previous question.

Q6: What are some practical differences between a consistent and an inadmissible heuristic, in terms of path cost and number of expanded nodes? When and why would you prefer an inadmissible heuristic over a consistent one? When and why would you prefer a consistent heuristic over an inadmissible one?

Consistency is about the cost of the path so when the cost of the result is important consistency should be a key point to consider in an algorithm. A consistent algorithm is also admissible so we can't talk about a case where there is consistency but no admissibility. For admissibility we care whether there is an overestimation for reaching a node so it affects the search nodes. If we want our algorithm to be faster and search for less nodes we can dismiss admissibility but when this is then we may miss a better result. If we value the quality of the result in terms of cost we should choose consistency rather than inadmissibility.