CS 404 – Artificial Intelligence

HW 2 – Blind Search – AIMA– Chp. 3

YASIN AYDIN

75pt

Late homeworks accepted for 2 days (no penalty in the first late day; -10pts off when late for 2 days)

Please type your answers and use only the allocated space.
You may color your answers blue for easy grading.

Objective: To deepend the understanding of time and space complexity in search algorithms and deciding on suitable algorithms for a given problem.

Type your answers, but you can draw any illustrations by hand (if so you can send the scanned document).

1) 30pts –**Answer the following using the general Tree Search algorithm** (remove front node from the fringe/queue – goal test – expand).

Reminder: You can use the following equality for compactness:

$1 + b + b^2 + \dots b^d = (b^{d+1}-1)/(b-1)$

a) 15pt - How many nodes are ***visited*** (chosen from the queue, goal tested and expanded) in the worst case using Breadth-First search, when the solution is at depth d, and the branching factor is b, and the depth of the maximum branch is m?

Give a formula.

$$((b^{d+1}-1) / (b - 1)) - 1$$

b) 15pt- How many nodes are ***generated*** (added to the queue as a result of expanding the parent) in the worst case using Breadth-First search, when the solution is at depth d, and the branching factor is b, and the depth of the maximum branch is m?

$$((b^{d+2} - 1)) / (b - 1) - b$$

2) 45pt – You are given the problem of finding whether 6-degrees of separation holds between a particular 2 people in the world. E.g. given two people – say you and your favorite celebrity - the software should decide whether they are connected in at most 6 friendship edges (e.g. you-f1-f2-f3-f4-f5-celebrity).

Let`s assume you have the list of all friendships for all people in the world and that everyone has exactly b=100 friends and that there are 6 billion people in the world.

a) **18pts)** State **whether the following algorithms are <u>complete</u>** (if there is a up to 6-degree path, does it find it?) **and <u>optimal</u>** (defined here as 'does it find the shortest path connecting two people') **for this problem**.

b) **12pts) If an algorithm is BOTH complete AND optimal, comment on its time and space complexity with a one line summary about its suitability (e.g. "will take too much time/space: O(b^d)").** If an algorithm would take too much time or space to be feasible, indicate as such; if it is suitable but is an overkill, you should indicate that also.

Algorithm	**Complete (answer as Yes or No)**	**Optimal (answer as Yes or No)**	**Feasibility (add a one line comment)**
Breadth first search	**Yes**	**Yes**	Will take too much time and space since time/space: O(b^(d+1)). It is **feasible**. (It is not feasible in space for personal computers since it requires GBs of RAM)
Depth first search without repeated state checking	No	No	
Depth first search with repeated state checking	Yes	No	
Depth limited search DFS with a depth limit of ……6……	Yes If $l >= d$	No	

Iterative deepening DFS	**Yes**	**Yes** if step cost = 1	Will take too much time: O(b^d), but space is linear, space: O(bd). It is **feasible.**
Bidirectional search	**Yes**	**Yes**	Will take less time than others: O(b^(d/2)), but space is not linear, space: O(b^(d/2)). It is **feasible**.

c) **15pts) Which blind search algorithm** (among the ones listed above**) would be best for this problem? Explain your answer.** Consider space, time complexities and completeness and optimality.

If two algorithms are the same or similar, you may choose the one which is easier to implement or state that they are both as good / suitable.

(Nowadays it is possible to have enough memory for this problem since the b and d are not large, so I included this in my assumptions.)

With the assumptions: There is enough memory capacity and we want the complete/optimal solution. I think best one would be the bidirectional search since the time complexity is the best among others and the space complexity is not too much, it is O(b^(d/2)).

But I think I would change my answer if the conditions are different.
In the cases of low memory capacity Iterative deepening DFS would be better since the space complexity is linear (O(bd)) which is so much lower than O(b^(d/2)). However, there is a trade of between time and space so that it will take more time (O(b^d) instead of O(b^(d/2))).

If there is enough memory capacity to store b^(d/2), bidirectional search would be preferable since the time complexity is the lowest among others and it is complete/optimal.