# Hw2 - Homework for cs404

Artificial Intelligence (Sabanci Üniversitesi)

# CS 404 – Artificial Intelligence
# HW 2 – Blind Search – AIMA– Chp. 3

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.

**1 + b + b^2 + b^3 + b^4 + … + b^d = O(b^d)**

**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 + b^2 + b^3 + b^4 + … + b^d + b^((d + 1) – b) = O(b^(d + 1))**

**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 complete** (if there is a up to 6-degree path, does it find it?) **and optimal** (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** | BFS is only optimal if the cost solution is a non-decreasing function. Also takes too much space. (O(b^d)) |
| Depth first search without repeated state checking | **NO** | **NO** | Can get stuck in a loop. |
| Depth first search with repeated state checking | **YES** | **NO** | May not always find the optimal solution. |
| Depth limited search DFS with a depth limit of ……6…… | **YES** | **NO** | It may lost to much time looking at the higher nodes of a tree however, most of the nodes are at lower level so it is OK. |
| Iterative deepening DFS | **YES** | **YES** | It requires O(b*d) memory and O(b^d) time. |

| | | | |
|---|---|---|---|
| Bidirectional search | **YES** | **YES** | If using BFS, edge weights needs to be same or not-existent so, the goal should be well defined. (Not feasible in chess for example) |

**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.

DFS is not good since the tree is very big so, BFS seems like the best option. However, there is a better alternative to BFS for this situation:

I would use **Iterative Deepening DFS**.

Firstly, I would represent the list of the people with a graph. Every person becomes a node and there is an edge between two people if they know each other. Then use IDS.

Time complexity of this algorithm is O(b^d) and space complexity is O(b*d). This means that IDS needs less memory than BFS (O(b^d)).

This method is complete. It will eventually find its way to the other node.

Is it optimal? This IDS essentially becomes Dijkstra's Algorithm on graph where every edge cost is 1 (path cost is a non-decreasing function of depth). Therefore, I can say that this method is optimal as well.

Bidirectional search with BFS is suitable as well but it is harder to implement.