Assignment 1

Part 1:

Explanation of the problem. (1 point)

Given the 0-1 knapsack problem, the goal is to apply Breadth-first search and Depth-first search to search for the best combination of items present inside the bag. Each item can either be included, corresponding to the value “1”, or excluded, corresponding to the value “0”.

a. Give the representation of a solution (answer) of the problem, as explained during

the course. (0.5)

Given an input of N objects, the solution to the problem can be represented as a binary vector

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

Where v_i is set to 1 if the object i is included in the optimal bag configuration and to 0 if it is excluded.

b. Give the equation of the objective function (what we want to maximize). (0.25)

The objective function is given by summation of the objects’ benefit values b_i multiplied by their corresponding inclusion/exclusion flags from the vector v.

$$\sum_{i=1}^{n} b_i \cdot v_i$$

c. Give the equation for the restriction(s) of the problem. (0.25)

The equation for the restrictions of the problem is given by the following inequality, where on the left hand side contains the summation of the weights' of the objects multiplied by their inclusion/exclusion flag from vector v, and on the right hand side the weight limit of the bag, W_L

$$\sum_{i=1}^{n} w_i \cdot v_i <= W_L$$

2. Comparison of the algorithms. (1 point)

a. Comparison of the time expended by the algorithms. (0.5)

Time complexity of both DFS and BFS in my approach is O(2^n), since they branch out at every item by making a decision to include or exclude (branching factor of 2), and explore all of the possibilities before terminating with the optimal answer in the worst case.

b. Comparison of the space used in memory at a time by the algorithms. (0.5)

DFS algorithm utilizes stack to store the search states, and it grows proportionally to the the depth of the search space, which in this case is bounded by n, number of items. As DFS only stores one path at a time, its memory complexity is given by O(n)

BFS algorithm on the other hand utilizes a queue and expands to full width at each depth level, storing 2^n nodes in the worst case (last level of the tree). Therefore, the memory complexity of the BFS algorithm is given by O(2^n) for the 0-1 knapsack problem.