0/1 Knapsack Problem

For this coding project, I use the pseudocode example from the lecture’s slide to implement the knapsack problem with breach bound. I use the priority queue to store all my queue. At first, I create two nodes u and v, then I set the v as my root node first. I then set up the level, profit, weight, and bound. After entering the while loop I dequeue the root node and store it to v again. I use v as my memory to increase child node’s level, profit, weight and bound. Next, the code start creating child nodes left and right. Only the left node includes the first object, and the right side doesn’t. The code

If(u.weight <= W && u.profit > maxprofit);

will check if the child node weight is lower than the capacity and profit is greater than the max profit. If it is true then it will be added into the solution set. Next If the bound is greater than the max profit then it will be added into the queue. The next cycle queue will start to dequeue the node that has the greater bound and stores it to node v again. The while loop will keep repeating the same process to check every child. For the bound method if the knapsack still has a room then it will calculate the bound of the node. The code

result += ((W – totalWeight)\*(profit[k]/weight[k]));

This calculation is the same as the fractional knapsack problem. If the next object can’t fit into the knapsack then it will only add part of the profit that still allows fitting into the knapsack. The time complexity for this program is around O(2^n). I gave 4 items for this program to solve, and the result shows that there are 11 visited nodes. Each node has 2 children to compare, and 2^4 is also closed to 11.