Phuc Dang
Ptd328
EE360C
Lab 2

Lab 2 Report

Part 1:

My graph contains an adjacency list to represent the keys graph. The reason why I choose adjacency list to represent the graph is because the graph will most likely be sparse as well as having houses that contain no keys of other houses. In the case of having a lot of houses, the difference will be significant. Thus, it will be more beneficial in term of space complexity to use adjacency list compared to adjacency matrix

Part 2:

My algorithm: Topological order sorting using lecture slides

```
Let S be the set of unlocked house
While S is not empty
    Pick a house N in S, add it to the end of the result list L, and remove it from S
    For each house M with an edge coming from house N
        Remove the key needed to get to house M
        If the house has no other keys, or no other incoming edges then
        Add house n to set S
If graph has edges, then
    Return false (can't rob all house)
Else
    Return L
```

The Time complexity is O (V + E) as I am going through each node in the list, and for each node I am going through all the edges that coming from that node. Thus, I am going through all the node and all the edges.

The graph is stored in an adjacency list, which has space complexity of O(V+E). There is an array list to store a result, which has a space complexity of O(V). Thus, the space complexity of the algorithm is O(V + E)

Part 3:

My algorithm:

Sort the list based on the value of the items.

```
Looping through the list from the most valuable item or till reaching maximum weight
    if the remaining weight is more than the weight of the item
        Take all the weight of that item
    Else if remaining weight is less than the weight of the item
        Take the item based on the remaining weight
```

Run Time: Sorting the list could be done in O(nlogn) time. Looping through the list take O(n) with n is the number of the items. Comparing the weight will only take O(1). Thus, the time complexity of this algorithm will be done in O(nlogn)

The algorithm will not work if Fruitcake can't steal a fraction of the device. Say we have a list of Laptop 2.5 7000, IPad 2 5000, Phone 1 3000, and max weight is 3. If we applying the algorithm Fruitcake will take the laptop which max loot of 7000 while he could have taken Ipad and Phone which costs 8000.

Part 4:

My algorithm: Using greedy algorithm described in class, always choose the one with the earliest finishing time for each step.

```
Sort the list in term of finishing time from earliest to latest
A is the result
Looping through the list from j = 1 to n
    If buyer j is not overlapping with A
        Add j to the result
Return L
```

Sort the list could be done in O(nlogn) time, looping through the list take n time, checking overlapping take O(1) as we comparing end time and start time, and add to the result can take O(1) (Adding to the end of the list). Thus the time complexity of this algorithm is O(nlogn).

Proving correctness:
Assume the algorithm is not optimal
Let i1, i2,….ik denotes the buyers selected by this algorithm
Let j1, j2,…..jm denotes the buyers selected by an optimal solution where j1 = i1, j2 = i2,…jr = ir for the largest number of r
Since for every step, I am picking the buyer with earliest finishing time. Thus, ir+1 will finish before jr+1 in the optimal solution
Therefore, we can replace buyer jr+1 with buyer ir+1 without affecting the optimality of the remaining solution.
Thus, the algorithm is optimal.