1) a)

```
Function cluster(Arr, max) → Find max of current, and before current
                             then save the new array the
                             value then return,
                             max val.
    For i, stop in enumerate(stops) → $T_1(n)$
        Arr[i] = max(dictStopsVal[stop] + Arr[i-1], Arr[i])
        if Arr[i] > max
            max = Arr[i]
        return max
```

time complexity: $T(n) = T_1(n) + 1$ → $T(n) = O(n)$

Space complexity: $O(n)$ → temporary array

1) b) In homework 3 time complexity is $\theta(n^3)$ of this question I made up it using loop and here the time complexity is $O(n)$ so there is big difference between two of them,

CamScanner ile tarandı

```
2) Function candy
    for i range (1, n+1) → $T_1(n)$
        max_val = -1
        for j in range(i) → $T_2(n)$
            max_val = max(max_val, arr[j] + Arr1[i-j-1])
        Arr1[i] = max_val
    return Arr1[n]
```

Time complexity : $T(n) = T_1(n) * T_2(n) + 1$ → $T(n) = \theta(n) * \theta(n) + 1$

$T(n) = \theta(n^2)$ //

Space Complexity : $\theta(n)$ → Temporary Array

3) when I see the question knapsack problem come to my mind I implement the algorithm using greedy algorithm. I calculated the item with the heighest ratio and add. then until we can't add the next item as a whole and at the end add the next item as much as we can.

Time complexity:

$$T(n) = \underbrace{T_1(n)}_{\text{Sort Algorithm}} + \underbrace{T_2(n)}_{\text{For loop}} + 1$$

$$T(n) = O(n\log n) + O(n) + 1$$

$$T(n) = O(n\log n) //$$

CamScanner ile tarandı

```
4) function maxMeeting
    for i in range (1, n)
        if( l[i].start > time_limit)
            ans.append(l[i].pos)
            time_limit = l[i].end
    for i in ans:
        print()
```

$$T(n) = T_1(n) + T_2(n) + T_3(n)$$

$$\begin{array}{ccc} \downarrow & \downarrow & \downarrow \\ \text{Sort Algorithm} & \text{for loop} & \text{for loop} \end{array}$$

$$T(n) = O(n\log n) + O(n) + O(n)$$

$$T(n) = O(n\log n)_{//}$$

let the given set of activities be $S = \{1, 2, 3 \ldots n\}$ and activities are sorted finish time. The Algorithm of greedy always pick activity 1. then there is a solution A of the same size with activity 1 as the first activity.