CS301 HOMEWORK 3

Ceren Dinç 28220

Question 1a)

→ The possible longest palindrome will include all letters. From 1st to last elements.
→ L[1, n]

Question 1b)

0	if i > j
1	if i = j
2 + L[i + 1, j - 1]	if i < j and ai = aj
max(L[i + 1, j], L[i, j - 1])	if i < j and ai != aj

→ When i > j:

- we can say that it is an empty sequence

→ When i = j:

- there is only 1 element in the subsequence which has length 1

→ When i < j && ai = aj:

- 1st and last elements are same so we already have length 2
- we add remaining length L[i+1, j-1] to 2

→ When i < j && ai != aj:

- Here 1st and last elements are different and we should decide the longest among:
  - L[i + 1, j] here we do not include i
  - L[i, j – 1] here we do not include j

Question 1c)

→ Θ(n^2)

→ n denotes the length and in order to compute all possible combinations of (i,j) for L[i, j] we need n*n calculations.

Question 2a)

→ We sould have the full capacity W by considering all of n items

→ P[n, W] where i = n , j = W

Question 2b)

0	if i = 0
P[i – 1, j]	if i > 0 and j < wi
max{P[i - 1, j] , P[i – 1 , j - wi] + vi}	if i > 0 and j >= wi

→ for i = 0 then we do not have items

→ for i > 0 && j < wi then with capacity of j we can use i-1 items

→ for i > 0 and j >= wi then we find maximum of P[i - 1, j] , P[i – 1 , j - wi] + vi

Question 2c)

→ Θ(n*W)

→ here n denotes objects count and W denotes knapsack capacity.

→ table entries are computed in constant time (each) and in total we make (n + 1)*(W + 1) computations which results in Θ(n*W) in worst case.

Question 3a)

→ First, we should sort objects' value-to-weight ratios in descending order

→ Second, fill the knapsack with those objects.

→ We continue to fill the knapsack and we stop when

- we do not have remaining knapscak capacity,
- or we do not have any remaining objects.

→ So, the greedy choice is picking the object which has highest value-to-weight ratio

Question 3b)

→ After picking oi object by greedy choice of selection with value-to-weight ratio,

→ Now remaining subproblem has:

- objects (o1, o2, …, o(i-1), o(i+1), …, on)
- capacity (W - wi)

→ remaining objects decreased after we pick oi

→ capacity decreased by wi