Okan Torun
1801042662

Q-1) Dynamic Programming is mainly an optimization over plain recursion. whenever we see a recursive solution that has repeated calls for same ~~[illegible]~~ inputs, we can optimize it using Dynamic Programming. The idea is to simply store the results of subproblems, so that we don't have to recompute them when needed later. This simple optimization reduces time complexities from exponential to polynomial.

M(i) = Max sum in index i.

Arr(i) = Array index

Recurrence solutions is = $M(i) = Max(M(i-1) + Arr(i), Arr(i))$

Analyze

$$\sum_{i=1}^{n} 1 = n \Rightarrow \boxed{T(n) = \Theta(n)}$$

b) in our previous assignment we used brute-force algorithm and found time complexity $\Theta(n^3)$. in this one, we found $\Theta(n)$ With dynamic programming. Time complexit became more efficient.

Okun Torun
180102.2662

Q-2) candy(n) : best possible Price

Recurrence Solution = candy(n) = max(Val(i), maxValue)

$$\sum_{i=1}^{n+1} \sum_{j=0}^{i} 1 = \sum_{j=1}^{n+1} i+1 = 2+3+4 \ldots n+2 \Rightarrow \frac{(n+2)(n+3)}{2}$$

$$\boxed{T(n) = \Theta(n^2)}$$

CamScanner ile tarandı

Okan Torun
1801042662

Q-3) In the sorting algorithm, we sort the cheeses according to their unit price.
Then we fill it with the most expensive amount of cheese without exceeding the volume of the box.

Anlyze

sorting

$$\sum_{1}^{n-1} \sum_{1}^{n-i-1} 1 \Rightarrow n^2$$

PutBox

$$T_B = \Theta(1)$$

$$T_W = \sum_{i=0}^{n} 1 = \Theta(n)$$

$$\boxed{T(n) = \Theta(n^2) + \Theta(n) = \Theta(n^2)}$$

Okan Torun
1801042662

(Q-4) First, we sorted the table according to the course hours. Then we found the maximum number of courses according to the start and end times.

Sorting

$\Theta(n^2)$

Find Course

$\sum_{i=0}^{n} 1 = n \Rightarrow \Theta(n)$

$T(n) = \Theta(n^2) + \Theta(n) = \Theta(n^2)$