Ozan Argıt
Örceken
1901042259

Q1: Basic operation = comparing 2 elements.

$$\sum_{i=1}^{n} 2 = \underbrace{2 + 2 + \dots + 2}_{n-1 \text{ times}} = 2(n-1) = 2n-2 \approx \Theta(n)$$

In homework 3 the complexity equals $O(n^2)$ because in that algorithm we have 2 for loop (one is inside of other one).

On the other hand that algorithm calculates same values more than one time but dynamic programming does not allows that.

Brute force algorithm calculates all possibilities but dynamic programming algorithm calculating like:

3	-5	2	11	-8	9	-5

↑ (-8)

For example in here algorithm compares current and maximum value and 13-8, 13>5 ⇒ current value equals 5 but maximum does not changes.

after that algorithm wents 9 and current goes 9+5=14 and compares 13 and 14 14>13 so maximum equals 14.

Q_2: Basic operation = + (Sum)

$$\sum_{i=1}^{n} \sum_{j=0}^{i} 2 = \sum_{i=1}^{n} \underbrace{2+2+\dots+2}_{i \text{ times}} \Rightarrow \sum_{i=1}^{n} 2i = 2\sum_{i=1}^{n} i$$

$$\therefore \sum_{i=1}^{n} i = 1+2+\dots+n = \frac{n\cdot(n+1)}{2} = \frac{n^2+n}{2} \approx \underline{\underline{O(n^2)}}$$

Because of dynamic programing instead of calculating all values algorithm calculates maximum profit of every cm for one time.

maxVal array holds the maximum price of every lenght and calculating new maximum value adds an individual value and any maximum value for test new maximum value. In inner for if statement looks for maxValue[i+j-1] + arr[j]. i+j-1+j = i-1 so it means that adding maximum values to every individual costs for finding the best. (because of i+j-1+j = i-1, every new values lenght equals the wanted lenght.)

Q_3: For basic operation = + (Sum) [illegible]

$$\sum_{i=0}^{n} 2 = \underbrace{2+2+\dots-2}_{n \text{ times}} = 2n \approx O(n)$$

But I used l.sort and sorting algorithm of Python equals $O(n\cdot\log(n))$ so Time complexity of algorithm $\approx O(n\cdot\log(n))$

In greedy algorithm I calculate the value of every kg of cheeses. Because greedy algorithm says put most valuable thing at first then second, third...

I put most valuable things and check the left space for cutting and filling of bag.

Q_4: Basic operation = lessons.append

$$\sum_{i=0}^{n} 1 = \underbrace{1+1+\dots+1}_{n \text{ times}} \Rightarrow n \approx O(n)$$

But lessons.sort takes $O(n \cdot \log(n))$ times.

In that algorithm I searched first finished lesson and any lesson that starts after that and finishes first. array was sorted for that with their finishing time.