Homework 6

2] Let $S_i S_{i+1} \dots S_k$ denote the substring. OPT(k) denotes whether $S_{i,k}$ can be segmented. OPT(k) = 1 if it can be segmented and 0 otherwise. Segmentation is possible if and only if the last word in the substring is a part of the dictionary and the remaining can be segmented.

$$OPT(k) = \max_{0 < i < k \text{ and } S_{i+1,k} \text{ is a part of the dictionary}} OPT(i)$$

We initialize OPT(0) = 1 and then compute for k = 1 .... n. The answer corresponding to OPT(n) is the optimal one and it takes $O(n^2)$ to compute.

3] Let OPT(l, r) be the maximum no. of coins we can obtain by bursting the balloons. Over here we have to determine the balloon to burst last. Let's say k is the balloon is want to burst last. Therefore all the other balloons have to be bursted before k.

$$OPT(l, r) = \max_{l < k < r} (nums(k) * nums(l-1) * nums(l+1)$$

$+ OPT(l, K-1) + OPT(K+1, l))$

We initialize $OPT(l,l) = 0$ if $l < l$
To compute it takes $O(n^2)$ and we have a total of $O(n^2)$ ∴ The total running time $= O(n^3)$.

4] Let length $[0, 1 \ldots n]$ and price $[p_0 \ldots p_n]$ be the two arrays having length and price of rods of different length.

$OPT(i, a)$ denote the rod of length $[1 \ldots i]$ ~~and~~ with max allowed length of a.

If i is a part of the optimal sol^n:

$$OPT(i, a) = p_i + OPT\left(i-1, a - len(i)\right)$$

~~else~~ Else

$$OPT(i, a) = OPT(i-1, a)$$

$$OPT(i, a) = \max\left(p_i + OPT(i-1, a - len(i)), OPT(i-1, a)\right)$$

5] (a)

	Min 1	Min 2	Min 3	Min 4
A	2	1	1	200
B	1	1	20	100

Here the algorithm finds 122 as the optimal answer whereas the actual optimal answer should have been 204. We are not suppose to choose B at allbeit the greedy algorithm chooses A for the first minute and then moves and chooses B for the final two steps.

(b) Let OPT(i, A) denote the maximum value of a plan in minutes 1 ... i that ends at machine A and similarly let OPT(i, B) denote the same for machine B.

If in the $i-1^{th}$ minute we were on machine A then
$OPT(i, A) = a_i + OPT(i-1, A)$
and suppose we used the $i-1^{th}$ minute in moving from B to A then $OPT(i, A) = a_i + OPT(i-2, B)$

$\therefore OPT(i, A) = a_i + \max(OPT(i-2, B), OPT(i-1, A))$

On the same grounds we will be

writing OPT(i, B),

$$OPT(i, B) = b_i + \max(OPT(i-1, B), OPT(i-2, A))$$

We will initialize OPT(1, a) as a and OPT(1, B) as b_1.
We are going to do this for n-1 values and computing this takes O(1) so the time complexity would be O(n).

7] Or the same grounds on which we use OPT(K, w) for the 0-1 knapsack in the class.

OPT(K, w) denote the value of the optimal soln with max. allowed weight w.
Over here if K is included in the soln then we can still use the same K for the next filling of weights as we have infinitely many items of each type. And if K is not a part of the optimal soln then

$$OPT(K, w) = OPT(K-1, w)$$

$$OPT(K, w) = OPT(K, w - w_K) + w_K$$

$$OPT(k) = \max(OPT(k, w - w_k) + w_k, OPT(k-1, w))$$

We initialize $OPT(0,0) = 0$

5] Let $W = [w_1, \ldots w_n]$ be the set of ordered words. Let's say in the optimal solution we are able to insert only the first k words in the first line.
∴ The remaining lines would be filled by $(w_{k+1} \ldots w_n)$ words

Opt(i) denote the sum of squares of the slacks for the optimal solution.
If we are able to put first p words from $(w_i \ldots w_n)$ then

$$\sum_{j=i}^{p+i-1} c_j + p - 1 \leq L \quad \text{and} \quad \sum_{j=i}^{p+i} w_j + p > L$$

The extra spaces characters would be

$$sc(i,k) := L - k + 1 - \sum_{j=i}^{k+i-1} c_j$$

∴ Our optimal solution recurrence formula becomes,

$$Opt(i) = \min_{1 \le k \le p} \{ BC(i,k)^2 + Opt(i-k) \}$$

$$= 0 \quad \text{if } p < n-i+1$$

$$\text{if } p \ge n-i+1$$

In order to find the minimized value of Opt(p) we trace back the value of k to get the no. of words to be printed.
The total time $O(nL)$