Comp 160: Algorithms — Jiawei Wang
Tufts University, Fall 2019 — Homework 10

Question 1:

- Recurrence without paying c each cut:

$$C(n) = \max_{1 \leq i \leq n} \{C(n-i) + p_i\} \tag{1}$$

- So the new question becomes: find the max profit with c cost each cut:

$$C(n) = max \left\{ \max_{1 \leq i \leq n} \{C(n-i) + p_i - c\}, p_n \right\} \tag{2}$$

- Base case: $C(0) = 0$
- each probelm takes n time and there are n subproblems so: $total = \theta(n^2)$

Question 2:

2.1

Since every element is positive, then the whole array is bigger than any segment:
go through the array and sum up all the elements: $\theta(n+1) = \theta(n)$

2.2

Since every element is negative, then we choose the biggest single element:
Go through the array and find the biggest one: $\theta(n)$

2.3

Use brute force: for i in array: for j in array[i:]: this takes $O(n^2)$ time
Then sum up the elements between i and j, this takes $O(n)$ time
$Total = O(n^3)$

2.4

- Use $C(i)$ as the max subarray with array[i] as the ending element

$$C(i) = max\left\{array[i], array[i] + C(i-1)\right\} \tag{3}$$

- Base case: $C(0) = array[0]$
- Use str[i] as the starting element in the max subarray C(i):

$$str[i] = \begin{cases} i & \text{if } C(i-1) < 0 \\ str[i-1] & \text{if } C(i-1) \geq 0 \end{cases} \tag{4}$$

- Runtime: $O(n)$ to fill in + $O(n)$ to go through and find the max = $O(n)$ in total
- Space useage: C(i) takes O(n) and str(i) also takes O(n) = O(n) in total