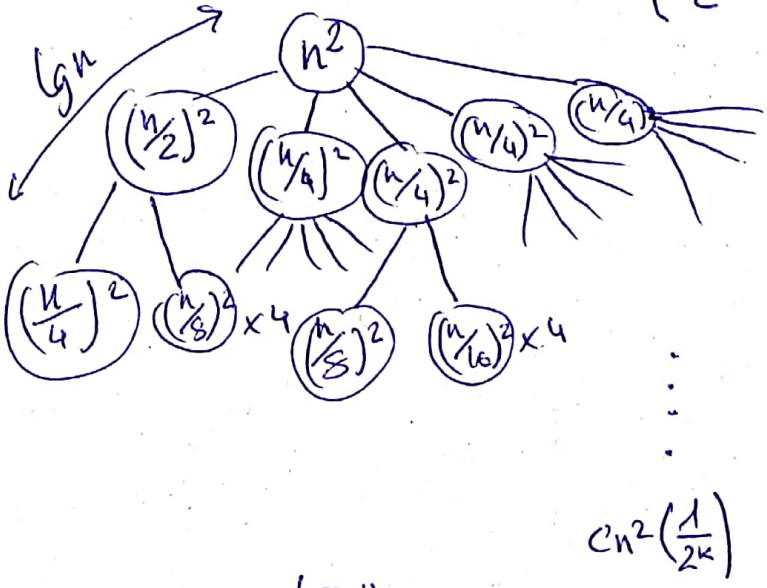
Alejandro Martínez Serrat
S202468

Mandatory 1

1. Recurrence

$$T(n) = \begin{cases} T(n/2) + 4T(n/4) + cn^2 & \text{for } n \geq 2 \\ c & \text{otherwise} \end{cases}$$



cn^2

$c(n/2)^2 + 4c(n/4)^2 = cn^2\left(\frac{1}{4} + \frac{1}{4}\right) = cn^2\left(\frac{1}{2}\right)$

$c(n/4)^2 + 8c(n/8)^2 + 16c(n/16)^2 =$

$= cn^2\left(\frac{1}{4^2} + \frac{8}{8^2} + \frac{16}{16^2}\right) =$

$= cn^2\left(\frac{1}{16} + \frac{8}{64} + \frac{16}{256}\right) =$

$= cn^2\left(\frac{1}{16} + \frac{1}{8} + \frac{1}{16}\right) =$

$= cn^2\left(\frac{4}{16}\right) = cn^2\left(\frac{1}{4}\right) = cn^2\left(\frac{1}{2^k}\right)$

$cn^2\left(\frac{1}{2^k}\right)$

$$T(n) \leq \sum_{k=0}^{\log n} cn^2\left(\frac{1}{2^k}\right) =$$

$$= cn^2 \sum_{k=0}^{\log n} \left(\frac{1}{2^k}\right)^k = cn^2 \sum_{n=0}^{\infty} \left(\frac{1}{2}\right)^n = cn^2 \frac{1}{1 - \frac{1}{2}} = 2cn^2 \Rightarrow O(n^2)$$

Assume: $T(m) \leq 2cn^2$, $m < n$

To find cn: $T(n) \leq T(n/2) + 4T(n/4) + cn^2 \leq$

$$\leq 2c\left(\frac{n}{2}\right)^2 + 4\left(2c\left(\frac{n}{4}\right)^2\right) + cn^2 = \frac{cn^2}{2} + \frac{2cn^2}{2} + cn^2$$

$$= 2cn^2 \quad \text{QED.}$$

~~Fibonacci numbers~~

2 Chocolate agency

2.1 $n=6$, maximum possible benefit

s_i	b_i	OPT
500	700	200
150	300	150
500	400	150
300	300	~~100~~
450	600	200
200	300	~~100~~ 100

$OPT = \max(b_j - s_i - (j-i)100)$, $i \leq j$

the pair of days that offer the maximum benefit for the agency is to buy and sell in the day 1 or ~~sell~~ (s_4, b_5)

2.2

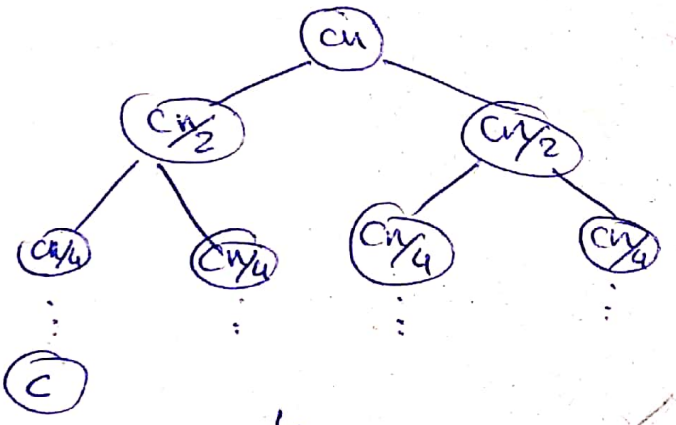
The day x sets the maximum day in which the product can be bought from the producer and the $(x+1)$ day is the first day it can be purchased by the stores. Thus, ~~it should be actualized to~~ to be able to compare the profits, they should be actualized to day x. Given x, the values in $\{S\}$ will be added 100$ each day of distance to day x $S_i = (x-i)100 + S_i$. While actualizing the values, the minimum will be stored. After covering x days we will have the minimum sell price.

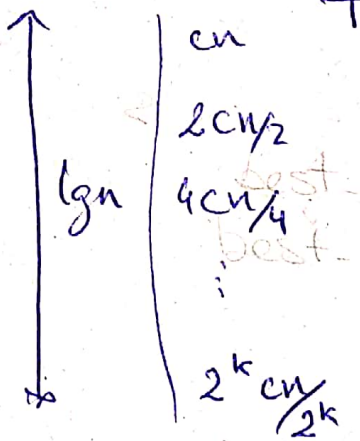
Similarly, ~~values greater than~~ values of $\{B\}$ after day x will be deducted 100$ each day of distance to day x and storing the day of the maximum price. After n actualizations, both index will be known, the smallest sell price and the maximum buy price, which will give the maximum profit. $O(n)$

2.3

The algorithm will divide the input arrays until they are split in single days $[s_i, b_i]$. Recursively, it will return the biggest profit of the considered days. At first, when it is only one day, the profit is $(b_i - s_i)$. The next two sets' profits will be compared, and the biggest will be selected. there is another option to consider, which is that the best sell-buy pair are from different sets. For that the algorithm in 2.2 will be used. In $O(n)$ will search for the best pair of sell-buy prices from different sets, divided by the boundary day. the divide-and-conquer division to single days takes $O(\lg n)$ and then $O(1)$ to compare profits. Check the "cross-profits" with 2.2 algorithm $O(n)$ so the complete algorithm will take $O(n \lg n)$.

2.4.





$T(n) = 2T(n/2) + cn$

$$T(n) \leq \sum_{k=0}^{\lg n} 2^k cn/2^k = cn \lg n$$

Assume $T(m) \leq cm \lg m$ for $m < n$

$$T(n) \leq 2T(n/2) + cn$$

$$T(n) \leq 2c(n/2)\lg(n/2) + cn =$$

$$= cn(\lg n - 1) + cn = cn \lg n \quad \text{(QED)}.$$