Stav Yosef, 316298876

Advanced Algorithms - Homework 1

Question 1 - Proof by Contradiction

Let's assume that there is a smart man called Neil, matched with a less smart woman.
For it to happen all smart women need to reject Neil (we have k smart women and k smart men).
We know that smart women prefer smart men and smart men prefer smart women therefore we know if Neil is matched with a less smart woman, Neil went to every smart woman and got rejected every time! Hence there is at least k other smart men that all the smart women preferred over Neil, therefore contradiction! there are k smart men and Neil is a smart man → we have at least k+1 smart men which is impossible.
Every smart man is matched with a smart woman.

Stav Yosef, 316298876

Question 2

In order to reach the maximum days, we need to make a situation where only 1 man is moving every day.
If n = 1, nothing to do maximum days = 1
If n = 2, maximum days = 2

1	
1	2
a	
b	

2	
1	2
a	b

The generalization works for n >= 3:
Let's start with an example,
Assume we have 4 women = {1, 2, 3, 4} and 4 men = {a, b, c, d
Maximum days = 10

1			
1	2	3	4
a	d		
b			
c			

2			
1	2	3	4
a	d	c	
b			

3			
1	2	3	4
a	d	c	
	b		

4			
1	2	3	4
a	d	c	
		b	

5			
1	2	3	4
a	d	b	
	c		

6			
1	2	3	4
a	c	b	
		d	

7			
1	2	3	4
a	c	b	
d			

8			
1	2	3	4
d	c	b	
	a		

9			
1	2	3	4
d	c	b	
		a	

10			
1	2	3	4
d	c	b	a

Stav Yosef, 316298876

Generalization:

Preferences for the men:

man n - {2, 3, 4, ..., n, 1}
man (n - 1) - {1, 3, 4, 5, n - 1, 2}
⋮
man (n - k) - {1, 2 + k, 2 + k + 1, ..., n - 1, 2, 2 + k - 1} where k > 0

Preferences for the women:

women 1 - {n, all other men}
women 2 - {n - 1, n, all other men}
⋮
women k - {n - k + 1, n - k + 2, all other men} where k > 0

Let's understand why the upper bound is $n^2 - 2n + 2 = (n(n - 2n) + 2)$
As we can see in the example above, each man moved $n - 2$ women (each day only 1 man moved = $n - 2$ days for each man).
The first man (in my example, man 'a') moves 1 extra day ($n - 1$) because it takes the last woman place.
So far, we have $n * (n - 2) + 1 = n^2 - 2n + 1$
The last day we are missing in our math is the first day so +1.
In the end we have $n^2 - 2n + 2$

Stav Yosef, 316298876

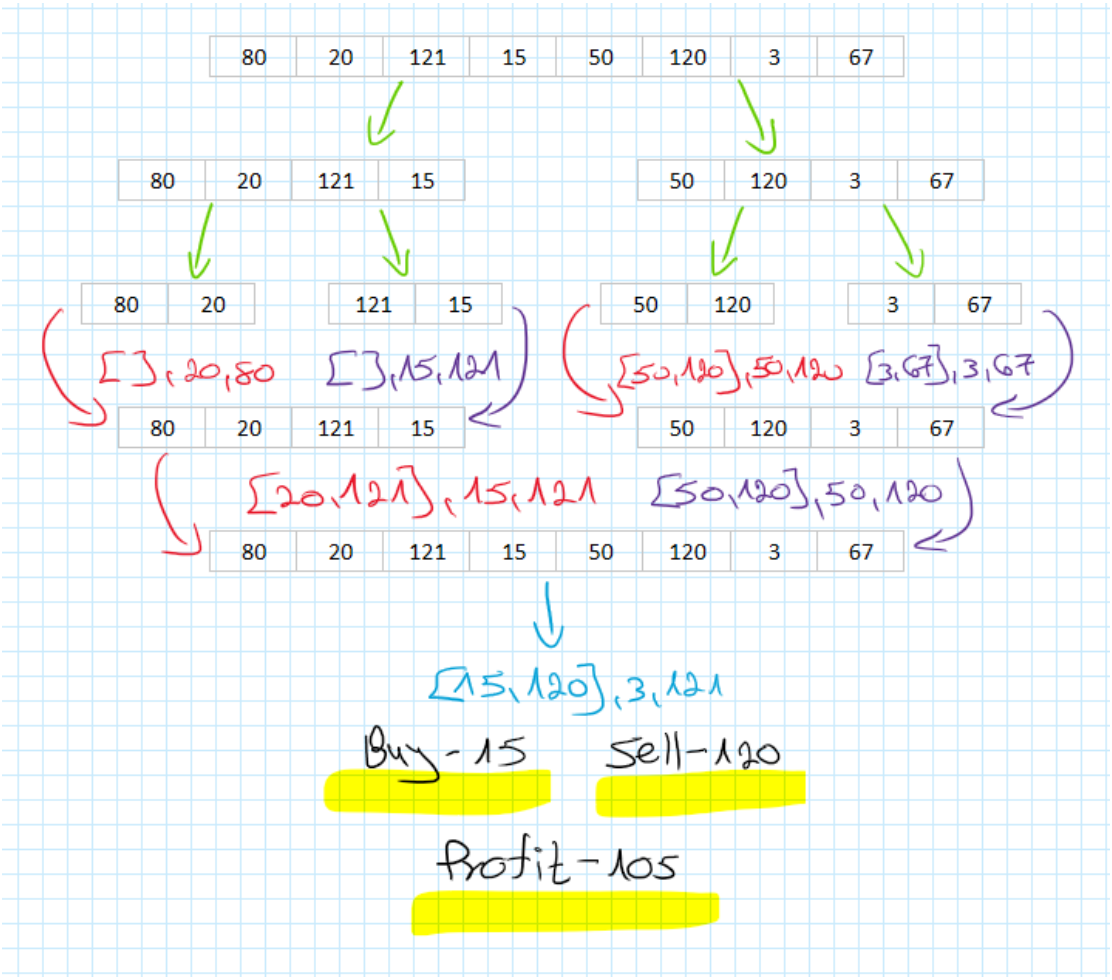
Question 3

1. $a = 6, b = 4, k = 1 \rightarrow \theta\left(n^{\log_4 6}\right) \approx \boldsymbol{\theta(n^{1.292})}$
2. $a = 16, b = 2, k = 4 \rightarrow \theta(n^4 \log n)$
3. $a = 3, b = 27, k = 0 \rightarrow \theta\left(n^{\log_{27} 3}\right) \approx \boldsymbol{\theta(n^{0.333})}$
4. $T(n) = T(n-1) + n^c = T(n-2) + (n-1)^c + n^c$
   $= \cdots = T(1) + 1^c + 2^c + \cdots + n^c \leq n^c + \cdots + n^c \; (n \; times)$
   $= n(n^c) = n^{c+1} = \boldsymbol{O(n^{c+1})}$
5. $T(n) = T(n-1) + c^n = A \; geometric \; progression.$
   $T(n) = T(n-2) + c^{n-1} + c^n = \cdots = T(1) + c^1 + c^2 + \cdots + c^n$
   $c^1 + c^2 + \cdots + c^n = \frac{c^1 - c^{n+1}}{1-c} = \frac{c^1}{1-c} - \frac{c^{n+1}}{1-c} = \frac{c^1}{1-c} + \frac{c^{n+1}}{c-1}$
   $\frac{c^{n+1}}{c-1} = \frac{c^{n+1}}{c} = \boldsymbol{\theta(c^n)}$

Stav Yosef, 316298876

Question 4

Before we dive into the algorithm, I'll present an example of my algorithm (you can check the python code if you find it interesting, https://pastebin.com/SKzvFH1w)



The algorithm:

The algorithm receives an array of prices (real numbers) and returns

1. An array of 2 numbers that represent the buy & sell (=B&S) signals.
2. The minimum value that the algorithm saw so far.
3. The maximum value that the algorithm saw so far.

The algorithm divides the input for 2 arrays and stops when the length of the array is 1 or 2.

If the array length is 1 so the function returns None (= stock should not be traded), and the number in the array as the minimum and the maximum value.

If the array length is 2 so the function checks if arr[1] > arr[0] if so, there is a B&S signal, plus returns the minimum & maximum value of the array.

The closing part of the algorithm involves a lot of if statements, but I'll try to keep it short.

When we have the results of the left partition and the right partition, we'll need to combine the results.

If the left results have a B&S we can try to improve it by checking if the maximum value of the right results is bigger than the current sell signal.
The same goes for the right results B&S with the minimum value of the left results.
We'll return the B&S that produces us the maximum profit.
If one of the results (left or right) is None for the B&S we check if there is a B&S by combining the left minimum and right maximum.
All the exact statements can be seen on the link I added above.

Complexity explanation:

The function receives an array (length = n)
Each divides the function divide into 2 arrays, n/2 n/2
All the other statements are O(1) therefore,

$$T(n) = 2T\left(\frac{n}{2}\right) + c$$

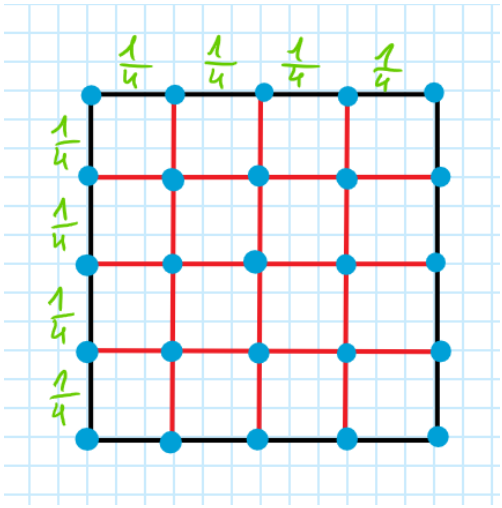
With the Master theorem we can see that

$a = 2, b = 2, k = 0 \rightarrow \theta(n) \rightarrow O(n \log n)$

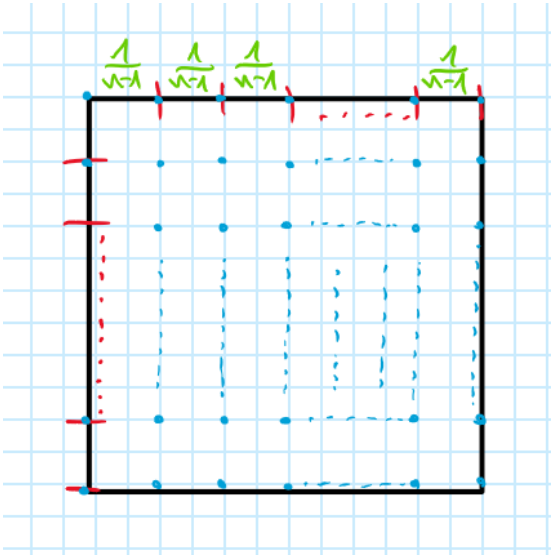
Stav Yosef, 316298876

Question 5

First, let's understand what scenario causes maximization of the closest pair of points. To create a maximum scenario as we want we need to place the points in a way that every two neighbors points have the same distance between them, the correct way to set the points is on a grid as the example below



In the example, we can place 25 points ($N = 25$) in which the distance between the closest pair (there are many) is $\frac{1}{4}$ if we try to move a specific point with one neighbor we'll have a bigger distance but with the other neighbor we'll have smaller distance (less than $\frac{1}{4}$), so this is the best setup of points we can have. Also, we have 5 rows and 5 columns, let's mark it with k, k = 5.
Next, we'll generalize this method for bigger N.



With the same approach, we can place N points over k rows and k columns with the distance between every two neighbors of $\frac{1}{\text{n-1}}$.

Stav Yosef, 316298876

We are talking about big N and let's assume it's a power of two (25, 36, 49, 64, and so on...)

As we can see $\mathrm{n} = \sqrt{\mathrm{N}}$ so, the maximum distance between the closest pair is $\frac{1}{\mathrm{n\text{-}1}} \rightarrow \frac{1}{\sqrt{\mathrm{N}}\text{-}1}$ and in big O notation is equals to $\mathrm{O}\left(\frac{1}{\sqrt{\mathrm{N}}}\right)$

If N is not a power of 2 is bounded between two numbers 2^{k_1} and 2^{k_2} (k_1 is the biggest below N and k_2 is the smallest over N) that sustain the claim we proved before, therefore, the claim is still correct.

Q.E.D