CSE 321 - Homework 4

Seniha Sena TOPKAYA
131044020

(1) I used <u>decrease-by-a-constant-factor</u> method to find minimum number of cuts.

I divide $n/2$ pieces the wire. I put these multiple pieces on top of each other and divided them into 2 at the same time.

<u>Time Complexity:</u>

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

$n^{\log_b a} = n^{\log_2 1} = 1$
$f(n) = c$
$\gg$ They are the same. So: // Master Theorem

$$T(n) = \Theta(n^{\log_b a} \cdot \log n)$$

$$\boxed{= \Theta(\log n)}$$

(2) I assumed the n experiments is an array of size n. Then I assumed the rates of the experiments are values of the array. I implemented 2 functions. One of them finds the worst rate and the other one finds the best.

I called both functions recursively to find the worst/best rate in the left part of the list, to find the worst/best rate in the right part of the list, and to determine which one is worse/better.

<u>Time Complexity:</u>

$$T(n) = [2T(n/2) + 1] + [2T(n/2) + 1]$$
$$= 2T(n/2) + 1$$

<u>Master Theorem</u>

$n^{\log_b a} = n^{\log_2 2} = n$
$f(n) = 1$, constant $\Rightarrow n >$ constant. So,

$$\boxed{T(n) = \Theta(n)}$$

(3) I used Variable-size-decrease approach. I choose the first element of the array as pivot. As we see in the course, I used any partition algorithm and quick select algorithm together. In partition:

- I divided the array 2 halves. Then I compared pivot and the elements in left/right sides. Counters increased/decreased. When I determine that the left is smaller than right element, I swapped.
- In quick selection I called partition function. In terms of k value I found the position of the kth element. I called quick select for left part and the right part of the array recursively.

Time complexity:

n+1 comparison in #1
n " " #2

$T(1) = \Theta(1)$
$T(2) = \Theta(1) + \Theta(2)$
$T(3) = \Theta(1) + \Theta(2) + \Theta(3)$

$$T(n) = \sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

$$= \frac{n^2 + n}{2} \Rightarrow \boxed{T(n) = \Theta(n^2)}$$

(4) I used the algorithm we learned in the lesson. I divided the array into 2 halves I recursively called the function for both halves. Then I defined 2 pointers for 2 halves. Then I compare 2 element from 2 halves to find the smaller one. I created a union set. I appended smaller one into the union set until the other element of the other part of the array becomes the smaller one. ✱ Larger numbers than the larger one are written in the union set without comparison.

Time Complexity: $T(n) \leq 2T(n/2) + n$ // Master Theorem

$n^{\log_b a} = n^{\log_2 2} = n \gg T(n) = \Theta(n^{\log_b a} \cdot \log n) = \boxed{\Theta(n \log n)}$

$f(n) = n$

(5) Brute Force: I compute n times multiplication in a for loop. Each of them constant time. But runs n times in total.

Time Complexity: $T(n) = \sum_{i=0}^{n} 1 = \boxed{\Theta(n)}$

Divide & Conquer: I divided the problem into 2 halves. I check whether the exponential number is even/odd. If it is even I multiplied 2 halves, otherwise I multiplied 2 halves and the number.

Time Complexity: $T(n) = T(n/2) + 1$

$\vdots$

$T(n) = T(n/2^k) + 1 \cdot k \quad (k = \log_2 n)$

$T(n) = T(1) + \log_2 n = \boxed{\Theta(\log_2 n)}$