CSCI 570 - Homework 05

- **Author:** Boyang Xiao
- **Due Date:** Sep. 28^{th} 2022
- **USC id**: 3326-7302-74
- **Email**: boyangxi@usc.edu

Problem 1

(a) Answer: $T(n) = \Theta(n^2 log^2 n)$

$a = 4, b = 2$, and $n^{log_2 4} = n^2$

$f(n) = n^2 logn$

Therefore, $T(n) = \Theta(n^2 log^2 n)$

(b) Answer: $T(n) = n^{log_6 8}$

$a = 8, b = 6$, and $n^{log_6 8} > f(n) = nlogn$

Therefore: $T(n) = \Theta(^{log_6 8})$

(c) Answer: $T(n) = \Theta(n^{\sqrt{6000}})$

$a = \sqrt{6000}, b = 2$, and $n^{log_2 \sqrt{6000}} < f(n) = n^{\sqrt{6000}}$

Therefore, $T(n) = \Theta(n^{\sqrt{6000}})$

(d) Answer: $T(n) = \Theta(2^n)$

$a = 10, b = 2$, and $n^{log_2 10} < f(n) = 2^n$

Therefore: $T(n) = \Theta(2^n)$

(e) Answer: $T(n) = T(log_2 nloglog_2 n)$

Let $k = log_2 n$, then $T(2^k) = 2T(2^{k/2}) + k$

$T(k) = 2T(k/2) + k$

That is: $a = 2, b = 2$ and $k^{log_2 2} = k == f(k) = k$

Therefore: $T(k) = T(klogk)$

And: $T(n) = T(log_2 nloglog_2 n)$

Problem 2

We cut the whole cards into two halfs recursively and decide if there are more than half of these "halfs" are equivalent to each other.

Then we combine these two halfs:

If neither of the two halfs has more than a half that are equivalent, then the combined cards have no more than a half that are equivalant.

If the two halfs both have more than a half that are equivalent, and they all belong to one single account, then the combined cards have more than a half that are equivalent.

If the two halfs both have more than a half that are equivalent, but they belong to different account, then we have to compare each other card with these two big accounts to see if either one of them have more than a half in these combined cards that are equivalent to them.

We do these steps recursively, cut into half, compare and combine and we shall get the result.

Problem 3

We divide all lines into two halfs recursively, find the visible lines in these two halfs recursively and combine these two halfs.

For the combining step, we sort the two halfs in a slope increasing order and find the intersections for the two uppermost boundaries for these two halfs. Then we can extract the upper-left and upper-right part from these two boundaries as the visible lines. This step should take $\Theta(n)$

The runtime complexity should be:

$$T(n) = 2T(n/2) + \Theta(n)$$

That is: $T(n) = \Theta(nlogn)$

Problem 4

We divide a by half recursively, compute each part (actually the two halfs are the same) and combine these two parts together.

That is, for each step of recurssion:

$x^a = x^{a/2} \times x^{a/2} \times x$ or $x^a = x^{a/2} \times x^{a/2}$

We reduce the multiplication times by one for the conqueror part and add three multiplications for the combine part

So the runtime complexity should be:

$T(n) = T(n-1) + 3$

That is: $T(n) = \Theta(n)$

Problem 5

- **Part I: Prove only strings with same length can be J-similar to each other**

Suppose that there are two strings str1 and str2, where str1 is shorter than str2 by one char, and str1 is J-similar to str2. Then these two strings can only comply to the second case of J-similar definitions.

To comply these two strings to the second case, we divide these two strings into two halfs: $str1_a, str1_b, str2_a, str2_b$. Since str1 is one char shorter than str2, then $str1_a$ should have a same size with $str2_a$ and $str1_b$ should be one char shorter than $str2_b$.

Since str1 and str2 are J-similar:

- Case (a): If $str1_a$ is J-similar to $str2_a$, then $str1_b$ and $str2_b$ should also be J-similar too. However, this statement can hold only if the grand statement for this whole problem can hold. Then this is a recursive problem and we have to divide $str1_b$ and $str2_b$ into halfs over and over again. Since $str1_b$ is one char shorter than $str2_b$, when one of them is divided into only one char left and the other has no char left, this statement cannot hold. And the recursion will return false level by level. So this case can not stand.

- Case (b): If $str1_a$ is J-similar to $str2_b$, then $str1_b$ and $str2_a$ should also be J-similar too. Since $str1_a$ and $str2_b$ are of the same size and $str1_b$ is one char shorter than $str2_a$, this is the same problem as we solve in the case(a) and this statement cannot stand.

Therefore, we can only have strings of the same size that can be J-similar to each other.

- ### Part II: Designing an algorithm

FIrst, we divide the given two strings into two halfs. We compare if the first half of the string-1 is J-similar to eihter two halfs of string-2 respectively. If it does, we compare if the remaining two halfs are J-similar. Or if not, then these two string are not J-similar. This step can only have two operations at most for each time.

We do this step recursively until there are only one char left in the divided string. If these two chars are equal, we return true and return the result level by level. Finally we will get the final results for this problem.

The runtime should be:

$T(n) = 2T(n/2) + \Theta(n)$ -> $T(n) = \Theta(nlogn)$

Problem 6

- ### (a)

We find this fixed point in a similar way to binary searching. We divide the array into two halfs, and dicide if the mid point $arr[m]$ is **{greater than / equal to / less than}** m.

- If $arr[m]$ is greater than m, than we do the same search in the first half recursively.
- If $arr[m]$ is less than m, than we do the same search in the second half recursively.
- If $arr[m]$ is equal to m, than we return the mid point and stop searching.

- ### (b)

The runtime: $T(n) = T(n/2) + \Theta(1)$

That is: $a = 1, b = 2$ and we got $n^{log_2 1} = n^0 = 1 == f(n) = 1$

Therefore: $T(n) = \Theta(logn)$

- **(c)**

If we find a fixed point, say $arr[m]$, we only have to check if $arr[m-1]$ and $arr[m+1]$ are fixed points too. If either of them is or both of them are, then $arr[m]$ is not unique. If neither of them is, then none of the other elements could be and $arr[m]$ is definitely unique. This makes a $O(1)$ runtime comlexity.