Problem Set 1

Name: Neeraj Pandey

Collaborators: Raunak Basu

Problem 1-1.

(a)

- $f_1 = 6006_n$
  $\implies$ As the function is asymptotically the largest and comes first in the order.

- $f_2 = (log_n)^{6006}$
  $\implies$ As the function is of the order $O((logn)^x$, it comes third in the order here.
- $f_3 = 6006n$
  $\implies$ As the function is linear function, so it is asymptotically the least and comes last in the order.

- $f_4 = n^{6006}$
  $\implies$ As the function is polynomial, so it is asymptotically comes fourth in the order.

- $f_5 = nlog(n^{6006})$
  $\implies$ As the function is of order $O(nlogn)$, it comes second in the order
  Therefore, the final order is

$$f_3, f_5, f_2, f_4, f_1$$

(b)

- $f_1 = n^{3logn}$
  $\implies$ This function is asymptotically the greatest because it is of the order $O(n^{3logn})$

- $f_2 = (log(logn))^3$
  $\implies$ This function asymptotically comes second in the order because it is of the order $O((log(logn))^3)$
- $f_3 = log((logn)^3)$
  $\implies$ This function is asymptotically the least as it is of the order $O(log(logn))$

- $f_4 = log((3^n)^3)$
  $\implies$ The function is of the order $O(n^3)$, so it is asymptotically fouth in the order.

- $f_5 = (logn)^{log(n^3)}$
  $\implies$ As the function is of order $O(logn^{3logn})$, it comes third in the order
  Therefore, the final order is

$$f_3, f_2, f_5, f_4, f_1$$

(c)

- $f_1 = 3^{2^n}$
  $\implies$ This function comes second asymptotically in the order.

- $f_2 = 2^{2^{n+1}}$
  $\implies$ This function comes first asymptotically in the order.
- $f_3 = 2^{2n}$
  $\implies$ This function comes first asymptotically in the order.

- $f_4 = 8^n$
  $\implies$ This function comes first asymptotically in the order.

- $f_5 = 2^{n^3}$
  $\implies$ This function comes third asymptotically in the order.

The final order is:

$$\{f_2, f_3, f_4\}, f_1, f_5$$

(d)

- $f_1 = \binom{n}{2}$
  $\implies$ The function is of the order $O(2^n)$

- $f_2 = n^2$
  $\implies$ The function is of the order $O(n^2)$
- $f_3 = 2^{2n}$
  $\implies$ The function is of the order $O(n^2)$

- $f_4 = \binom{n}{\frac{n}{2}}$
  $\implies$ Here, using Stirling Approximation

$$\left(\frac{n!}{\frac{n}{2}!\frac{n}{2}!}\right)$$

$$\implies O(2^n/\sqrt{n})$$

- $f_5 = 2(n!)$
  $\implies$ The function is of the order $O(n!)$

Therefore, the final order is

$$\{f_2, f_3\}, f_4, f_1, f_5$$

Problem 1-2.

(a) For k = 5

$$T_n = \begin{cases} O(1) \text{ if } n = 1, 5T(\frac{n}{5}) + C_n \text{ if } n > 1 \end{cases}$$

(b) Given, $a = 5$ and $b = 5$ and $f(n) = n$
According to case 2 of Master Theorem,

$$n^{log_a^b} = n^{log_5^5} = 5 = f(n)$$

Therefore, solution to the recurrence is of the order $O(nlogn)$

(c) For $k = \sqrt{n}$,

$$\begin{cases} O(1) \text{ if } n = 1, \sqrt{n}(T(\sqrt{n}) + C_n) \text{ if } n > 1 \end{cases}$$

Problem 1-3.

(a) $T(n) = 2T(\frac{n}{2}) + O(2^n)$
Using Master algorithm,

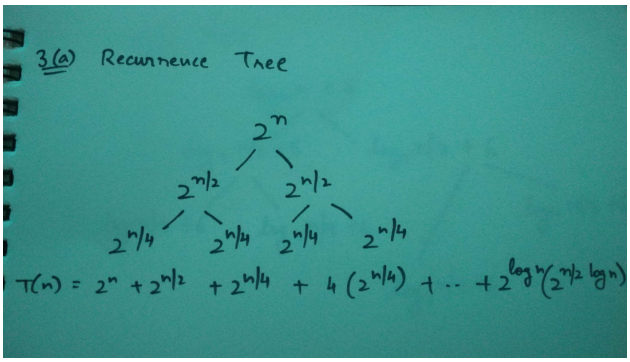
$$a = 2 = 2$$

$$n^{log_{a^b}} = n^{log_{2^2}} = n$$

$f(n) = 2^n \implies$ This is a polynomial and asymptotically it is greater than n

Using case 3 of masters theorem

$\implies$ solution to recurrence is $O(2^n)$



(b) $T(n) = 3T(\frac{n}{9}) + O(\sqrt{n})$
Using Master Theorem,

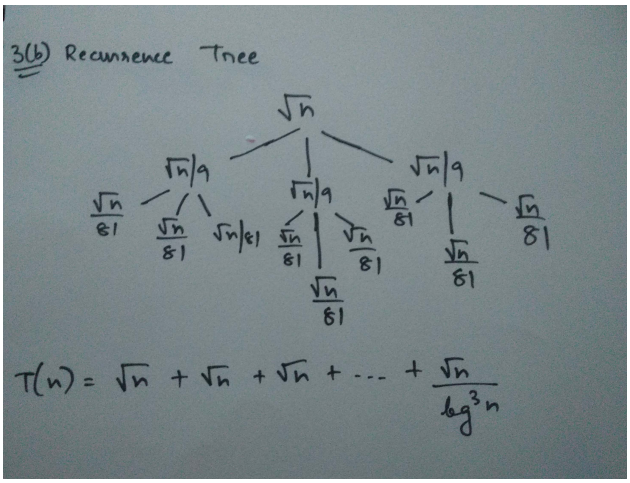
$$a = 3, b = 9$$

$$n^{log_{b^a}} = n^{log_9 3} = n^{\frac{1}{2}}$$

$f(n) = C\sqrt{n} \implies$ it is asymptotically equal to n , $i.e : n^{log_{b^a}}$

Using case 2 of Master Theorem

$\implies$ solution to recurrence is $O(\sqrt{n}logn)$



(c) $T(n) = 2T(\frac{n}{3}) + log_7 n + 6$
Using Master Theorem,

$$a = 2, b = 3$$

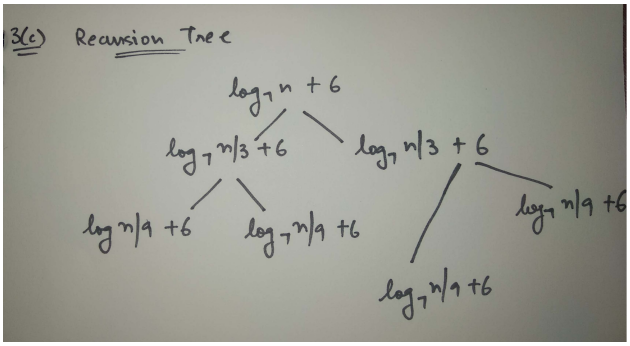
$$n^{log_b a} = n^{log_3 2}$$

$$f(n) = log_7 n + 6$$

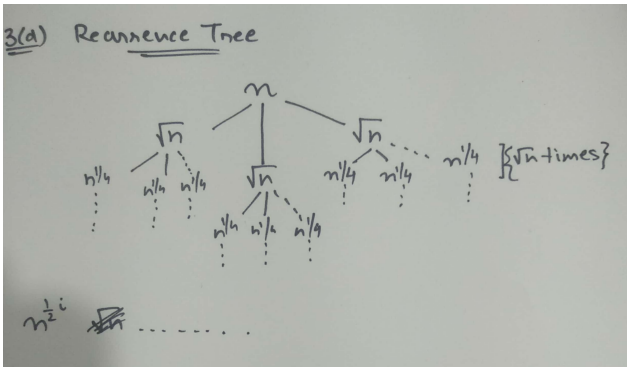
f(n) is polynomially greater than $n^{log_b a}$

Using case 3 of master theorem,

solution to recurrence is $O(log_7 n + 6)$



(d) Recursion tree is below:



Problem 1-4.

(a) Here, we are taking the skylines as geometrical rectangle shapes. The naive incremental algorithm will run with a time complexity of $O(n^2)$ for n steps. Starting from some point, we mark all points for the buildings separately as keys at the start of the horizontal line segment (rectangle). Now, we add all the points to update the skyline. The kth element is unknown upto the k - 1 lines in the skyline which is formed by k - 1 rectangles, so the list requires $O(k)$ in it's worst case scenario to update the list. So, the resulting time complexity will be $O(n^2)$. This approach will require more time as it will run the program many times to add one rectangle.

(b) Start with a point (rectangle), and take it's height as the skyline height (the largest building). Move to the end of the building, and remove the building from the list. If this end point is the highest point then it will be the skyline point as well and add it to the priority queue. If the buildings in front are taller then don't add the skyline point. An easy understanding can be, make a list and add each element to it. If we have a start or end position of the building (rectangle) and it is taller/bigger than the right rectangle, remove the current element from the data type (priority queue) and add the other element to the datatype. If the height of the last added element is different from the height of the first element, add the new point as skyline point as height has been changed. This algorithm will have a time complexity of O(n).

(c) The divide and conquer method is the best case to produce a skyline in O(n log n) time. The way we implement the divide and conquer algorithm is as follows.

First, the program will sort all the index points in the array x_left and x_right. The program will divide the buildings into two halves and then recursively create a skyline for those two halves. After the skyline for both halves is made, it will merge those skylines to create a final result. Starting from the left most point of both created skylines, if the left most points are not the same, it will update the height to the max height of the pointer in the skyline

Now, as the pointer moves from left to right, if and when the next index overlap, it shall update the height to the maximum of the two and then and then append the pointer to this index. To finish the program, it will proceed with the same steps given above until the end for both skylines is reached.

Since this algorithm is in a similar format to that of the merge sort algorithm, it will take O(n log n) time to create a skyline with n buildings.

(d) Submit your implementation online.