21k-3881

Design And Analysis (Algo)

ASSIGNMENT 02 BSE-5A

(Q1) Problem 1

We have two lists i.e weekly sales, target sales

Now, we start by sorting the array based on the time complexity O(nlogn) search algorithm under divide and conquare aproach to find the sales of two weeks which represents our target sale value.

To Find items from array, we apply O(nlogn) search on the sorted array by traversing the array from start to end (low to high) by dividing the array in two sub parts. or subarrays.

The left sub half array is traversed to find the definite weeks sales or target value, then it will check for the right half subarray and afterwards combining both the subarrays.

If the sales for two definite weeks is equal to the target sale value, the function returns true and if not

21k-3881

then return false.

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

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + O\left(\frac{n}{2}\right)\right) + O(n)$$

$$T(n) = 4T\left(\frac{n}{4}\right) + O\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = 2\left[4T(n/8) + O(n/4)\right] + O\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = 8T\left(n/8\right) + O\left(n/4\right) + O\left(n/2\right) + O(n)$$

$$\Rightarrow T(n) = 2^3 T\left(\frac{n}{2^3}\right) + O\left(\frac{n}{2^2}\right) + O\left(\frac{n}{2^1}\right) + O(n)$$

$\therefore$ let $k = 3$

$$\therefore T(n) = 2^k T\left(\frac{n}{2^k}\right) + O\left(\frac{n}{2^{k-1}}\right) + \cdots + O(n)$$

$\because n = 2^k$

$k = \log_2 n$

$$\Rightarrow T(n) = 2^{\log_2 n} T\left(\frac{n}{2^{\log_2 n}}\right) + O\left(\frac{n}{2^{\log_2 n - 1}}\right) + \cdots + O(n)$$

21k-3881

$T(n) = O(n \log_2 n)$

Problem 2:

We sort the bakery sales items list by using the time complexity $O(n \log n)$ i.e merge sort and sort out the elements based on their priority. For bakery 1 and 2 we have two separate lists for each bakery.

Now, sort the lists from the bakeries by dividing them into two sub arrays one for each bakery and traverse both lists of bakery finding suitable values from the respective lists and pairing them to meet the target sales value. Check recursively for suitable values in both subarrays.

To search pair, $O(n \log n)$ search is applied and traverses both ~~arrays~~ bakeries lists to find the target value of items. If the pair with target value of sales is found so the function returns true and other wise false;

21k-3881

After recursively traversing the whole lists of array using divide and conquer approach.

$$T(n) = 2T\left(\frac{n}{2}\right) + 2O(n)$$
$$T(n) = 2\left[2T(n/4) + c\frac{n}{2}\right] + cn$$

$$T(n) = 4T(n/4) + cn + cn$$
$$T(n) = 4T(n/4) + 2cn$$
$$T(n) = 4\left[2T[n/8] + \frac{cn}{4}\right] + 2cn$$
$$T(n) = 8T(n/8) + cn + 2cn$$
$$T(n) = 8T(n/8) + 3cn$$
$$T(n) = 2^3 T\left(n/2^3\right) + 3cn$$

Let $3 = k$

$$T(n) = 2^k T\left(n/2^k\right) + kcn$$

$\therefore 2^k = n$

$\Rightarrow k = \log_2 n$

$$T(n) = nT\left(n/n\right) + cn\log_2 n$$

$$T(n) = nT(1) + cn\log_2 n$$
$$T(n) = n + cn\log_2 n$$
$$O(n\log_2 n)$$

21k-3881

Q2)

To search a single character in the word string where all other letters appears twice consecutively, we apply binary search algorithm.

Identify the midpoint in the given array. In the example xxyyzzwwii, here the midpoint is 6th value i.e (z) so for the next value of midpoint, if both characters are same so the given function is true and by using pointer based approach update the left pointer to mid+2 b/c the single character lie in right array half. In the example of word = xxyyzzwwii, the midpoint = z is not equal to element of next position so update right pointer as mid b/c no single character exists.

Now, the same steps are repeated until both the pointers are equal to each other and that position, single character lies.

21k-3881

$$T(n) = T(n/2) + O(1)$$

$$T(n) = T(n/4) + O(1)$$

$$T(n) = T(n/8) + O(1)$$

$$T(n) = T(n/2^3) + O(1)$$

$$\Rightarrow 3 = k$$

$$T(n) = T\left(\frac{n}{2^k}\right) + O(1)$$

Let $n = 2^k$

$k = \log n$

$$T(n) = n/2^{\log n}$$

$$T(n) = n - 2^{\log n}$$

$$T(n) = n - \log n \quad (2)$$

$O(\log n)$

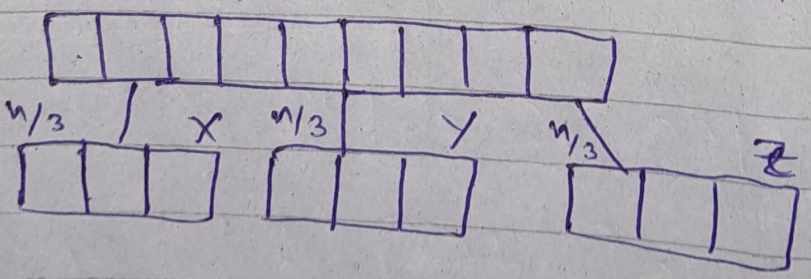
21k-3881

(Q3)

(a) 3 jars, heaviest = 3^n, $n/3$ jars
Complexity $< O(n)$

By applying divide and conquer algorithm, we try to divide the 3^n size of jar into 3 groups, each of $\frac{n}{3}$ size. To determine the heaviest of the 3 jars, we compare them and further on eleminating the remaining jars and until we are left with only remaining jar which is heaviest.

(b) Now, we start ~~&~~ sorting by defining the array by reducing into 3 sub arrays in functions $\{[l-h]/3\}$



We ~~subdivide~~ iterate the whole 3^n array in 3 groups where $X = \{h \ldots 2+g-size\}$,

21k-3881

$Y = \{h+g-size+1 \dots h+2*g-size\}$,
$Z = \{h+2*g-size+1 \dots 1\}$, where X shows sub array of left to mid, Y shows sub array of mid to right and Z shows the whole array of left to right.

If weight of $(X == Y)$, we will obtain 3^n sized array, if $(X > Y)$, we will obtain the array of left to mid and if $(A < B)$ we will obtain the array of mid to right.

(C) $T(n) = T(n/3) + O(1)$

$T(n) = T(n/9) + 1 + 1$

$T(n) = T(n/9) + 2$

$T(n) = T(n/27) + 2 + 1$

$T(n) = T(n/27) + 3$

let $k = 3$

$T(n) = T(n/3^k) + 3$

$\Rightarrow k = \log_3 n$

$T(n) = T(n/n) + \log_3 n$

$T(n) = 1 + \log_3 n$

It is proved that

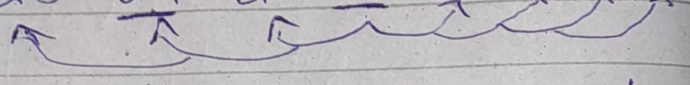
$O(\log_3 n) < O(n)$

21k-3881

Q4)

Meta Binary Search:

21 22 23 24 25 26 27 28 29 30



=> Now, if mid value = target value, search terminates

=> If mid value > target value, so find new interval to left half of 24 (previous interval)

=> If key to search = 26, so index value is = 25 (out put is found)
A(9) = 30, if target not found update previous position

Ternary Search :-

21 22 23 24 25 26 27 28 29 30

mid1 mid2

=> Now find 26, [l+h/3] -> traversing every 3 elements

21k-3881

Now, we check about whether key = mid1 value
If it is not equal then compare with mid2
Check if key > value at mid1 and key < value at mid2, keep on checking recursively until the point where (mid1 = mid2), key = Mid 1

→ By using ternary search, decreases the time complexity as compared to meta binary search b/c it involves n iterations which make $O(\log_3 n)$. For Meta Binary Search, reduces no. of comparisions needed in binary searching which enhances its time complexity.

21k-3881

Q5)

We can handle O(n+k) size by using count sort algorithm.

First of all, the whole array with 0's. The array having (k+1) size.

Treverse the whole array calculating iterations of array. Further increment the count arr[i] by 1.

Update count[i] after calculating the sum of values in count iterating from (0..i).

Return count[b] if (a==0) for all the values in array.

If (a != 0), return count(b) - count[a-i] for all values in an array.

By traversing the whole array, we obtain time complexity of O(1) time.

21k-3881

Q6)

ihab, abid, afif, fadi, adib, hadi, ibad

By using Bucket sort algorithm:

Divide into 2 parts.

(1, 2, 3, 4) (5, 6, 7)

(1) a → abid, afif

b

c

d

e

f → fadi

g

h

i → ihab

(2) a → adib

b

c

d

e

f

g

h → hadi

i → ibad

Apply sorting in ascending order

a → abid, adib, afif

b

c

d

e

f → fadi

g

h → hadi

i → ibad, ihab

21k-3881

If we sort the list in ascending order, the output will be
abid, adib, afif, fadi, hadi, ibad, ihab

O(n) complexity is obtained