Class starts @ 7:07 am

DSA: Lab Session on Heaps and Greedy

Aravind Kumar S

DSA SME @ Scaler.

↳ Real time Problem Solving Situations.

↳ Solving problems.

Structure:

Pick a problem

↓

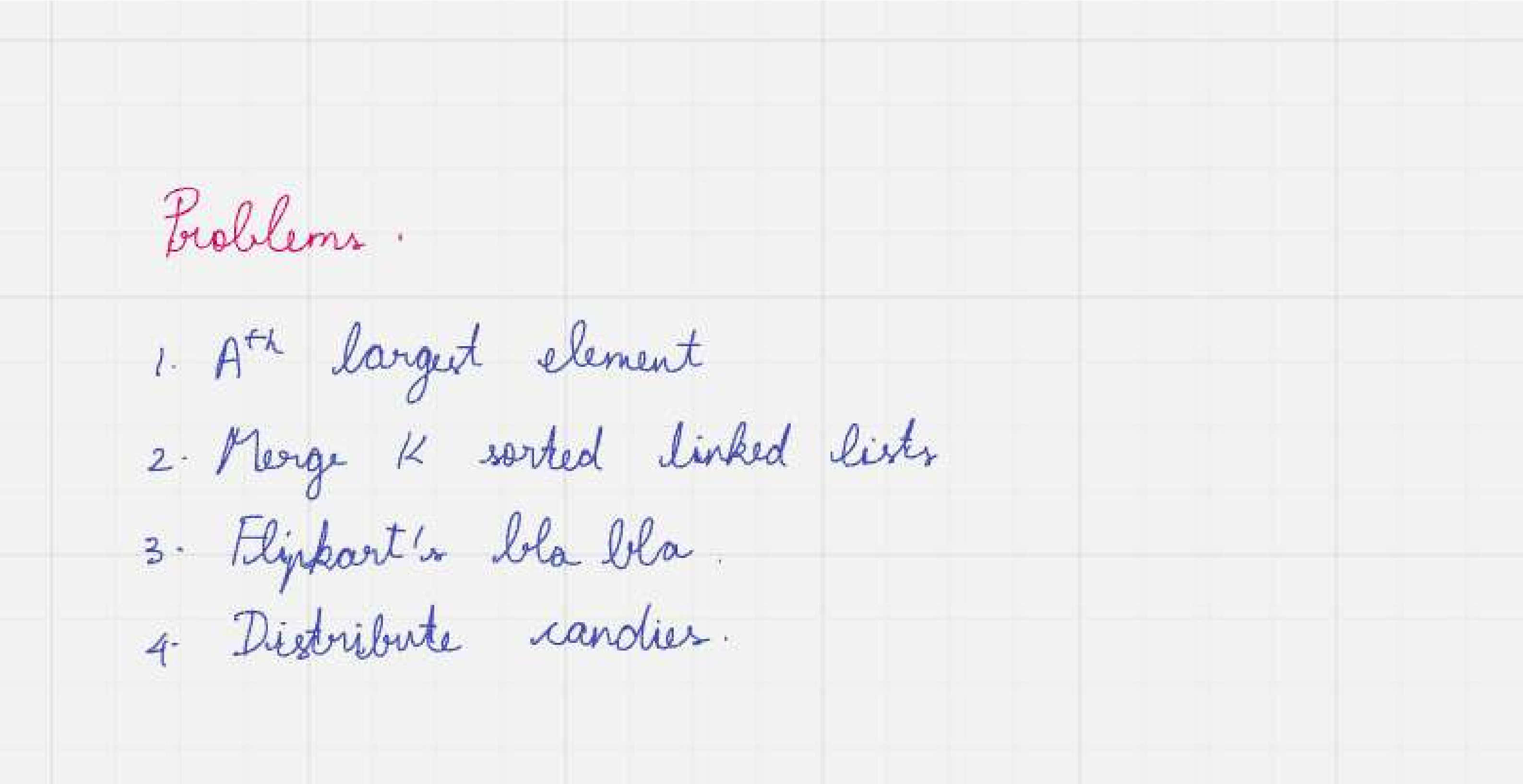
Quick read

↓

Try to solve it → (Doubt, app dis)

↓

Discussion of Solutions }



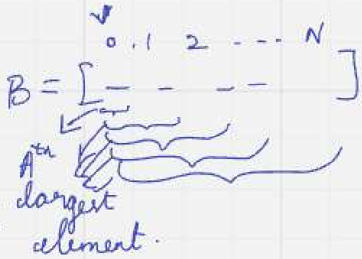
Problems .

1. Ath largest element
2. Merge K sorted linked lists
3. Flipkart's bla bla .
4. Distribute candies .

7: 33 am

Question 1) A^{th} Largest Element

Given an array B of length N, find then A^{th} largest element of every window that starts from index 0, and ends at all indices.

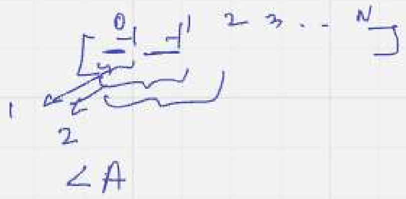


(i) s = 0 ✓

(ii) $0 \le e < N$

(iii) If window size is less than A, return -1

A = 3



res = [A^{th} A^{th} A^{th} A^{th}]

Example 1)

A = 3 ✓ 3rd

$$B = \left[\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 10 & 18 & 7 & 5 & 16 & 19 & 3 \end{array}\right]$$

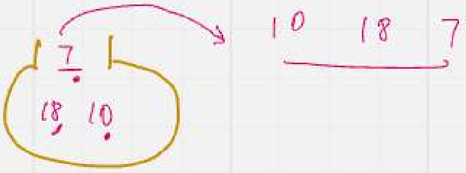
res = [-1 -1 7 7 10 16 16]

A = 3

7:38

HINT 1 ☆

Min Heap



HINT 2 ☆

☆ maintain the size of the heap at A.

Pseudocode)

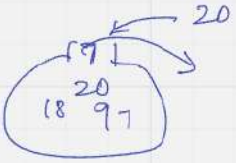
```
int[] solve (int[] B, int A){
    int N = B.length;
    int[] res = int[N];
    res.fill(-1);  — N

    PriorityQueue<Integer> q = new PriorityQueue<>();

    for (int i = 0; i < N; i++){  → N
        q.add(B[i]);
        If (q.size() < A) continue;
        If (q.size() > A) q.remove();
        res[i] = q.peek();
    }

    return res;
}
```

3

-1 -1 -1

A = 3

(S > A
S == A) } 2 log(N)



S == A

TC ⇒ O(NlogN)

SC ⇒ ~~O(N)~~ O(K) because the question has demanded the auxillary array.

Try to not use the auxillary array.

Question 2) Merge K sorted linked lists

Given K sorted linked list, merge them together as a single sorted linked list.

8 : 19 am

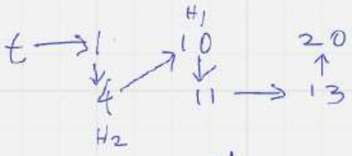
comparator
object / class → (Pair)
Parse Int ("234")
slicing
accessing DS
lambda expressions.

Example 1)

NK (log K)

A = [
K { L1 → 1 → 10 → 20,
L2 → 4 → 11 → 13,
L3 → 3 → 8 → 9
]

→ N (upper bound)



N

K

- 1 → 10 → 20,
- 4 → 11 → 13,
- 3 → 8 → 9

→ 2N

2N + N ⇒ 3N

$2N + 3N + 4N + \ldots\ldots KN$

$N(\underbrace{2+3+4 \ldots\ldots k})$

$N\left(\underbrace{K \times \frac{(k+1)}{2}}\right)$

$N K^2$

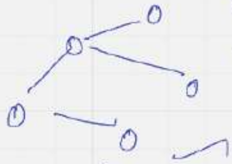
$TC = (NK^2) >>>$ ✗

HINT 1 : min Heap

HINT 2 ✶ :

min (P_1, P_2)

min (P_1, P_2, P_3 ..... P_k)



auxillary array / list ✗

( ) C++

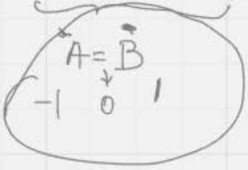
( ) → Pyt

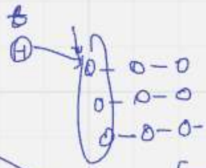
8 : 20 am

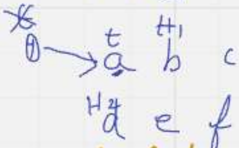
Pseudocode)

```
ListNode solve (List <ListNodes> nodes) {

    PriorityQueue <ListNode> q = new PriorityQueue<> ((a, b) -> a.val - b.val)
    ListNode dummy = new ListNode (-1);  -> hack
    ListNode temp = dummy;

    while (!q.isEmpty()) {
        ListNode smallNode = q.remove();
        temp.next = smallNode;
        temp = temp.next;
        If (smallNode.next != null) q.add(smallNode.next);
    }

    return dummy.next;
}
```

{add all first elements in the pq}







K -> no. of lists

N -> length of each list

TC => O(NK logk)

SC => O(K)

Break till 8:45

9 : 03

Question 3) Flipkart's Challenge.

You are given with 2 arrays A and B of size N. Here A[i] is the expiry time of a particular product, and B[i] is the profit margin of that product if sold.

$P_0 \quad P_1 \quad P_2 \cdots P_N$

$A = [ET_0 \; ET_1 \; ET_2 \cdots ET_N]$

$B = [PM_0 \; PM_1 \; PM_2 \cdots PM_N]$

Your task is to maximize the profit by selling the products before the expiry time.

ct >= Et.

P_0 ct = 2, 3, 4 ....

Et 2

PM = 1000

(i) Selling a product takes 1 unit of time.

(ii) A product cannot be sold if the current time >= expiry time.

ct >= 3 ✓

Example 1)

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & \overset{v}{4} \\ A = & [\,1 & \underline{3} & \underline{3} & 3 & 2\,] \\ & & & & \times & \times \\ B = & [\,\underline{5} & 6 & 9 & \cancel{3} & 1\,] \end{array}$$

ct = ~~0~~ ~~1~~ ~~2~~ 3

P = 0 + 5 + 6 + 9

₹20 ⇒ max ?

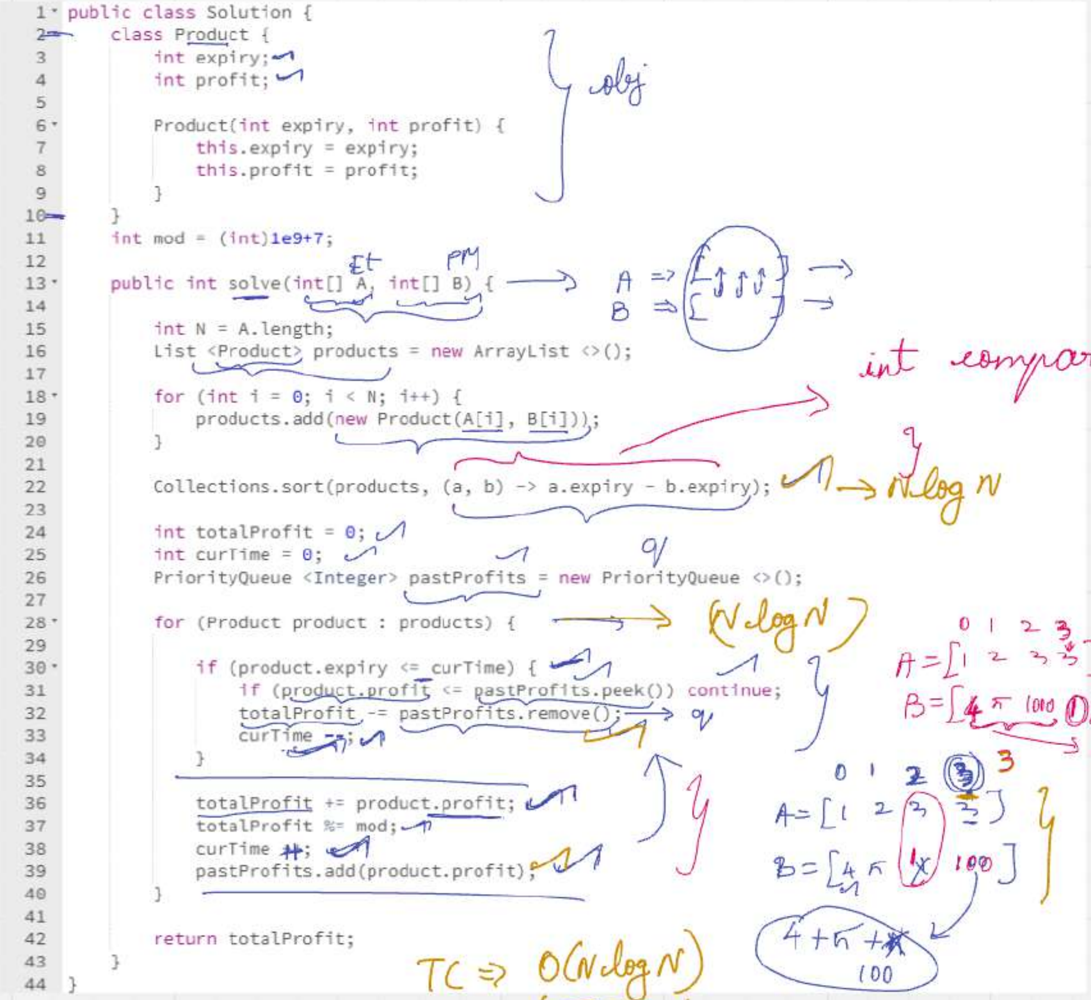
HINT 1 ☆ : Sorting

HINT 2 ☆ : ✗DP ≈ LIS ✗ ~~DP~~

HINT 3 ☆ : Sort based on expiry time

HINT 4 ☆ : minHeap

Pseudocode)



```
public class Solution {
    class Product {
        int expiry;
        int profit;

        Product(int expiry, int profit) {
            this.expiry = expiry;
            this.profit = profit;
        }
    }
    int mod = (int)1e9+7;

    public int solve(int[] A, int[] B) {

        int N = A.length;
        List <Product> products = new ArrayList <>();

        for (int i = 0; i < N; i++) {
            products.add(new Product(A[i], B[i]));
        }

        Collections.sort(products, (a, b) -> a.expiry - b.expiry);

        int totalProfit = 0;
        int curTime = 0;
        PriorityQueue <Integer> pastProfits = new PriorityQueue <>();

        for (Product product : products) {

            if (product.expiry <= curTime) {
                if (product.profit <= pastProfits.peek()) continue;
                totalProfit -= pastProfits.remove();
                curTime --;
            }

            totalProfit += product.profit;
            totalProfit %= mod;
            curTime ++;
            pastProfits.add(product.profit);
        }

        return totalProfit;
    }
}
```

java.util.*

int compare(—,—){

TC ⇒ O(N log N)

SC ⇒ O(N)

Qustion 4) Distribute Candies

N children are standing in a line.
Each child is assigned with a rating.
Your task is to distribute candies to the children in such a way that:

(i) Every child ~~has~~ must have atleast 1 candy.

(ii) Children with higher ratings, should have more candies than their immediate neighbors.

Find the minimum number of candies you have to distribute to satisfy the above criteria.

$C_1 > C_0$ $\quad C_1 > C_2$

more cand - - -

$$A = \begin{array}{ccccc} 0 & 1 & 2 & \cdots & N \\ \left[C_0 \right. & C_1 & C_2 & \cdots & \left. N \right] \end{array}$$

marks

Example 1)

$$A = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ \left[1 \right. & 5 & 2 & \left. 1 \right] \end{array}$$

$$res = \begin{bmatrix} 1 & \cancel{1} & \cancel{1} & 1 \\ & \cancel{2} & 2 & \\ & 3 & & \end{bmatrix}$$

$1 + 3 + 2 + 1 \Rightarrow 7$

$[1 \quad 5 \quad 2 \quad 1]$

ry $[\_ \quad \_ \quad \_ \quad \_]$

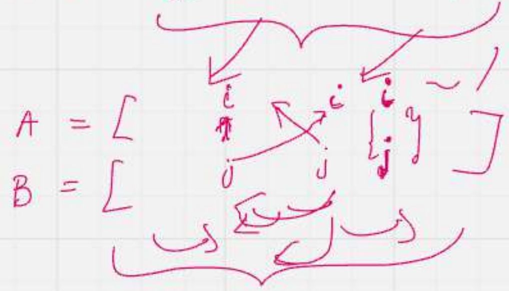
$[- \quad - \quad - \quad -]$

$[1 \quad 5 \quad 2 \quad 1]$

$$\begin{bmatrix} 1 & \cancel{1} & \cancel{1} & 1 \\ & \cancel{2} & 2 & \\ & 3 & & \end{bmatrix}$$

int[][] => [[Et, PM] [Et, PM]]

Pseudocode)



```
int solve ( int[] A){
    int N = A.length;
    int[] res = new int[N];
    res.fill(1);
                                   left → right
    for (int i = 1; i<N; i++) {                   ⟵ neigh
        If (A[i] > A[i-1]) {
            res[i] = res[i-1] + 1; → stringent (greedy).
        }
    }
                                   right → left
    for (int i = N-2; i > -1; i--) {              ⟶ neigh
        If (A[i] > A[i+1]) {
    If (res[i] <= res[i+1]) res[i]= res[i+1] + 1; → stringent
        }
    }                                   res[i] => candies
    return sum(res);   res  sum
}
```

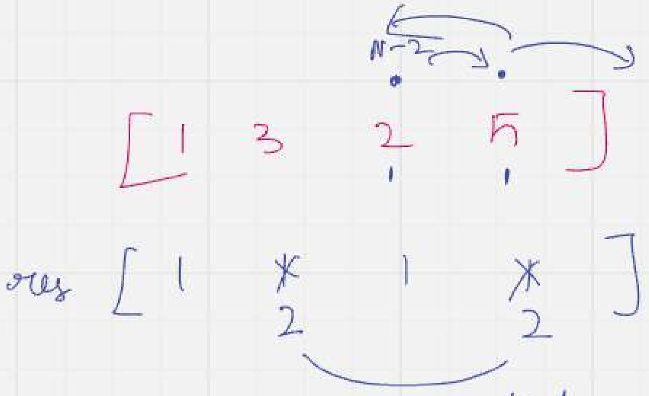
A[i] >A[i+1]

res[i] = res [i+1]+1

res[i] > res [i+1]

$TC = O(N)$

$SC = O(N)$



If (i == N-1) skip

X immediate neighbors .