Todays Content:

1) N Sorted arrays
   - print completed data in sorted order
   - store complete sorted data in single array

2) Given 2 sorted array find k smallest pair sum

1) Given 2D mat[N][M], every row is sorted, merge entire data into 1D sorted list & return sorted list

N * M

Eg: mat[4][6]

	0	1	2	3	4	5
0	2	7	10	17	25	34
1	-6	0	1	8	11	16
2	3	4	6	14	21	26
3	7	10	14	19	23	27

merged 1D sorted array

-6	0	1	2	3	4	.....

Idea

1) Insert all elements in a list sort & return it
   // elements = N*M
   TC: NM log(NM) SC: O(1)

2) Apply merge ~ N times
   TC: TC: O(MN^2)

r_1 } 2M
r_2 }
r_3 — M } 3M
r_4 ..... M } 4M
r_5 ..... M } 5M
:
r_{n-1} ..... } (N-1)(M)
r_n ..... M
N * M

2M + 3M + 4M + .. N*M

TC: M(2 + 3 + .. N]

: M(1 + 2 + 3 .. N - 1)

: $M\left[\frac{(N)(N+1)}{2} - 1\right]$

TC: O(MN^2)

ideal: Pointer approach

0	1	2	3	4	5	
2	7	10	17 (p_1)	25	34	
-6	0	1	8	9	10	(p_2)
3	4	6	14 (p_3)	21	26	
7	10 (p_4)	14	19	23	27	

Idea:

N pointers, every iteration print pointer with min val & move pointer. continue process till all pointers reaches end

output: -6 0 1 2 3 4 6 7 7 8 9 10 10 . . . ?

How to implement? ⟶ Use min heap

1. for every number we should also store its row number.
2. for every number we should also store its col number.

0	1	2	3	4	5
2	7	10	17	25	34
-6	0	1	8	11	16
3	4	6	14	21	26
7	10	14	19	23	27

<2,0,0> <0,1,1>
<-6,1,0> <1,1,2>
<3,2,0> <8,1,3>
<7,3,0> <7,0,1>

output:

-6 0 1 2 ...

r c r c r c r c

<-6,1,0> <0,1,1> <1,1,2> <2,0,0>

⟶ Each data point:

: pair< data, pair< row, col >>

↳ pair< int, pair< int, int >>

↳ // please check how to do in your language of choice?

data row, col

minheap < pair< int , pair< int, int > > > mh.

a) Single element in minheap
b) It is sorted based on data

a) Custom class
b) Override comparator

```
int[]  Merge(int mat[N][M]){

    minheap< pair<int, pair<int,int>>> mh
    for(i=0; i< N; i++){
      // value we need insert : mat[i][0]
       mh.insert( pair( mat[i][0], pair(i, 0)))
    }
    list<int> ans;
    while(mh.size()>0){
                                  row  col
        pair<int, pair<int,int>> data = mh.getMin()
             first     second
        mh.deleteMin()
        int val = data.first
        int r = data.second.first, c = data.second.second
        ans.add(val) // insert in list

        if(c+1<M){
          mh.insert( pair( mat[r][c+1], pair(r, c+1)))
        }
    }
    return ans;
}
```

{ 8:50 }

Obs1: heap size = N TC: $N*M\{\log N\} + N*M\{\log N\} \Rightarrow O(NM\log N)$

SC: $O(N)$

28) K Smallest Pair sums:

Given 2 sorted arrays, print k smallest Pair sums, a single pair only 1 time

A[i] + B[j]

N	0	1	2	3	4	5	6
a[5]:	2	5	8	11	13		
M							
b[7]:	3	7	9	12	15	16	20

K = 5 output

Pairs:	Sum	A[] i	B[] j
1	5	0	0
2	8	1	0
3	9	0	1
4	11	0	2
5	11	2	0

Ideal: Store all pair sum in a list sort it & get first k elements.

- pairs = N*M
- TC: N*M log N*M SC: O(N*M)

Ideal: Optimization using [maxheap & size = k]

Insert all pairs in maxheap & get only min k pair sums in heap

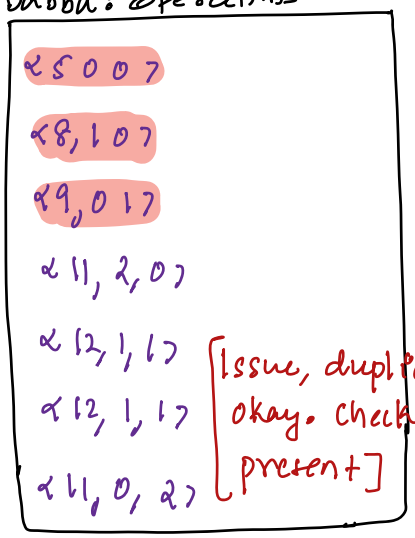
- TC: N*M log k SC: O(k)

Idea:

	0	1	2	3	4	5	6
a[5]:	2	5	8	11	13		
b[7]:	3	7	9	12	15	16	20

Pairs:	Sum	i	j
1	5	0	0
2	8	1	0
3	9	0	1

Dabba: operations



[Issue, duplicate insertion is not okay. Check if pair is already present]

delete i j	Insert i+1 j	i j+1
0, 0 →	1, 0	0, 1
1 0 →	2 0	1 1
0 1 →	1 1	0 2

Dabba:

- getmin()
- deletemin()
- insert()
- search() → hashset

} minheap

} ② Use both minheap & hashset

// Store i, j as a pair

// Convert pair into single string

→ String s = "i _ j"

```
void kpairs(int a[], int b[]){
    int n = a.length, m = b.length;
    minheap< pair<int, pair<int, int>>> mh;
    hashset<String> hs // To store pairs
    mh.insert( pair( a[0]+b[0], pair(0,0)) ); // Insert min pair :
    hs.insert (0+" "+0); // inserting in hashset
    for (int i=1; i<=k; i++){
        pair<int, pair<int, int>> d = mh.getmin();
        mh.deletemin()
        int val = d.first, r= d.second.first, c= d.second.second
        print(val)
        // for r, c ---posibilities---> {r+1, c}, {r, c+1}
        if ( c+1 < M && hs.search( to_string(r) + " " + to_string(c+1) == false)
            val = a[r]+b[c+1]
            mh.insert( pair(val, pair(r, c+1))
            hs.insert( to_string(r) + " " + to_string(c+1))
        }
        if ( r+1 < N && hs.search(to_string(r+1) + " " + to_string(c) == false)
            val = a[r+1]+b[c]
            mh.insert( pair(val, pair(r+1, c))
            hs.insert (to_string(r+1) + " " + to_string(c))
        }
    }
}
```

TC: $k * [\log k + \log k + \log k] = O(k \log k)$ SC: k

2 insertions, 1 deletion