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) Merge N Sorted lists into single sorted list

N M → col index : [0 M-1]

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

Solutions:

1) Store all N*M elements in a 1D array & directly sort it

TC: $CNM + NM\log(NM)$

SC: $N*M$ → Merge sort: $O(N*M)$ / Quick sort: $O(1)$

2) Store all N*M elements in a min-heap, & delete 1 by 1 element insert in new 1D array (Insert, delete)

TC: $N*M\log(N*M) + N*M\log(N*M)$

SC: $N*M$ → max heap size

3) Apply merge on arrays itself,

In N rows → merge first N/2 rows → 1D array ($n/2$)
merge last N/2 rows → 1D array ($n/2$)
} merge them

Total assume n elements

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

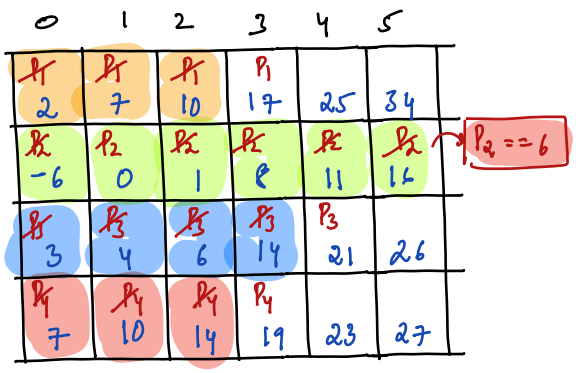
$\boxed{T(n) = 2T(n/2) + n}$

$T(n) = n\log n$

// $n = N*M$

$T(N*M) = N*M\log(N*M)$

idea: Pointer approach



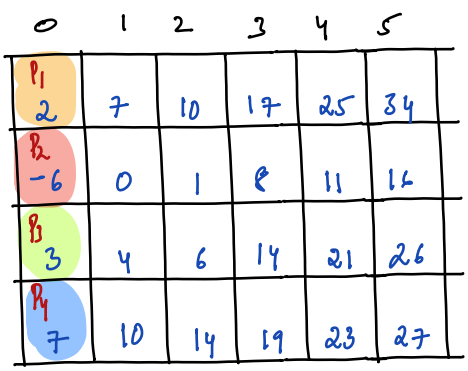
→ Idea: N pointers, at every step pick pointer with min value store it in 1D array & increment the pointer.

Repeat the above process till all elements are inserted in 1D-array

output

-6 0 1 2 3 4 6 7 7 8 10 10 11 14 14 16 ....

How to implement?



→ Idea, min_heap

obs1: We need to know which row an element belongs to

obs2: We need to know current index in its row, col number

Data we need to insert:

val, rownumber, colnumber

For tracing say we are storing using 1st way

1) pair<int, pair<int, int>>, as this a single element in heap

2) Implement your custom class

```
class data{
    int val;
    int row;
    int col;
};
```

minheap<data> mh;

3) pair<int, int> — val, [row*M + col]

/M ⇒ row number

%M ⇒ colnumber

Trace:

	0	1	2	3	4	5
→	P_1 2	P_1 7	10	17	25	34
→	P_2 -6	P_2 0	P_2 1	P_2 8	11	16
	P_3 3	P_1 4	6	14	21	26
	P_4 7	10	14	19	23	27

minheap < pair< int, pair<int, int>>

(val, r, c)

<2, 0, 0> <2, 0, 1>

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

<3, 2, 0> <8, 1, 3>

<7, 3, 0> <7, 0, 1>

<4, 2, 1>

Step:

	getMin() val	r	c	insert() val	r	c
1	-6	1	0	0	1	1
2	0	1	1	1	1	2
3	1	1	2	8	1	3
4	2	0	0	7	0	1
5	3	2	0	4	2	1

idea: do the above approach N*M

TC: O(NM log N + NM log N)

insert in heap

getmin in heap

SC: N → max size of heap

```
int[] Merge(int mat[][]){
    int n = mat.length; int m = mat[0].length;
    int in[N*M]; int ind = 0 { at which index in in[] we are going to insert}

    minheap< pair<int, pair<int, int>>> mh;
    // Step-1 insert entire 0th col
    for(i=0; i<N; i++){
        ele = mat[i][0]
        mh.insert(pair(ele, pair(i, 0)));
    }

    while(mh.size()>0){
        pair<int, pair<int, int>> d = mh.getmin();
        mh.deletemin()
        int val = d.first, r = d.second.first, c = d.second.second
        in[ind] = val; ind++
        // We need to insert next element in that row
        if(c+1<M){
            val = mat[r][c+1]   // We need to check if next element is present
            mh.insert(pair(val, pair(r, c+1)))
        }
    }
    return in;
}
```

10:30 - 10:40

TC: $N*M \log N$ SC: N

28) K Smallest Pair sums:

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

A[i] + B[j]

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

$N <= 10^5$
$M <= 10^5$

K = 5 output

Pairs:	Sum	i	j	
1	5	0	0	
2	8	1	0	
3	9	0	1	
4	11	0	2	? both are diff pairs
5	11	2	0	?

ideas:

1) Store all pairs sum in 1D array sort & get 1st k elements

given a[N], b[M]
int c[N*M], ind = 0
i = 0; i < N; i++) {
  j = 0; j < M; j++) {
    c[ind] = a[i] + b[j]
    ind++
  }
}
Sort c[] & get [0, k-1]

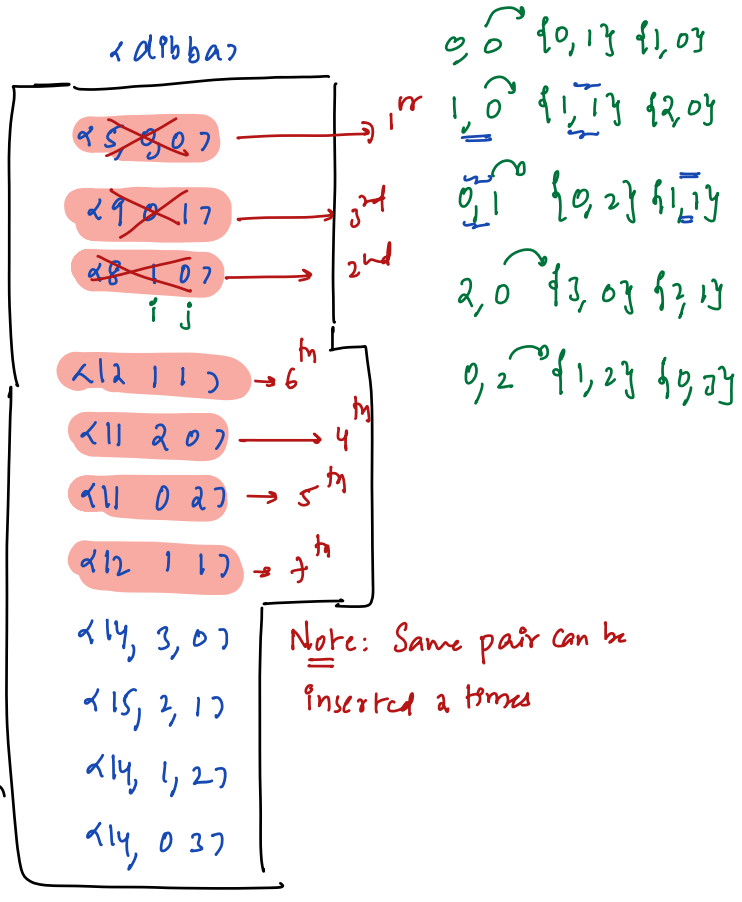
TC: $N*M \log(N*M)$
SC: $N*M$

2) Generate all pairs but insert in maxheap of size k

TC: $N*M \log k + N*M \log k$
(insert, delete)
SC: k

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

Pairs	val	i	j
1st	5	0	0
2nd	8	1	0
3rd	9	0	1
4th	11	2	0
5th	11	0	2
6th	12	1	1
7th	12	1	1



0, 0 {0, 1} {1, 0}

1, 0 {1, 1} {2, 0}

0, 1 {0, 2} {1, 1}

2, 0 {3, 0} {2, 1}

0, 2 {1, 2} {0, 3}

Note: Same pair can be inserted 2 times

Data Structure

1) getmin()
2) insert()
3) deletemin()

} Heap

4) Search if a pair is already present or not } HashSet<String>

i j

to check if pair is there or not → {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
    // Insert min pair:
    mh.insert( pair( a[0]+b[0], pair(0,0)));
    hs.insert (0+" "+0);
    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  possibilities -> {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: $k * \log k + k * \log k$ (insert + delete)

SC: $O(k + 2k)$ → min heap size, → hashset size