↳ Sudhesh

↳ DTU grad.

↳ 5* on codechef. (ICPC) → yellow

↳ Content creator. {CP, graph, Hashmap} → >1M views → CP

↳ PEPcoding

↳ 3 yrs+ teaches

↳ SDE & Instructor

↳ Advanced batch end.

Today's agenda

↳ N sorted array

↳ Given 2 sorted array find K smallest pair sum.

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

Ex: mat[4][6] (N * M)

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

N*M

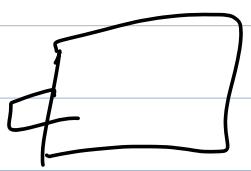
-6 0 1 2 3 4 6 7 7

//Idea1

↳ Insert all the elements in a dynamic array and sort.

T.C: O(N*M) + (N*M) log (N*M)

= O(N*M log (N*M))



(N*M)

N*M log (N*M)

↳ 2000+

1000 ↙ leetcode

↘ 1000 cc+ef

N Pointer approach

	0	1	2	3	4	5
0	~~2~~	7 p1	10	17	25	34
1	~~-6~~	~~0~~	~~1~~	8 p2	11	16
2	3 p3	4	6	14	21	26
3	7 p4	10	14	19	23	27

idea:

N Pointers, every iteration Point Pointer with min val and move that Pointer.

ans: -6 0 1 2

How to implement?

	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

min heap

~~(11,1,1)~~

~~(16,1,5)~~

(2,0,0) ~~(8,1,3)~~

~~(-6,1,0)~~ ~~(0,1,1)~~

(3,2,0) ~~(1,1,2)~~

(7,3,0) (7,0,1)

ex: (2,0,0)

-6 0 1 2

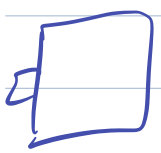
Pair {
int val;
int row;
int col;
}

& Comparator

// Pseudo Code

```
void merge(int mat[N][M]){

    minheap < Pair<int val, int row, int col>> mh;

    for(int i=0; i<N; i++){
        mh.insert(Pair(mat[i][0], i, 0));
    }

    while(mh.size()>0){

        Pair rem = mh.getmin();   // on the basis of val
        mh.deletemin();
        int v= rem.val;
        int r= rem.row;
        int c= rem.col;
        Print(v);
        if(c+1<m){
            mh.insert(Pair(mat[r][c+1], r, c+1));
        }
    }
}
```

log(size of PQ)



T.C: O(N*M logN)

S.C: O(N)

dry run

m=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

return

mh

{2, 0, 0}

~~{-6, 1, 0}~~ ~~{0, 1, 1}~~

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

{7, 3, 0}

```
void merge(int mat[n][m]){
    minheap < Pair<int val, int row, int col>> mh;
    for(int i=0; i<n; i++){
        mh.insert(Pair(mat[i][0], i, 0));
    }
    while(mh.size()>0){
        Pair rem = mh.getmin(); // on the basis of val
        mh.deletemin();
        int v = rem.val;
        int r = rem.row;
        int c = rem.col;
        Print(v);
        if(c+1 < m){
            mh.insert(Pair(mat[r][c+1], r, c+1));
        }
    }
}
```

rem: {0, 1, 1}

v: 0

r: 1

c: 1

-6 0

Google interview (MAANG)

1 round

Telephonic round (Screening round)

1 (1 tech)

10:15 pm

4 onsite rounds

3 tech

1 HR (Googly)

4 DSA

1 round

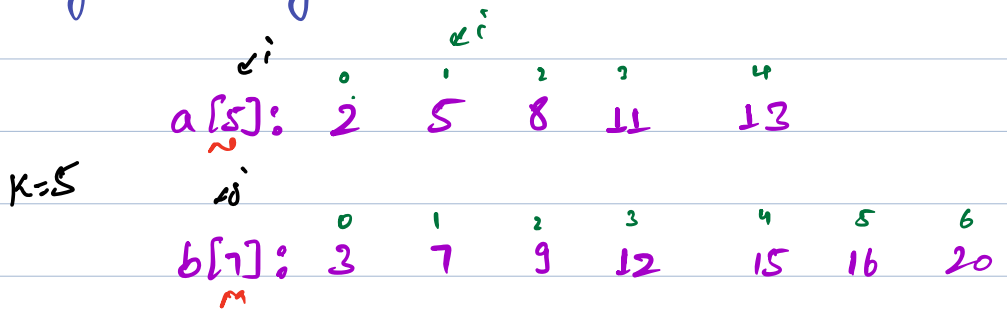
Heap follow up

3 DSA + 1 SD

1 HM

Q) K Smallest Pair Sums:

↳ Given 2 sorted arrays, Print K Smallest Pair Sums, a single Pair only 1 time.



Pairs	:	Sum	i	j
1	:	5	0	0
2	:	8	1	0
3	:	9	0	1
4	:	11	0	2
5	:	11	2	0

Idea

↳ Create all the Pairs and Store in a list, Sort that and Print K Smallest Pairs.

5 9 11 14 17 18 22

⇓

Sort ( )

T.C : O(N*m) + O(N*m log (N*m)) + O(K)

a[5]: 2 5 8 11 13

K=5

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
4	11	0	2

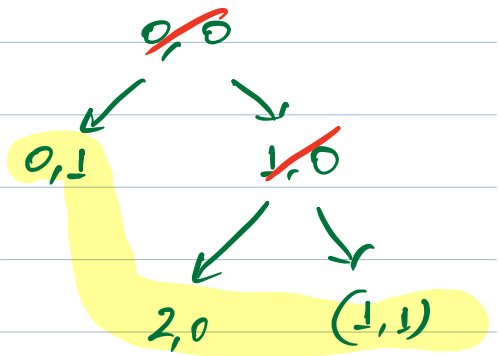
1,2 0,3

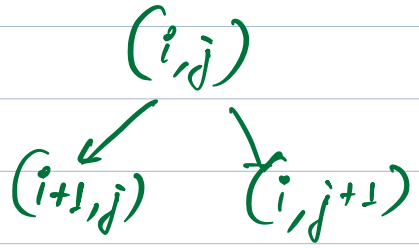
(5, 0, 0) (11, 0, 2)

(9, 0, 1) (14, 1, 2)

(8, 1, 0) (14, 0, 3)

(11, 2, 0) (12, 1, 1)





↳ To take care of duplicacy

↳ hashset

<String>

"i @ j"

// Pseudo code

```
void KPairs (int a[N], int b[M], int K){

    minheap < Pair(val, i, j) > mh;

    HashSet <String> hs;

    mh.insert (Pair(a[0]+b[0], 0, 0));
    hs.insert (0 + " " + 0);

    for(int i=0; i< K ; i++){
        Pair rem = mh.getmin();
        mh.deletemin();
        //int val= rem.val;
        //i = rem.i      //j = rem.j
        Print (val);

        if (i+1<N && hs.search((i+1) + " " + (j)) == false){
            mh.insert(a[i+1]+b[j], i+1, j);
            hs.insert( (i+1) + " " + (j) );
        }

        if (j+1<M && hs.search( (i) + " " + (j+1)) == false){
            mh.insert(a[i]+b[j+1], i, j+1);
```

hs.insert((i)+" "+(j+1));

}

T.C: O(K * log(K))

}

}

mh.insert(Pair(a[0]+b[0], 0,0));
hs.insert(0+" "+0);

for(int i=0; i<K; i++){
  Pair rem = mh.getmin();
  mh.deletemin();
  //int val = rem.val;
  //i = rem.i //j = rem.j
  Print(val);

  if(i+1<N && hs.search((i+1)+" "+(j))==false){
    mh.insert(a[i+1]+b[j], i+1, j);
    hs.insert((i+1)+" "+(j));
  }

  if(j+1<M && hs.search((i)+" "+(j+1))==false){
    mh.insert(a[i]+b[j+1], i, j+1);
    hs.insert((i)+" "+(j+1));
  }

}

rem: (5, 0, 0)

5 8

Dry run

K=3

a[5]: 0: 2, 1: 5, 2: 8, 3: 11, 4: 13

b[7]: 0: 3, 1: 7, 2: 9, 3: 12, 4: 15, 5: 16, 6: 20

mh: ~~(5,0,0)~~, ~~(8,1,0)~~, (9,0,1)

hs: "0 0" | "1 0" | "0 1"

↳ 1 add PQ

(1 rem → 2 add) → K times

-1 +2 ⇒ K add^n

↳ Pseudo code

Doubt session

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

```
PriorityQueue<Pair> mh = new PriorityQueue<>(){
    Comparator (Pair a, Pair b){
        return a.val - b.val;
    }
}
```

```
Pair n1 = new Pair (a[0]+b[0], 0, 0);
mh.insert(n1);
```

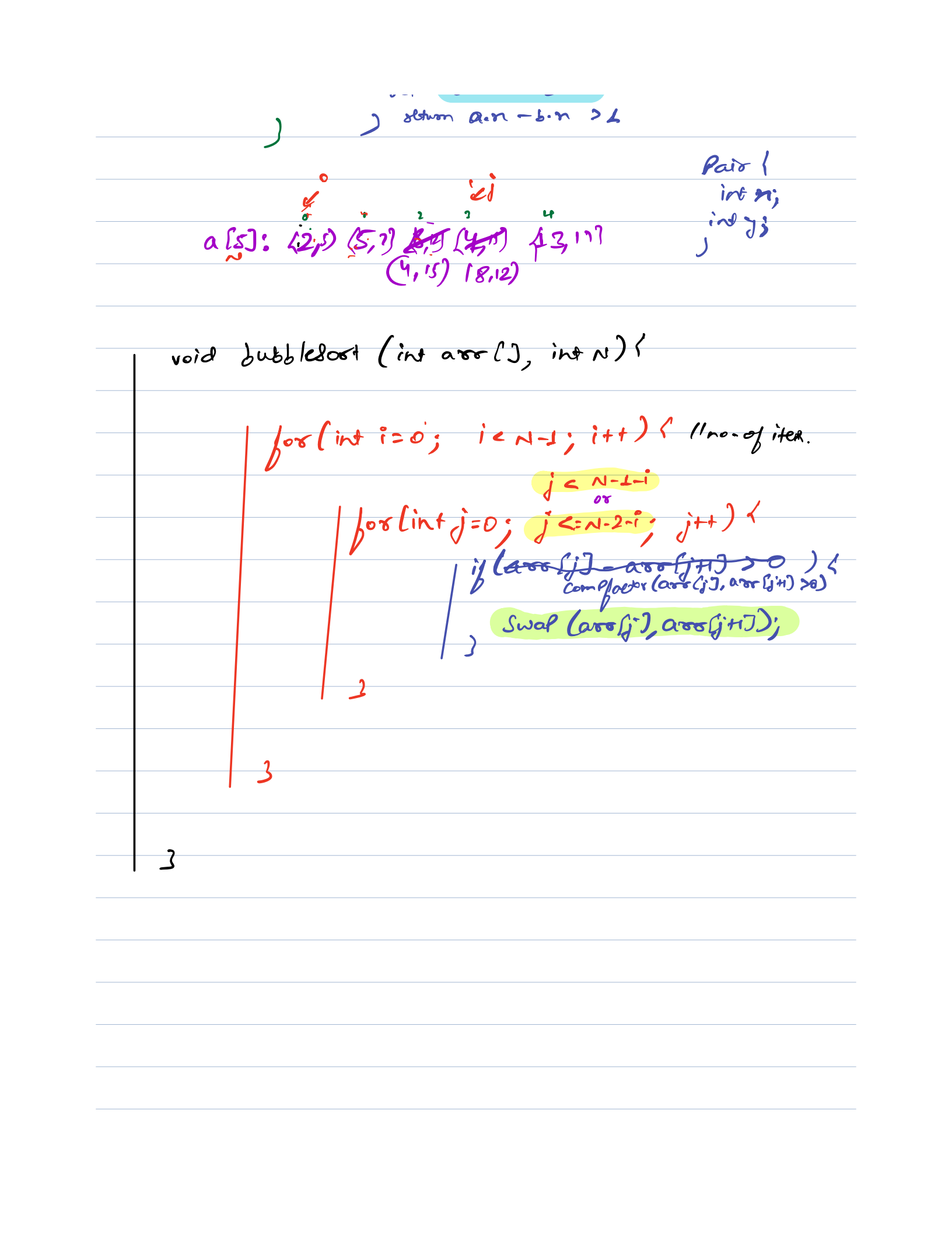
bubblesort

{-1, 0, 1}

Arrays.sort (a) {

Comparator (Pair a, Pair b) { → 1, 0, -1

return a.or - b.or


return a.n - b.n > 1
Pair {
int x;
int y;
}
a[5]: (2,5) (5,7) (8,15) (4,15) (3,17)
(4,15) (8,12)
void bubbleSort (int arr[], int N){
for(int i=0; i<N-1; i++){ //no. of iter.
j < N-1-i
or
for(int j=0; j<=N-2-i; j++){
if(arr[j] - arr[j+1] > 0){
compFactor(arr[j], arr[j+1]) > 0)
swap(arr[j], arr[j+1]);
}
}
}