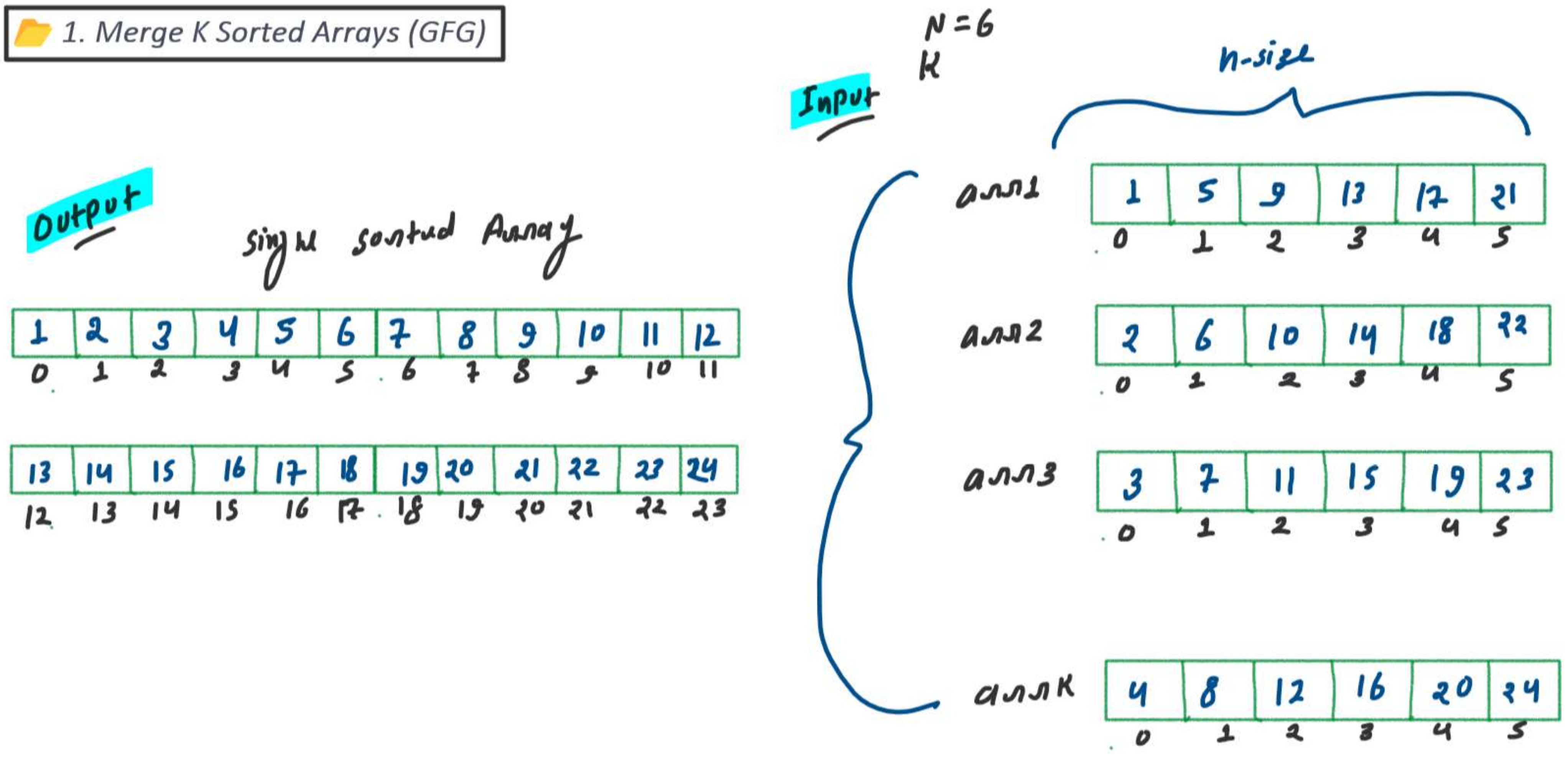
1. Merge K Sorted Arrays (GFG)



Output: Single sorted Array

index	0	1	2	3	4	5	6	7	8	9	10	11
value	1	2	3	4	5	6	7	8	9	10	11	12
index	12	13	14	15	16	17	18	19	20	21	22	23
value	13	14	15	16	17	18	19	20	21	22	23	24

Input: N = 6, K

	0	1	2	3	4	5
arr1	1	5	9	13	17	21
arr2	2	6	10	14	18	22
arr3	3	7	11	15	19	23
arrK	4	8	12	16	20	24

ALGORITHMS

STEP1 Find First min Element of K-Arrays

Note when we merge two sorted Array
↳ To HUM phle First min element find karte hai OR ussi element ko Ans Array me pahle push karte hai
Right - YES

↳ we find min-element using min-heap with Time complexity O(1)

EX1 **INPUT** N=4
K=3

arr1

1	4	8	10
0	1	2	3

arr2

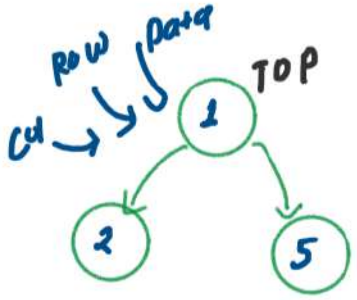
2	3	6	9
0	1	2	3

arr3

5	7	11	12
0	1	2	3

(I) CREATE MIN-HEAP USING First Elements of K-Arrays

Iteration 1



(II) FIND MIN ELEMENT

MIN = 1 RIndex = 0 CIndex = 0

(IV) Update TOP and Heapify it

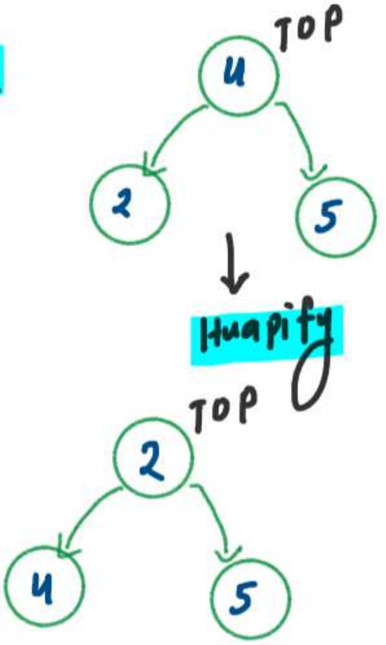
TOP = 4

		0	1	2	3
arr1	0	1	4	8	10
arr2	1	2	3	6	9
arr3	2	5	7	11	12

(III) Push MIN into ANS ARRAY

	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1											

Iteration2



II FIND MIN ELEMENT

MIN = 2   RIndex=1   CIndex=0

IV Update TOP and Heapify it

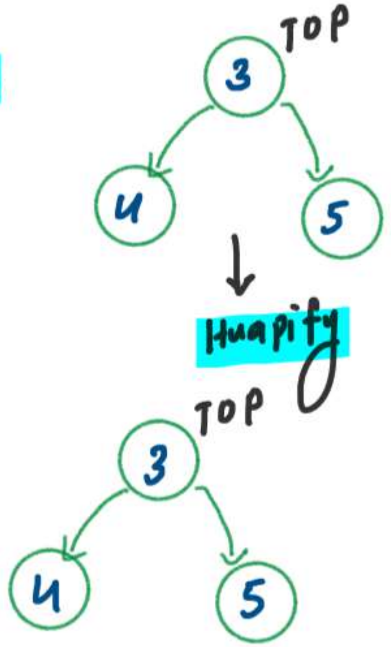
TOP= 3

		0	1	2	3
arr1	0	1	4	8	10
arr2	1	2	3	6	9
arr3	2	5	7	11	12

III Push MIN into ANS ARRAY

	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1	2										

Iteration3

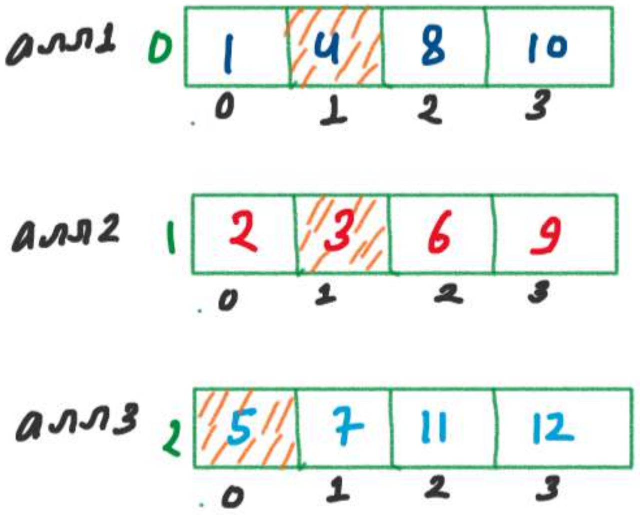


Ⅱ FIND MIN ELEMENT

MIN = 3   RIndex = 1   CIndex = 1

Ⅳ Update TOP and Heapify it

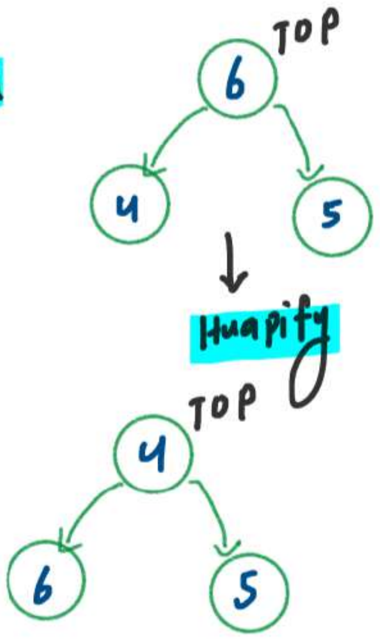
TOP = 6



Ⅲ Push MIN into ANS ARRAY

ANS	1	2	3									
	0	1	2	3	4	5	6	7	8	9	10	11

Iteration 4



Ⅱ FIND MIN ELEMENT

MIN = 4 RIndex = 0 CIndex = 1

Ⅳ Update TOP and Heapify it

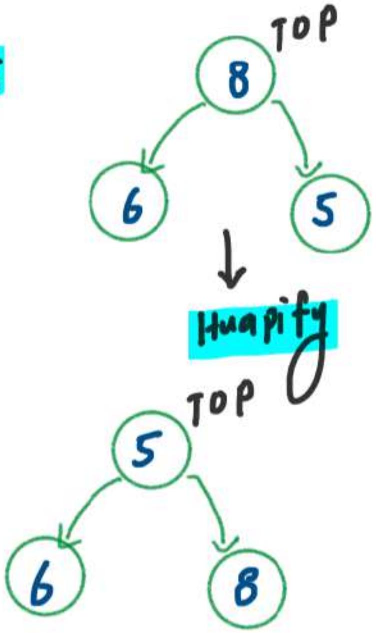
TOP = 8

		0	1	2	3
arr1	0	1	4	8	10
arr2	1	2	3	6	9
arr3	2	5	7	11	12

Ⅲ Push MIN into ANS ARRAY

	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1	2	3	4								

Iterations



II Find MIN ELEMENT

MIN = 5 RIndex = 2 CIndex = 0

IV Update TOP and Heapify it

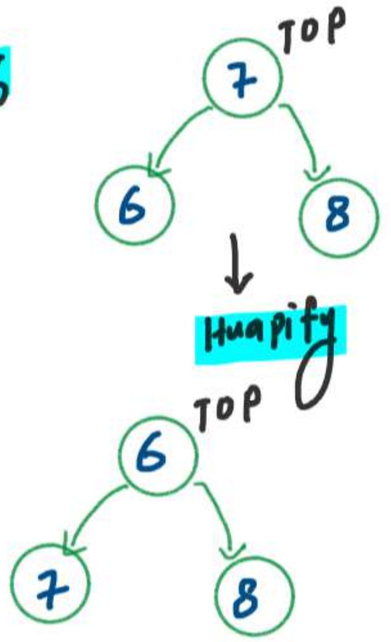
TOP = 7

		0	1	2	3
arr1	0	1	4	8	10
arr2	1	2	3	6	9
arr3	2	5	7	11	12

III Push MIN into ANS ARRAY

	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1	2	3	4	5							

Iteration 6



(II) FIND MIN ELEMENT

MIN = 6 RIndex = 1 CIndex = 2

(IV) Update TOP and Heapify it

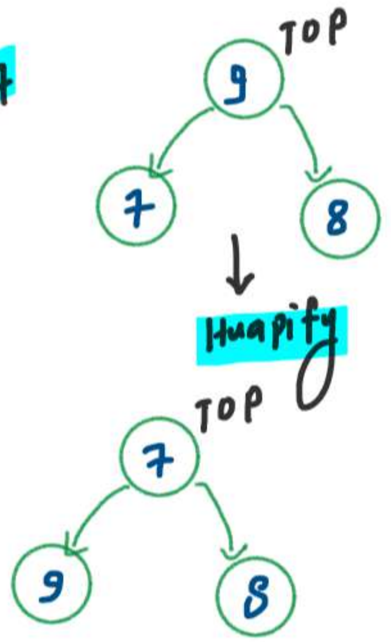
TOP = 9

		0	1	2	3
arr1	0	1	4	8	10
arr2	1	2	3	6	9
arr3	2	5	7	11	12

(III) Push MIN into ANS ARRAY

	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1	2	3	4	5	6						

Iteration 7



II FIND MIN ELEMENT

MIN = 7 RIndex = 2 CIndex = 1

IV Update TOP and Heapify it

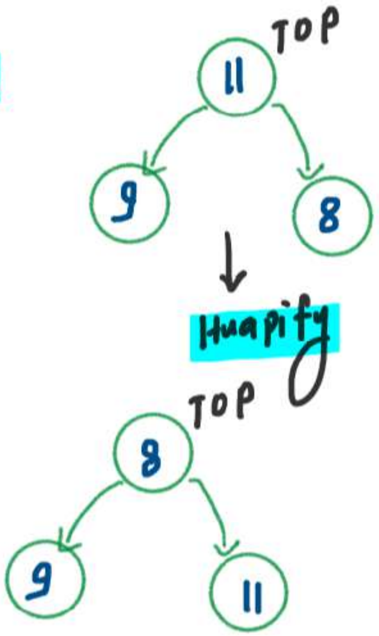
TOP = 11

		0	1	2	3
arr1	0	1	4	8	10
arr2	1	2	3	6	9
arr3	2	5	7	11	12

III Push MIN into ANS ARRAY

	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1	2	3	4	5	6	7					

Iteration8



Ⅱ FIND MIN ELEMENT

MIN = 8 RIndex= 0 CIndex = 2

Ⅳ update TOP and Heapify it

TOP= 10

arr1 0	1	4	8	10
	0	1	2	3

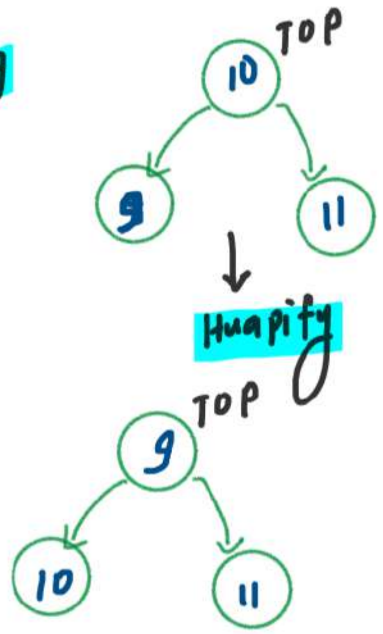
arr2 1	2	3	6	9
	0	1	2	3

arr3 2	5	7	11	12
	0	1	2	3

Ⅲ Push MIN into ANS ARRAY

ANS	1	2	3	4	5	6	7	8				
	0	1	2	3	4	5	6	7	8	9	10	11

Iteration 9



(II) FIND MIN ELEMENT

MIN = 9 RIndex = 1 CIndex = 3

(IV) Update TOP and Heapify it

TOP = 10

CIndex < N
3 < 3 ✗

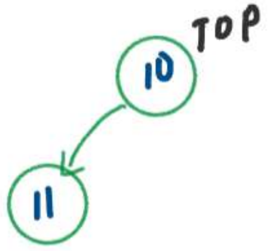
(GALTI YAHAN PAR HOTI HAI)

		0	1	2	3
arr1	0	1	4	8	10 ▨
arr2	1	2	3	6	9 ▨
arr3	2	5	7	11 ▨	12

(III) Push MIN into ANS ARRAY

	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1	2	3	4	5	6	7	8	9			

Iteration 10



Ⅱ FIND MIN ELEMENT

MIN = 10 RIndex = 0 CIndex = 3

Ⅳ Update TOP and Heapify it

TOP = 11

CIndex < N
3 < 3 ✗

(GALTI YANHA PAR HOTI HAI)

arr1	0	1	4	8	▨ 10
		0	1	2	3
arr2	1	2	3	6	9
		0	1	2	3
arr3	2	5	7	▨ 11	12
		0	1	2	3

Ⅲ Push MIN into ANS ARRAY

ANS	1	2	3	4	5	6	7	8	9	10		
	0	1	2	3	4	5	6	7	8	9	10	11

Iteration 11

$\boxed{11}$ TOP

(II) FIND MIN ELEMENT

MIN = 11 RIndex = 2 CIndex = 2

(IV) update TOP and Heapify it

TOP = 12

		0	1	2	3
arr1	0	1	4	8	10
arr2	1	2	3	6	9
arr3	2	5	7	~~11~~	12

(III) Push MIN into ANS ARRAY

	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1	2	3	4	5	6	7	8	9	10	11	

Iteration 12

12 TOP ✗

(II) FIND MIN ELEMENT

MIN = 12   RIndex = 2   CIndex = 3

(IV) Update TOP and Heapify it

TOP = TOP UPDATE NAHI KAR SAKTE HAI

Heap is Empty Now   AND   colIndex 3<3 ✗

FINAL OUTPUT

		0	1	2	3
arr1	0	1	4	8	10
arr2	1	2	3	6	9
arr3	2	5	7	11	12

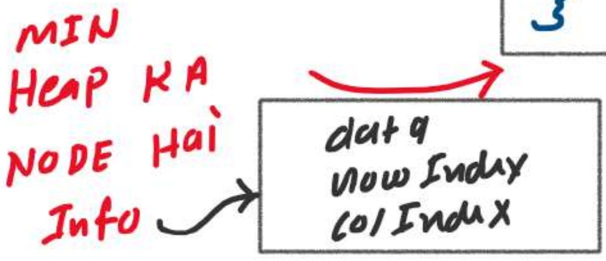
(III) Push MIN into ANS ARRAY

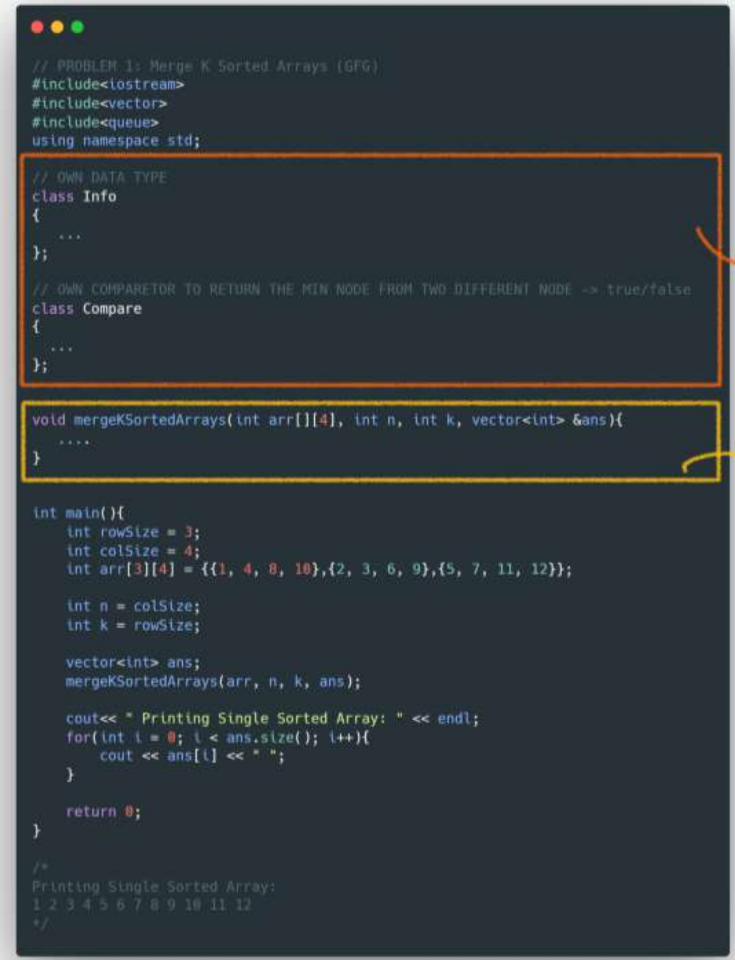
	0	1	2	3	4	5	6	7	8	9	10	11
ANS	1	2	3	4	5	6	7	8	9	10	11	12

(IV) Array main jis min value ko push kar raha hai uski next element k dubara Heap ke TOP Element ko update karne ke liye home RowIndex, colIndex and data ki need hogi Right → YES

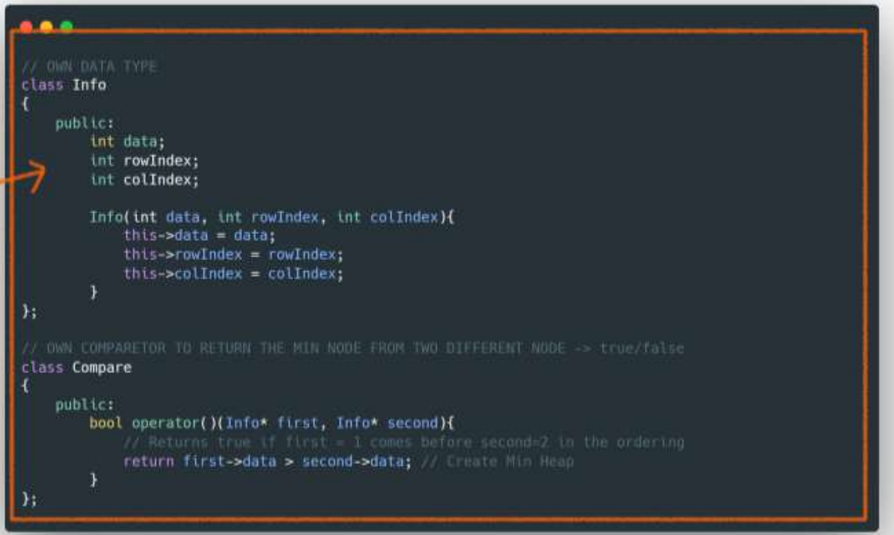
↳ To hum apna khud ka ek new data type create kar lenge jisme yeh three properties milenge.

```
class Info {
    public:
        int data;
        int rowIndex;
        int colIndex;

        Info(int data,
             int rowIndex,
             int colIndex){
        this->data = data;
        this->rowIndex = rowIndex;
        this->colIndex = colIndex;
    }
}
```

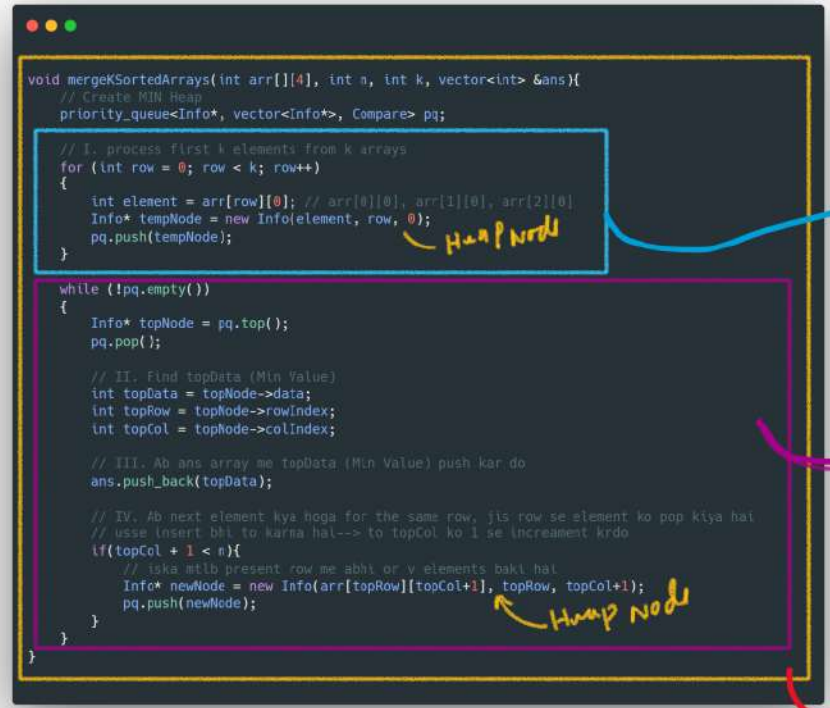




```cpp
// PROBLEM 1: Merge K Sorted Arrays (GFG)
#include<iostream>
#include<vector>
#include<queue>
using namespace std;

// OWN DATA TYPE
class Info
{
   ...
};

// OWN COMPARETOR TO RETURN THE MIN NODE FROM TWO DIFFERENT NODE -> true/false
class Compare
{
  ...
};

void mergeKSortedArrays(int arr[][4], int n, int k, vector<int> &ans){
   ....
}

int main(){
    int rowSize = 3;
    int colSize = 4;
    int arr[3][4] = {{1, 4, 8, 10},{2, 3, 6, 9},{5, 7, 11, 12}};

    int n = colSize;
    int k = rowSize;

    vector<int> ans;
    mergeKSortedArrays(arr, n, k, ans);

    cout<< " Printing Single Sorted Array: " << endl;
    for(int i = 0; i < ans.size(); i++){
        cout << ans[i] << " ";
    }

    return 0;
}

/*
Printing Single Sorted Array:
1 2 3 4 5 6 7 8 9 10 11 12
*/
```



```cpp
// OWN DATA TYPE
class Info
{
    public:
        int data;
        int rowIndex;
        int colIndex;

        Info(int data, int rowIndex, int colIndex){
            this->data = data;
            this->rowIndex = rowIndex;
            this->colIndex = colIndex;
        }
};

// OWN COMPARETOR TO RETURN THE MIN NODE FROM TWO DIFFERENT NODE -> true/false
class Compare
{
    public:
        bool operator()(Info* first, Info* second){
            // Returns true if first = 1 comes before second=2 in the ordering
            return first->data > second->data; // Create Min Heap
        }
};
```



```cpp
void mergeKSortedArrays(int arr[][4], int n, int k, vector<int> &ans){
    // Create MIN Heap
    priority_queue<Info*, vector<Info*>, Compare> pq;

    // I. process first k elements from k arrays
    for (int row = 0; row < k; row++)
    {
        int element = arr[row][0]; // arr[0][0], arr[1][0], arr[2][0]
        Info* tempNode = new Info(element, row, 0);
        pq.push(tempNode);
    }

    while (!pq.empty())
    {
        Info* topNode = pq.top();
        pq.pop();

        // II. Find topData (Min Value)
        int topData = topNode->data;
        int topRow = topNode->rowIndex;
        int topCol = topNode->colIndex;

        // III. Ab ans array me topData (Min Value) push kar do
        ans.push_back(topData);

        // IV. Ab next element kya hoga for the same row, jis row se element ko pop kiya hai
        // usse insert bhi to karna hai--> to topCol ko 1 se increament krdo
        if(topCol + 1 < n){
            // iska mtlb present row me abhi or v elements baki hai
            Info* newNode = new Info(arr[topRow][topCol+1], topRow, topCol+1);
            pq.push(newNode);
        }
    }
}
```

Time complexity

Heap ki T.C. = O(log(K))
For loop ki T.C. = O(K)
} O(K * log(K))

K = No. of Arrays

+

While loop ki T.C. = O(N)
Heap ki T.C. = O(log(K))
} O(N * log(K))

N = Total Elements of All arrays

overall T.C.

$O(K * \log(K)) + O(N * \log(K))$

SPACE complexity

(MIN Heap) Priority queue ki S.C. = $O(K)$

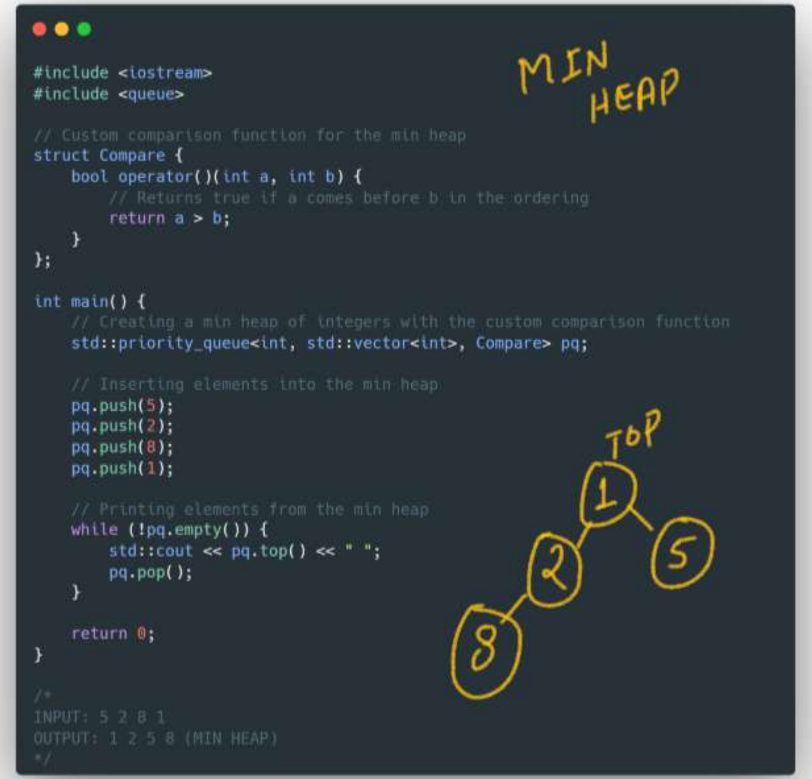
K = No. of arrays

(Ans) Vector Array ki S.C. = $O(N)$

N = Total Elements of All arrays

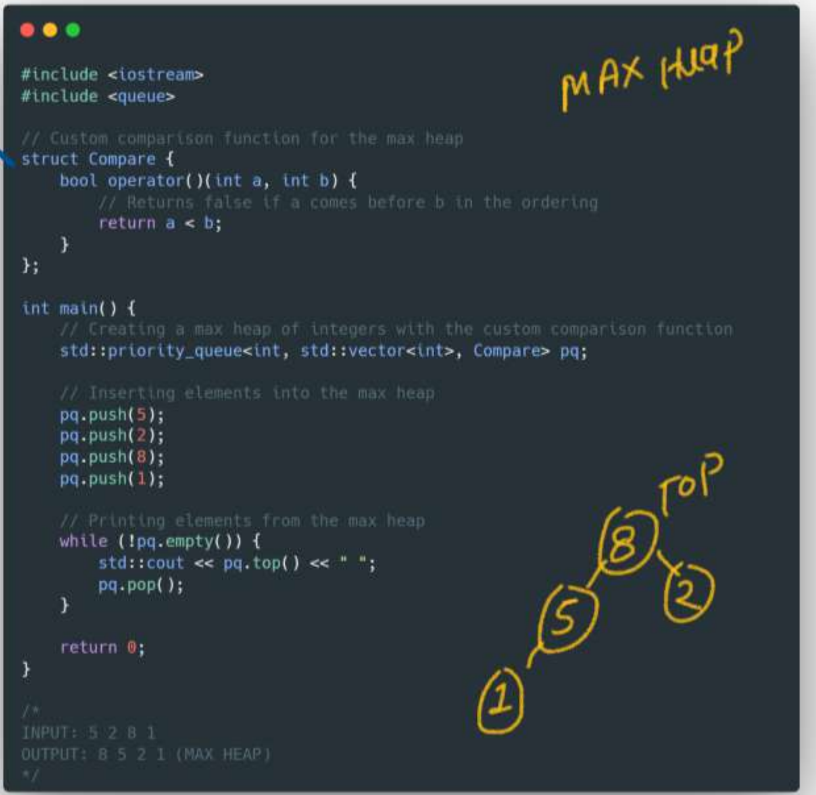
Overall S.C.

$O(K) + O(N)$



```cpp
#include <iostream>
#include <queue>

// Custom comparison function for the min heap
struct Compare {
    bool operator()(int a, int b) {
        // Returns true if a comes before b in the ordering
        return a > b;
    }
};

int main() {
    // Creating a min heap of integers with the custom comparison function
    std::priority_queue<int, std::vector<int>, Compare> pq;

    // Inserting elements into the min heap
    pq.push(5);
    pq.push(2);
    pq.push(8);
    pq.push(1);

    // Printing elements from the min heap
    while (!pq.empty()) {
        std::cout << pq.top() << " ";
        pq.pop();
    }

    return 0;
}

/*
INPUT: 5 2 8 1
OUTPUT: 1 2 5 8 (MIN HEAP)
*/
```

Struct ke place par Class v Use kar sakte ho



```cpp
#include <iostream>
#include <queue>

// Custom comparison function for the max heap
struct Compare {
    bool operator()(int a, int b) {
        // Returns false if a comes before b in the ordering
        return a < b;
    }
};

int main() {
    // Creating a max heap of integers with the custom comparison function
    std::priority_queue<int, std::vector<int>, Compare> pq;

    // Inserting elements into the max heap
    pq.push(5);
    pq.push(2);
    pq.push(8);
    pq.push(1);

    // Printing elements from the max heap
    while (!pq.empty()) {
        std::cout << pq.top() << " ";
        pq.pop();
    }

    return 0;
}

/*
INPUT: 5 2 8 1
OUTPUT: 8 5 2 1 (MAX HEAP)
*/
```