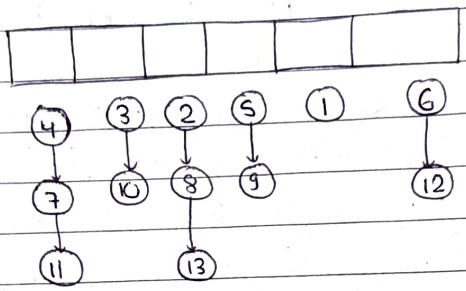
Date _____ Page _____

Day - 191

Heap-6

* Merge k Sorted LL:



⇒ We have given some LL & we have to merge all the LL.

⇒ We also solved this question previously ~~that~~ where we solved this question using approaches.

⇒ First, we will merge first two LL then next and so on.

⇒ Second, we will use merge sort here → O(nk log k)

⇒ Now, in third approach, we will use heap.

⇒ So, we can do that insert all the data into the min heap then pop all the elements & create a LL.

⇒ T.C. → nk log nk

S.C. → nk

Date _____ Page _____

⇒ This is not the most optimized approach.

⇒ Now, we can do that: we will store first elements of every LL then when any element popped, we will push their next element.

⇒ T.C. → $O(nk \log k)$
S.C. → $O(k)$

Code

⇒
```
class Node{
    public:
    int data;
    Node *next;
    Node(int x){
        data = x;
        next = NULL;
    }
};

priority_queue<Node*, vector<Node*>, compare> pq;

class compare{
    public:
    bool operator()(Node* a, Node* b){
        return a->data > b->data;
    }
};
```
↳ for min heap

Date _____ Page _____

* Merge K sorted Array

K = 4

3	4	5	11
1	6	7	14
4	5	8	13
6	9	10	12

⇒ We have to return a sorted array.

⇒ In the brute force approach, we can put all the elements inside a vector.

⇒ Then sort it —

→ if we use merge sort →

T.C. → $O(k^2 \log k)$

S.C. → $O(k^2)$

→ if we use heap sort →

T.C. → $O(k^2 \log k)$

S.C. → $O(1)$

⇒ In the next approach, we can use min heap & push all the elements —

⇒ Then pop elements & push into a array.

T.C. → $O(k^2 \log k)$

S.C. → $O(k^2)$

⇒ We can also use the previous day approach

⇒ Push first col then pop ~~min~~ top element from the min heap.

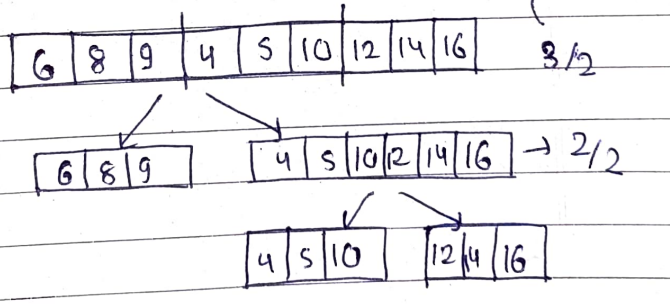
Date _____ Page _____

⇒ And push the adjacent element into the min heap.

T.C. → $k^2 \log k$
S.C. → k

⇒ We can also solve this question by using merge sort.
⇒ We know that every row is sorted.
⇒ So, we can convert the matrix into into single array then every k size window is sorted.
⇒ So we can apply merge function of merge sort on these all 'k' windows.

⇒ Now, when the value of k is odd then we will divide the portions in this way → k/2. manner



⇒ int mid = start + ((portion)/2) * k) - 1