17/03/2024

Page No. | Date

Day 20 of DSA.

Task.

CheckBox | Heap.

- ✓ Kth Largest Sum Subarray.
  - ✓ – Approach #1
  - ✓ – Approach #2
- ✓ Merge K Sorted Arrays
  - ✓ – Approach #1
  - ✓ – Approach #2
- ✓ Merge K linked list.
  - ✓ – Approach #1
  - ✓ – Approach #2

Page No.
Date

(1) kth Largest Sum Sub Array !-

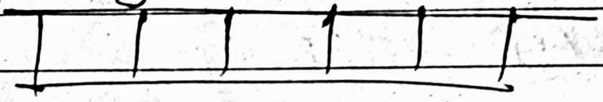
(1) Approach (1)-

* 1st Find out all subarray sum
put it in vector

* Sort that vector

* & Finally return (n-k).

sorted Array -



n-3 n-2 n-1th largest
n-k th largest

(2) Approach (2).

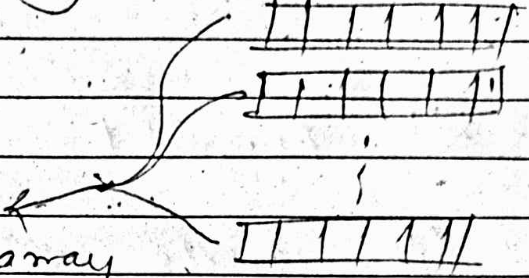
* declare min heap using priority queue

* calculate sum of subarray
2 condition after calculating sum

(1) min-heap size < k
then push it in min heap (sum)

(2) else {
```
    if (sum > mini.top()) {
        mini.pop();
        mini.push(sum);
    }
}
```

② Merge k sorted Arrays.



Approach 1.

① create an array

② Insert all elements into an array.

③ Sort an Array.

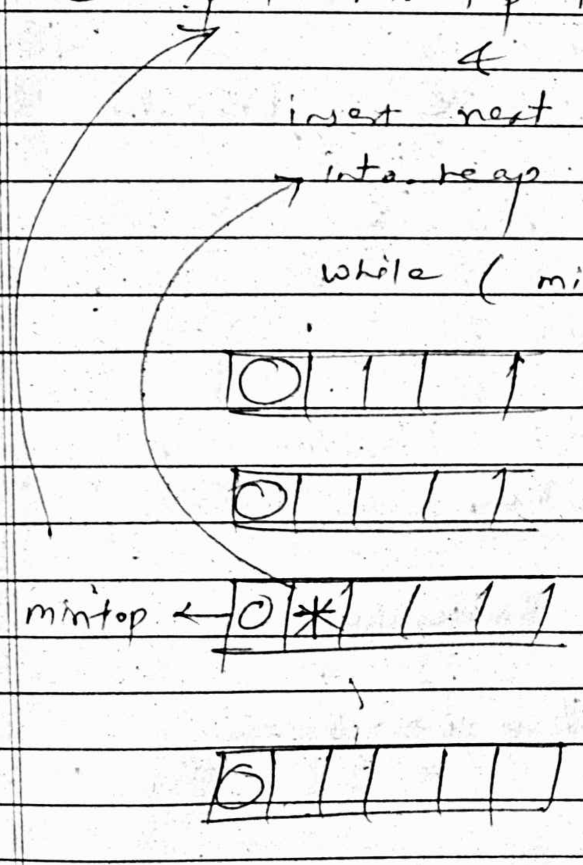
✓

Approach 2.

① Create min heap &
store 1st element of all arrays.

② put min top in an array
&
insert next element of same array
into heap

while (minHeapsize > 0).



Page No.
Date

(3) Merge k sorted linked list.

Approach 1.

(1) Create vector ans.
push all linked node data
into ans.

(2) Sort that ans vector

(3) join all liked lists.
& replace linked list node data
with vector data

High Time Complexity

Approach 2.

(1) Create minheap.
& push First element of linked list
in it.

```
(2) while (!minHeap.empty())
    {
        if( head == NULL)
        {
            head = tail = minHeap.top();
            minHeap.pop();
        }
        if( heap -> next != NULL)
        {
            min Heap insert 4
        }
        else{
            tail -> next = minHeap.top();
              minHeap.pop()
              tail = tail -> next;

            if (tail -> next != NULL)
            {
                 minHeap insert)
            }
        }
          return head;
    }
```