Day 20

Thursday, 27 April 2023 2:28 PM

P1 : Merge K-Sorted Lists

⇒ Use merge fⁿ from merge sort two lists repeatedly and store result (two merged list in head node.

① Brute Force :

- Traverse all linked list → store nodes value in array
- Sort and traverse over array to get value of nodes
- Create new LL and extend it with new values.

Time : $O(N \log N)$; Space : $O(N)$

② Compare 1 by 1 :

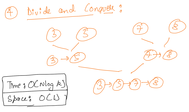
- Compare all K heads and get the node with smallest value.
- Extend LL with the selected nodes

Time : $O(KN)$ Space : $O(N)$

③ Merge List one by one :

Time : $O(nxk)$
Space : $O(1)$

④ Divide and Conquer :



Time : $O(N \log k)$
Space : $O(1)$

P2 : MAXIMAL RECTANGLE

O/P : 8

4 x 2 = 8

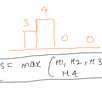
⇒ We will need to make histogram of given matrix :-

↳  → H①

↳  → H②

↳  → H③

↳  → H④



no element at ground lvl so we will ignore these two.

ans = max (H①, H②, H③, H④)

⇒ Apply MAH at every lvl of (Row of) Matrix. starting from Top.

↓

Maximum Area of Histogram

P3 : LONGEST INCREASING PATH IN MATRIX

Observation : We cannot reuse a variable

9 9 4
↑
6 6 8
↑
2 ← 1 1

↳ Let us say we found LIP from here is of length 4.

Hence will there be any case that longest increasing path from will be more than 3 ?

↳ (NO)

① Brute Force

↳ Try for every cell to find LIP and store it in matrix. Find max of that matrix ← That is our answer.

② Dynamic Programming

↳ Use that matrix caching to reduce no. of DFS calls.

[Time : $O(n \cdot m)$
Space : $O(n \cdot m)$]

TRIE

efficient for following operations :-

- ↳ Search
- ↳ Insert
- ↳ Delete
- ↳ Prefix Search
- ↳ lexographically ordering of words

→ Implementing dictionaries

→ Search : O (word_len)
Insert : O (word_len)
Delete : "
Prefix Search : O (prefix_len + output_len)
Lexographical Ordering : O (output_len)

→

```
struct TrieNode {
    TrieNode *child[26];
    bool isEnd;

    TrieNode() {
        isEnd = false;
        for (i = 0; i < 26; i++)
            child[i] = NULL;
    }
};
```