Q1) TikTok Reel Impact

In the dynamic landscape of TikTok, creators are in a constant race to boost their videos' engagement and reach by leveraging new features that enhance their content.

Each creator starts with a set of mvideos represented by initialReelimpacts, which Indicates the baseline popularity of each reel. For next n days, Tik Tok releases new trending features represented by newReelimpacts, with each feature offering an additional boost to the creator's existing reels.

On each of the next in days, the following takes place:

The creator appends the new features represented by new Reelimpacts[i] (where 0 <= i \le n) to their current reels

Review the updated lineup, select the most impactful reel based on its popularity, and add its impact value to their total impactscore

For example, if k = f\_{s} the creator selects the reel with the highest popularity; if k = 2 the creator selects the reel with the second highest popularity, and so on.

Given two integer arrays, initialReellmpacts and newReelimpacts of length m and n respectively and an integer k. The task is to calculate the creator's total impact score after incorporating all elements from newReelImpacts across n days.

Note: The initial impact score of the creator is considered to be the kth highest Impact value from the initial set of initialReelimpacts.

Example m = 2 initialReelImpacts = [2, 3]

n = 3 newReelimpacts =[4, 5, 1]

k = 2

Day 1: The creator appends a new reel with an impact of 4, resulting in [2, 3, 4], and selects the 2nd highest impact of 3, making impact score 2+3=5.

Day 2: The creator appends a new reel with an impact of 5, resulting in [2, 3, 4, 5], and selects the 2nd highest impact of 4, bringing total impact score to 5+4= 9.

Day 3: The creator appends a new reel with an impact of 1, resulting in [1, 2, 3, 4, 51, and selects the 2nd highest impact of 4, making total impact score to 9 + 4 = 13.

Therefore the creator's total impact score after incorporating all elements from newReelimpacts across given n days is 13.

Hence return 13 as the answer

Q2) Maximize Throughput

in ByteDance's vast network of data centers, millions of interconnected servers process content requests, handle user interactions, and deliver data to users globally. Each server is part of a dynamic task execution flow, represented by an array server Tasks, where each entry in the array indicates the next server in the chain that will handle a task.

Optimizing the data pipeline requires careful management of these task handoffs. Once a task is picked up by server & it triggers a dependency on server server Tasks[i]. transferring the load there. However, this data transfer disables both servers/and server Taskisfil from participating in any further task handoffs, as they are locked due to processing the current load. Therefore, selecting the right servers and managing the chain reactions of these task handoffs is crucial for maximizing throughput.

Each server at index/points to the next server server Tasks[i], where the task is transferred. Once this transfer occurs, both the sending server and the receiving server become unavailable for subsequent tasks. Your challenge is to select servers in such a way that maximizes the overall throughput score. The throughput score is determined by the sum of the indices of the servers where tasks are successfully handed off.

Your task is to analyze this network of server-to-server task handoffs, navigate the

dependencies, and determine the maximum possible throughput score that can be achieved by optimally choosing the task handoffs.

Given the array serverTasks, calculate the maximum throughput score achievable by performing these operations in the most efficient way.

Example

n-3

server Tasks = [0, 1, 2]

We first select the server at index 0. It points to itself (server 0), and its throughput is 0. So, the current total throughput is 0. Note that server O is now blocked.

Next, we select the server at index 1. It points to itself (server 1), and its throughput is 1. So, the current total throughput becomes 0+1=1. Note that both servers 0 and 1 are now blocked.

Then, we select the server at index 2. It points to itself (server 2), and its throughput is 2. So, the current total throughput becomes 0+1+2=3.

All servers are now blocked.

calculate Max Processing Throughput has the following parameter(s): serverTasks[n]: An array of integers where each element indicates the next server where the task is to be transferred.

Returns

long: The maximum throughput score achievable.

Constraints

1 <= n <= 2 \* 10 ^ 5

0<senverTasks \{i\} < n