Find Kth Largest Sum of Sub Array CodeStudio

Given an array of integers, find the Kth largest sum subarray For example, given the array [1, -2, 3, -4, 5] and K = 2, the 2nd largest sum subarray would be [3, -4, 5], which has a sum of 4.

Please note that a subarray is the sequence of consecutive elements of the array.

Example: [3,4,2,8,9,1], k = 4

Output: The 4 largest sum of subarray: 23 (4 + 2 + 8 + 9)

Approach 1: Function to find the kth largest sum of subarrays using a brute force approach

- **Function Purpose:** This approach is designed to find the Kth largest sum of subarrays within an array. It calculates all possible subarray sums and returns the Kth largest sum.
- **Explanation:** The brute force approach computes all subarray sums through two nested loops. It iterates through each possible subarray, calculates the sum, and stores it in a list. This list of sums is then sorted in ascending order. Finally, it returns the Kth largest sum from the sorted list.
- **Time Complexity: O(n^2)**
    - ***Explanation:*** **Calculating all possible subarray sums takes O(n^2) time because for each starting index, we compute the sum for each ending index.**
- **Space Complexity: O(n^2)**
    - ***Explanation:*** **All subarray sums are stored in a vector, resulting in a space complexity of O(n^2).**

Approach 2: Function to find the kth largest sum of subarrays using a max heap

- **Function Purpose:** This approach aims to find the Kth largest sum of subarrays efficiently by maintaining a max heap (priority queue) with the K largest sums.
- **Explanation:** In this approach, all possible subarray sums are calculated and inserted into a max heap. To ensure that the heap only contains the K largest sums, as the elements are inserted into the heap, we pop elements if the heap size exceeds K. After processing all subarrays, the Kth largest sum is retrieved from the max heap.
- **Time Complexity: O(n^2 * log n)**
    - ***Explanation:*** **The time complexity arises from inserting subarray sums into the max heap, which takes O(log n) time for each insertion. As there are O(n^2) subarray sums, the overall time complexity is O(n^2 * log n).**

- **Space Complexity: O(n^2)**
    - ***Explanation:*** **The max heap stores all possible subarray sums, leading to a space complexity of O(n^2).**

Approach 3: Function to find the kth largest sum of subarrays using a min heap

- **Function Purpose:** Similar to Approach 2, this approach efficiently finds the Kth largest sum of subarrays by maintaining a min heap (priority queue) with the K largest sums.
- **Explanation:** All subarray sums are computed and inserted into a min heap. As in Approach 2, the heap size is controlled, ensuring that it contains only the K largest sums. Finally, the Kth largest sum is retrieved from the min heap.
- **Time Complexity: O(n^2 * log n)**
    - ***Explanation:*** **The time complexity is similar to Approach 2, O(n^2 * log n), due to the insertion of subarray sums into the min heap.**
- **Space Complexity: O(n^2) (worst-case)**
    - ***Explanation:*** **The space complexity depends on K, but in the worst case, it's O(n^2) because the min heap stores all possible subarray sums.**

Approach 4: Function to find the kth largest sum of subarrays using an optimized approach

- **Function Purpose:** This approach efficiently identifies the Kth largest sum of subarrays by maintaining a max heap with the K largest sums while iterating through subarrays.
- **Explanation:** In this approach, a max heap is maintained to keep track of the K largest sums as subarrays are considered. During iteration, when a new subarray sum is computed, it is compared with the smallest sum in the max heap. If the new sum is larger, it replaces the smallest sum. After processing all subarrays, the Kth largest sum is present in the max heap.
- **Time Complexity: O(n^2 * log k)**
    - ***Explanation:*** **The time complexity is determined by the number of elements in the max heap, which is limited to K. Since there are O(n^2) subarray sums, the time complexity is O(n^2 * log k), where K <= n^2.**
- **Space Complexity: O(K)**
    - ***Explanation:*** **The space complexity depends on the value of K, representing the size of the max heap. In the worst case, it's O(n^2) when K = n^2.**

Conclusion

In conclusion, we have explored four approaches for finding the Kth largest sum of subarrays within an array. Each approach has its strengths and weaknesses, and the choice depends on the specific requirements and constraints of your problem. **Approach 4 is the most efficient when K is relatively small compared to the array size.**