Sum of Elements (medium)

We'll cover the following

Problem Statement

Given an array, find the sum of all numbers between the K1'th and K2'th smallest elements of that array.

Example 1:

```
Input: [1, 3, 12, 5, 15, 11], and K1=3, K2=6
Output: 23
Explanation: The 3rd smallest number is 5 and 6th smallest number 15. The sum of n
umbers coming
between 5 and 15 is 23 (11+12).
```

Example 2:

```
Input: [3, 5, 8, 7], and K1=1, K2=4
Output: 12
Explanation: The sum of the numbers between the 1st smallest number (3) and the 4t
h smallest
number (8) is 12 (5+7).
```

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
def find_sum_of_elements(nums, k1, k2):
  # TODO: Write your code here
  return -1


def main():

  print("Sum of all numbers between k1 and k2 smallest numbers: " +
        str(find_sum_of_elements([1, 3, 12, 5, 15, 11], 3, 6)))
  print("Sum of all numbers between k1 and k2 smallest numbers: " +
        str(find_sum_of_elements([3, 5, 8, 7], 1, 4)))


main()

```

Solution

This problem follows the Top ‘K’ Numbers (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5728885882748928/) pattern, and shares similarities with Kth Smallest Number (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5696381570252800/).

We can find the sum of all numbers coming between the K1’th and K2’th smallest numbers in the following steps:

1. First, insert all numbers in a min-heap.
2. Remove the first `K1` smallest numbers from the min-heap.
3. Now take the next `K2-K1-1` numbers out of the heap and add them. This sum will be our required output.

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```
from heapq import *


def find_sum_of_elements(nums, k1, k2):
  minHeap = []
  # insert all numbers to the min heap
  for num in nums:
    heappush(minHeap, num)

  # remove k1 small numbers from the min heap
  for _ in range(k1):
    heappop(minHeap)

  elementSum = 0
  # sum next k2-k1-1 numbers
  for _ in range(k2 - k1 - 1):
    elementSum += heappop(minHeap)

  return elementSum


def main():

  print("Sum of all numbers between k1 and k2 smallest numbers: " +
        str(find_sum_of_elements([1, 3, 12, 5, 15, 11], 3, 6)))
  print("Sum of all numbers between k1 and k2 smallest numbers: " +
        str(find_sum_of_elements([3, 5, 8, 7], 1, 4)))

```

Time complexity #

Since we need to put all the numbers in a min-heap, the time complexity of the above algorithm will be $O(N * logN)$ where 'N' is the total input numbers.

Space complexity #

The space complexity will be $O(N)$, as we need to store all the 'N' numbers in the heap.

Alternate Solution

We can iterate the array and use a max-heap to keep track of the top K2 numbers. We can, then, add the top `K2-K1-1` numbers in the max-heap to find the sum of all numbers coming between the K1'th and K2'th smallest numbers. Here is what the algorithm will look like:

Java | Python3 | C++ | JS

```
from heapq import *


def find_sum_of_elements(nums, k1, k2):
  maxHeap = []
  # keep smallest k2 numbers in the max heap
  for i in range(len(nums)):
    if i < k2 - 1:
      heappush(maxHeap, -nums[i])
    elif nums[i] < -maxHeap[0]:
      heappop(maxHeap) # as we are interested only in the smallest k2 numbers
      heappush(maxHeap, -nums[i])

  # get the sum of numbers between k1 and k2 indices
  # these numbers will be at the top of the max heap
  elementSum = 0
  for _ in range(k2 - k1 - 1):
    elementSum += -heappop(maxHeap)

  return elementSum


def main():

  print("Sum of all numbers between k1 and k2 smallest numbers: " +
        str(find_sum_of_elements([1, 3, 12, 5, 15, 11], 3, 6)))
  print("Sum of all numbers between k1 and k2 smallest numbers: " +
        str(find_sum_of_elements([3, 5, 8, 7], 1, 4)))
```

Time complexity

Since we need to put only the top K2 numbers in the max-heap at any time, the time complexity of the above algorithm will be $O(N * logK2)$.

Space complexity

The space complexity will be $O(K2)$, as we need to store the smallest 'K2' numbers in the heap.

← Back

Maximum Distinct Elements (medium)

Next →

Rearrange String (hard)

✓ Mark as Completed

Report an Issue

Ask a Question
(https://discuss.educative.io/tag/sum-of-elements-medium__pattern-top-k-elements__grokking-the-coding-interview-patterns-for-coding-questions)