Array Splitting

(https://codeforces.com/contest/1197/problem/C)

Nipun Goel(IMT2018052)   Vinayak Agarwal(IMT2018086)

Pseudo Code:

Input: n – size of the array
k – number of non-empty consecutive subarrays in which array is to be divided.
arr[] - elements of the array

Output: Minimum cost of division that can be obtained by dividing the array ‘arr’ into ‘k’ non-empty consecutive subarrays. **Cost of division is Summation(max(i)-min(i)) over all ‘k’ subarrays.**

Algorithm(Assuming 1-based indexing) :

```
//Compute the difference array as follows
1.  for i=1 to n-1
2.     diff[i] = arr[i] – arr[i+1]
3.  Sort the diff[] in increasing order
4.  ans = arr[n]-arr[1]
//Add the first (k-1) elements of the sorted diff[] to ans
5.  for i=1 to k-1
       ans = ans + diff[i]
6.  return ans
```

TIME COMPLEXITY:

O(n) for iteration and O(nlogn) for sorting the array.
Therefore, T(n) = theta(nlogn)

Proof of correctness :

Array ‘a’ : a1 - - - - an is in sorted order.

1. So, for any subarray a_i, a_{i+1},....,a_j where $1 \leq i \leq j \leq n$. a_i is the minimum element of the subarray and a_j is the maximum element of the subarray.
   Therefore, contribution of this subarray to cost of division is (a_j-a_i).

2. Subarrays are ordered such that i^{th} subarray occurs before $(i+1)^{th}$ subarray.
   Now we are supposed to divide the array ‘a’ into ‘k’ consecutive subarrays.
   Suppose the leftmost element of subarray ‘i’ is denoted by $b_{l,i}$ and the rightmost element is denoted by $b_{r,i}$.
   Note that: ***$b_{l,1}$=a_1 (leftmost element of 1^{st} subarray) $b_{r,k}$=a_n (rightmost element of k^{th} subarray)***

3. Now contribution to the cost of division by i^{th} subarray is $b_{r,i} - b_{l,i}$ (max i – min i).

4. Also note if $b_{r,i} = a_j$ (for some $1 \leq i < k$ and $1 \leq j < n$) then $b_{l,i+1} = a_{j+1}$ (since subarrays are consecutive) and so this uses the fact that the next subarray starts from the very next element where the previous subarray ends.

Contribution from -

1st subarray:	$b_{r,1} - b_{l,1}$
2nd subarray:	$b_{r,2} - b_{l,2}$
|	|
(k-1)th subarray:	$b_{r,k-1} - b_{l,k-1}$
kth subarray:	$b_{r,k} - b_{l,k}$

Total cost of division :-

$$= - b_{l,1} + (b_{r,1} - b_{l,2}) + (b_{r,2} - b_{l,3}) + \text{---} + (b_{r,k-1} - b_{l,k}) + b_{r,k}$$

$$= b_{r,k} - b_{l,1} + {}^{k-1}\Sigma_{i=1}(b_{r,i} - b_{l,i+1})$$

$$= (a_n - a_1) + {}^{k-1}\Sigma_{i=1}(b_{r,i} - b_{l,i+1}) \quad \text{(from (2) } b_{r,k} = a_n \text{ and } b_{l,1} = a_n \text{)}$$

Now, in order to minimize the total cost of division we need to minimize the quantity under summation.
$(b_{r,i} - b_{l,i+1})$ is nothing but $(a_j - a_{j+1})$ for some j. (from (4)).

So basically, we need to find the minimum (k-1) values of (a_j-a_{j+1}) in order to minimize summation.

And that is what the algorithm does.
Stores a_j-a_{j+1} in diff[j] for all j such that $1 \le j < n$
and then the output becomes sum of minimum (k-1) values of this diff[] array plus $(a_n - a_1)$.

Data Structures used:

Integer array is used to store numbers.

Inbuilt functions used:

Inbulit “sort” function of C++ is used to sort the array in ascending order.

Syntax: sort(startaddress,endaddress)

Instructions to run and compile the code:

1- Open a new terminal window.
2- Change the directory to the directory in which array_splitting.cpp is present.
3-After this enter the following command in the terminal to compile the source file assuming gcc compiler is installed. -
“g++ array_splitting.cpp”
4- Now run the program using the following command -
“./a.out”.
5- Enter the input in the format specified to get the required output.

Individual Contribution:

Pseudo code,implementation of Algorithm and expected input and output format was written by Nipun Goel(IMT2018052).

Sample test cases , Code Script for generating large test cases and time complexity analysis was done by Vinayak Agarwal(IMT2018086).

Proof of Correctness was discussed by both of us and was written by both of us together.
We both discussed the problem before starting and divided our parts after a thorough discussion of what all needs to be done and how it is supposed to be done including the algorithm, test cases, etc.