Dog Walking

locked

by *IEEEXtreme*

Statistics

Difficulty: Hard
Publish Date: Jul 14 2016

First consider the restriction that each group must be nonempty. If we have an optimal solution where a group is empty, then we can always move one of the dogs from the other groups to this group (since $K \le N$) without increasing the value of the arrangement. Hence we don't have to worry about this restriction.

Now take three dogs of size a, b and c, where $a \le b \le c$. Say that a and c are in the same group, but b is not. Notice that we can move b into the group containing a and c without increasing the value of the arrangement (again, we don't have to worry about c's group becoming empty). This suggests that in the optimal arrangement, each group forms a consecutive subsequence of the sorted list of dog sizes.

So now we've reduced the problem to partitioning the sorted list of dog sizes into K consecutive subsequences, so that the sum of max(S) - min(S) over each subsequence S is minimized. One standard way to solve this kind of problem is to use dynamic programming, but unfortunately N and K are too large for such an approach.

Consider the following test case:

```
8 4
7
5
1
30
24
25
26
31
```

After sorting the dog sizes, we want to partition the following list into 4 consecutive subsequences:

```
1 5 7 24 25 26 30 31
```

Here is the optimal solution, where a | marker is used to denote how the list is partitioned:

```
| 1 | 5 7 | 24 25 26 | 30 31 |
```

This gives us the total (1-1) + (7-5) + (26-24) + (31-30). But now think about this sum in terms of the markers. The first and last markers are special: they are always there in any arrangement. The first marker contributes -1 to the sum, and the last marker contributes 31 to the sum. The second marker contributes -5+1, the third marker contributes -24+7, and the fourth marker contributes -30+26. Summing up the contributions from the markers will give you the same total as summing up the ranges of the subsequences; after all this is just another way of summing up the same numbers.

This suggests yet another way of looking at the problem. Take the original sorted list, and between each pair x, y of adjacent numbers, write the number x - y in parenthesis.

```
1 (-4) 5 (-2) 7 (-17) 24 (-1) 25 (-1) 26 (-4) 30 (-1) 31
```

Putting a marker in between two of these numbers will contribute the corresponding value within parenthesis. We need to put down *K*-1 markers such that the sum is minimized, and so we will pick the *K*-1 minimum values within parenthesis. Adding the last number and subtracting the first one (corresponding to the mandatory first and last marker), we get the required minimum sum of group ranges.

In Python 2:

```
t = int(raw_input())
for tc in range(t):
    n,k = map(int,raw_input().split())
    arr = sorted([ int(raw_input()) for _ in range(n) ])
    res = arr[-1] - arr[0]
    arr = sorted([ x-y for (x,y) in zip(arr,arr[1:]) ])
    print res + sum(arr[:k-1])
```

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Scoring | Environment | FAQ | Terms Of Service | Privacy Policy