Venkatesha H.D

graphikaa
Date
Page

We are given:

* An Array arr[] of size N, where each element represents the height of a tower

* A positive integer K, which can be added or subtracted from each tower's height exactly once.

Goal

We need to minimize the difference b/w the tallest and shortest towers after modification

① Sorting the Array: Sorting helps in efficiently picking minimum and maximum hights after modification

② Two choices for Each Tower:
* Increase height by K
* Decrease height by K

③ Handling Negtive Heights: we cannot have negtive heights so decreasing is only possible if the result remains non-negative

④ Finding Minimum and maximum Heights:
* The smallest tower should be increased to get closer to the others
* The largest tower should be decreased to get closer to the others.

⑤ Tracking the Minimum Difference: The goal is to minimize the gap b/w the tallest and shortest towers

Venkatesha H.D

Code in C

```
#include <stdio.h>
#include <stdlib.h>

int compare(const void *a, const void *b){
    return (*(int*)a - *(int*)b);
}

int getMinDifference(int arr[], int n, int k){
    if(n==1) return 0;

    qsort(arr, n, sizeof(int), compare);

    int initial_diff = arr[n-1] - arr[0];

    int min_height, max_height;
    int result = initial_diff;

    for(int i=1; i<n; i++){
        if(arr[i] - k < 0) continue;

        min_height = (arr[0]+k < arr[i]-k) ? arr[0]+k :
                     arr[i]-k;
        max_height = (arr[n-1]-k > arr[i-1]+k) ? arr[n-1]-
                     : arr[i-1]+k;

        result = (result < (max_height - min_height)) ?
                 result : (max_height - min_height);
    }

    return result;
}
```

```
int main(){
    int arr[] = {1,15,10};
    int n = sizeof(arr) / sizeof(arr[0]);
    int k = 6;

    printf("Minimum Difference : %d\n", getMinDiffern(arr, n, k));

    return 0;

}
```