Крави

locked

Problem | Submissions | Leaderboard | Discussions

Краварят Курт живеел на село и гледал крави. В ранчото имало N колиби и K крави. Всички колиби били построени на една линия и Курт знаел координатите на всяка една от колибите. Колибите не били големи и затова най-много една крава можела да се побере в една колиба. По неясни причини кравите започнали да се карат една с друга. В следствие на това млекодобивът намалял. За да се справи с възникналата ситуация, Курт решил, когато прибира кравите в колибите им, да ги подреди по такъв начин, че минималното разстояние между две крави да е максимално. С това си действие той се надявал кравите да се успокоят и да започнат отново да дават мляко. След няколко дена тежка мисловна дейност Курт разбрал че проблема с нареждането на кравите не бил по неговите сили. Вашата задача е да напишете програма, която ще помогне на Курт да нареди кравите.

Submissions: 120
Max Score: 100
Difficulty: Medium

Rate This Challenge:

More

Input Format

На първия ред са зададени две числа – N и K - броят на колибите и броят на кравите. На втория ред има N числа, описващи координатите на всяка една от колибите.

Constraints

$1 \le N \le 100000$

$2 \le K \le N$

Координатите на колибите са в интервала [1, 2000000000].

Output Format

Търсеното максимално разстояние между двете най-близки крави

Sample Input 0

```
5 2
5 8 12 32 1
```

Sample Output 0

```
31
```

Explanation 0

Можем да поставим двете крави на позиции 1 и 32. Така разстоянието между тях ще е 31 и ще е максималното възможно.

Sample Input 1

```
7 3
1 15 35 10 69 60 28
```

Sample Output 1

```
34
```

Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
int partition(int * array, int start, int end) {
    int pivot = array[end];
    int pivotIndex = start;

    for (int i = start; i <= end; i++) {
        if (array[i] < pivot) {
            swap(array[i], array[pivotIndex]);
            pivotIndex++;
        }
    }

    swap(array[end], array[pivotIndex]);
    return pivotIndex;
}
void quickSort(int * array, int start, int end) {
    if (start < end) {
        int pivot = partition(array, start, end);
        quickSort(array, start, pivot - 1);
        quickSort(array, pivot + 1, end);
    }
}
bool isPlacingPossible(int* arr,int mid,int size,int cows){
    int pos=arr[0];
    int elements=1;
    for(int i=1;i<size;i++){
        if(arr[i]-pos>=mid){
            pos=arr[i];
            elements++;
        }
         if(elements==cows){
        return 1;
    }
    }
    return 0;
}


int minSpace(int* arr,int size,int cows){
    int minSpace=-1;
   long int left=arr[0];
    long int right=arr[size-1];
    while(left<right){
      long long int mid=(left+right)/2;
        if(isPlacingPossible(arr,mid,size,cows)){
            left=mid+1;
            minSpace=mid;
        }
        else{
            right=mid;
        }
    }
    return minSpace;
}
int main() {
    int hutsNumber;
    cin>>hutsNumber;
    int cowsNumber;
    cin>>cowsNumber;
    int* coordinates=new int[hutsNumber];
    for(int i=0;i<hutsNumber;i++){
        cin>>coordinates[i];
    }
    quickSort(coordinates,0,hutsNumber-1);
    cout<<minSpace(coordinates,hutsNumber,cowsNumber);
    return 0;
}

```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code