Equal

Christy has erred by distributing chocolates unevenly among her coworkers and now has to correct her mistake by equalizing chocolates of coworkers using minimum number of operations.

Christy has to equalize the number of chocolates for all the coworkers. The only action she can make at every operation is to increase the count of every others' chocolate by 1,2 or 5 except one of them. This is equivalent to saying, christy can take away the chocolates of one coworker by 1, 2 or 5 while keeping others' chocolate untouched.

Let's consider decreasing a coworker's chocolate as an operation. To minimize the number of operations, we should try to make the number of chocolates of every coworker equal to the minimum one in the group(min). We have to decrease the number of chocolates the i^{th} person A[i] by (A[i] - min). Let this value be x. For this you may consider Coin change algorithms.

we now follow a greedy algorithm so number of operations required is minimum. This can be done in k operations.

```
k = x/5 +(x%5)/2 + (x%5)%2
```

Let f(min) be sum of operations performed over all coworkers to reduce each of their chocolates to min.

However, sometimes f(min) might not always give the correct answer. It can also be a case when

```
f(min) > f(min-1)
```

but it is safe to assume that

```
f(min) < f(min-5)
```

as f(min-5) takes N operations more than f(min) where N is the number of coworkers.

Therefore, if

```
A = {min,min-1,min-2,min-3,min-4}
```

then,

```
f(A) <= f(min) < f(min-5)
```

Compute f(y) ∀ y ∈ A and print the minimum as the answer.

Set by amitiitkgp

Problem Setter's code:

C++

```
#include<iostream>
#include<cstdio>
#define INT_MAX 2000000000

using namespace std;
```

```
long long int functn (long long int temp)  // similar to greedy Coin-change
{
    long long int x=0;
    if(temp>=5)
    {
        x = temp/5;
        temp = temp%5;
    }
    if(temp>=2)
    {
        x += temp/2;
        temp = temp%2;
    }
    x += temp;
    return x;

}

int array_smallest(long long int array[],int array_length)
{
    long long int smallest = INT_MAX;
    long long int i;
    for (i = 0; i < array_length; i++)
    {
        if (array[i] < smallest) {
            smallest = array[i];
        }
    }
    return smallest;
}

long long int mod(long long int x)
{
    if(x>0)
        return x;
    else
        return (-1)*x;
}

int main()
{
    long long int T,N,i,j,min,sum,temp;
    cin>>T;
    while(T--)
    {
        min = 1000000;
        cin>>N;
        int A[N];
        for(i=0 ; i< N ; i++)
        {
            cin>>A[i];
            if(A[i]<min)
                min = A[i];
        }
        long long int sum[6];
        for(j=0 ; j<=5 ; j++)
        {
            sum[j]=0;
            for(i=0 ; i< N ; i++)
            {
                temp = mod(A[i] - (min-j));
                sum[j] = sum[j] + functn(temp);
            }
        }
        cout<<array_smallest(sum,6)<<endl;
    }
    return 0;
}
```

Problem Tester's code:

C++

```
#include<cassert>
#include<iostream>
#include<algorithm>
using namespace std;

int T, N;
int d[100010];

int main() {
    cin >> T;
    assert(T <= 100);
    while(T--) {
        cin >> N;
        assert(N <= 10000);
        for(int i = 0; i < N; i++)
        {cin >> d[i]; assert(d[i] < 1000);}
        int M = *min_element(d, d + N);
        int r = 1e9;
        for(int t = M - 4; t <= M; t++) {
            int s = 0;
            for(int i = 0; i < N; i++) {
                int D = d[i] - t;
                s += D / 5, D %= 5;
                s += D / 2, D %= 2;
                s += D;
            }
            r = min(r, s);
        }
        cout << r << endl;
    }
    return 0;
}
```

Feedback

Was this editorial helpful?

Yes | No