***OPERATE ONCE***

Given an array A with elements from set {1,2,3}. we need to perform **one operation only once**.

The operation is to decrease element at any index by value 1 and increase value of any element at some other index by 1

Now find out the maximum size of a subarray with all equal elements. Print that maximum size.

**Input:**

The first line of the input contains a single integer T denoting the number of test cases. The description of T test cases follows.

The first line of each test case contains an integer N denoting number of elements in Array

The second line contains N space-separated integers A1, A2, ..., AN denoting the elements of array A. each element belongs to the set {1,2,3}

**Output:**

For each Test case, print the Maximum possible Size of Subarray with all equal elements, after performing operation only once.

**Constraints:**

1 ≤ T ≤ 15

2 ≤ N ≤ 105

1 ≤ A[i] ≤ 3

**SAMPLE INPUT**

2

6

1 2 2 1 2 3

2

1 1

**SAMPLE OUTPUT**

5

1

**Explanation**

In First Testcase if we decrease the element at index 6 by 1 and increase the element at index 4 by 1 then the final array becomes { 1, 2 ,2 ,2 ,2 ,2}.

hence max. length is 5.

In Second Testcase if we decrease the first element by 1 and increase the second element by 1 then the array becomes {0,2}

hence the max.length is 1

**Time Limit:**1.0 sec(s) for each input file.

**Memory Limit:**256 MB

**Source Limit:**1024 KB

#include<bits/stdc++.h>

#define pp pop\_back

#define pb push\_back

#define int long long int

#define INF 1e18

#define vec vector<int>

#define pii pair<int,int>

#define REP(i,a,b) for(i=a;i<b;i++)

using namespace std;

vector<int> a;

bool check(int x)

{

int ar[4]={0},n=a.size();

for(int i=0;i<x;i++)

ar[a[i]]++;

if(ar[1]==1 && ar[3]==1)

return true;

else if((ar[1]==x || ar[2]==x || ar[3]==x) && n-x>=2)

return true;

else if(((ar[2]==1 && ar[3]==x-1) || (ar[3]==1 && ar[2]==x-1)) && n-x>=1)

return true;

else if(((ar[1]==1 && ar[2]==x-1) || (ar[2]==1 && ar[1]==x-1)) && n-x>=1)

return true;

int i=x;

while(i<n)

{

ar[a[i]]++,ar[a[i-x]]--;

if(ar[1]==1 && ar[3]==1)

return true;

else if((ar[1]==x || ar[2]==x || ar[3]==x) && n-x>=2)

return true;

else if(((ar[2]==1 && ar[3]==x-1) || (ar[3]==1 && ar[2]==x-1)) && n-x>=1)

return true;

else if(((ar[1]==1 && ar[2]==x-1) || (ar[2]==1 && ar[1]==x-1)) && n-x>=1)

return true;

i++;

}

return false;

}

int bsta()

{

int l=1,r=a.size(),ans=1;

while(l<=r)

{

int mid=l+(r-l)/2;

if(check(mid))

ans=max(ans,mid),l=mid+1;

else

r=mid-1;

}

return ans;

}

int32\_t main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

a.clear();

for(int i=0;i<n;i++)

{

int q;

cin>>q;

a.pb(q);

}

cout<<bsta()<<"\n";

}

}