Globos

Autor: Darío Martínez Ramírez

El primer paso es çomprimir"los segmentos consecutivos de globos del mismo color. Creamos un nuevo vector donde cada elemento representa un segmento y contiene un entero identificando el color y otro diciendo cuantos globos contiene (para hacer la última subtask, comprimir no es necesario, pues todos los segmentos son de tamaño 1).

Vamos a crear un vector $ans[i]$ que nos diga la respuesta pinchando el color i.

Primero miramos cuáles son los segmentos más grandes que hay antes de pìnchar globos. Si el segmento más grande es del color a y longitud l_1, y el segmento más grande de otro color es de color b y longitud l_2, inicializaremos $ans[i] = l_1$ para $i \neq a$ y $ans[a] = l_2$.

Ahora miramos los segmentos que resultan de juntarse varios al pinchar globos. Nótese que para que al pinchar el color a se junten todos los segmentos en el intervalo $[l, r]$ de color b en uno solo, los colores de la array de segmentos comprimidos en $[l, r]$ tendrán que ser $baba \ldots bab$, es decir, que solo hay esos dos colores alternándose.

Para calcular esto iteraremos por los segmentos comprimidos de izquierda a derecha. Mantenemos 4 variables:

1. c_1: El color del último segmento visto
2. c_2: El color del penúltimo segmento visto
3. s_1: La suma de los tamaños de los segmentos de color c_1 desde que c_1 y c_2 comenzaron a alternarse
4. s_1: La suma de los tamaños de los segmentos de color c_1 desde que c_1 y c_2 comenzaron a alternarse

Claramente podemos mantener estas cuatro variables. Después de actualizarlas, actualizaremos el vector respuesta con $ans[c_2] = \text{máx}(ans[c_2], s_1)$.

Por tanto, lo primero que hacemos se encarga de los segmentos que no se agrandan al pinchar globos, y la segunda cosa que hacemos se encarga de los que sí, resolviendo el problema.

```cpp
#include<bits/stdc++.h>

using namespace std;

#define int long long
#define vi vector<int>
#define vvi vector<vi>
#define vvvi vector<vvi>
#define pii pair<int,int>
#define vii vector<pii>
#define vvii vector<vii>
#define fi first
#define se second
#define pb push_back

#define sz(x) (int)(x).size()

#define piiii pair<pii,pii>

signed main(){
    int t;
    cin >> t;
    while (t--){
        int n,k;
        cin >> n >> k;
        vi v(n);
        for (int &x:v) cin >> x;
        for (int &x:v) x--;

        vii seginfo(n); //iniciseg, finalseg
```

```
        int l=0;
        while (l<n){
            int r=l;
            while (r<n-1 && v[r+1]==v[l]) r++;
            for (int i=l;i<=r;i++) seginfo[i]={l,r};
            l=r+1;
        }

        vi ans(k,0);

        /// INTERSECTING

        piiii col1, col2; //firstpos, lastpos, col, number
                        // dos ultimos colores
        col1=piiii{{0,-1},{-1,-1e9}};
        col2=piiii{seginfo[0],{v[0],seginfo[0].se-seginfo[0].fi+1}};
        int ac=seginfo[0].se+1;
        while (ac<n){
            if (v[ac]==col1.se.fi){
                col1.fi.se=seginfo[ac].se;
                col1.se.se+=seginfo[ac].se-seginfo[ac].fi+1;
                swap(col1,col2);
            }
            else if (v[ac]==col2.se.fi){
                col2.fi.se=seginfo[ac].se;
                col2.se.se+=seginfo[ac].se-seginfo[ac].fi+1;
            }
            else{
                if (col1.se.fi!=-1) ans[col1.se.fi] = max(ans[col1.se.fi],col2.se.se);
                ans[col2.se.fi] = max(ans[col2.se.fi],col1.se.se);

                col2.fi.fi = col1.fi.se+1;
                col2.se.se = col2.fi.se-col2.fi.fi+1;

                col1=col2;
                col2={seginfo[ac],{v[ac],seginfo[ac].se-seginfo[ac].fi+1}};
            }
            ac=seginfo[ac].se+1;
        }
        ans[col1.se.fi] = max(ans[col1.se.fi],col2.se.se);
        ans[col2.se.fi] = max(ans[col2.se.fi],col1.se.se);

        //// NON-INTERSECTING

        vi longestseg(k);
        for (int i=0;i<n;i++) longestseg[v[i]]=max(longestseg[v[i]],seginfo[i].se-seginfo[i].fi+1);

        int xd=0;
        for (int i=0;i<k;i++){
            ans[i]=max(ans[i],xd);
            xd=max(xd,longestseg[i]);
        }

        xd=0;
        for (int i=k-1;i>=0;i--){
            ans[i]=max(ans[i],xd);
            xd=max(xd,longestseg[i]);
        }

        ///// PRINT

        for (int i=0;i<k;i++) cout << ans[i] << ' ';
        cout << endl;
    }
}
```