**Reunion of 1's**

9186

88%

30

★★★★★

**Problem**

You are given a string of size **n** consisting of **0s** and/or **1s**. You have to perform total **k** queries and there are **two** types of queries possible:

1. "**1**" (without quotes): Print length of the longest sub-array which consists of all '1'.
2. "**2 X**" (without quotes): where **X** is an integer between **1** to **n**. In this query, you will change character at the Xth position to '1' (It is possible that the character at i-th position was already '1').

**Input Format**

* First line of input contains **n** and **k**, where **n** is the length of the string, **k** is the number of queries.
* Next line contains a string of 0's and/or 1's of length **n**.
* Each of next **k** lines contains query of any one type(i.e 1 or 2).

**Input Constraints**

* 1≤n,k≤105
* 1≤X≤n
* String contains only 0s and/or 1s.
* Each query is of one of the mentioned types.

**Output Format**

* For each query of type **1**, print in a new line the maximum size of the subarray with all 1's

**Sample Input**

5 7

00000

1

2 3

1

2 5

1

2 4

1

**Sample Output**

0

1

1

3

#include<bits/stdc++.h>

#define ll int64\_t

#define mod 1000000007

#define f(i, in, n) for(int64\_t i=in; i<n; i++)

#define fl(i, in, n) for(int64\_t i=n-1; i>=in; i--)

#define lim(i, a, b) for (int64\_t i=a; i<b; i++)

#define siz 1000001

using namespace std;

ll find(ll n, vector<ll> &parent, vector<ll> &rank) {

    if (parent[n]==n) return n;

    return parent[n]=find(parent[n], parent, rank);

}

void unionByRank(ll a, ll b, vector<ll> &parent, vector<ll> &rank) {

    ll x=find(a, parent, rank);

    ll y=find(b, parent, rank);

    if (x==y) return;

    if (rank[x]>=rank[y]) {

        parent[y]=x;

        rank[x]+=rank[y];

    }

    else {

        parent[x]=y;

        rank[y]+=rank[x];

    }

}

int main() {

    ios\_base::sync\_with\_stdio(false);

    cin.tie(NULL);

    //SieveOfEratosthenes();

    ll t=1;

    //cin>>t;

    while (t--) {

        ll n, k;

        cin>>n>>k;

        string s;

        cin>>s;

        vector<ll> parent(n), rank(n, 0);

        f(i, 0, n) parent[i]=i;

        ll length=0;

        f(i, 0, n-1) {

            if (s[i]=='1') rank[i]++;

        }

        f(i, 0, n-1) {

            if (rank[i]>=1 and rank[i+1]>=1) unionByRank(i, i+1, parent, rank);

        }

        f(i, 0, n) length=max(length, rank[i]);

        while (k--) {

            ll que;

            cin>>que;

            if (que==2) {

                ll in;

                cin>>in;

                in--;

                if (s[in]=='1') continue;

                s[in]='1';

                rank[in]++;

                ll left=0, right=0;

                if (in-1>=0) left=rank[find(in-1, parent, rank)];

                if (in+1<n) right=rank[find(in+1, parent, rank)];

                if (left>=1) unionByRank(in, in-1, parent, rank);

                if (right>=1) unionByRank(in, in+1, parent, rank);

                length=max(length, rank[find(in, parent, rank)]);

            }

            else cout<<length<<"\n";

        }

    }

    return 0;

}