

문제해결기법 - 201921725 안성현

---

# Double Rainbow

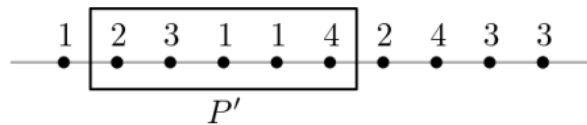
<2021/12/25>

## 문제 소개 및 접근법

### 1-1] Double Rainbow

#### ① 문제

Let  $P$  be a set of  $n$  points on the  $x$ -axis and each of the points is colored with one of the colors  $1, 2, \dots, k$ . For each color  $i$  of the  $k$  colors, there is at least one point in  $P$  which is colored with  $i$ . For a set  $P'$  of consecutive points from  $P$ , if both  $P'$  and  $P \setminus P'$  contain at least one point of each color, then we say that  $P'$  makes a *double rainbow*. See the below figure as an example. The set  $P$  consists of ten points and each of the points is colored by one of the colors 1, 2, 3, and 4. The set  $P'$  of the five consecutive points contained in the rectangle makes a double rainbow.



Given a set  $P$  of points and the number  $k$  of colors as input, write a program that computes and prints out the minimum size of  $P'$  that makes a double rainbow. The names of your program file and report file are rainbow.cpp(c) and rainbow.pdf, respectively, and the time limit of your program is 1 second. There is no partial score.

#### Input

Your program is to read from standard input. The input starts with a line containing two integers  $n$  and  $k$  ( $1 \leq k \leq n \leq 1,000,000$ ), where  $n$  is the number of the points in  $P$  and  $k$  is the number of the colors. Each of the following  $n$  lines consists of an integer from 1 to  $k$ , inclusively, and the  $i$ -th line corresponds to the color of the  $i$ -th point of  $P$  from the left.

#### Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum size of  $P'$  that makes a double rainbow. If there is no such  $P'$ , print 0.

#### ② 설명

▶ 투 포인터 기법을 이용해서 문제를 풀었다. 투 포인터는 배열 위의 두 개의 원소를 가리키는 두 포인터를 말한다. 그렇다고 원소를 직접 가리키는 포인터 변수를 만들 필요는 없고 배열의 인덱스 두 개를 저장하는 변수 두 개를 이용하면 된다. 필자는 arr배열(입력된 색이 저장된 배열) 위에 투 포인터로 작용하는 l변수와 r변수를 만들었다. l은 P'배열의 왼쪽 끝 부분, r은 P'배열의 오른쪽 끝 부분이라고 생각했다. 그리고 Double Rainbow를 판정하기 위해 배열 두 개와 변수 두 개를 만들었다.

color1, color2배열은 각각 P'의 색을 저장하는 배열, P-P'의 색을 저장하는 배열이다. nclr1,

nclr2 변수는 각각 P'의 색 개수(중복 제외), P-P'의 색 개수(중복 제외)이다. n을 10, k를 4, 입력을 (1,2,3,1,1,4,2,4,3,3)이라고 가정하면 초기에 { color1:(0,0,0,0), color2:(3,2,3,2), nclr1=0, nclr2=4 } 형태로 저장된다. 이후 처음 l과 r을 1,0으로 초기화하고 P'의 색 개수(nclr1)가 전체 색 개수(k)와 같아질 때까지 r을 증가시킨다. l과 r이 바뀌면서 배열과 변수의 상태도 변하게 되는 구조이다. ex) { l=1, r=0, color1:(0,0,0,0), color2:(3,2,3,2), nclr1=0, nclr2=4 } -> { l=1, r=1, color1:(1,0,0,0), color2:(2,2,3,2), nclr1=1, nclr2=4 }

P'가 k개의 색을 가졌으면 P-P'도 k개의 색을 가졌는지 확인한다. 즉 nclr1과 nclr2가 동시에 k와 같은지 확인한다. 만약 같다면(Double Rainbow이면) 그 때의 P' 크기를 정답 후보에 저장한다. Double Rainbow 여부를 확인했으면 P'가 아직 k개의 원소를 가진 상태이니 l을 1만큼 증가시켜도 k개의 원소를 가지는지, Double Rainbow인지 확인한다. 만약 k개의 원소를 가지지 않는다면 k개의 원소를 가질 때까지 r을 다시 증가시킨다. (l과 r이 n을 넘지 않을 때까지 하늘색 과정을 반복) 반복문을 벗어나면 정답 후보들 중 가장 작은 값을 리턴한다. 만약 정답 후보가 없다면 Double Rainbow를 하나도 못 찾았다는 의미이니 0을 리턴한다.

이런 방식으로 풀지 않고 단순하게 모든 경우(가능한 모든 P': nH2개)에 대해 Double Rainbow를 확인한다면 절대로 시간 안에 해결할 수 없다.

투 포인터로 일부 조건(One Rainbow)을 만족시킨 다음 어떻게 전체 조건(Double Rainbow)을 만족시킬지 고민한다면 훨씬 성능이 좋은 프로그램을 구현할 수 있을 것이다.

TMI) 아래는 이해를 돕기 위한 배열과 변수 상태 예시이다.

(l, r / color1, nclr1 / color2, nclr2), find: double rainbow찾음, min:정답 후보 중 최소값

```

Microsoft Visual Studio 디버그 콘솔
10 4
1 2 3 1 1 4 2 4 3 3
l=1, r=1 / 1 0 0 0 nclr1=1 / 2 2 3 2 nclr2=4
l=1, r=2 / 1 1 0 0 nclr1=2 / 2 1 3 2 nclr2=4
l=1, r=3 / 1 1 1 0 nclr1=3 / 2 1 2 2 nclr2=4
l=1, r=4 / 2 1 1 0 nclr1=3 / 1 1 2 2 nclr2=4
l=1, r=5 / 3 1 1 0 nclr1=3 / 0 1 2 2 nclr2=3
l=1, r=6 / 3 1 1 1 nclr1=4 / 0 1 2 1 nclr2=3
l=2, r=6 / 2 1 1 1 nclr1=4 / 1 1 2 1 nclr2=4
find!! min =5
l=3, r=6 / 2 0 1 1 nclr1=3 / 1 2 2 1 nclr2=4
l=3, r=7 / 2 1 1 1 nclr1=4 / 1 1 2 1 nclr2=4
find!! min =5
l=4, r=7 / 2 1 0 1 nclr1=3 / 1 1 3 1 nclr2=4
l=4, r=8 / 2 1 0 2 nclr1=3 / 1 1 3 0 nclr2=3
l=4, r=9 / 2 1 1 2 nclr1=4 / 1 1 2 0 nclr2=3
l=5, r=9 / 1 1 1 2 nclr1=4 / 2 1 2 0 nclr2=3
l=6, r=9 / 0 1 1 2 nclr1=3 / 3 1 2 0 nclr2=3
l=6, r=10 / 0 1 2 2 nclr1=3 / 3 1 1 0 nclr2=3
5
C:\Users#tjdu#source#repos#Project1#Debug#Project4.exe(12660 프로세스)이(가) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하여
꼭 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.

```

## 소스 코드와 실행 결과

### 2-1] 소스 코드와 실행 결과

#### ① 코드

```
#include <iostream>
#include <vector>
#define MAX 1000001
using namespace std;

int n, k; // size of array, size of color

vector<int> arr; // array with k color
vector<int> color1; // p' color check array
vector<int> color2; // p-p' color check array

int nclr1, nclr2; // not duplicated color num of color1, color2

int find_min_dr() {
    nclr1 = 0; // initial nclr1 is 0;
    nclr2 = k; // initial nclr2 is k;

    int l, r; // left, right idx of arr => p' array
    l = 1; r = 0; // initial l,r

    int min_size = MAX; // answer candidate

    // find double rainbow and min size of p'
    while (l <= n && r <= n) {
        // we filled all elements of color1
        if (nclr1 == k) {
            // we filled all elements of color2 (dr)
            if (nclr2 == k) {
                int size = r - l + 1;
                if (size < min_size)
                    min_size = size;
            }
            // update l and variables, arrays
            l++;
            if (l > r)
                break;
            color1[arr[l - 1]]--;
            if (color1[arr[l - 1]] == 0)
                nclr1--;
            if (color2[arr[l - 1]] == 0)
                nclr2++;
            color2[arr[l - 1]]++;
        }
    }
}
```

```

        else {
            // update r and variables, arrays
            r++;
            if (r > n)
                break;
            color2[arr[r]]--;
            if (color2[arr[r]] == 0)
                nclr2--;
            if (color1[arr[r]] == 0)
                nclr1++;
            color1[arr[r]]++;
        }
    }

    // can't find double rainbow
    if (min_size == MAX)
        return 0;
    // find double rainbow
    else
        return min_size;
}

int main() {
    cin >> n >> k;

    // init
    arr.assign(n + 1, 0);
    color1.assign(k + 1, 0);
    color2.assign(k + 1, 0);

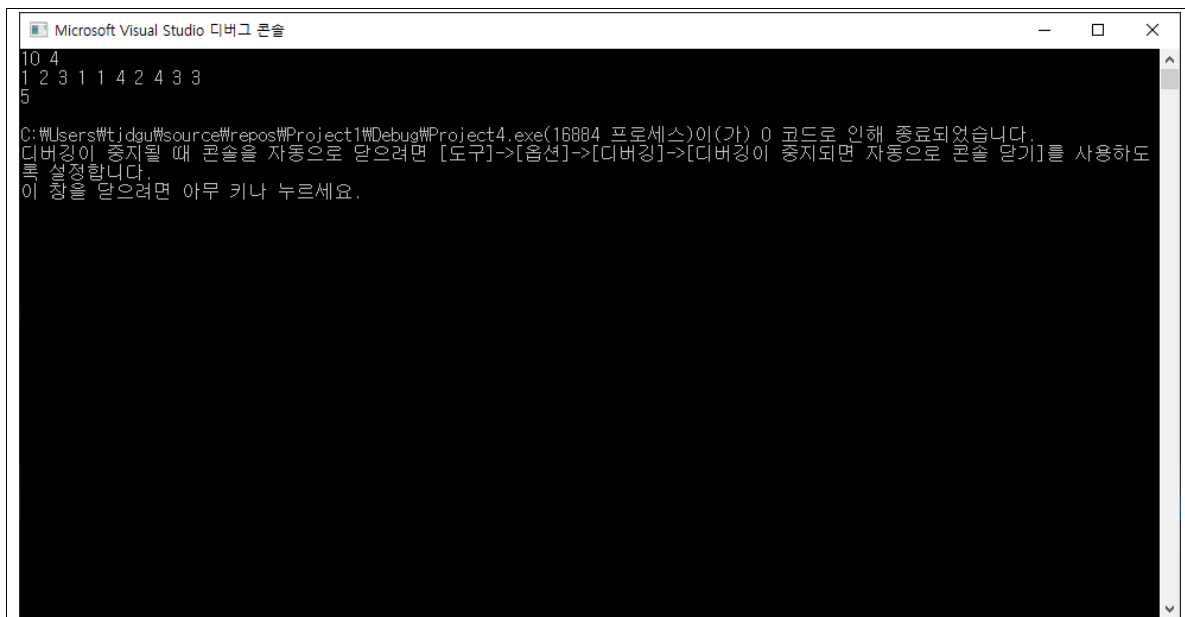
    // fill arr, color2
    for (int i = 1; i <= n; i++) {
        int num;
        cin >> num;
        arr[i] = num;
        color2[num]++;
    }

    // find min size of p' with double rainbow
    cout << find_min_dr() << endl;

    return 0;
}

```

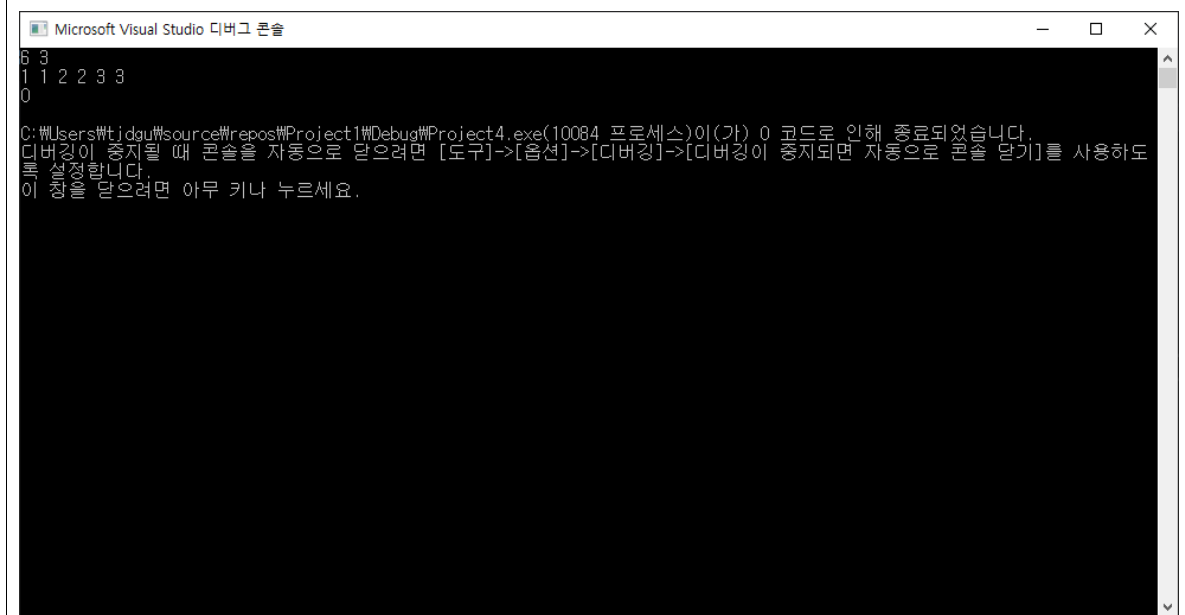
## ② DEV-C++ 컴파일러 이용 실행 결과



```
Microsoft Visual Studio 디버그 콘솔
10 4
1 2 3 1 1 4 2 4 3 3
5
C:\Users\#tjdg#\source#repos#Project1#Debug#Project4.exe(16884 프로세스)이(가) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도
록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.
```

입력: 10 (입력할 색의 개수:n), 4 (전체 색 개수:k), 색 모음 (arr)

출력: 5 (Double Rainbow를 만족하는 P'의 최소 크기)



```
Microsoft Visual Studio 디버그 콘솔
6 3
1 1 2 2 3 3
0
C:\Users\#tjdg#\source#repos#Project1#Debug#Project4.exe(10084 프로세스)이(가) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도
록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.
```

입력: 6 (입력할 색의 개수:n), 3 (전체 색 개수:k), 색 모음 (arr)

출력: 0 (Double Rainbow를 만족하는 P'는 존재하지 않음)