[백준]20444_색종이와 가위

상태	완료
날짜	@2022년 4월 12일
공부유형	스터디
알고리즘	이진탐색
사이트	백준
깃허브	☑

20444번: 색종이와 가위

오늘도 역시 준성이는 어김없이 색종이와 쿼리 를 푸는 데 실패하였다!! 색종이에 열등감을 느낀 준성이는 가위로 눈에 보이는 색종이를 모두 잘라 버리려고 한다!! 색종이를 자를 때는 다음과 같은 규칙을 따른다. 색종이는

https://www.acmicpc.net/problem/20444

개념

이진탐색 : **정렬된 배열**에서 탐색 범위를 절반씩 좁혀가며 데이터를 찾는 것

시간복잡도 :$O(logN)$

- 탐색 범위가 2000만을 넘어가면 이진 탐색으로 접근해보자.

풀이

색종이는 직사각형, 자를 때는 한변에 평행하게

자르기 시작하면 경로 상 모든 색종이 자를 때까지 멈추지 않음

이미 자른곳은 또 자르지 못함

n번의 가위질로 k개의 색종이 조각으로 만들 수 있는지 YES , NO 출력

// 이게 왜 이분탐색 문제일까.....
// 자르는 횟수 N이 고정되어있는데 가로로 자르는것, 세로로자르는것 두개밖에 없기 때문에

// **가로컷 횟수 + 세로컷 횟수 = N**
// 하나씩 늘이고 줄이면서 해보면 됨!

//근데 사실 가로나 세로나 90도 돌리면 똑같기 때문에 n/2번까지만 해보면 된다.

// 색종이 조각 개수 = (가로컷 횟수 +1) * (세로컷 횟수 +1)

[testcase1]. N=4, K=9

1️⃣start = 0 ~ end = N/2 = 2. mid = 1

mid(가로컷) = 1 , 세로컷= 4-1=3

(mid+1)*(N-mid+1) {= 2*4=8} < K {=9}

start = mid +1

2️⃣ start = 2, end = 2, mid = 2

가로컷 = 2, 세로컷 = 4-2 = 2

(mid+1)*(N-mid+1) {= 3*3} = K { =9} : YES!

[testcase2] N=4, K=6

1️⃣ start = 0, end = 2, mid = 1

가로컷 = 1, 세로컷 = 4-1=3

(mid+1)*(N-mid+1) {= 2*4=8} > K {=6}

end = mid - 1

2️⃣ start = 0, end = 0, mid = 0

가로컷 = 0 , 세로컷 = 4-0 = 4

(mid+1)*(N-mid+1) {= 1*5=5} < K {=6}

start = mid +1

3️⃣ start = 1, end = 0, mid =

→ 이진탐색 종료

근데 못찾았으니 답 없음 : NO!

코드

```
package BinarySearch;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class BinarySeach_BOJ_20444_LJE { // 색종이와 가위
  // 이게 왜 이분탐색 문제일까......
  // 자르는 횟수 N이 고정되어있는데 가로로 자르는것, 세로로자르는것 두개밖에 없기 때문에
  // 가로컷 횟수 + 세로컷 횟수 = N 이니까
  // 하나씩 늘이고 줄이면서 해보면 됨! 근데 사실 가로나 세로나 90도 돌리면 똑같기 때문에 n/2번까지만 해보면 된다.
  public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    StringTokenizer st = new StringTokenizer(br.readLine());
    long N = Integer.parseInt(st.nextToken());
    long K = Integer.parseInt(st.nextToken());

    //가로기준으로 해보쟝
    long start = 0; // 가로의 시작 인덱스
    long end = N/2;  //가로의 끝 인덱스

    while(true) {
      if(start>end)break;
      long mid = start +(end-start)/2; // =(start+end)/2   Overflow 막기위한 방법
      //mid가 가로 자르는 횟수
      //N-mid 는 세로 자르는 횟수
      if((mid+1)*(N-mid+1)==K) {
        System.out.println("YES");
        return;
      }else if((mid+1)*(N-mid+1)> K){ // 너무 많이 잘랐으면
        end = mid -1;
      }else if((mid+1)*(N-mid+1)< K) { // 좀 덜잘랐으면 가로로 더잘라줘
        start = mid +1;
      }
    }
    System.out.println("NO"); // 끝까지 돌았다는건 답이 없다는 뜻

  }

}
```

참고

Binary Seach 이진 탐색, 이진탐색트리