〈 매개변수 탐색 (Parametric Search) 〉 → 정해가 있는지, 있으면 X/O의 경계찾기.

: Binary Search를 이용하여 문제를 해결하는 테크닉

예제) 구간의 합. : N개의 숫자와 S가 주어질 때, 몇 개의 연속된 값을 합해야 그 합이 S 이상이 되는가? (단 1<=N<=100,000)

↳ 즉, 구간의 길이를 구하라는 문제.

입력의 예) 9 (N) 14 (S)

2 1 8 1 3 7 2 6 3

↓

이렇게 연속 합 = 15 > 14

출력의 예) 3

∴ 3개

알고리즘 아이디어) 답을 정해두고, 그 답이 맞는지 확인

구간 길이 1은 가능? X

2는 ? X

3은 ? O → 그래서 답이 3이다

⇒ 이것의 시간복잡도

1. 구간길이 'x'가 가능한지 판단하는데 걸리는 시간 : O(n)

2 1 8 1 3 7 2 6 3

↓ 내가. O(n)

2. 시도 해 봐야하는 구간 길이의 갯수 : n개.

∴ 총 시간복잡도 $O(n^2)$ //

근데 n<=100,000 이므로 1초 안에 해결 할수 X.

⇒ 그래서 새로운 알고리즘 아이디어.) 정말 끝까지 다 해보아야 하나? NO

ex) 구간길이 : 1 2 3 4 5 6 7 8 9 10 에서

X X O 이면, 이 뒤부턴 당연히 다 O이다.

↑

이게 N의 최소값이 된다.

즉, 이 문제는 ↓ X와 O의 '경계'를 찾는문제O

여기서 'Binary Search'를 활용하는 것이다.

그리고 Binary Search를 이용하여 경계를 찾는 것을 "매개변수탐색"이라 한다.

활용해서 풀어보자.) 구간길이 : 1 2 3 4 5 6 7 8 9

1. start와 end를 잡는다

이때 ☆☆ start는 무조건 'X'이고, end는 무조건 'O'이다.

2. mid가 X/O 뭔지 확인

3. X이면 start가 mid로 (mid+1이 아니라 mid!)

O이면 6부턴 해볼필요X ∴ end가 mid로 (×mid-1이 아니라 mid!)

4. 반복.

5. 그러다가 start end 가 나란히 있는경우 X O 나옴 "아- 여기가 경계구나!"

⇒ 이때의 시간복잡도 : O(nlogn)

```
#include <stdio.h>

//구간의 합
//N : 9 , S : 14
//2 1 8 1 3 7 2 6 3

//구간 길이 : 1 2 3 4 5 6 7 8 9중 어떤 길이부터 S가 만들어지는지 판단

//어떻게?? binary search로 X, O의 경계 찾기

const int MAX = 100010;
int n;
int s;
int data[MAX];

bool check(int interval){
//구간의 길이 interval만큼 정했을 때, 그 합이 S이상인 경우가 있는가 판단 하는 함수
//있으면 true, 없으면 false

int sum=0;

for(int i=0; i<interval; i++){
  sum += data[i];
}
if(sum>=s)
  return true;
```

```
  else
    for(int i=0; i<n-interval; i++){
      sum = sum-data[i]+data[i+interval];
      if(sum>=s)
        return true;
    }
  return false;
}

int main(){

  scanf("%d %d", &n, &s);

  for(int i=0; i<n; i++)
      scanf("%d", &data[i]);

  int start = 1; //start는 무조건 x를 가리키고
  int end = n; //end는 무조건 0를 가리킴

  if(check(1)){ //가능한 경우의 최소값일 때로 되면, 무조건 된다는 거니까
      printf("1\n");
      return 0;
  }
  if(!check(n)){ //가능한 경우 중 최대 일 때도 안된다는건, 다른어떤 경우이든 안된다는 거니까
      printf("-1\n");
      return 0;

  }

  while(start+1 < end){
    int mid = (start+end)/2;

    if(check(mid))//했는데 0라면
      end = mid;
    else
      start = mid;//X라면
  }

  printf("%d\n", end);

  return 0;

}
```

>=S 가아니면,

이렇게 해보다.

∴ data[i]를 빼고 Sum에서 data[i+interval]을 더한다.

다른경우는

return 1,, → 이정에 미리 끝낼수있는것부터 걸러줌

return -1,,

그게 아니라면

BinarySearch로 정말정석 들어갑니다 ^^