이분탐색과 누적합

이분탐색, 누적합, 슬라이딩 윈도우

Dosawas 2024-04-19

목차

차례

- 3회차 리뷰
- 이분탐색 <- 긴 배열에서 내가 원하는 정보 빨리 찾기
- Prefix sum <- 긴 배열에서 부분 합 빨리 찾기
- 슬라이딩 윈도우 <- 긴 배열에서 고정된 길이의 윈도우를 잘 처리하기

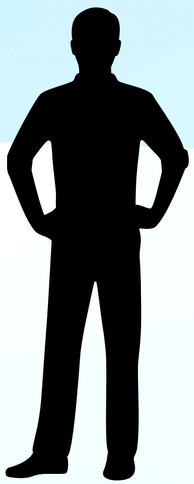
3회차 리뷰

- 스택, 큐, 덱, 리스트 -> 선형 자료구조
  - 어떤 자료구조인지 설명할 수 있으면 됨!!
- Set, map -> 오늘 log시간이 뭔지 잘 깨달아보자
  - Stl 쓰는법
  - 언제 set, map을 쓰는지 알고 있자

- 저번 시간엔 자료를 잘 관리하는 도구!! Vs 이번시간엔 자료를 잘 사용, 관리하는 방법!!

이분탐색

Up-down 게임



1부터 100까지 숫자 중
제가 생각한 숫자를
맞춰보세요!!
up-down으로
힌트를 드립니다.

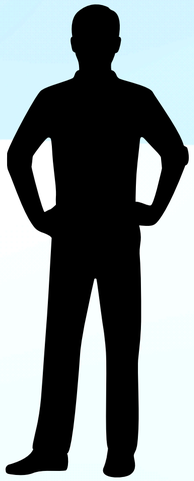
전략 1 : 1 부터 100까지 차례대로 불러본다.

전략 2 : 1 부터 100사이 숫자 중 랜덤으로 하나씩 말해본다.

전략 3 : 이분탐색

이분탐색

Up-down 게임



Up

Down

Down

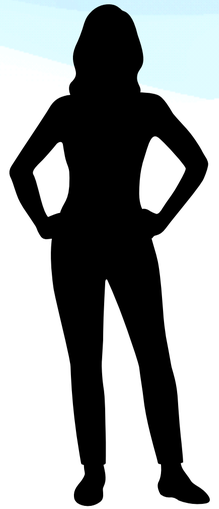
정답...

50

75

67

58

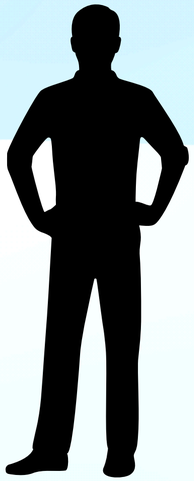


검사하는 범위가 절반씩

줄어든다!! $O(logN)$

이분탐색

Up-down 게임



Up

Down

Down

정답...

Low = 1
Hi = 100

Low = 50
Hi = 100

Low = 50
Hi = 75

Low = 50
Hi = 67

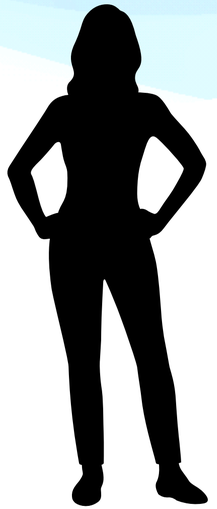
Low = 50
Hi = 58

50

75

67

58



이분탐색

STL 사용

정렬된 배열에서 어떤 정보를 찾고 싶을때!!!!

```
13  int main(){
14      fast_io
15      // 9가 어디에 들어갈지 찾기
16      int arr1[] = {1, 3, 5, 11};
17      cout << lower_bound(arr1, arr1+4, 9) - arr1 << '\n'; // 3이 출력됨
18      vector<int> arr2 = {1, 3, 5, 11};
19      cout << upper_bound(arr2.begin(), arr2.end(), 9) - arr2.begin() << '\n'; //3이 출력
20
21      // 4가 어디에 들어 갈 수 있는지 찾기
22      int a[] = {4, 4, 9, 11, 11};
23      // 4가 처음 등장하는 곳의 위치
24      cout << lower_bound(a, a+5, 4) - a << '\n'; // 0이 출력됨
25      // 4보다 처음 커지는 곳의 위치
26      cout << upper_bound(a, a+5, 4) - a << '\n'; // 2가 출력됨
27
28      vector<int> b = {4, 4, 9, 11, 11};
29      cout << *lower_bound(b.begin(), b.end(), 4) << '\n'; // 4가 출력됨
30      cout << *upper_bound(b.begin(), b.end(), 4) << '\n'; // 9가 출력됨
31  }
```

이분탐색

이분탐색

Arr라는 배열에
14보다 처음으로
커지는 위치는?

Arr라는 배열에 14가 처음 들어갈 수
있는 곳은??

Index	0	1	2	3	4	5	6	7
Arr	1	14	15	18	22	22	25	52

lo = 0 mid = 3 hi = 7

기저 사례 : lo + 1 == hi

즉, lo + 1 < hi인 동안

이분탐색

이분탐색

Arr라는 배열에
14보다 처음으로
커지는 위치는?

Arr라는 배열에 14가 처음 들어갈 수
있는 곳은??

Index	0	1	2	3	4	5	6	7
Arr	1	14	15	18	22	22	25	52

lo = 0　　mid =1　　hi = 3

기저 사례 : lo + 1 == hi

즉, lo + 1 < hi인 동안

이분탐색

이분탐색

Arr라는 배열에
14보다 처음으로
커지는 위치는?

Arr라는 배열에 14가 처음 들어갈 수
있는 곳은??

Index	0	1	2	3	4	5	6	7
Arr	1	14	15	18	22	22	25	52

lo = 1 mid = 2 hi = 3

기저 사례 : lo + 1 == hi

즉, lo + 1 < hi인 동안

이분탐색

이분탐색

Arr라는 배열에
14보다 처음으로
커지는 위치는?

Arr라는 배열에 14가 처음 들어갈 수
있는 곳은??

Index	0	1	2	3	4	5	6	7
Arr	1	14	15	18				

lo = 1 hi = 2

기저 사례 : lo + 1 == hi

여기서 답을 lo로 할지 , hi로 할지 잘 선택해야함

```
int bi_search(){
    int lo = 가능한 값;
    int hi = 불가능한 값;
    // int lo = 불가능한 값;
    // int hi = 가능한 값
    while(lo + 1 < hi){
        int mid = (lo + hi) >> 1;
        if(can(mid)) lo = mid;  // hi = mid;
        else hi = mid; // lo = mid;
    }
    return lo; // return hi;
}
```

이분탐색

Parametric search(매개변수 탐색)

- 1. 특정 조건을 만족하는 최댓값/최솟값을 구하는 문제
  - 최댓값을 구하는 경우 그 값보다 작은 값은 모두 가능해야함.
    - 예를 들면, 음주운전이 아닌 최대 혈중 알코올 농도가 0.03이라고 하면 0.03보다 작은 경우는 모두 음주운전이 아니다!!!
- 2. 이산적인 답
  - 위 예시에서 음주운전이 0.03이상이라고 하면
  - 0.2999999999999….가 음주운전이 아닌 최대 혈중 알코올 농도가 되는데 이는 정확한 값이 아니다..

이분탐색

Parametric search(매개변수 탐색) - 공유기설치 (G4)

문제

도현이의 집 N개가 수직선 위에 있다. 각각의 집의 좌표는 x_1, ..., x_N이고, 집 여러개가 같은 좌표를 가지는 일은 없다.

도현이는 언제 어디서나 와이파이를 즐기기 위해서 집에 공유기 C개를 설치하려고 한다. 최대한 많은 곳에서 와이파이를 사용하려고 하기 때문에, 한 집에는 공유기를 하나만 설치할 수 있고, 가장 인접한 두 공유기 사이의 거리를 가능한 크게 하여 설치하려고 한다.

C개의 공유기를 N개의 집에 적당히 설치해서, 가장 인접한 두 공유기 사이의 거리를 최대로 하는 프로그램을 작성하시오.

입력

첫째 줄에 집의 개수 N ($2 \le N \le 200,000$)과 공유기의 개수 C ($2 \le C \le N$)이 하나 이상의 빈 칸을 사이에 두고 주어진다. 둘째 줄부터 N개의 줄에는 집의 좌표를 나타내는 x_i ($0 \le x_i \le 1,000,000,000$)가 한 줄에 하나씩 주어진다.

이분탐색

Parametric search(매개변수 탐색) -

문제

도현이의 집 N개가 수직선 위에 있다. 각각의 집의 좌표는 x_1, ..., x_N이고, 집 여러개가 같

도현이는 언제 어디서나 와이파이를 즐기기 위해서 집에 공유기 C개를 설치하려고 한다. 수
있고, 가장 인접한 두 공유기 사이의 거리를 가능한 크게 하여 설치하려고 한다.

C개의 공유기를 N개의 집에 적당히 설치해서, 가장 인접한 두 공유기 사이의 거리를 최대

입력

첫째 줄에 집의 개수 N (2 ≤ N ≤ 200,000)과 공유기의 개수 C (2 ≤ C ≤ N)이 하나 이상 x_i ≤
1,000,000,000)가 한 줄에 하나씩 주어진다.

1. Naive하게 어떻게 풀까?
    1. $\binom{N}{C} \times N$의 시간 복잡도로 풀 수 있다.
2. 지금 문제가 “특정 조건을 만족하는 최댓값”을 구하는 문제
    1. -> 그럼 매개변수 탐색을 생각!!
    2. 만약 가장 인접한 공유기 사이의 거리가 D라고 하면 N개의 집에 가장 인접한게 D도록 설치할 수 있을까? - greedy한 생각 필요
    3. 그럼 설치가 가능한 최대 D를 찾아보자!!

이분탐색

Parametric search(매개변수 탐색) -

<코드 리뷰>

1. 이분탐색을 하려고 맘 먹었으면 - **sort해야함**
2. 값이 될 수 있는 범위를 잘 설정하고 시작
    1. hi에 답이 될 수 없는 적당히 큰 값
    2. lo에 답이 될 수 있는 적당히 작은 값
    3. **이 둘은 문제에 따라 반대가 되기도 함**
3. can함수로 따로 빼는게 보기 더 좋음

```cpp
int n, c;
vector<int> house;

bool can(int d){
    int cnt = 1; // 첫번째 집에 공유기 박고 시작
    int prev = house[0]; // 이전에 공유기 설치한 곳 위치
    for(int i=1;i<n;i++){
        if(house[i] - prev >= d){
            cnt++;
            prev = house[i];
        }
    }
    return (cnt >= c) ;
    // cnt가 c보다 크거나 같으면 d의 간격으로 설치하고도 남는다는 뜻이니까 설치가능
    // cnt가 c보다 작으면 d의 간격으로는 최대 cnt개 만큼의 공유기밖에 설치 못한다는 뜻이니 설치 불가능
}

int bi_search(){
    // 최대간격 D가 될 수 있는 범위 : lo <= D < hi
    int lo = 0, hi = house[n-1] + 1;
    while(lo + 1 < hi){
        int mid = (lo + hi) >> 1;
        if(can(mid)) lo = mid;
        else hi = mid;
    }
    return lo;
}

int main(){
    fast_io
    cin >> n >> c;
    house = vector<int> (n);
    for(int i=0;i<n;i++) cin >> house[i];
    sort(house.begin(), house.end());
    cout << bi_search();
}
```

Prefix Sum

- Prefix : 접두사 (미리 고정한다)
- Sum : 합
- 누적 정보, 앞에서부터 더한 값
- 예를 들면,
  - 우리가 물건을 팔아서 하루단위로 지금까지 번 돈을 기록한다고 해봅시다.
  - 1일차 : 50만원 , 2일차 : 130만원 , 3일차 : 240만원, 4일차 : 260만원
  - 이렇게 누적으로 정보를 저장해두면 편한 부분이 생깁니다.
    - 1. 지금까지 번 돈을 한 눈에 확인할 수 있다.
    - 2. L~R일차에 얼마를 벌었는지 쉽게 확인할 수 있다. (3~4일차 = 4일차 - 2일차 = 130만원)

Prefix Sum

Index	1	2	3	4	5	6	7	8
Arr	1	14	23	4	-10	10	10	2
PS	1	15	38	42	32	42	52	54

Arr[i] : i일차에 번 돈 <- 손실(minus 값)이 있어도 상관없음

PS[i] : i일차까지 번 돈

재밌는 사실 : 만약 -값이 없다고 생각하면 PS는 정렬된 상태일 것이다.

Prefix Sum

Index	1	2	3	4	5	6	7	8
Arr	1	14	23	4	-10	10	10	2
PS	1	15	38	42	32	42	52	54

당신이 L일차 부터 R일차까지 얼마의 돈을 벌거나 잃었나요?

부분합 모르시는 분 : L ~ R까지 Arr값을 계속 더해주면서 계산

부분합 아는 분 : PS[R] - PS[L-1]로 구하기!!

Prefix Sum

구현

```
13  int main(){
14      fast_io
15      int N; cin >> N;
16      vector<int> arr(N+1), ps(N+1);
17      for(int i=1;i<=N;i++){ // 1-based indexing (1일차 부터를 쉽게 구하기 위함)
18          cin >> arr[i];
19          ps[i] = ps[i-1] + arr[i];
20      }
21      int Query; cin >> Query;
22      while(Query--){
23          int l, r; cin >> l >> r; // l일차부터 r일차의 수익 또는 손실이 뭐야?
24          cout << ps[r] - ps[l-1] << '\n'; // l-1이 등장하기때문에 1-based가 유리
25
26          // 부분합 모르는 사람
27          int res = 0;
28          for(int i=l;i<=r;i++){
29              res += arr[i];
30          }
31          cout << res << '\n';
32      }
33  }
```

Prefix Sum
응용

```
for (int i = 1; i <= N; i++){
    cin >> arr[i];
    ps[i] = ps[i - 1] ^ arr[i];
    // 결합법칙 성립, 연산 순서가 어찌 되었든 가능한 상황
}
ps[r] ^ ps[l-1] // r부터 l까지 xor한값
```

합만 가능한 것이 아니다.

부분 xor도 구할 수 있다.

부분 곱도 구할 수 있다. (하지만 overflow조심)

2차원 배열에서도 가능하다!!

응용할 수 있는 상황이 매우매우 많다!!!

Prefix Sum

문제 - 리그전 오브 레전드(G3)

1. Naive하게 생각하기
    1. L~R까지의 배열의 값이 {a, b, c}라고 하면 a*b + a*c + b*c를 계산한다.
    2. $\binom{n}{2} \times Q$의 연산이 필요할 것이다.
2. a*b + a*c + b*c를 구하는 법
    1. $(a+b+c)(a+b+c) - (a^2+b^2+c^2) = 2(ab+ac+bc)$
    2. 그냥 합 ps1과 제곱한것의 합 ps2를 각각 구해서 처리

출력

Q개의 줄에 걸쳐 주어진 순서대로, 각 후보 디비전에서 진행되는 리그전의 재미를

예제 입력 1 복사

```
5 3
5 1 2 3 2
2 4
4 5
1 5
```

```
11
6
63
```

- $l = 2, r = 4$: 인기가 $[1, 2, 3]$인 세 팀이 리그전에 참가한다. 이 리그전의 재미는 $(1 \times 2) + (1 \times 3) + (2 \times 3) = 11$이 된다.
- $l = 4, r = 5$: 인기가 $[3, 2]$인 두 팀이 리그전에 참가한다. 이 리그전의 재미는 $(3 \times 2) = 6$이 된다.
- $l = 1, r = 5$: 인기가 $[5, 1, 2, 3, 2]$인 다섯 팀이 리그전에 참가한다.

Prefix Sum

문제 - 리그전 오브 레전드(G3)

부분합을 제대로 응용할줄 알아야

저런 사고 방식이 나올 수 있음

1. l~r까지 합은 쉽게 구할수 있는데…

2. l~r까지 제곱의 합은 쉽게 구할 수 있는데…

-> 이 2개를 이용하면 식을 정리할 수 있구나!!

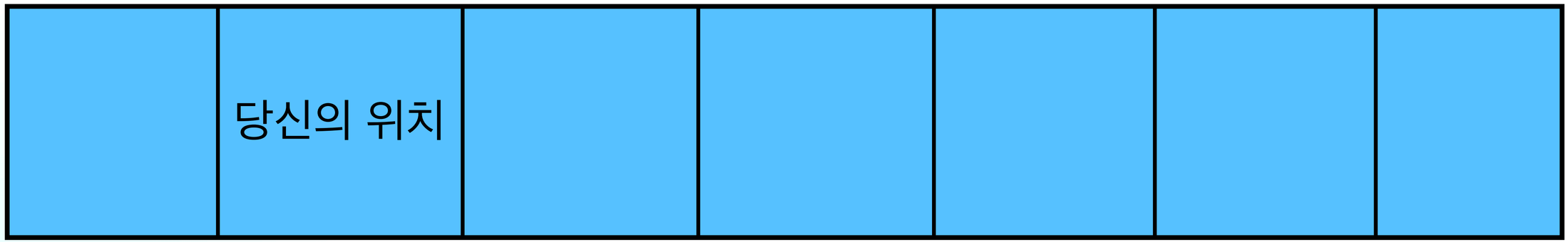
제곱의 합이니까 integer overflow 주의!

```cpp
int main(){
    fast_io int n, q;
    cin >> n >> q;
    vector<int> a(n + 1);
    vector<ll> ps(n + 1), pps(n + 1);
    for (int i = 1; i <= n; i++){
        cin >> a[i];
        // 그냥 합
        ps[i] = a[i] + ps[i - 1];
        // 제곱 합
        pps[i] = a[i] * a[i] + pps[i - 1];
    }

    while (q--){
        int l, r;
        cin >> l >> r;
        ll k = ps[r] - ps[l - 1];
        ll kk = pps[r] - pps[l - 1];
        // (a+b)(a+b) - a^2 - b^2 = 2(ab)
        ll res = (k * k - kk) / 2;
        cout << res << '\n';
    }
}
```

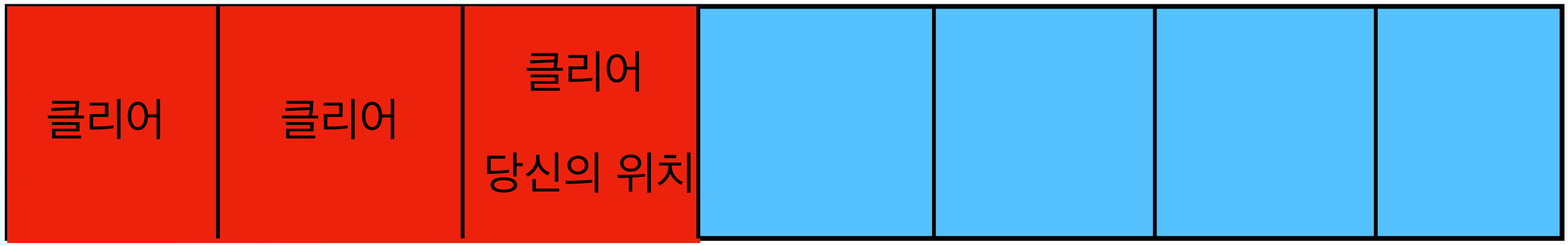
슬라이딩 윈도우(Sliding window)

- 간단하게만 할 거임!!
- 당신이 홍익대학교 건물에 매달려서 일자로 쭉 늘어진 창문을 닦는다고 해보자!!
  - 당신의 손은 가용범위가 당신의 위치에서 양옆으로 1개까지 커버가 된다.



슬라이딩 윈도우(Sliding window)

- 이동하고 나서도 나의 청소가능 범위에 이미 닦은 곳이 있다.
  - 당신은 빨간색으로 된 이미 청소한 자리를 다시 닦을 것인가??
    - “NO” 이미 청소한 곳을 다시 청소할 이유가 없다.



슬라이딩 윈도우(Sliding window)

2559 - 수열

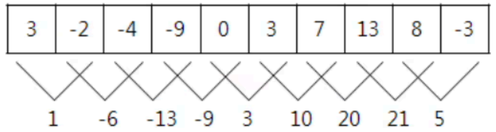
문제

매일 아침 9시에 학교에서 측정한 온도가 어떤 정수의 수열로 주어졌을 때, 연속적인 며칠 동안의 온도의 합이 가장 큰 값을 알아보고자 한다.

예를 들어, 아래와 같이 10일 간의 온도가 주어졌을 때,

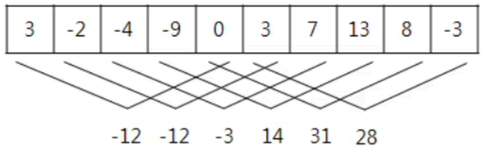
3 -2 -4 -9 0 3 7 13 8 -3

모든 연속적인 이틀간의 온도의 합은 아래와 같다.



이때, 온도의 합이 가장 큰 값은 21이다.

또 다른 예로 위와 같은 온도가 주어졌을 때, 모든 연속적인 5일 간의 온도의 합은 아래와 같으며,



이때, 온도의 합이 가장 큰 값은 31이다.

매일 측정한 온도가 정수의 수열로 주어졌을 때, 연속적인 며칠 동안의 온도의 합이 가장 큰 값을 계산하는 프로그램을 작성하시오.

슬라이딩 윈도우(Sliding window)

주의점 :

- 윈도우 사이즈가 되기 전에는

결과를 갱신하지 않음

- 답이 음수가 될 수 있으니까

매우 작은 값을 넣어둠

```
13  int main(){
14      fast_io
15      int n, window_size;
16      cin >> n >> window_size;
17      vector<int> a(n);
18      int res = -987654321;
19      int sum = 0;
20      int w_sz = 0;
21      for(int i=0;i<n;i++) {
22          cin >> a[i];
23          if(w_sz == window_size){
24              sum -= a[i-window_size];
25              sum += a[i];
26              res = max(res, sum);
27          }else{
28              w_sz++;
29              sum += a[i];
30              if(w_sz == window_size) res = max(res, sum);
31          }
32      }
33      cout << res;
34  }
```

마무리

- 오늘 배운 알고리즘은 매우 간단한 알고리즘입니다.
  - 하지만, 응용범위는 정말 넓고도 넓습니다.
- 시간 초과가 난다
  - 정렬이 되어있는 정보라서 값을 빠르게 찾을 수 있지 않을까? -> 이분탐색
  - 내가 같은 원소에 여러번 접근해서 더하고 있는가 ? -> 누적합, 슬라이딩 윈도우
  - 문제의 작은 요소로 등장하는게 부지기수다.

마무리

추천문제

- 28449 누가 이길까 : 이분탐색, 누적합 둘 다 이용 가능
- 2143 두 배열의 합 : 누적합을 한 후에 이분탐색을 하는 문제
- 2559 수열 : 슬라이딩 윈도우 기본 문제
- 29718 줄줄이 박수 : 2차원일때 슬라이딩 윈도우
- 15318 새로운 수열 :
  - 복잡한 연산을 누적합으로 하는 문제 (좀 어려움..)
  - b[l]를 b의 이전항과 a의 전체합과 a로 나타낼 수 있음
  - 홀수와 짝수의 경우가 다름… 규칙성을 따로 찾아보자