이분 탐색과 Parametric Search

190527(월) - 전현승

경북대학교 알고리즘 문제해결 동아리 GORI

목차

- 이분 탐색
- 1920 – 수 찾기
- Parametric search
- 2805 – 나무 자르기
- 1654 – 랜선 자르기
- 2110 – 공유기 설치
- 2022 – 사다리

이분 탐색

- Binary search (이진 탐색, 이분 탐색, …)

- 정렬된 배열에서 원하는 값을 빠르게 찾아내는 방법
- 반씩 쪼개서 찾아나가는 방식 → 분할 정복의 일종 (후술)

이분 탐색

2	5	7	8	9	15	21	24	29	30

- N=10짜리 정렬된 수열에서 특정한 수를 찾고 싶다

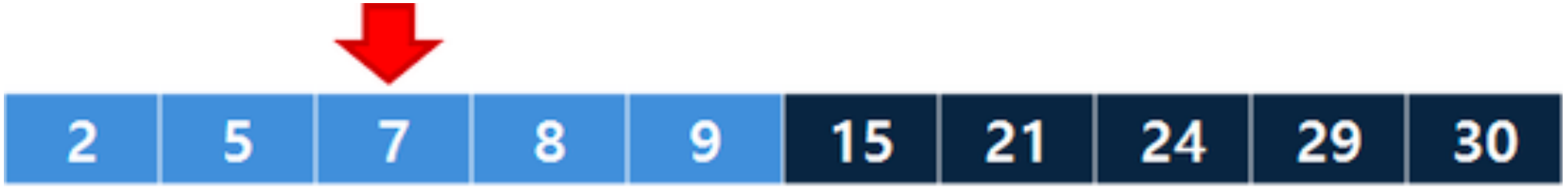
- 전형적인 Sequential search → O(N)
  - 원하는 수가 있는지 앞에서부터 쭉 찾아보는 방법
- Binary search → ?
  - 어떻게 작동하는가?

이분 탐색

- 예시 : 8을 찾고 싶다

- 업 다운 게임과 유사
- 일단 중간 원소를 보자. 15는 8보다 크다.

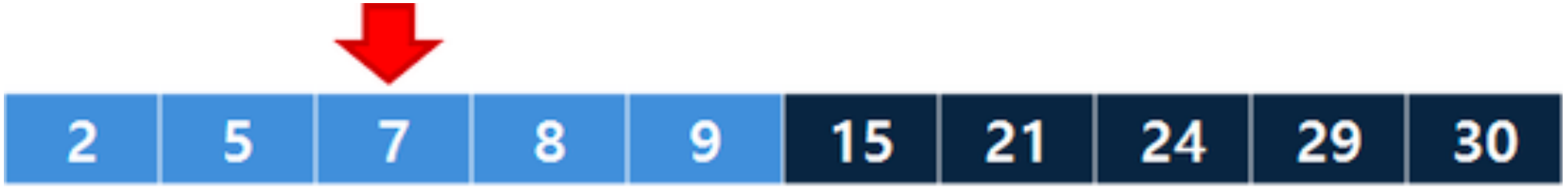
이분 탐색



- 오름차순 정렬되어있기 때문에, 15 오른쪽으로는 우리가 원하는 값 (8)이 있을 수가 없다
- 문제 크기가 반으로 줄어들었다 (수 찾을 범위가 반으로 줄었다)

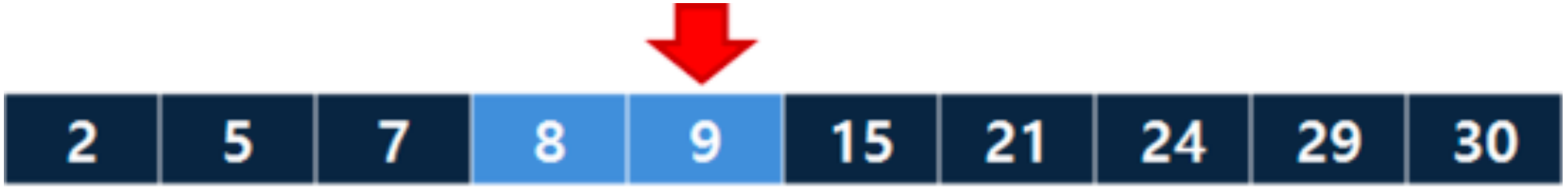
- 15 오른쪽 부분은 날리고, 왼쪽 부분에서 다시 똑같은 작업을 수행

이분 탐색



- 왼쪽 부분의 중간 원소를 다시 보자.
- 7은 8보다 작다.

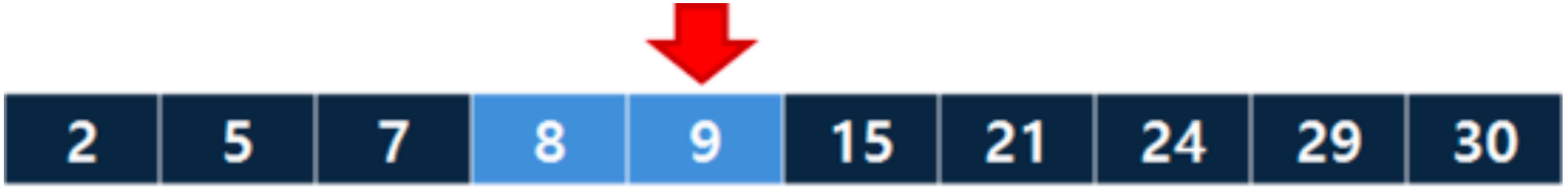
이분 탐색



- 7보다 왼쪽에 있는 원소들은 모두 8보다 작을 것이기 때문에, 마찬가지로 볼 필요가 없다

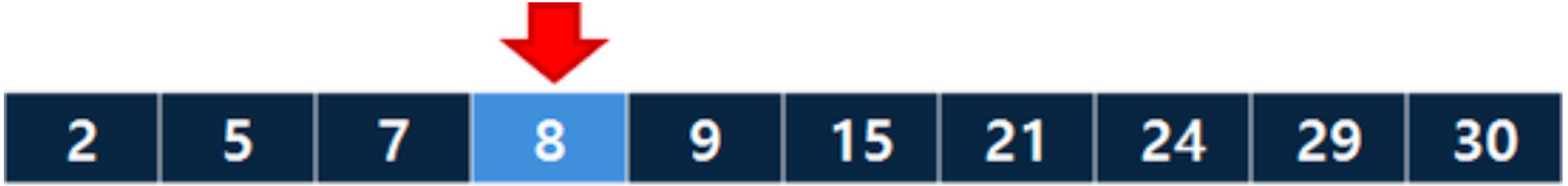
- 7 왼쪽 부분은 날리고, 7 오른쪽 부분에서 다시 탐색 진행

이분 탐색



- 중간 원소 9는 8보다 크다
- 마찬가지로 왼쪽 부분으로 이동

이분 탐색



- 중간 원소가 8이므로 원하는 값 찾기에 성공, 탐색 종료

이분 탐색

- 반복문 이용 버전

```
// a : 정렬된 배열, val : 찾으려는 값, n : 배열 길이
bool binary(int a[], int val, int n) {
    int left = 0, right = n - 1;  // left: 배열의 맨 왼쪽 idx, right: 배열의 맨 오른쪽 idx
    while (left <= right) {
        int mid = (left + right) / 2;  // 중간 idx
        if (a[mid] > val)  right = mid - 1;  // 중간값이 더 크다면, 오른쪽은 볼 필요 X
        else if (a[mid] < val) left = mid + 1;  // 중간값이 더 작다면, 왼쪽은 볼 필요 X
        else return true; // 중간값이 찾는 값 자체라면 탐색 종료
    }
    return false;  // left == right 될 때까지 못 찾았다면, 찾으려는 값이 없는 것
}
```

이분 탐색

- 재귀 이용 버전

```
// a : 정렬된 배열, val : 찾으려는 값, left: 배열의 맨 왼쪽 idx, right: 배열의 맨 오른쪽 idx
bool binary(int a[], int val, int left, int right) {
    if (left > right) return false;

    int mid = (left + right) / 2;  // 중간 idx
    if (a[mid] > val) return binary(a, val, left, mid - 1);
    else if (a[mid] < val) return binary(a, val, mid + 1, right);
    else return true;
}
```

- int형 등으로 선언해서
  다른 정보를 반환할 수도 있음 (찾으려는 수의 idx 등)

이분 탐색

- 절반씩 쪼개면서 찾아가므로
- 시간 복잡도 : O(lgN)

- left, right, mid 정의와 갱신, 반복문 탈출 조건 등 헷갈리기 쉬움
- 자신만의 기준 (구현)을 외우고 있으면 편리

이분 탐색

- 구현을 자세히 써 놨지만…
- 사실은 STL에 binary_search() 함수가 있다
- 정렬된 container에서 원하는 원소가 있는지 bool형 반환

- 그러나 사용처가 매우 한정적이어서 거의 쓰이지 않음
- 찾는 원소의 유무만 bool형으로 반환하기 때문에,
  문제가 조금만 바뀌어도 사용이 어려움
- 뒤에서 할 Parametric search 때문에라도 직접 작성하는 걸 추천

- (+ 찾는 원소의 위치를 반환하는 lower_bound(), upper_bound())

1920 – 수 찾기

- N개의 정수가 주어져 있을 때, 이 안에 X라는 정수가 존재하는지 알아내는 프로그램을 작성하시오.
- 1≤N≤100,000
- 1≤M≤100,000 (찾으려는 X 개수 == 탐색 횟수)

- Sequential search
  - O(NM) : 탐색 1번에 O(N), 탐색 총 M번 → TLE
- Binary search
  - O(NlgN + MlgN) : 정렬 1회에 O(NlgN) + 탐색 1회 O(lgN) 총 M번

1920 – 수 찾기

- http://boj.kr/ffd3bcd0239d4a598ea80366d195178c
  - binary_search() 사용
- http://boj.kr/66cb619100b04f8c8970879f6820765e
  - 이진 탐색 직접 작성

Parametric search

- 지금까지 했던 이분 탐색 아이디어로
- 최적해를 찾는 문제를 풀 수 있다
- 최소 비용, 최단 시간 등

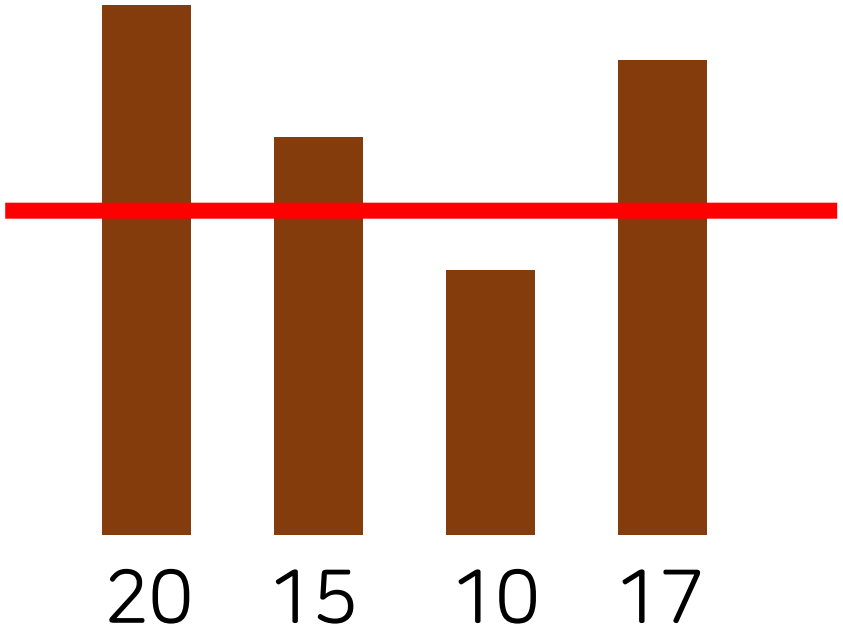
- **정답을 구하는 것은 어렵지만**
- **특정 값에 대해 가능/불가능을 따지는 건 쉬운 문제들이 있다**
- 가능/불가능 문제 + 이분 탐색 아이디어

- int overflow 주의 → long long 등 사용

GORI

2805 – 나무 자르기

- 나무 N그루, 가져가야 할 나무 길이 M
- 절단기 높이를 H로 설정하면, 모든 나무에 대해 H 윗부분을 모두 가져갈 수 있다
- M미터를 가져가기 위한 절단기 높이의 최댓값?



H = 13

가져가는 나무 길이
= 7+2+0+4

= 13

2805 - 나무 자르기

- M미터 가져가기 위한 절단기 높이의 최댓값을 바로 구하는 건 어렵지만
- 절단기 높이가 주어지면 M미터를 가져갈 수 있는지, 없는지는 알 수 있다

- 절단기 H미터에서 잘랐는데, 나무가 더 많이 나왔다면?
  - 절단기를 더 내려봤자 더 많이 나올 뿐이다. 절단기를 올려야 한다.
- 절단기 H미터에서 잘랐는데, 나무가 부족하다면?
  - 절단기를 더 올려봤자 더 적게 나올 뿐이다. 절단기를 내려야 한다.

- 절단기 높이를 가지고 이분 탐색을 해 보면 된다

2805 – 나무 자르기

```
typedef unsigned long long ull;

ull ans = 0;
// left : 절단기 높이의 하한 (0), right : 절단기 높이의 상한 (나무들 중 가장 긴 것의 높이)
ull left = 0, right = max_len;  // max_len : 나무들 중 가장 긴 것의 높이
while (left <= right) {
    ull mid = (left + right) / 2;
    if (check(mid)) {  // 높이를 mid로 해 봤더니, m미터 이상 가져갈 수 있다
        ans = max(ans, mid);
        left = mid + 1;  // 절단기를 더 올려도 된다
    } else {  // 높이를 mid로 해 봤더니, m미터 이상 가져갈 수 없다
        right = mid - 1;  // 절단기를 내려야 한다
    }
}

cout << ans << '\n’;
```

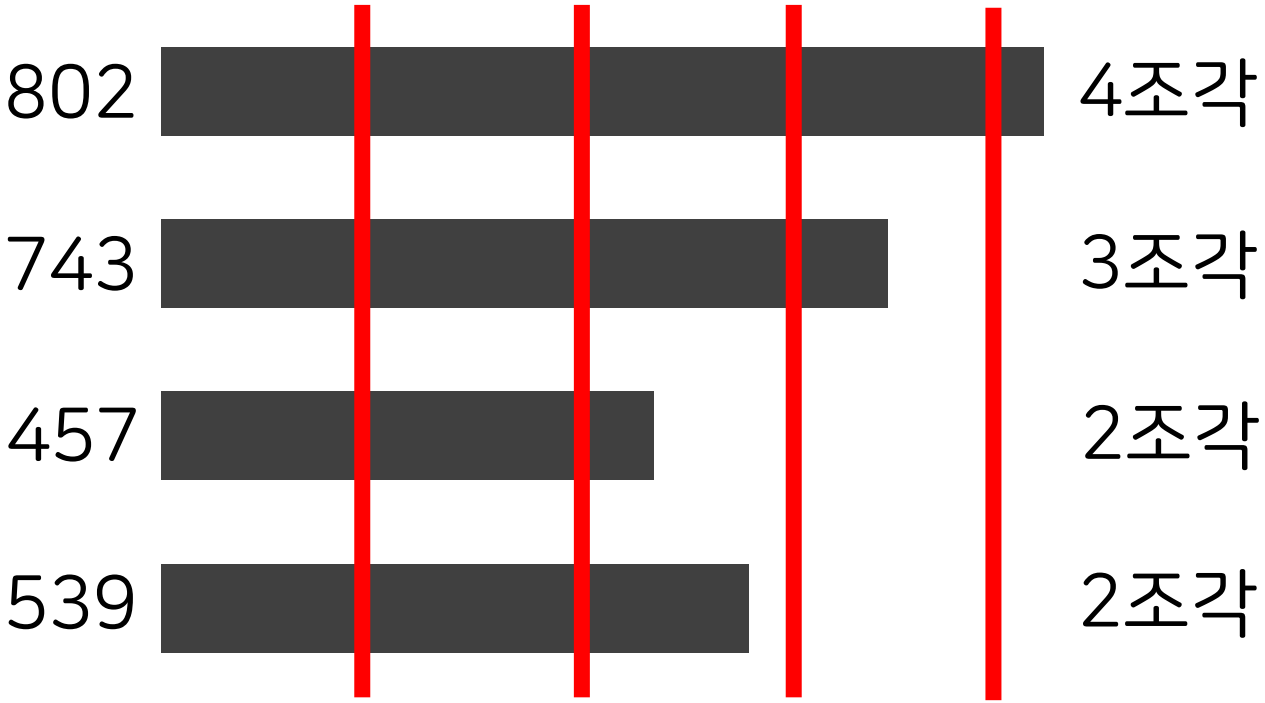
2805 – 나무 자르기

```
bool check(ull x) {  // 높이를 x로 했을 때 m미터 이상 가져갈 수 있는가
    ull total = 0;
    for (int i = 0; i < n; i++) {
        total += (len[i] >= x ? len[i] - x : 0);
    }
    return total >= m;
}
```

- http://boj.kr/22259ad08c1342f9bdfcd18880441746

1654 – 랜선 자르기

- 랜선 K개에서 동일한 길이의 랜선 N조각을 만들어야 한다.
- 만들 수 있는 최대 랜선 조각의 길이는?



K = 4, N = 11

11조각을 만들어야 한다면,
길이 200으로 자르면
총 11조각을 만들어낼 수 있고,
이 때가 최대 길이이다.

1654 – 랜선 자르기

- 최대 랜선 조각 길이는 한번에 못 구하겠지만
- 랜선 조각 길이를 X로 했을 때, 몇 조각이 나오는지는 알 수 있다

- 랜선 조각 길이를 X로 했더니, 조각이 필요한 것보다 더 나왔다면?
  - X를 더 늘려보자
- 랜선 조각 길이를 X로 했더니, 조각이 필요한 것보다 덜 나왔다면?
  - X를 더 줄여보자

- 랜선 조각 길이로 업 다운 게임

1654 – 랜선 자르기

- http://boj.kr/611ea7b9e50c4e0ba0c4c01284dba6dd

2110 – 공유기 설치

- 수직선 위 점 N개 중 C개에 공유기를 하나씩 설치하려고 한다
- 가장 인접한 두 공유기 사이의 거리를 가능한 크게 하여 설치하려고 한다.
- 그렇게 설치했을 때 가장 인접한 두 공유기 사이의 최대 거리를 출력한다.

- 가장 인접한 두 공유기 사이의 최대 거리를 바로 구하기는 어렵지만
- 가장 인접한 두 공유기 사이의 최대 거리를 X라 했을 때
- 그렇게 설치할 수 있는지는 쉽게 판별할 수 있다

- 역으로 X를 주고, 설치 가능/불가능 여부로 업다운 게임

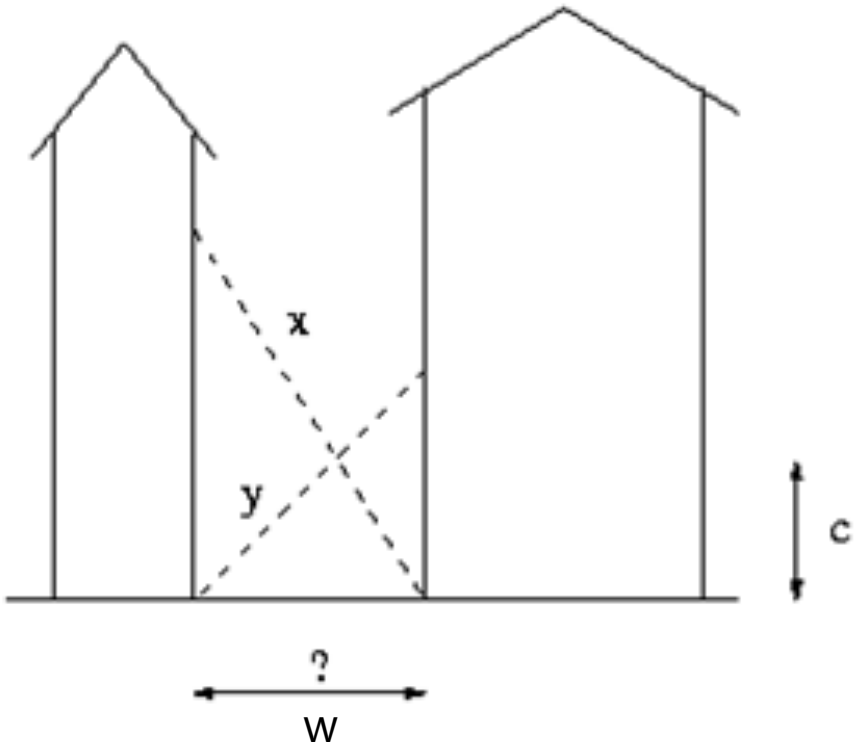
2110 - 공유기 설치

- 가장 인접한 두 공유기 사이의 최대 거리가 X가 되게 설치할 수 있다면?
  - X를 더 늘려보자
- 최대 거리가 X가 되게 설치할 수 없다면?
  - X를 더 줄여보자

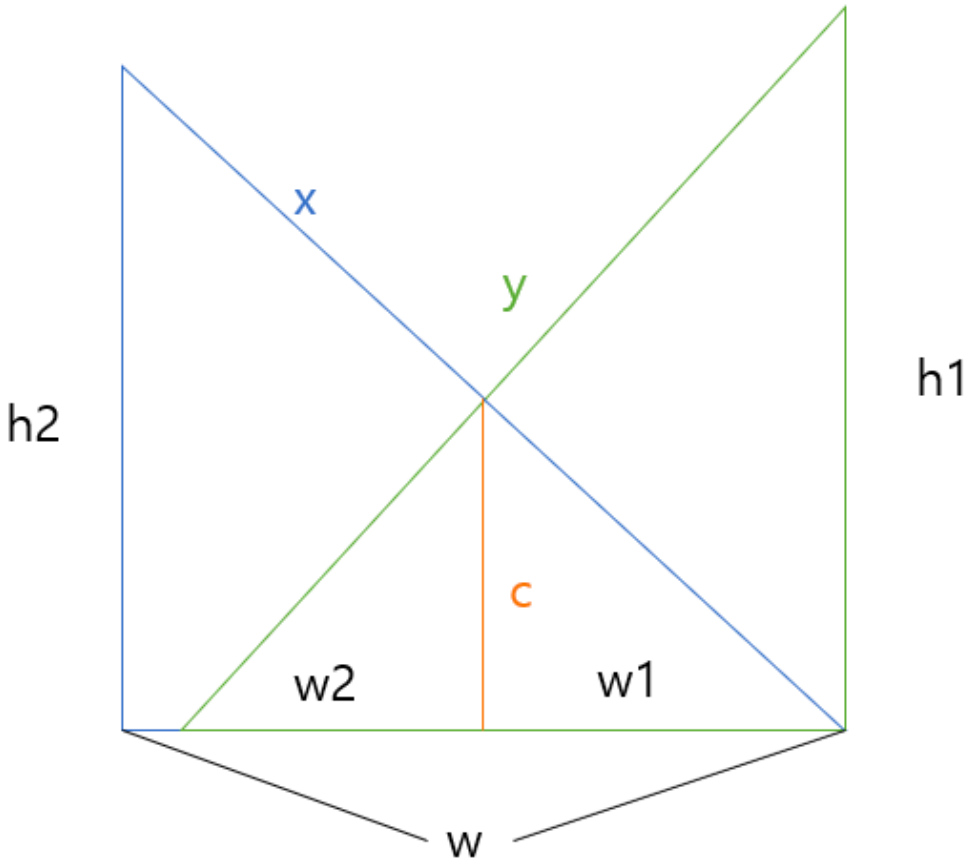
- http://boj.kr/1e80abbff38f417d9fdbfe3445cfb2e2

2022 – 사다리

- x, y, c가 주어질 때, w를 구해라.
- 절대/상대 오차는 10^(-3)까지 허용



2022 – 사다리



$$w = w_1 + w_2$$

$$h_1/w = c/w_2$$

$$h_2/w = c/w_1$$

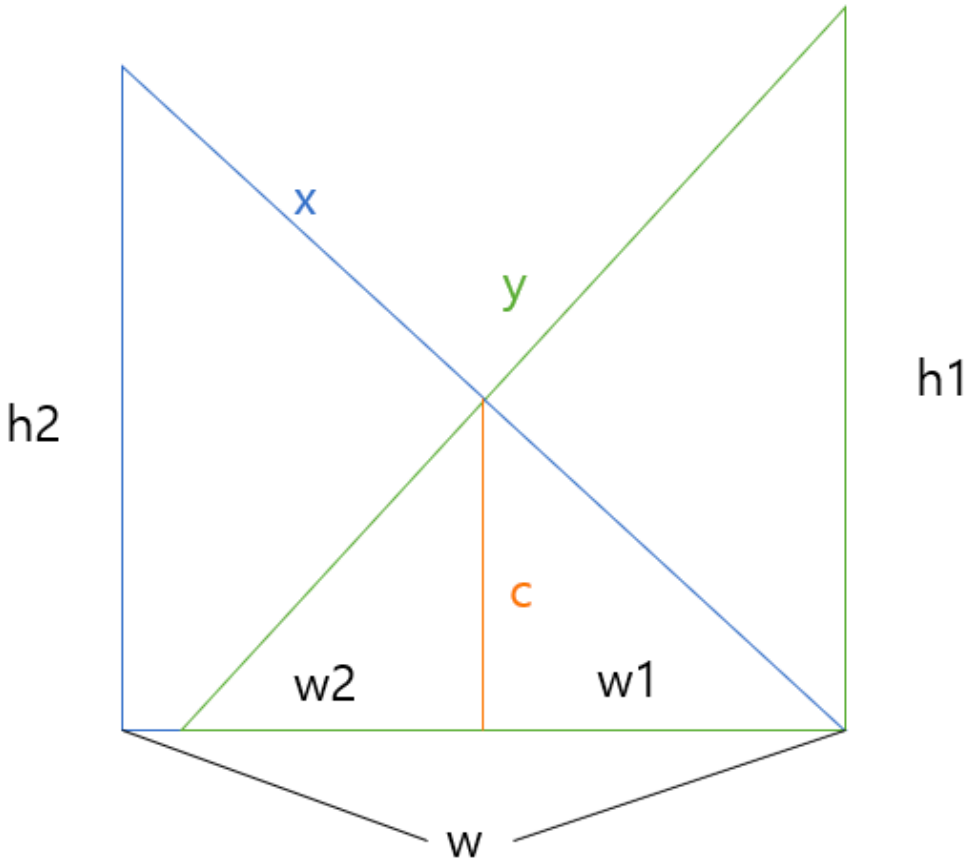
$$w = w * c/h_2 + w * c/h_1$$

$$1 = c/h_2 + c/h_1$$

$$1 = c(h_1 + h_2)/(h_1 * h_2)$$

$$c = (h_1 * h_2)/(h_1 + h_2)$$

2022 – 사다리



- w와 c는 반비례 관계이고
- w가 임의의 값으로 결정되면 c도 결정된다
- 따라서 w를 결정하는 데에
  c를 기준으로 삼을 수 있다

- 임의의 w에 대해 계산값이 c보다 크다면?
  - w가 너무 작다. w를 늘리자.
- 임의의 w에 대해 계산값이 c보다 작다면?
  - w가 너무 크다. w를 줄이자.

2022 – 사다리

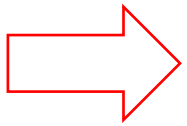
- 실수 범위에서의 이분 탐색 / Parametric search는 조금 다르다

```
while (left <= right) {
    if (…) left = mid + 1;
    else right = mid – 1;
}
```

- left, right, mid가 정수가 아니고 실수이기 때문에
  위와 같은 코드를 짜면 안 된다
- 문제가 되는 것은 종료 조건과 left, right 갱신
- 실수에서 left == right이려면? left = mid + 1의 문제는?

2022 – 사다리

```
while (left <= right) {
    if (…) left = mid + 1;
    else right = mid – 1;
}
```



```
while (abs(right – left) > 1e-6) {
    if (…) left = mid;
    else right = mid;
}
```

- 종료 조건을 상대 오차를 기준으로 바꾸어 줄 수 있고
- left, right 갱신도 바꾸면 된다

- (참고 : 1e-6 == 10^(-6) == 0.000001)

2022 – 사다리

- http://boj.kr/01d00ed768d54252a3b452f60f81b8df

참고

- http://kks227.blog.me/220776241154
- http://kks227.blog.me/220777333252
- https://jaimemin.tistory.com/1129

끝