정윤성

이진 탐색

Binary Search

By POSCAT

Search Algorithm

- 검색 알고리즘: 주어진 자료들 안에서 특정 조건을 만족하는 것을 찾고 싶다.
- 정렬된 배열 A에서 x 이상인 원소 중 최솟값의 index를 찾는 문제를 생각해 보자. (같은 값이라면 먼저 오는 것으로 찾자)
- 이를 lower_bound(array A, int x)라 하자.

Search Algorithm

- A = [1, 5, 7, 7, 8, 12, 12, 15]에 대해 (편의상 1-based index 사용)
- lower_bound(A, 10) = 6. (arr[6] = 12)
- lower_bound(A, 22) = NULL
- lower_bound(A, 5) = 2. (arr[2] = 5)

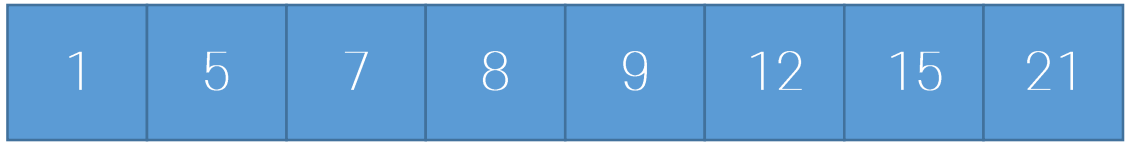
- 길이가 N인 배열에서 모든 원소를 다 순회하면 O(N)의 시간 필요
- 더 빠른 방법이 있을까?

Step Size

- Step Size를 2로 늘리면 (N/2+1)번만 비교하면 된다.
- Step Size를 K로 늘리면 O(N/K + K)가 필요하다.

- Step Size가 상수라면 O(sqrt N)일 때 최적이긴 하다.
- 하지만 남은 size에 따라 Step Size를 바꾸면 어떨까?

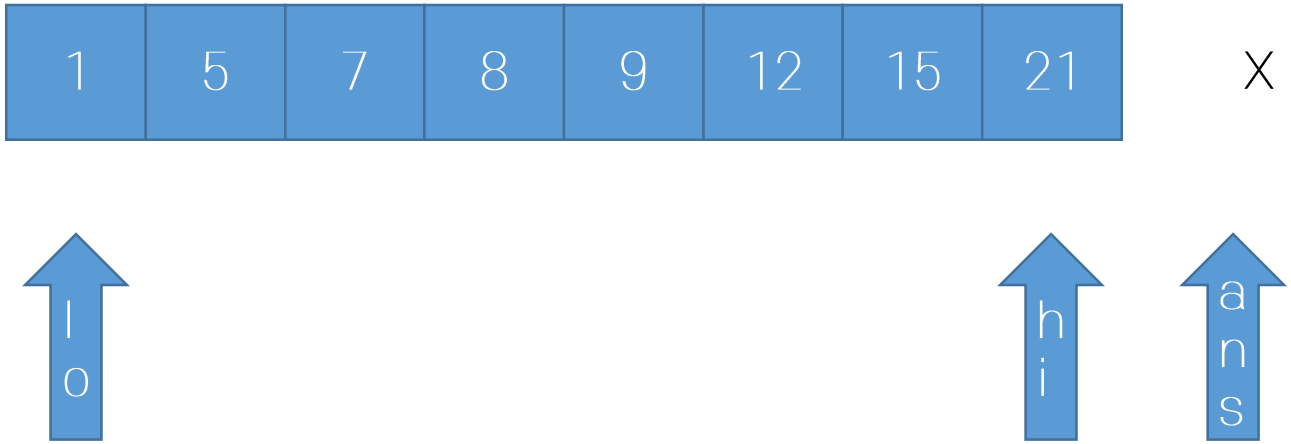
최초로 arr[i*K]>=X이면 [(i-1)*K+1 ~ i*K] 사이 답 존재



-> lower_bound를 찾
자

Binary Search

- Step Size를 현재 남은 원소의 절반으로 한다면?

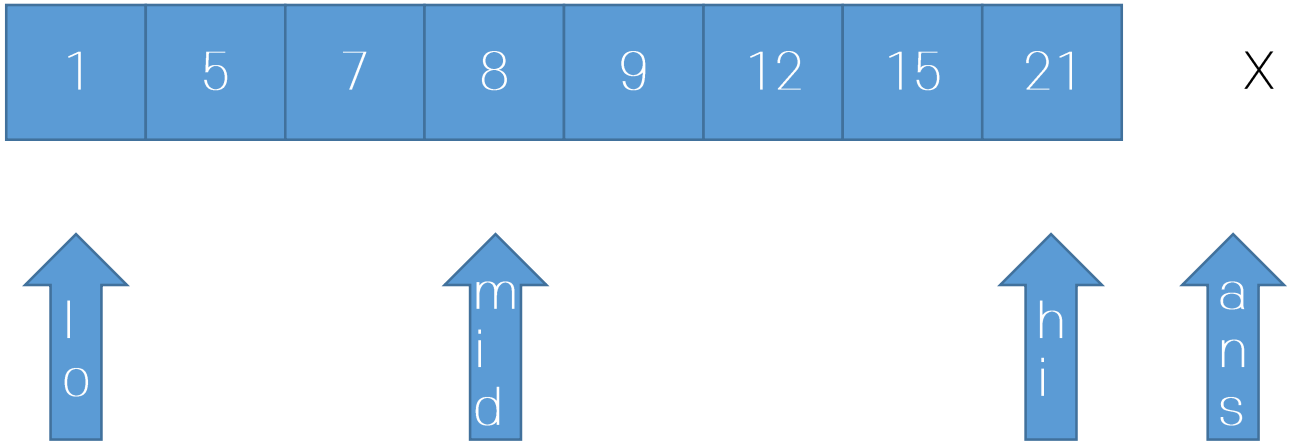


X 이상인 수 중 최솟값을 찾자 (X = 10)

lo: 가능한 최솟값, hi: 가능한 최댓값, ans: 최종 lower_bound(arr, X)

Binary Search

- Step Size를 현재 남은 원소의 절반으로 한다면?

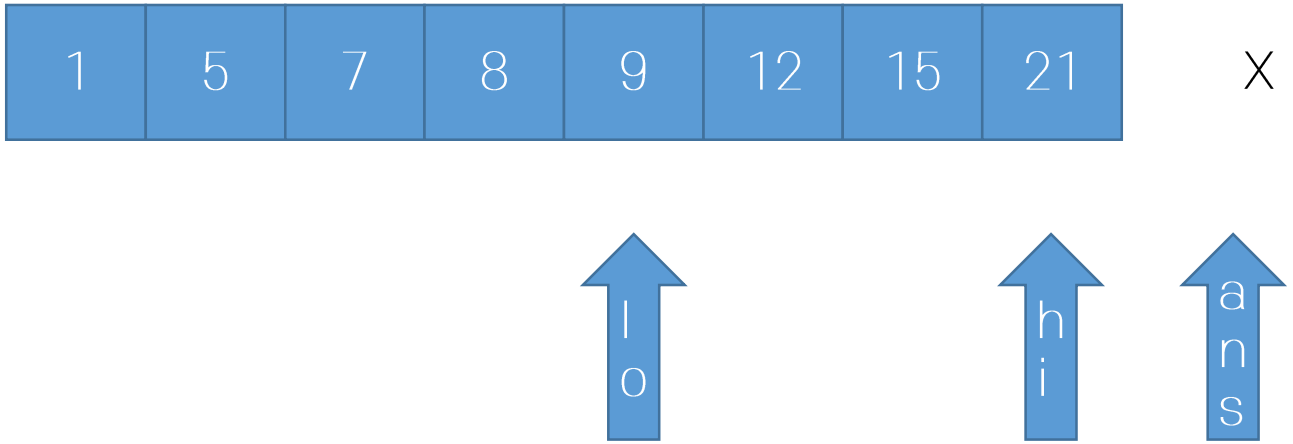


X 이상인 수 중 최솟값을 찾자 (X = 10)

arr[mid] < X 이므로 답은 [mid+1, hi]에 존재

Binary Search

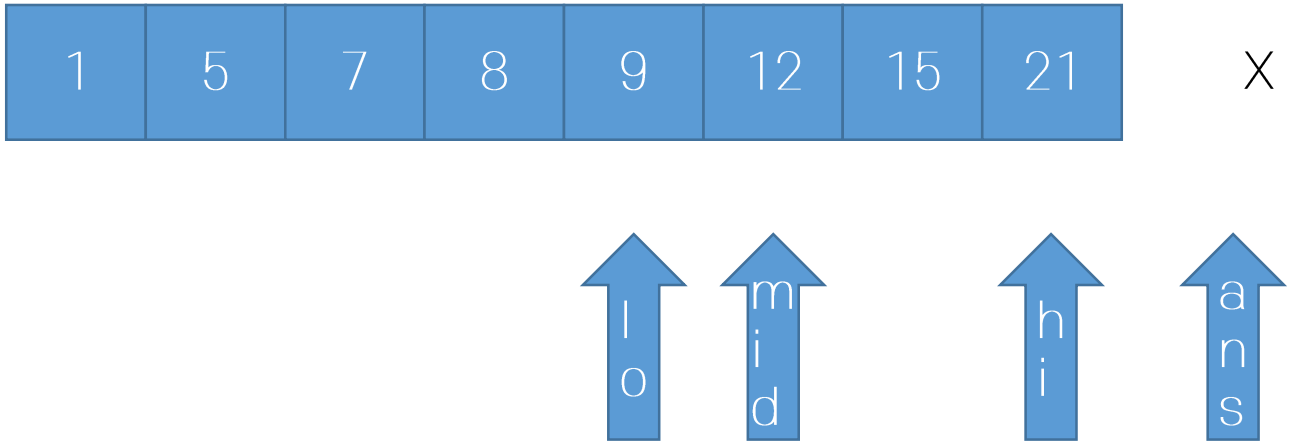
- Step Size를 현재 남은 원소의 절반으로 한다면?



X 이상인 수 중 최솟값을 찾자 (X = 10)

Binary Search

- Step Size를 현재 남은 원소의 절반으로 한다면?

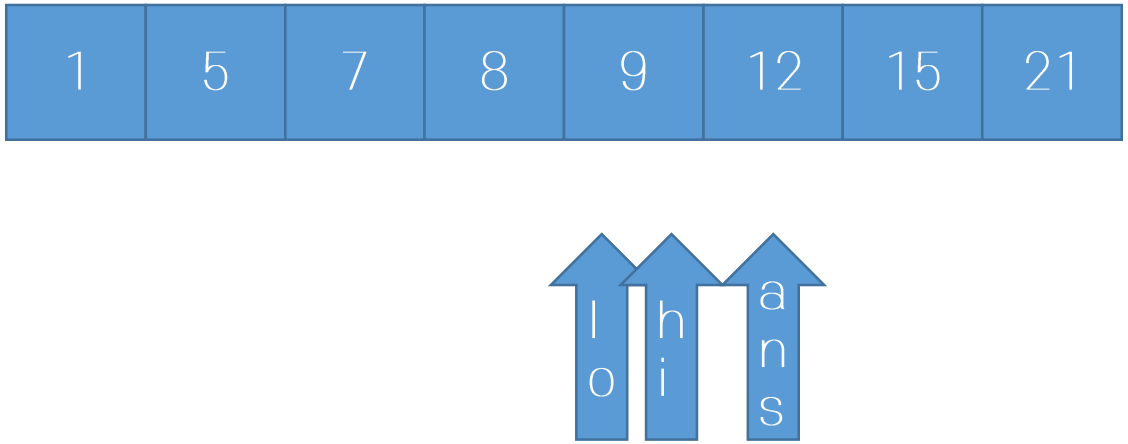


X 이상인 수 중 최솟값을 찾자 (X = 10)

arr[mid] >= X 이므로 답은 [lo, mid]에 존재, mid는 확실히 X 이상
일단 ans = mid로 두고, [lo, mid-1]에 X 이상인 수가 있다면 그것으로 갱신

Binary Search

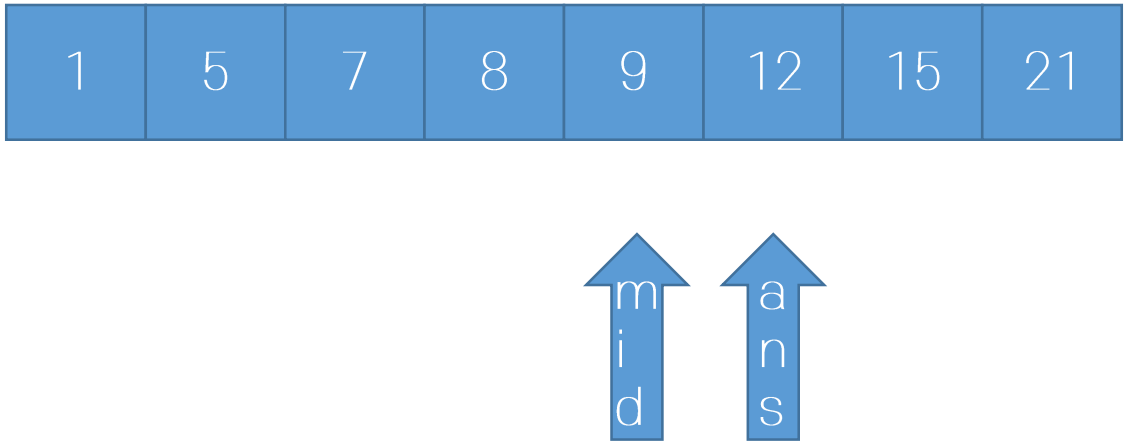
- Step Size를 현재 남은 원소의 절반으로 한다면?



X 이상인 수 중 최솟값을 찾자 (X = 10)

Binary Search

- Step Size를 현재 남은 원소의 절반으로 한다면?



X 이상인 수 중 최솟값을 찾자 (X = 10)

이제 lo = mid+1을 하면 lo>hi이므로 종료, ans를 출력

Binary Search

- 결과적으로 한 번 비교할 때마다 남는 개수가 절반으로 준다.
- $T(N) = T(N/2)+O(1)$
- $T(N) = O(\log N)$

Binary Search

Code Explanation

```
//길이가 N, 시작 인덱스가 1인 정렬된 배열. lower_bound(arr, x)가 존재하면 그 인덱스 값을, 없다면 “-1”을 print하자.

int lo = 1, hi = N, ans = hi + 1;

while((lo<=hi){

    int mid = (lo+hi)/2;

    if(arr[mid] >= x) hi = mid-1, ans = mid;

    else lo = mid+1;

}

printf(“%d\n”, ans > N ? -1 : ans);
```

Binary Search

Code Explanation

C++의 STL을 사용하면 한 줄만에 가능하다. std::lower_bound()

#include <algorithm>이 필요하다.

포인터를 반환하므로 초기 주소만큼 빼면 인덱스가 나온다.

int lb = lower_bound(arr+1, arr+N+1, X) - (arr); //arr[1]~arr[N] 중 값이 X **이상**인 arr 인덱스 최솟값.

int ub = upper_bound(arr+1, arr+N+1, X) - (arr); //arr[1]~arr[N] 중 값이 X **초과**인 arr 인덱스 최솟값.

이 바로 전 인덱스인 --lower_bound(~)을 하면 X **미만** 중 최댓값, --upper_bound(~)는 X **이하** 중 최댓값

답의 범위 정하고 좁히기

- 정렬된 배열의 예시를 잘 생각하면 lower_bound를 많은 곳에 응용할 수 있다.

- 아래와 같이 (N x M) 사각형의 (1, 1)에서 출발해 (N, M)까지 도달할 때(최단경로일 필요 없음), 경로 상에 있는 모든 C[i][j] 중 최댓값을 가능한 최소화 시키면 얼마일까?
- N, M <= 1,000, C[i][j] <= 1,000,000,000

2	11	4	17
7	5	10	6
12	3	9	8

(1, 1) -> (2, 1) -> (2, 2) -> (3, 2) -> (3, 3) -> (3, 4)로
가면 경로 상 C의 최댓값이 9가 된다.
(다른 어떠한 경로를 잡아도 9 미만으로 빠져나갈 수는 없다)

답의 범위 정하고 좁히기

- 어떤 X에 대해, 배열 arr을 아래와 같이 정의한다.
- X 이하의 타일만 밟고 (1, 1) ~ (N, M)으로 갈 수 있다면 arr[X] = 1, 아니면 arr[X] = 0.
- 이때 주목할 점은, arr[X]가 0이라면 당연히 모든 Y>X에 대해 arr[Y] = 0이다.
- 즉, 어떤 X를 기준으로 arr 값이 0 -> 1로 변하게 된다. (정렬된 상태)
- 답이 lower_bound(arr, 1)과 같다. (arr[X] >= 1이 되게 하는 인덱스 X의 최솟값)

2	11	4	17
7	5	10	6
12	3	9	8

답의 범위 정하고 좁히기

- 어떤 한 값 X에 대해, X 이하의 타일만 밟고 (1, 1) ~ (N, M)으로 갈 수 있는지는 O(NM)에 알 수 있다. (DFS or BFS)
- 이제 lo = 1, hi = 1e9로 두고 이진탐색을 하면 답을 찾을 수 있다.
- 배열 arr에서 실제로 계산해볼 값은 log(1e9) 개이고, 하나 계산하는데 O(NM)
- 총 O(NM * log(1e9))에 해결된다.

2	11	4	17
7	5	10	6
12	3	9	8

오늘의 문제

- https://www.acmicpc.net/problem/1920 lower_bound(arr, X) 구해서 X와 같은 값인지 비교
- https://www.acmicpc.net/problem/10816 lower_bound, upper_bound 적절히 빼기
- https://www.acmicpc.net/problem/2805 답의 범위 정하고 좁히기
- https://www.acmicpc.net/problem/1300 (Hard)