* 이진 탐색 접근(The Binary Search Approach)

- 이진 검색(binary Search)은 정렬된 리스트의 중간부터 비교해나가는 접근법이다. 즉 리스트를 반으로 나눠 검색한다.

- 이진검색에는 아래와 같은 세가지 경우의 수가 있다.

◆ 만약 키(key)가 리스트의 중간 요소보다 작을 경우, 리스트의 중간 이전까지만 탐색하면 된다.

◆ 만약 키(key)가 리스트의 중간 요소와 일치한다면(찾고자 하는 요소가 중간요소라면), 검색은 끝이 난다.

◆ 만약 키(key)가 리스트의 중간 요소보다 클 경우, 리스트의 중간 이후만 탐색하면 된다.

- 탐색중 리스트는 비교가 끝날 때 마다 검색 범위가 절반으로 줄어 든다.

- 첫번째 인덱스와 마지막 인덱스를 나타내는 low와 high 변수가 필요하다. low가 0, high가 n-1이다.

- 중간 인덱스를 나타내는 mid도 필요한데, mid는 (low + high) // 2이다.

- 알고리즘을 차근차근 살펴보자.

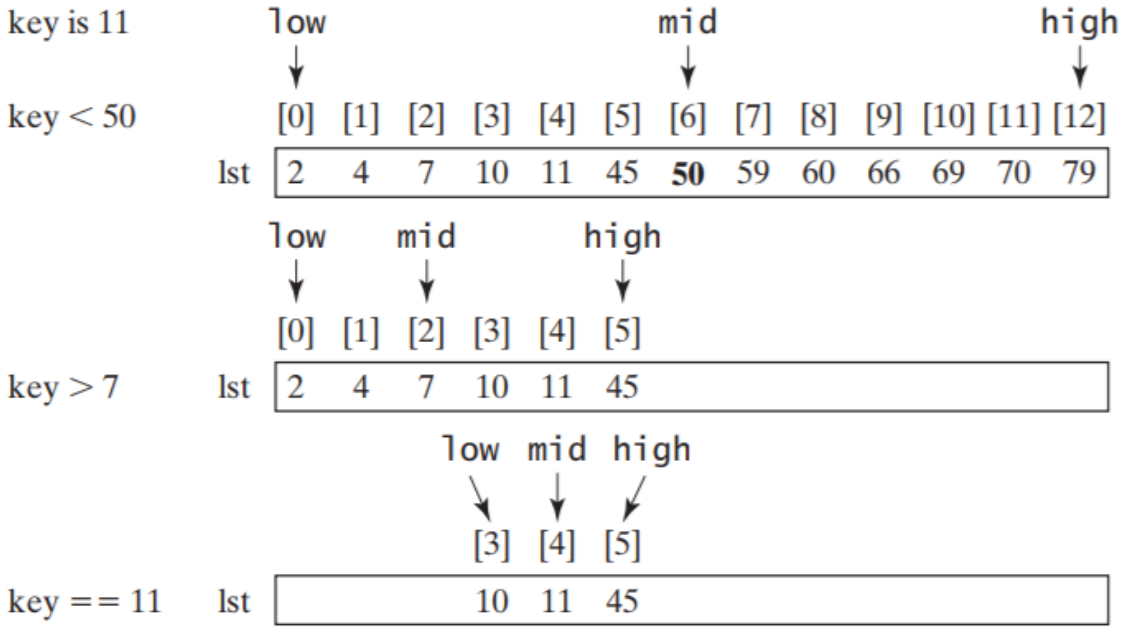
- 먼저 키(key)와 low 인덱스가 0이고 high 인덱스가 len(lst)-1인 리스트의 중간 요소를 비교한다

- 만약 key < lst[mid] 라면, high 인덱스 값을 mid - 1로 바꾼다.

- 만약 key == lst[mid] 라면, 검색하고자 하는 요소를 찾은것이므로 mid 변수를 반환(return)한다.

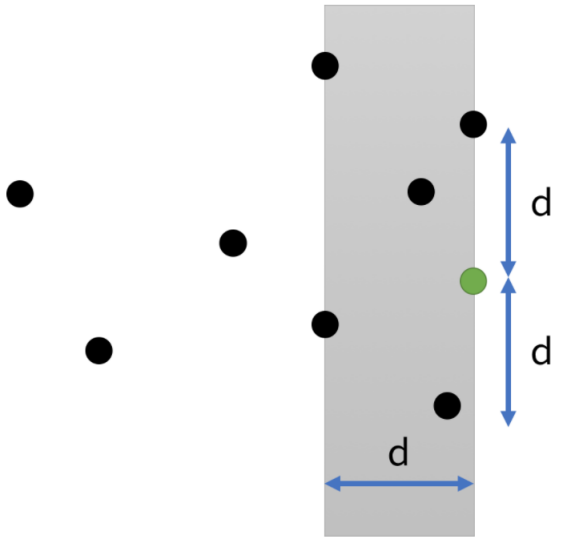
- 만약 key > lst[mid] 라면, low 인덱스를 mid + 1로 바꾼다.

- 이 과정을 반복해 key 값을 찾거나, 키값을 못찾았을 때(low > high) 탐색을 종료한다.



- 출처 : https://andamiro25.tistory.com/100

* Line Sweep 알고리즘



- 여러 점들 중 가장 가까운 두 점을 모두 비교할 경우 시간 복잡도는 O(N^2)
- 이러한 문제를 해결하기 위해 Line Sweep 알고리즘을 사용 이 알고리즘은 시간 복잡도 O(NlongN)까지 가능
- 한 점을 기준으로 point.x, point.y 좌표에서 최단 거리를 -+영역에 존재하는 후보들을 추출
- 위의 그림과 같이 후보들을 추출하는 과정 속에서 최단거리를 갱신
- 기본적인 아이디어는 x,y축을 나누어서 생각한 방식
- 즉 x,축을 먼저 고려한 후 y축을 고려하는 방식

순서

1. x 값이 증가하는 순으로 정렬
2. 두 점사이의 거리를 최단거리라고 가정
3. x축의 차이가 최단거리보다 크다면 비교할 필요가 없기에 후보자제거
4. y축을 기준으로 정렬된 후보자들을 통해 최단거리를 갱신

* 4번의 과정에서는 최단 거리를 갱신하기 위해 y-d ~ y+d 영역을 구해야 함

* 출처

- https://mygumi.tistory.com/294
- https://www.acmicpc.net/blog/view/25
- https://en.wikipedia.org/wiki/Sweep_line_algorithm