**이진 탐색 (Binary search) 개념 및 구현**

이진 탐색은 정렬된 리스트에서 검색 범위를 줄여 나가면서 검색 값을 찾는 알고리즘입니다.

이진 탐색은 정렬된 리스트에만 사용할 수 있다는 단점이 있지만, 검색이 반복될 때마다 검색 범위가 절반으로 줄기 때문에 속도가 빠르다는 장점이 있습니다.

## **동작 방식**

이진 탐색 알고리즘은 리스트의 중간 값과 비교하여 검색값을 찾습니다. 중간 값을 찾아야 하기 때문에  반드시 정렬된 배열에서만 사용할 수 있습니다.

이진 탐색의 동작 방식

1. 배열의 중간 값을 가져옵니다.
2. 중간 값과 검색 값을 비교합니다.

2-1 중간 값이 검색 값과 같다면 종료.

2-2 중간 값보다 검색 값이 크다면 중간값 기준 배열의 오른쪽 구간을 대상으로 탐색합니다. (mid<key)

2-3 중간 값보다 검색 값이 작다면 중간값 기준 배열의 왼쪽 구간을 대상으로 탐색합니다. (mid>key)

1. 값을 찾거나 간격이 비어 있을 때 까지 반복한다.

## **문제**

n개의 서로 다른 양의 정수 a1, a2, ..., an으로 이루어진 수열이 있다. ai의 값은 1보다 크거나 같고, 1000000보다 작거나 같은 자연수이다. 자연수 x가 주어졌을 때, ai + aj = x (1 ≤ i < j ≤ n)을 만족하는 (ai, aj)쌍의 수를 구하는 프로그램을 작성하시오.

## **입력**

첫째 줄에 수열의 크기 n이 주어진다. 다음 줄에는 수열에 포함되는 수가 주어진다. 셋째 줄에는 x가 주어진다. (1 ≤ n ≤ 100000, 1 ≤ x ≤ 2000000)

## **출력**

문제의 조건을 만족하는 쌍의 개수를 출력한다.

## **해결 과정**

배열에서 두 요소의 합이 x가 되는 경우의 수를 구하는 문제이다.

O(N^2)인 완전 탐색방법으로 풀 수는 있겠지만 배열의 길이가 10만까지 들어오므로 O(N^2)로 처리하면 1초내로 연산이 다 불가능하고 시간 초과가 발생한다.

해결 방법은 크게 2가지가 있다.

**1. 투 포인터**

배열을 정렬한 뒤, left, right 포인터를 배열의 시작지점, 끝부분에 위치시킨다. 그 후 배열의 left번째 index의 원소 + right번째 index의 원소를 x와 비교한 후, x보다 작으면 left를 전진, 0보다 크면 right를 후진, x와 같으면 left, right를 전진, 후진 시킨 뒤 정답을 1 늘려주면 된다.

**2. 배열로 자연수 존재 여부 판단**

(x - 배열의 원소) 라는 수가 나온 적이 있는지를 체크하는 배열을 두고, 등장한 적이 있으면 합해서 x가 되므로 답의 개수를 하나씩 늘리면 된다.

**3. 소스코드**

import sys

n = int(sys.stdin.readline())

number = list(map(int, sys.stdin.readline().split()))

x = int(sys.stdin.readline())

number.sort()

start, end = 0, n-1

cnt, tmp = 0, 0

while start < end:

tmp = number[start] + number[end]

if tmp < x:

start += 1

elif tmp > x:

end -= 1

else:

cnt += 1

start += 1

end -= 1

print(cnt)

투 포인터 문제인데 start 포인터와 end 포인터의 초기값이 왜 리스트의 시작과 끝을 가리키는지 이해해야 한다.

문제에서 ai + aj = x이고 범위는 (1 <= i < j <= n)이기 때문에 정렬된 리스트에서 끝과 끝에서 하나 하나 비교하며 모든 쌍을 만족해야하고 start포인터가 end포인터보다 크면 안된다.

두 지점의 합을 구하는 문제이기 때문에 부분수열의 누적합이 아니어서 끝과 끝에서 서로를 향해 가까워지며 반복문을 수행해야한다.