Two Pointer

sianke1991

#2230: 수 고르기

N개의 정수로 이루어진 수열 A[1], A[2], …, A[N]이 있다. 이 수열에서 두 수를 골랐을 때(같은 수일 수도 있다), 그 차이가 M 이상이면서 제일 작은 경우를 구하는 프로그램을 작성하시오.

예를 들어 수열이 {1, 2, 3, 4, 5}라고 하자. 만약 M = 3일 경우, 1 4, 1 5, 2 5를 골랐을 때 그 차이가 M 이상이 된다. 이 중에서 차이가 가장 작은 경우는 1 4나 2 5를 골랐을 때의 3이 된다.

입력: 첫째 줄에 두 정수 N, M이 주어진다. 다음 N개의 줄에는 차례로 A[1], A[2], …, A[N]이 주어진다.

출력: 첫째 줄에 M 이상이면서 가장 작은 차이를 출력한다. 항상 차이가 M이상인 두 수를 고를 수 있다.

제한

$1 \le N \le 100,000$

$0 \le M \le 2,000,000,000$

$0 \le |A[i]| \le 1,000,000,000$

M = 6, arr = [2, 3, 9, 13, 22]일 때 먼저 생각할 수 있는 방법은 2중 for loop 이다. 배열 내 두 원소를 선택할 수 있는 모든 방법에 대해 조사하여 최적을 선택한다.

```
for (int i=0; i<arr.length; i++) {
    for (int j=i; j<arr.length; j++) { //j<i에 대해서는 고려할 필요가 없다.
        if (Math.abs(arr[j]-arr[i])>=M) {
            ans = Math.min(ans, Math.abs(arr[j]-arr[i]));
        }
    } //j loop
} //i loop
```

이 방법은 배열의 길이 N에 대해 $O(N^2)$의 시간복잡도를 갖는다. 이 보다 더 빠른 방법을 생각해 보자.
우선 배열 arr이 정렬이 되어 있다면, (혹은 정렬을 시킨다면) 안쪽 for loop에 break를 걸 수 있다.

```
for (int i=0; i<arr.length; i++) {
    for (int j=i; j<arr.length; j++) { //j<i에 대해서는 고려할 필요가 없다.
        if (Math.abs(arr[j]-arr[i])>=M) {
            ans = Math.min(ans, Math.abs(arr[j]-arr[i]));
                  //배열의 뒤로 갈 수록 원소 값이 커지므로 한 번 Math.abs(arr[j]-arr[i])>=M
                  //조건을 만족하는 케이스를 발견했다면 그 뒤에서는 ans 값이 갱신되지 않는다.
        }
    } //j loop
} //i loop
```

또한 위 코드를 정리하여 다음과 같이 쓸 수도 있다. (안쪽 for loop에 사용되는 j는 매 번 i 로 초기화할 필요가 없다. 새로운 i에 대해 j가 기존에 갖고 있던 값에서 시작하여 오른쪽으로 밀려도 충분히 문제를 해결할 수 있다.)

```
int ed = 0;
for (int st = 0; st<arr.length; st++) {
    while (ed<arr.length-1 && arr[ed]-arr[st]<M)
        ed++;
    if (arr[ed]-arr[st]>=M)
        ans = Math.min(ans, arr[ed]-arr[st]);
} //st loop
```

이 방법을 사용하면 두 포인터 st와 ed가 0에서 arr.length까지 한 번 훑쓸면서 (필요한) 모든 경우를 탐색하게 된다.

두 포인터 st와 ed가 0에서 arr.length까지 한 번 휩쓸기 때문에 st loop의 시간복잡도는 O(N)이라고 할 수 있다. 만약 정렬을 수행하는 경우 전체 시간복잡도는 O(NlogN)이 되며 이는 기존 2중 for loop 보다 더 나은 시간복잡도라고 볼 수 있다.