BOJ 14504
수열과 쿼리 18

#sqrt_decomposition

난이도 – Diamond V

sorohue@sasa.hs.kr

BOJ 14504 수열과 쿼리 18

- 배열 위의 어떤 구간에서 k보다 큰 수의 개수를 찾아야 합니다.
- 배열의 각 수는 변화할 수 있습니다.

BOJ 14504 수열과 쿼리 18

- 수를 바꾸는 쿼리가 없는 경우에는 이분 탐색을 통해 O((Q+N) lg N)에 간단히 풀 수 있는 문제입니다.

- 하지만 수가 바뀌는 경우에는 배열을 정렬된 상태로 유지시키려면 매 쿼리마다 배열을 정렬해 주어야 합니다.
- 이 경우 시간 복잡도가 O(QN lg N) 까지 늘어나게 됩니다.

BOJ 14504 수열과 쿼리 18

- 시간 복잡도가 늘어나는 가장 큰 원인은
  정렬해야 하는 구간의 크기가 너무 크기 때문입니다.

- 현재 방식으로는 1번 쿼리가 2번 쿼리에 비해 상당히 빠르게 동작합니다.
- 1번 쿼리의 시간 복잡도를 일부 희생해서 2번 쿼리의 실행 속도를
  대폭 감축해 봅시다.

BOJ 14504 수열과 쿼리 18

- 배열을 적당한 크기로 분할해 줍니다. 보통 각 구간의 길이가 총 구간의 개수와 비슷해지도록 맞춥니다.

- 이는 제곱근 분할법이라 부르는 테크닉입니다.

- 분할한 각각의 구간 안에서만 정렬을 해 줍시다.

BOJ 14504 수열과 쿼리 18

- 1번 쿼리가 들어오면, 다음과 같이 쿼리를 처리해 줍니다.
  - 범위에 구간이 완전히 포함되는 경우 그 구간에서 이분 탐색을 해 줍니다.
  - 구간 중 일부만이 포함되는 경우 해당 부분에서 순차 탐색을 해 줍니다.

- 이 경우 해당 쿼리를 처리하는데 $O(N^{1/2} \lg N)$ 정도 걸립니다.
- 원래 방식대로는 $O(\lg N)$ 이 걸립니다.

BOJ 14504 수열과 쿼리 18

- 2번 쿼리가 들어오면, 쿼리로 들어온 인덱스의 수를 바꿔준 후
  해당 수가 포함된 구간만 새로 정렬해 주면 됩니다.

- 이 경우 해당 쿼리를 처리하는 데 역시 $O(N^{1/2} \lg N)$ 정도 걸립니다.
- 원래 방식대로는 $O(N \lg N)$이 걸립니다.

BOJ 14504 수열과 쿼리 18

- 최종적으로 시간 복잡도는 $O(QN \lg N)$ 에서 $O(QN^{1/2} \lg N)$으로 줄어들게 됩니다!