컴퓨터공학과 20211507 김동건

문제1) Largest N n umbers

우선, N^2개의 원소를 모두 배열에 넣고, 정렬하여 N개의 원소를 뽑는 풀이를 생각해볼 수 있다. 그럼 N^2 log(N^2)이기 때문에 N^2logN에 해결할 수 있다. 그러나, 다음과 같이 각 행에서 O(N)에 가장 큰 수를 찾아서 각 행의 맨 끝 수와 바꾸어 놓자(행위 1). 여기까지 각 행별로 O(N)이 들기 때문에 O(N^2)이다. 이제 “맨 끝 수들 중 가장 큰 값을 출력하고, 그 행에 대해서 남은 수들 중 가장 큰 수를 그 행의 끝에 둔다.”를 행위 2라고 하자. 행위 2를 N번 반복하면 가장 큰 N개의 수를 출력할 수 있다. 맨 처음 행위 1에 O(N^2)의 시간이 들고, 행위 2도 맨 끝 수들 중 최댓값을 찾는 데에 O(N), 제일 큰 수가 있는 행에 대해서 새로운 제일 큰 값을 찾는 데에 O(N)이 들기 때문에 행위 2 한 번에 O(N)의 시간이 든다. 이를 N번 하기 때문에 O(N^2)이 들고, 행위 1 1번과 행위 2 N번을 합쳐도 O(N^2)에 문제를 해결 할 수 있다. 자세한 방법은 아래 그림과 같다. 각 행 맨 끝에 있는 빨간 바의 역할은, 그 행에서는 그 바가 있는 곳까지만 보면 된다는 것이다. 빨간 동그라미는 그 수를 뽑았다는 뜻이다. 텍스트, 친필, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

문제2) Position of k

우선, N^2 개의 원소를 모두 확인하면 답을 구할 수 있다. 그러나, 각 행과 열이 nondecreasing하다는 것을 이용하면 O(NlogN)에 문제를 해결할 수 있다. 각 행마다 이분탐색으로 k 이상의 원소를 찾는 것이다. 이렇게 하면 최악의 경우에 N개의 행에 대해 이분탐색을 logN번 진행하므로 O(NlogN)에 문제를 해결할 수 있다. 그러나, 이 문제는 사실 (1, 1)에서 시작하여 k보다 크지 않을 때까지 오른쪽으로 가고, k보다 큰 값을 만나면 아래로 내려가는 것(단, 내려갔을 때 k보다 큰 곳이라면 k보다 작아지는 순간까지 왼쪽으로 가준다.)을 k를 만날 때까지 반복하면 O(N)에 문제를 해결할 수 있다. 예를 들어 다음과 같은 행렬에서 k가 17이라면 노란색 경로를 통해 17을 찾을 수 있다. 텍스트, 폰트, 번호, 친필이(가) 표시된 사진

자동 생성된 설명이제 위 알고리즘이 O(N)이라는 것을 증명해 보겠다. 우선, 위 풀이에서는 1) 우측으로 이동, 2) 좌측으로 이동, 3) 아래로 이동의 세 가지 이동만 있다. (위 풀이에서 위로 갈 일은 없기 때문) 이 세 이동 횟수의 합이 O(N)에 비례하는 것을 보이면 된다. 우선 위 알고리즘에서 다시 위로 가는 경우가 없으므로 아래로 이동하는 것은 N-1번이다. 이제 중요한 것은 우측, 좌측으로 이동하는 것이다. 우리는 위의 예시에서 18을 만났을 때 k보다 작은 수가 나올 때 까지 좌측으로 이동한다. 그렇게 되면, 18이 있는 4열을 포함하여 4열의 오른쪽으로는 절대 갈 일이 없다. 왜냐하면 18의 우하단은 모두 18보다 크거나 같은 수가 있기 때문이다. 즉, 만약 오른쪽에서 **왼쪽으로 진행**하면서 지난 열들은 내려가면서 다시 갈 일이 없다. 따라서 최악의 경우에 오른쪽으로 쭉 가고, 왼쪽으로 다시 쭉 오는 경우에도 우측과 좌측 이동의 합이 2N을 넘지 않는다. 즉, 3N보다 적은 횟수의 이동으로 k의 위치를 알아낼 수 있다. 참고로 위 방법을 진행하면서 행렬을 벗어나게 되면, 행렬에는 k가 없다.