컴퓨터공학과 20211507 김동건

문제1) Matrix Addition

문제를 읽어보면 R1과 R2 그리고 C1과 C2 사이의 대소관계가 주어지지 않았지만, R1 <= R2, C1 <= C2라고 가정하고 문제를 해결해도 된다. (만약 반대로 주어졌다면, swap하면 같기 때문이다.)

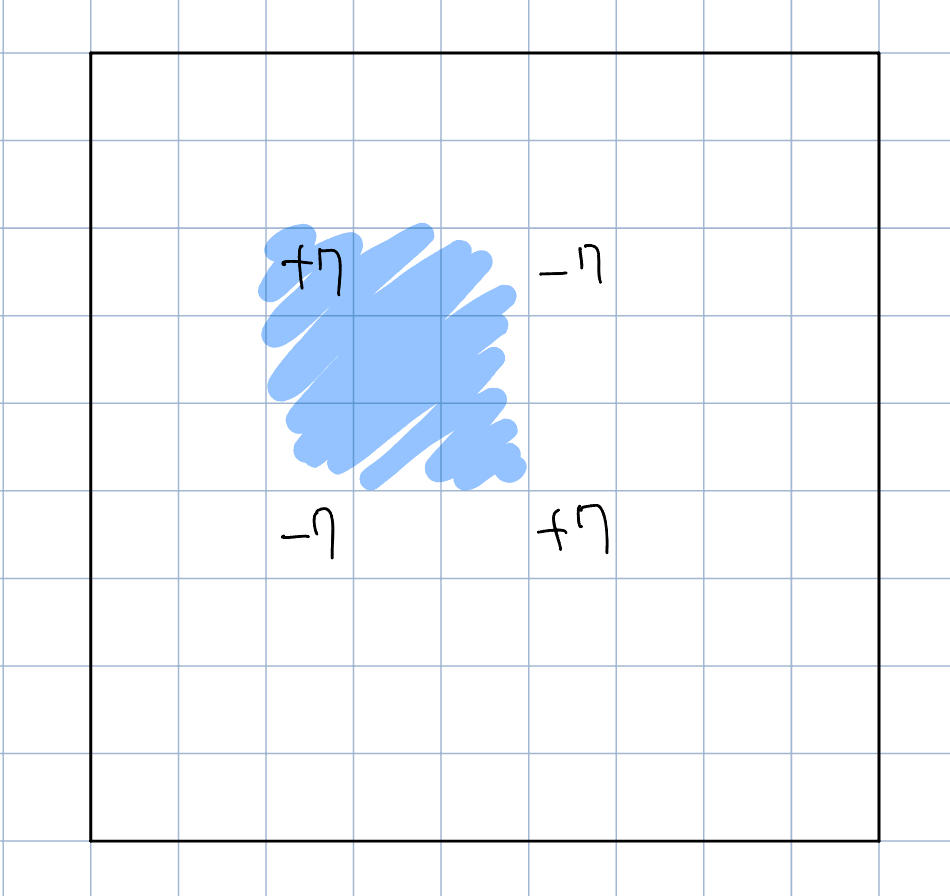
결국 문제를 해석하면 행렬의 어떠한 직사각형 영역을 골라 그 영역에 모두 v를 더해주는 행위를 N번 하는 것이다. 주어진 코드는 O(N^3)이다.

우선, 최악의 경우 무조건 N\*N개의 값을 바꿔야 하기 때문에 N\*N보다 빠른 시간에 문제를 해결할 수 없음은 생각할 수 있다. 따라서 O(N^2) 혹은 그것보다 느리면서 O(N^3)보다 빠른 해결 방법을 생각해 보아야 한다.

가장 먼저 떠오른 방법은 lazy segment tree를 활용한 N^2logN 풀이이다. 각 행별로 lazy segment tree를 만들면 read 한 번에 모든 행을 Update해야 하므로, NlogN이 걸리고(N개의 행 \* update 연산) read를 N번 하기 때문에 N^2logN이 걸린다. 최종 결과를 확인하는 것도, 각 칸마다 logN이 걸리기 때문에 N^2logN에 해결 가능하다.

그러나, 우리는 read를 한 번 할 때마다 그 값을 물어보는 쿼리같은 것이 들어오지 않는다는 점을 이용하면 더 빠르게 풀 수 있을 것이라 생각할 수 있다. 즉, 온라인으로 그때그때 처리하는 위의 풀이 대신, 결국 마지막에 한 번 최종 결과만 알면 된다는 점을 이용하는 오프라인 풀이를 고안해볼 수 있다는 것이다. 결론부터 말하면, O(N^2)에 문제를 해결할 수 있다.

이는 흔히 imos 법으로 알려져 있는 기법을 통해 해결할 수 있다. 이는 더해야 하는 구간의 끝 부분들에 표시를 해둔 뒤, 마지막에 차원의 수 만큼 sweeping하여 값을 구하는 기법이다. 표시는 4개를 하면 된다. 예를 들어, 파란색 영역에 +7을 해야 한다면, 아래 그림과 같이 4곳에 표시를 다음과 같이 해주자.



다음으로, 빨간색 영역에 +4가 들어온다고 하면 추가로 아래와 같이 마킹을 해줄 수 있다.

텍스트, 도표, 그래프, 스크린샷이(가) 표시된 사진

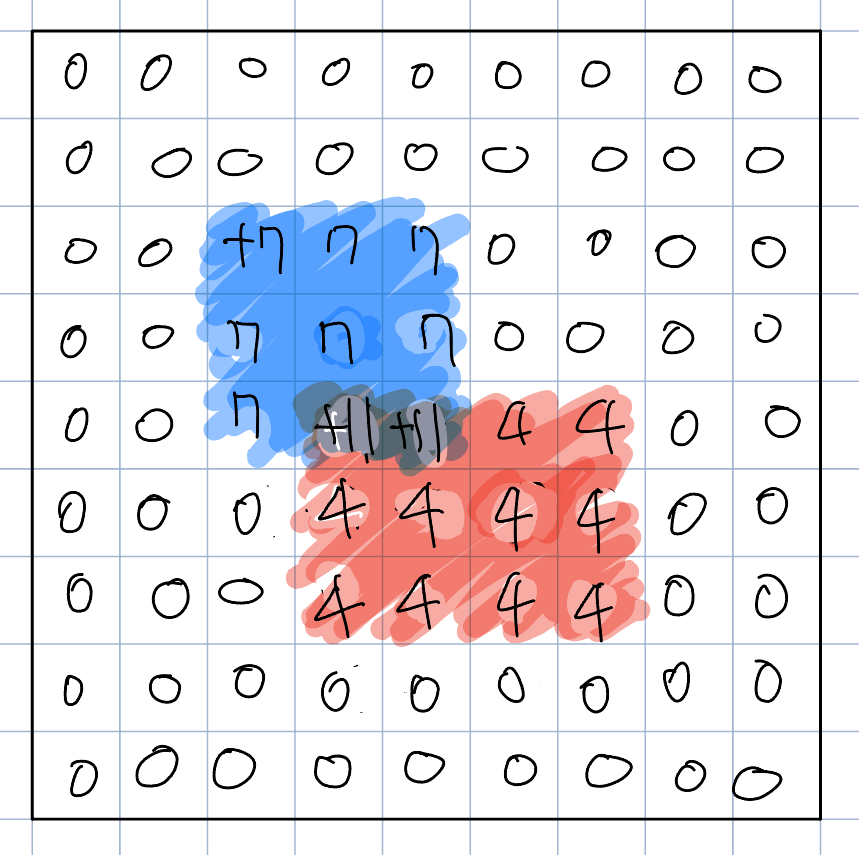
자동 생성된 설명

이제, 왼쪽에서 오른쪽으로 값을 누적해서 더해가며 sweeping 한 뒤, 위에서 아래로 sweeping해 보자.

먼저, 왼쪽에서 오른쪽으로 값을 누적해서 더해가며 sweeping하면 다음과 같다.

텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명 이제 위에서 아래로 sweeping하여 보자.

잘 반영된 것을 볼 수 있다. 우리가 처음에 표시를 한 것은 값이 더해지기 시작하는 곳과 끝나는 곳을 기록한 것이다. 이런식으로 표시를 해두고, 차원만큼 sweeping을 해주면, 원하는 값을 구간에 더할 수 있다.

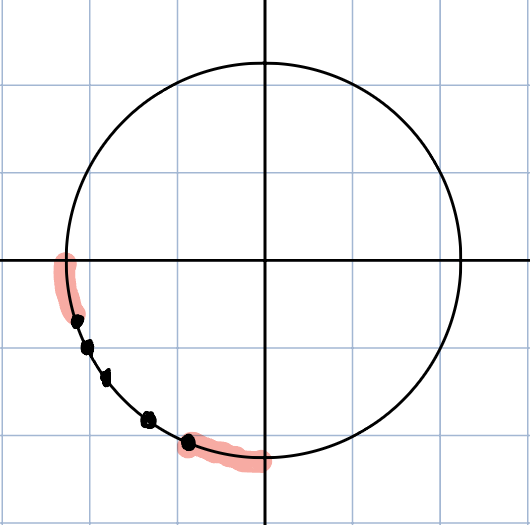
표시하는 데에 O(N), sweeping하는 데에 O(N^2)이 걸리기 때문에 총 시간복잡도 O(N^2)에 문제를 해결할 수 있다.

문제2) Half-Circle Property

우선, 가장 먼저 떠오른 쉬운 풀이는 점을 양의 x축의 방향과의 각도를 기준으로 정렬한 뒤, 두 영역으로 나눈 다음, sweeping하면서 푸는 풀이가 떠올랐다. 이는 주어진 점을 정렬하는 것에 O(NlogN), sweeping에 O(N)이 걸려 총 시간복잡도는 O(NlogN)이다. 그 후, O(N)에 풀 수 있는지 고민해 보았다.

아래 풀이 설명에서는 조건을 만족하는 직선이 1, 3 사분면을 지나는 경우만 설명하겠다. 이 경우에서 문제를 해결할 수 있으면, 주어진 상황을 4번 회전하여 각각 답을 구하면 전체 문제를 해결할 수 있기 때문이다.

결론부터 말을 하면, 우리는 각 사분면마다 x값이 가장 큰 점, x값이 가장 작은 점 두 개만 관리해도 된다는 것이다. 아래와 같이 3사분면에 점이 여러개 있다면, 결국 직선이 지나는 곳으로 가능한 영역은 빨간색 영역 뿐이다. 그 점 사이를 지나는 직선은 양쪽에 점들이 생기기 때문이다. 따라서 우리는 점이 2개 이상이라면 점을 2개로 축소하여 문제를 해결할 수 있다.



마찬가지로 두 빨간 영역 중 더 아래에 있는 빨간 영역만 생각해보자. (위쪽도 같은 방식으로 확인하면 되기 때문에 생략한다.) 3사분면의 영역은 아래 빨간영역으로 좁혔다. 다음은 마주보는 사분면인 1사분면을 살펴보자. 빨간 영역에서 x좌표가 가장 작은 점을 기준으로 원점과의 직선을 그려보자. 만약 이 경우에서 조건을 만족하려면 1사분면의 점은 전부 파란색 구간 위에 있어야 한다.

라인, 도표, 원, 그래프이(가) 표시된 사진

자동 생성된 설명

각 사분면별로 x가 최대인 것과 x가 최소인 것만 가지고 있어도 이를 확인하는 데에 전혀 문제가 되지 않는다.

만약, 여기까지 조건을 모두 통과하였다면 마지막으로 2, 4 사분면에 있는 점들을 해결해야 한다. 2사분면은 점이 있던 없던 조건을 만족하니 신경쓰지 않아도 좋고, 4사분면에는 점이 존재하면 불가능, 존재하지 않으면 가능하고, 문제를 해결하였다.

만약 3사분면의 점이 1개인 경우에는 그 점을 최대이자 최소로 생각하고 양쪽을 체크해주면 되니 걱정하지 않아도 좋다.

마지막으로 3사분면의 점이 0개라면 어떨까. 반대쪽 사분면(1사분면)에 점이 하나라도 있다면 1사분면을 기준으로 문제를 해결하는 과정에서 해결되니 3사분면은 따로 체크하지 않아도 된다. 만약, 1사분면에도 점이 없으면, 1,3 사분면을 지나는 경우가 답이 되려면 2 혹은 4 사분면에 아무점도 없어야 한다. 따라서 3사분면의 점이 0개일 때도 해결할 수 있다.

정리하면, 우리는 3사분면에 대해 점이 2개이상, 1개, 0개일 때 각각 문제를 해결할 수 있고, 이를 모든 사분면에 대해 진행하면 전체 문제를 해결할 수 있다.

각 사분면별로 최대, 최소 점을 기록하는 데에 O(N), 각 사분면마다 문제를 해결하는 데에는 상수시간이 걸리므로, 총 O(N)에 문제를 해결하였다.