1）

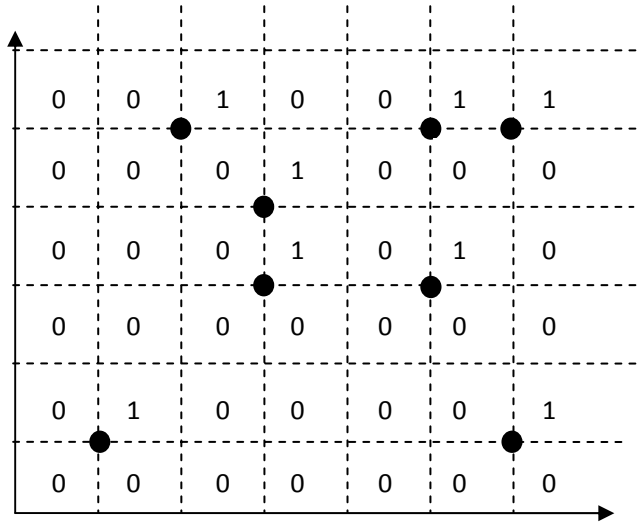
Preprocessing:

Bucket sort all points by their X-coordinates and Y-coordinates.

Create a matrix P of (N + 1) * (N + 1) with each entry (x, y) to be:

1, if exists one node whose coordinates is (x-1, y-1)

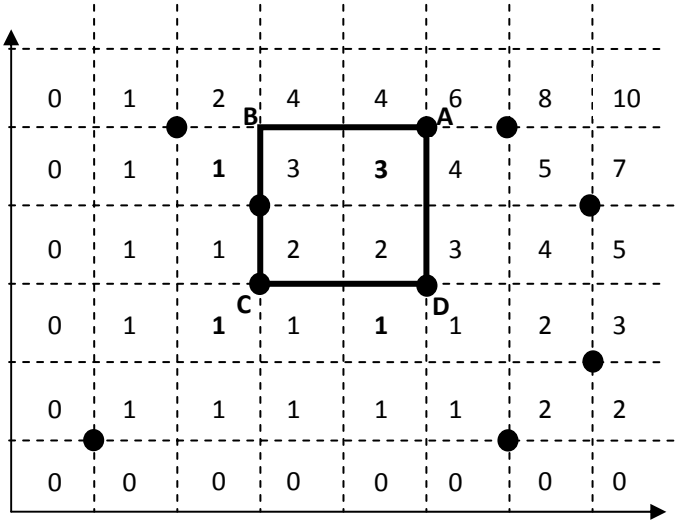
0, else




Calculate the dominate value for each cell in the array using:

P(x, y) = P(x, y) + P(x - 1, y) + P(x, y - 1) - P(x - 1, y - 1); (x > 0 && y > 0)

or P (x, y) (else)




Query:

Given a query request,

Binary search the corners of the request rectangle in the array (A, B, C and D in the figure above),

and the number of points in the rectangle is
P(A) - P(B) - P(D) + P (C) = 3 - 1 - 1 + 1 = 2
Note that in this algorithm, the upper and right boundary of the rectangle is considered out of the range.

2)
First we divide the plane with horizontal and vertical line passing through the points, and compute the number of points that are dominated by the top-right point of each cell.
According to the inclusion- exclusion principle, for a region ABCD, the number of points dominated equals to P(A) - P(B) - P(D) + P (C) (Proof omitted).
In this way, the range searching can be mapped to finding the number of dominated points in the rectangle.

3)
Hence we need a matrix of (N + 1) * (N + 1) so the space complexity is O(N*N);
To calculate the number of each cell, 3 steps are needed, and the time complexity is O(N*N) each. But they are performed one after another, so the overall time complexity is still O(N*N).
For query, binary search is performed for each corner, so the time complexity is O(logN)

4

A)

Sort the set S using any sort algorithm in O(NlogN)

For each element Y in S, we do:

Calculate the Y' = X – Y;

Binary search Y' in S

If found Y' and Y' is not the same element with Y, output Y and Y' to result

Return the result.

B)

It is obvious that for any X, if there are two elements of S whose sum is exactly X, say Y and Y', Y' must be equal to X – Y. So what we need to do is just to find out if the counterpart of each element is in the set S. And because the S has been sorted, a binary search will ensure that we will not ignore the proper value.

C)

Sorting takes time of O(NlogN)

And binary search one element takes time of O(logN) but we may have to repeat it N times in the worst case, so the overall time will be O(NlogN).

The two steps are performed sequentially, so the time complexity of the algorithm is O(NlogN).