February 2012 Problem 'planting' Analysis

There are several ways to approach this problem.

1) An approach is to extend every line segment in our scene, creating a grid of horizontal and vertical lines (the grid is easy to loop over if we first sort all the x and y coordinates in the scene). For each rectangular cell in the grid, we add its area into the total if some rectangle overlaps that cell (and this is easy to check by looping over all the rectangles in the input).
2) For the mathematically inclined, the problem can also be solved via the principle of inclusion-exclusion: loop over all 2^N subsets of rectangles, compute the area A of the intersection of the rectangles in each subset, and either add this area to the total if the subset contains an odd number of rectangles, or subtract it from the total if the subset contains an even number of rectangles (i.e., add the areas of all single rectangles, then subtract the areas of all pairwise intersections, then add back in the areas of all 3-wise intersections, etc.)
3) The simplest way to solve this problem is with a "sweep line" approach. First, sort all the y coordinates in the scene (2N of them) and use them to divide space up into horizontal slices. We store in an array the height of each slice as well as an "overlap count" for each one (described shortly). We then sort all the x coordinates in the scene (2N of them) and sweep across the plane from left to right. Any time we hit the leading vertical edge of a rectangle, we increment the overlap counts of all the slices covered by that rectangle, and any time we hit the trailing vertical edge of a rectangle, we decrement the overlap counts of all its slices. We therefore maintain, during our sweep, the current number of "active" rectangles within each slice. To compute the total area, we simply add up the area swept across consisting of slices having positive overlap counts. The total running time is O(N^2), although it can be reduced even further to O(N log N) using a fancier data structure to encode the array of overlap counts.

```
// sweepline solution
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <vector>
#include <utility>

using namespace std;

const int MAXN = 1111;

int N, x1[MAXN], y1[MAXN], x2[MAXN], y2[MAXN], all_x[2 * MAXN];

int main()
{
	FILE * w = fopen("planting.in", "r");
	FILE * o = fopen("planting.out", "w");

	fscanf(w, "%d", &N);
	for(int i = 0; i < N; i++)
	{
		fscanf(w, "%d %d %d %d", &x1[i], &y1[i], &x2[i], &y2[i]);
```

```
		all_x[2 * i] = x1[i];
		all_x[2 * i + 1] = x2[i];
	}
	sort(all_x, all_x + 2 * N);

	// sweep the x-coordinates
	long long answer = 0;
	for(int i = 0; i < 2 * N; )
	{
		int x = all_x[i];
		if(i != 0 && x == all_x[i - 1])
		{
			i++;
			continue;
		}
		vector<pair<int, int> > y;
		// look for relevant rectangles
		for(int j = 0; j < N; j++)
			if(x1[j] <= x && x2[j] > x)
			{
				y.push_back(make_pair(y2[j], 1));
				y.push_back(make_pair(y1[j], -1));
			}
		if(y.size() == 0)
		{
			i++;
			continue;
		}
		// sweep
		sort(y.begin(), y.end());
		long long cur_area = 0;
		int num_rectangles = 0, pos = 0;
		while(pos < y.size())
		{
			int bottom_y = y[pos].first;
			num_rectangles += y[pos].second; // num_rectangles == 1
			while(num_rectangles > 0)
				num_rectangles += y[++pos].second;
			int top_y = y[pos].first;
			cur_area += top_y - bottom_y;
			pos++;
		}
		// find the next x-coordinate
		int j = i + 1;
		while(all_x[j] == all_x[i])
			j++;
		answer += cur_area * (all_x[j] - x);
		i = j;
	}

	fprintf(o, "%lld\n", answer);
	printf("%lld\n", answer);

	return 0;
}
```