Analysis for task grid

The first thing that comes to mind is to create a matrix NxN. The ways to reach each cell are the sum of the ways to reach the cell above and the ways to reach the cell on the right. If any one of the stated cells is outside the array, we add 0 for it since it’s unreachable. The different ways to reach the first cell are always 1, unless it’s restricted. If we reach a cell with coordinates – the coordinates of a restricted cell we write 0 in that cell. Following these rules we achieve the result in the last cell on the last row.

Example for a matrix with no restricted dots:

1 1 1

1 2 3

1 3 6

Example for a matrix with one restricted dot:

1 1 1

1 2 0

1 3 3

Let’s take a look at the limits. For N = 6000 and modulo = 10000000019 we’ll need to use the type long long int. The memory used to do so is way over the memory limit. So we need to modify the idea.

Instead of N\*N matrix, we can just use an array a[N] with a first element a[0] = 1 and the rest – zeros. To reach the result in the final row of the NxN matrix, we need to do N iterations, where on each iteration, the cell a[j] = a[j] + a[j-1], for j = 1, 2, …, N-1. Don’t forget to use modulo, as well as to check if a restricted dot is reached – assign 0 to the respective cell if so. In case the cell with coordinates (1, 1) or (N, N) is restricted the final result is 0. Otherwise the result will appear in the last cell on the last iteration.

The solution has time complexity of O(N\*N).

Author: Delyan Dobrev