Като ця ло задачата може да се каже като: имаме множество от двойки числа {x, y} и за всяка двойка търсим броя на елементите на множеството, чиито координати са по-малки. Тоест за {x, y} трябва да преброим всички {a, b}, така че а<=x && b<=y.

Нека си представим задачата сякаш нямахме двуизмерни точки, а едноизмерни такива. Тогава условието ще ни бъде едноизмерно – трябва да намерим всички **a**<=**x.** Това може да се направи като сортираме звездите и така за една звезда бройката, която търсим, ще бъде просто индекса й (приемаме, че няма повтарящи се звезди).

Сега нека се върнем обратно в двуизмерната задача. Можем отново да сортираме звездите по едното измерение. Така какво се получава: за всяка звезда, всички звезди наляво от нея са евентуално „по-малки“ от нея, а всички надясно със сигурност не са такива. Също таа занем, че всички звезди наляво са със сигурност „по-малки“ от нашата спрямо измерението, което сме избрали и трябва само да се съобразим с другото измерение. Ако, да кажем, измерението което ни остава да проверяваме е Х, то тогава за нашата точка {x, y} трябва да преброим реално всички х-ове, които са по малки или равни на нашия или с други думи да видим колко са точките, които имат за х координата {1, 2, 3, … x}. Това можем да го направим чрез интервално дърво, което играе ролята на масив брояч.

Накратко, точките са ни дадени на входа сортирани и реално ние само ги четем. За една точка отговорът ще бъде query(0, x). След това в дърво отбелязваме, че още една точка има координата х, т.е. правим нещо като update(x, +1).

Между другото в тая задача координатите са много малки, ама принципно при такива работи е желателно да ги компресираме. Тук компресията работи, понеже на нас реално не ни пука кви са координатите, а кои са по-малки от кои други.