АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ГРАДИНА

Решение на подзадача 1:

Можем да пробваме всички O(N^4) варианта за четирите ъгли на правоъгълната градина.

Решение на подзадача 2:

Можем да фиксираме два противоположни ъгъла на градината. Така сме фиксирали и координатите на останалите два върха на правоъгълника, защото той трябва да е със страни успоредни на осите. Можем да държим всички точки в set и да постигнем сложност O($N^2 * log\ N$).

Решение на подзадача 3:

За да решим подзадача 3, трябва да достигнем до решение със сложност O(N^2). Нека сортираме всички точки спрямо X координатата, а при равни X координати, спрямо Y-координатата. Нека точките в сортираната редица *са* $P_1(X_1,Y_1)$, $P_2(X_2,Y_2)$, $P_3(X_3,Y_3)$, ..., $P_n(X_n,Y_n)$. И нека искаме да преброим правоъгълниците с дясна координата $X = X_i = X_{(i+1)} = X_{(i+2)} = ... = X_j$ и $X_{(i-1)} < X_i = X_j < X_{(j+1)}$. Нека $S = \{Y_i, Y_{(i+1)}, ..., Y_j\}$. Разглеждаме всички точки с X координати, по-малки от X_i. За всеки две точки P_k, P_l, такива че $X_k = X_l < X_i$ и Y_k, $Y_l \in S$, ние получаваме нов правоъгълник.

Нека фиксираме лявата X-координата на правоъгълниците, които броим: нека това е X': $X' < X_i$. Да преброим точките P_s с X-координата X'(т.е. $X_s = X'$) и Y_s е в S. Нека броят на тези точки е C => имаме C*(C-1)/2 правоъгълника с лява страна X' и дясна страна X_i. Тъй като $-1000000 \leq Y_i \leq 1000000$, можем да запазим масив с 2000000 елемента с 0 или 1, в зависимост дали съответната Y-стойност е в множеството S. Така можем константно да проверим за някое s дали $Y_s \in S$. За да оценим сложността на алгоритъма, нека го запишем по-формално:

1. Сортираме точките спрямо X координата, а при равни X координати – спрямо Y координата – сложност O($N*logN$)
2. Вървейки по сортирания масив с точки, за всяка група от точки с равни X координати, строим множество S с Y координатите на точките от групата. Множеството трябва да бъде построено така, че за константно време да може да се проверява дали някое цяло число между -1000000 и 1000000 принадлежи на него. За целта използваме масив с 2000000 елемента, който нулираме и след това записваме 1 в елементите, съответстващи на Y координатите на точките от групата. Ако приемем, че нулирането е със сложност O(1000000), то тази операция е с такава сложност. В действителност memset работи доста по-бързо и все пак не трябва да се рискува. По-добре е да се компресират координатите на точките. Тогава строенето на множеството S ще бъде със сложност O(N).

3. За всяка група точки с равни *X* координати, за която е построено множеството *S*, се връщаме назад и за всяка предшестваща група точки с равни *X* координати, броим ония точки от нея, чийто *Y* координати се съдържат в множеството *S*. След това, както беше описано по-горе, пресмятаме броя на правоъгълниците, чийто леви ъгли са в точки от тази група, а десните – в групата, за която е построено множеството *S*. Като се върнем през всички групи, ще сме преброили всичко правоъгълници, чийто десни върхове са в групата, за която построихме множеството *S*. Сложността на това преброяване е O(*N*), тъй като връщайки се през групите, минаваме през всички точки с по-малки *X* координати точно по веднаж. И така работата, която вършим за всяка група от точки с равни *X* координати е O(N) (строене на множеството S)+O(*N*) (минаване по групите назад и броене на правоъгълниците), т.е. тя е O(*N*).
   *Общата сложност на алгоритъма е O(брой групи точки с равни X координати * N). В най-лошия случай сложността е O(N^2).*

Решение на подзадача 4:

Както се вижда, решението на подзадача 3 дава много добри резултати, ако имаме малко групи с различни *X*-координати и много точки с равни X-координати. Когато групите са много и във всяка има малко точки, този алгоритъм не може да реши подзадача 4.

За решаването на задача 4 ще разгледаме още един алгоритъм, който работи добре при много на брой, малки групи от точки с равни *X* координати, но се бави при малко на брой, големи групи от точки с равни *X* координати (т.е. той е в известен смисъл е „обратен“ на предния). При този алгоритъм поддържаме структура от двойки точки с еднакви *X* координати, но с различни *Y* координати (по-точно поддържаме структура с двойки *Y* координати на точки, които са с еднакви X координати – самите *X* координати не ни интересуват). За всяка двойка пазим и колко пъти се е срещнала до текущия момент при последователно обхождане на групите от точки с равни *X* координати. Такава структура можем да дефинираме чрез map< pair, int >, което е двоично балансирано дърво. При това обхождане, за всяка група от точки с равни *X* координати правим следното:

1. Разглеждаме всяка двойка (Y_a, Y_b), където Y_a и Y_b са *Y* координати на точки от групата. Търсим в структурата колко пъти се е срещала тази двойка в предните групи. Всяко нейно срещане означава нов правоъгълник.
2. След като обработим по този начин поредната бройка, то я добавяме в структурата (или увеличаваме броя на срещанията и с единица, ако тя вече е в структурата)

След като обработим по този начин всички групи ще получим търсения брой правоъгълници. Да направим оценка на сложността на алгоритъма. Нека $m_1, m_2, \ldots, m_p$ са бройките на точките с еднакви *X* координати в групите. Броят на двойките от точки в група *i* е $m_i*(m_i-1)/2$, т.е. O(m_i^2). За всяка двойка правим търсене и обновяване в структурата, която към момента съдържа O($\sum_{l=1}^{i} m_l^2$) елемента. Това значи, че за всяка група изпълняваме O(m_i^2*log ($\sum_{l=1}^{i} m_l^2$)). Тогава сложността на целия алгоритъм можем да я оценим като O(*F*logF*), където $F=\sum_{l=1}^{p} m_l^2$).

Алгоритъмът, който решава подзадача 4 е съчетание от двата алгоритъма. Нека сме избрали някакво число *K* (какво точно ще видим след малко) и отново да разгледаме групите от точки с равни *X* координати. Да наречем такава група „голяма“, ако в нея има повече от *K* точки и „малка“, ако съдържа най-много *K* точки. Броят правоъгълници ще пресмятаме по следния начин: движим се по групите и:

- ако групата е „голяма“, за нея прилагаме модифициран алгоритъма от подзадача 3. Модификацията се състои в това, че пресмятаме всички правоъгълници с десни ъгли в групата и леви във всички групи (големи и малки) преди тази група **и всички правоъгълници с леви ъгли в тази група и десни само в МАЛКИТЕ групи след тази група.** Това че гледаме само малките групи след разглежданата голяма е с цел да не преброим някой правоъгълник два пъти. Както беше казано, тази обработка на една голяма група е със сложност O(*N*).
  Като минем през всички големи групи ще сме преброили правоъгълниците, които имат поне една страна в голяма група.
- ако групата е „малка“, то за нея прилагаме втория алгоритъм, като в структурата вкарваме само двойки от *Y* координати на точки от малки групи. Като минем през всички малки групи, ще сме преброили и правоъгълниците, на които и двете страни са в малки групи.

Сега да оценим сложността на такъв алгоритъм. За големите групи сложността е O(брой големи групи * *N*). Тъй като броят на големите групи е най-много *N/K*, то можем да кажем, че най-голямата сложност, произтичаща от големите групи е $O(N^2/K)$. За малките групи: най-големият брой двойки, които може да породи една малка група е *K*(K-1)/2*. Максималният брой двойки е (*N/K*)*(*K*(K-1)/2*) = O(*N*K*). При положение, че всички групи са малки и във всяка има по *K* точки, то сложността е O(*N*K*log(N*K)*), което е най-голямата сложност, която може да се получи от частта на алгоритъма за малките групи.

Остава да изберем такава стойност на *K*, че двете числа да са почти равни при N = 50000. Стойностите на *K* около 40-70 дават добри резултати.

Автор: Йордан Чапъров