АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
С ЛИНИЙКА И ПЕРГЕЛ

Състезателите би трябвало да са срещали основните построения с линия и пергел в училище (Учебна програма по математика за VII клас). Въпреки че знания за основни построения биха направили решението малко по-ефективно, задачата е решима и без предварителни знания.

Гледайки формулата, по която се оценяват резултатите, виждаме че ако можем да конструираме точката в теста с не повече от 45 стъпки, ще вземем всички точки за конкретния тест.

1. tary = 0, tarx <= 40, tarx целочислено.
Възможно решение е да генерираме всички точки с целочислени координати върху оста Ox последователно по следния начин:

Точките 0 и 1 са вече генерирани. Взимаме последната генерирана точка k (започваме с точка 1), и построяваме окръжност с център тази точка, и радиус 1 (това е разстоянието между точки с идентификатори 0 и 1). Пресичаме тази окръжност с правата Ox, дефинирана от точките 0 и 1. Така генерираме точките k – 1 и k + 1. Не се интересуваме от k – 1, продължаваме работа с точка k + 1. Продължаваме докато не генерираме точка tarx. Това решение отнема tarx стъпки.

2. tary = 0, tarx цяло
Решението по-горе ни показва как да добавим едно към число. Сега да видим как да удвояваме число. Ако имаме точка (k, 0) и искаме да генерираме (2*k, 0) е достатъчно да вземем окръжност с център (k, 0) и радиус k (разстоянието между (0, 0) и (k, 0)), и да я пресечем с оста Ox (правата дефинирана от точки 0 и 1).
След като можем да увеличаваме дадено число с единица, и да удвояваме дадено число, можем да генерираме tarx като вървим по двоичното му представяне. Например, за да генерираме числото 25 = $11001_{(2)}$ = ((1 * 2 + 1) * 2 * 2 * 2 + 1, можем да започнем от 1 и да направим операциите 0, (*2, +1), (*2, +1), (*2), (*2), (*2, +1). Като видим в двоичното представяне на числото 0, правим (*2), като видим 1, правим (*2, +1). Така винаги имаме генериран префикс на числото. Брой стъпки: 2 * log(tarx).

3. tary = 0
За да генерираме произволно реално число върху остра Ox е достатъчно да генерираме достатъчно голямо число k > tarx, и да направим двоично търсене за стойността на tarx, като на всяка стъпка генерираме средата на интервала.
За да построим точка на средата на отсечка AB, можем да построим симетралата на отсечката AB и да я пресечем с правата AB. Симетралата се строи като вземем две пресичащи се окръжности с равни радиуси, и центрове в точките A и B. В езика на задачата, е достатъчно да направим

```
define_circle(A, A, B);
define_circle(B, A, B);
intersect(id1, x1, y1, id2, x2, y2);
define_line(id1, id2);
define_line(A, B);
intersect(id1, x1, y1, id2, x2, y2);
```

и точката между A и B ще е (x1, y1) с идентификатор id1.

А сега двоичното търсене: започваме с интервал [l, r] = [0, k].
На всяка стъпка генерираме m = (l + r) / 2. tarx е в интервала [l, m] или в интервала [m, r].

Стесняваме работния интервал до интервала, съдържащ tarx. Когато нашият интервал е достатъчно малък, ще се доближим до tarx на следващата стъпка, и можем да докладваме точката.
Брой операции до достигане на необходимата точност: около 50 (по две операции за делене на отсечка на две). Ще вземем почти всички точки за теста, стига да генерираме стойността на k бързо (например с многократно използване на операция *2). Визуализация на намиране средата на отсечка: http://www.mathopenref.com/constbisectline.html

4. tarx > 0, tary > 0
След като построим отсечки с дължини tarx и tary, лесно можем да построим точката (tarx, tary):

1. Нанасяме отсечката tarx върху оста Ox (генерираме точка (tarx, 0)).
2. Издигаме перпендикуляр от (tarx, 0) спрямо оста Ox – на тази права ще лежи точката (tarx, tary). Перпендикуляра се построява като пресечем окръжност около (tarx, 0) с оста Ox, и вземем симетралата на получените пресечни точки.
3. Пресичаме току що построената симетрала с окръжност, с център (tarx, 0) и радиус tary.

Решение, което строи дължините tarx и tary с двоично търсене ще използва над 100 операции и ще хване около 60-70 точки.

Решение за 100 точки:
Идея: Ако имаме отсечки с дължини A и B, можем да генерираме отсечка с дължина A / B. Това се прави като вземем точките C = (B, 0) и D = (0, A), и построим права успоредна на CD през точката (1, 0). Тази права ще пресече оста Oy в точка (0, A/B). (Координатите на точката се виждат лесна като разгледаме подобни триъгълници). Построяването на права успоредна на друга права през точка се прави чрез издигане на два перпендикуляра (4 операции), или построяване на ромб (3 операции, както е показано на http://www.mathopenref.com/constparallelrhombus.html).
Лесно можем да извършваме аритметичните операции събиране (A+B) и изваждане (A-B) като вземем точката (A, 0) и построим окръжност с радиус B около нея. Като пресечем тази окръжност с оста Ox, построяваме точки (A-B, 0) и (A+B, 0).

Едно дробно число t може да се представи като t = a + b / c, където a е цялата част на числото, а b/c (b < c) представляват дробната част на числото, и a, b, c са цели числа. За да се доближим с исканата точност до точката-цел е достатъчно да ограничим 0 <= a, b, c <= 10000.
Необходимо ни е да генерираме две числа tarx = a + b / c, и tary = d + e / f. Можем да вземем c=f . Тоест, искаме да генерираме 5 цели числа бързо, за да можем да построим отсечки с дължина tarx и tary.
Генериране на числата чрез метода описан в 2. би отнел над 100 операции.

За да генериране бързо няколко цели числа можем първо да построим всички степени на двойката (1, 2, 4, 8, …., 2^{14} > 10000) като точки на оста Ox, като взимаме окръжности с център последната построена точка и радиус, разстоянието до (0, 0). Така всеки път удвояваме радиуса на окръжността, с която строим. Построяването на точка с целочислени координати (p, 0) става като вземем двоичното представяне на p и добавим всички дължини, съответстващи на степените на двойката, които участват в двоичното представяне на p. Този алгоритъм би отнел 14 + 5 * 14 = 84 стъпки.
Това може да се подобри като забележим, че може да генерираме числото със всички битове 1 (2^{14} - 1), като извадим 1 от 2^{14}. След това, ако p има над 7 единици в двоичното си

представяне, можем да тръгнем от $2^{14} – 1$ и да махаме битовете на позиции, които са 0 в p. Това ще отнеме най-много 7 операции. Тоест, може да генерираме 5 цели числа с 15 + 5*7 = 50 операции.
Допълнително подобрение може да се постигне като вместо числата (1, 2, 4, …) построим числата (1, -1, 2, -2, …, 2^{14}, -2^{14}) като вземем концентрични окръжности около точката (0, 0), и удвояваме радиусите им. Това би отнело 15 операции. Но забележете, че така генерираме всички дължини на отсечки, които са сума на две степени на двойката ($2^4 + 2^8$ е разстоянието между (2^4, 0) и (-2^8, 0)). По подобен начин, това ни позволява да слагаме или махаме по два бита на числото p. Това ни дава решение, което ще генерира 5 цели числа с 15 + 5 * 4 = 35 операции.
Сега числото c в представянето t = a + b / c, може да бъде взето равно на 2^{14}, което имаме генерирано. Това дава решение с 15 + 4 * 4 = 31 операции за генерирането на 4те цели числа.

Как да намерим целите a, b, c по зададено дробно t?
a = floor(t)
c = 2^{14}
b = floor((t – a) * c)
Грешката между a + b / c и t ще е по-малка от 1/c = $1/2^{14}$.

Заедно с всички допълнителни операции, необходими за генериране на частните b / c, e / c, и самите приближения до tarx и tary, и издигане на перпендикуляр, на авторовото решение са необходими най-много 44 операции, при ограниченията зададени в условието.

Автор: Йордан Чапъров