Покритие

Дадени са n отсечки върху една права (0 < n < 100). Всяка отсечка е зададена с координатите на двата си края ai и bi, i = 1, ..., n. Координатите са цели числа от интервала [-999, 999]. Някои от отсечките могат да се припокриват. Напишете програма, която премахва минимален брой от дадените отсечки така, че никои две от останалите отсечки да нямат обща вътрешна точка, т.е. точка, която принадлежи едновременно на двете отсечки и е вътрешна за поне едната от тях.

Жокер: Още една класика в динамичното програмиране.
Помощна Лема: Ако максималният брой незастъпващи се отсечки с координати (и леви и десни) по-малки или равни на X1 е C1 и X2 е координата по-голяма или равна на X1 (X2 >= X1), то максималният брой C2 незастъпващи се отсечки с координати (и леви и десни) по-малки или равни на X2 е по-голям или равен на C1. Доказателството е очевидно: всички незастъпващи се отсечки с координати по-малки или равни на X1 са такива и за X2, тъй като X2>= X1. Също така може и да има K отсечки, чиито десни граница са в интервала (X1, X2], а броя незастъпващи се отсечки до лявата им граница е C1. Тогава C2 = C1 + K

Ще направим динамично програмиране с едномерна таблица DP. Нейното единствено измерение ще е абсцисата X, по която се задават левия и десния край на отсечките. Стойностите на елементите в таблицата ще са максималния брой незастъпващи се отсечки, с координати по-малки или равни на текущия индекс от таблицата. Тъй като имаме отрицателни координати индекса на таблицата ще бъде изместен с base_offset = +1000 спрямо координатата на която той съответства, за да е винаги неотрицателен. Целта ни е да попълним таблицата в нарастваща посока на индекса и. Първият елемент (с индекс 0) в таблицата ще е началния инициализиран от нас елемент - една помощна нула (нула застъпващи се отсечки преди най-малката от всички координати на отсечки). Вторият елемент (с индекс 1) ще съответства на координата -999. Последният елемена на таблицата (съответстващ на координата +999) ще е отговорът на задачата.
Всички отсечки ще ги индексираме спрямо крайната им дясна координата. Примерно ще използуваме std::map<int, vector<int>> segment_map. Тук ключа key (int) ще е дясната координата на всяка една от отсечките, а стойността (vector<int> value) ще е масив с левите координати на всички отсечки (възможно е да са повече от една) завършващи на дясна координата текущия ключ в мапа key. Чрез това представяне си гарантираме, че segment_map е напълно еквивалентен на масива от всички отсечки (двойки лява и дясна координата - pair<int, int>), без

да губим никаква информация. Целевата ни функция за даден елемент от таблицата i (съответстващ на координата i-base_offset) ще е максимума от предната стойност на таблицата (според горната помощна лема) и 1 (заради текущата отсечка завършваща на i-base_offset) плюс колкото е бил максимума на незастъпващите се отсечки в началната лява точка на текущата отсечка завършваща на i-base_offset:
DP[i] = max(DP[i-1], max(1 + DP[segment_map[i-base_offset][j] + base_offset] | j = 0..segment_map[i-base_offset].size() - 1))

Input Format

Входните данни се четат от стандартния вход. От първия ред на стандартния вход се въвежда цяло число k – броя на тестовите примери. За всеки тестов пример данните се въвеждат по следния начин: От първия ред се въвежда цялото число n. Следват n реда, всеки съдържащ по две цели числа, разделени с един интервал. Всяка от тези двойки числа задава координатите на двата края на поредната от дадените отсечки.

Constraints

1 <= k <= 10
-999 <= ai, bi <= 999

Output Format

Резултатът от програмата трябва да бъде записан на стандартния изход, като за всеки тестов пример се извежда единствено цяло число, равно на броя на отсечките, които са останали, след като програмата е премахнала тези, за които това се изисква от условието на задачата.

Sample Input 0

```
2
3
6 3
1 3
2 5
6
1 30
1 2
2 3
4 5
5 6
100 200
```

Sample Output 0

```
2
5
```