直升機抓寶

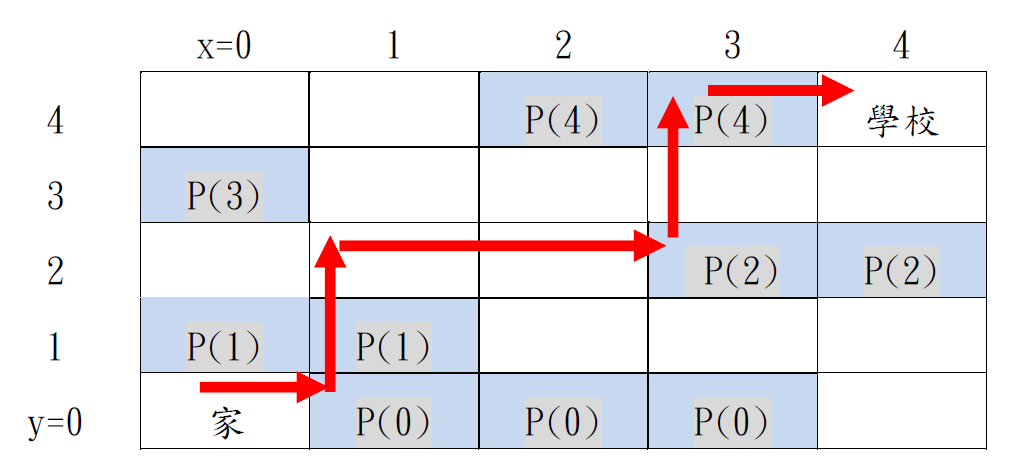
時間限制：2秒

**問題敘述：**

小智的學校在天空之城，他每天開直升機上學，現在他想要規劃一條路徑以便在從家裡到學校的路上可以抓到最多的寶貝，請你寫個程式幫助他。

我們將學校與他家之間所有的位置劃分成N\*N的格子，每個格子以坐標(x,y)表示，其中x代表水平距離，y代表高度，而小智家的坐標在(0,0)的位置，學校在(N-1,N-1)的位置，由於他的父母不准他上學途中貪玩繞路，直升機被設定成每次只能向前或向上一格，也就是說，如果從(x,y)到(x+1,y)或(x,y+1)，當然，他也不可以超過x<N且y<N範圍，否則會無法到達學校。

小智到學校的路途上一共有N隻寶貝，每隻寶貝可以補獲的範圍是某特定高度而水平座標在某連續區間的格子。明確的說，對於寶貝P(i), 0≦ i <N，要捕抓到P(i)，小智必須經過下列座標之一：{(x,y)| S(i)≦x≦T(i) and y=i}，其中所有的S(i)與T(i)都已經透過抓寶雷達得到資料了。



上圖是一個N=5的例子，藍色區塊顯示可以捕抓到寶貝的地方，請注意，每一個寶貝都是在一個水平連續區間。紅線所顯示的路徑是一條合乎規定的路徑，因為他每一步都只有向右或向上，沿這一條路徑可以捕抓到四隻寶貝，是所有可能路徑中可以捕抓到寶貝數最多的。

**輸入說明：**

輸入包含多個測試案例。每個測試案例的第一行是座標範圍的N，接下來N行，每一行有兩個整數S(i)與T(i)，依序是i=0,1,…N-1，其中0≦S(i)≦T(i)<N。一筆測試案例結束後是下一筆測試案例，若N=0代表輸入資料結束，不須處理這筆資料。

**輸出說明：**

針對每個測試案例的每個計算要求，以一行輸出小智最多可以抓到幾隻寶貝。

**子題(Subtask)說明：**本題採IOI模式，在此題中有5個子題，每個子題測試案例數不超過10，Time Limit都是2秒。最佳的演算法對每一個子題都可以在規定時限解出，如果你無法解決所有子題，也可以只解其中某些子題。你的成績將是你所繳交程式中分數最高者。每個子題的測資參數條件如下。

Subtask 1：（15%）N≦10，且對所有i，S(i)=T(i)。

Subtask 2：（17%）N≦5000。

Subtask 3：（17%）N≦100000，且對所有i，S(i)=T(i)。

Subtask 4：（20%）N≦100000。

Subtask 5：（31%）N≦250000。

**輸入範例1：(Subtask 1,3)**

5

2 2

1 1

0 0

2 2

4 4

2

1 1

0 0

0

**輸出範例1：**

3

1

**輸入範例2：(Subtask 2,4,5)**

5

1 3

0 1

3 4

0 0

2 3

2

1 1

0 0

0

**輸出範例2：**

4

1

Solution

The problem asks for a non-decreasing sequence to touch as many intervals as possible.

Let D(y,x) denote the maximum number of intervals for any path from (0,0) to (x,y). Then,

D(y,x)= max{

D(y-1,x)+if(S(y)<=x<=T(y)), // x in the y-th interval

D(y,x-1)+if(x=S(y)) // x is the starting point of y-th interval

}

Observe that the sequence D(y,\*) is surely non-decreasing, and we can easily compute D(,) for y from small to large as follows

Initially d[]=0;

For (i=0;i<n;i++) {// compute D(i,\*)

For (j=S(i); j<=T(i); j++) d[j]++;

For ( ; j<n;j++) d[j]=max(d[j-1],d[j]); // prefix maximum

}

The above two algorithms takes O(N^2) time and linear space.

Subtask 3: When each interval is only a point, the problem degenerates to the longest non-decreasing subsequence problem. Similar to LIS (longest increasing subsequence), it admits an O(NlogN) time algorithm.

Subtask 4:

An idea to improve the quadratic algorithm is to delay the evaluation of the d-values. It is similar to the case of adding intervals in a data structure such as segment tree.

Observe the above quadratic algorithm. When considering the interval [S(i),T(i)] at i-th stage, we in fact add an interval [S(i),R(i)], where R(i) is the maximum index j such that d[j]=d[T(i)]. Therefore, we have the next algorithm.

For each interval i from 0 to N-1 do

Find R(i) by binary search;

Add interval [S(i), R(i)];

Enddo

Output d[N-1];

For such a data structure, the value can be evaluated by adding the values on the path from a leaf to the root. So, it takes O(logN) time to evaluate one d[] value, and thus it takes O(log^2 N) time to find R(i).

The total time complexity is O(N(logN)^2).

Subtask 5

Further improvement.

Let p[i]=d[i]-d[i-1]. Then the sequence can be transformed to a sequence of (x,P[x]), removing those p[i]=0. By using a BST, the p-sequence can be easily maintained. When considering interval [S(i),T(i)], we just insert key S(i) and delete a key R(i) if R(i) exists, where R(i) is the smallest key>T(i) in the BST. This gives us an O(NlogN) time algorithm by using a balanced tree, or map in C++ STL.