**★★★☆☆**

**題組：Problem Set Archive** **with Online Judge**

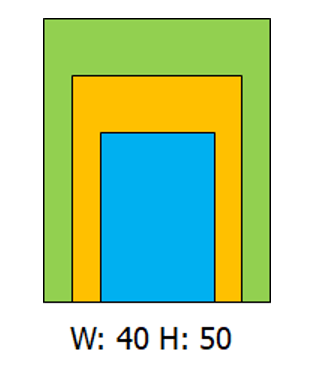
**題號：11368: Nested Dolls**

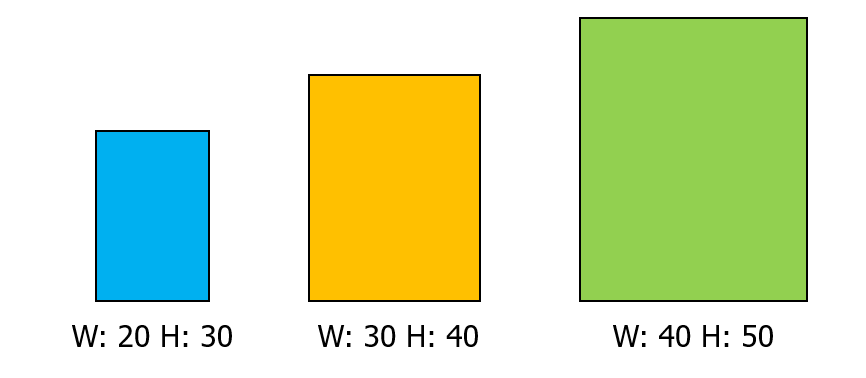
**解題者：嚴宇同**

**解題日期：2021年5月13日**

**題意：**

給定t組測試資料，每筆資料包含m個玩偶，每個玩偶都有其寬度和高度，請計算出每筆資料可以形成的最少組俄羅斯娃娃。

**題意範例：**

[Sample Input]

4 🡺 (4組測試資料)

3 🡺 (3個娃娃)

20 30 40 50 30 40

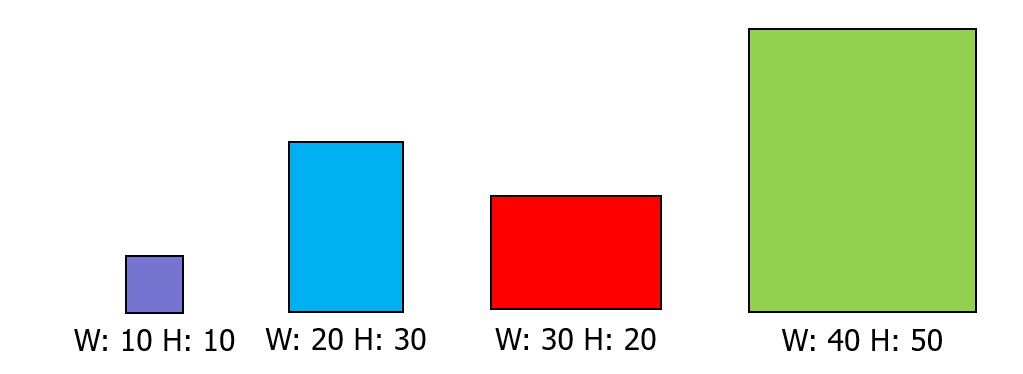
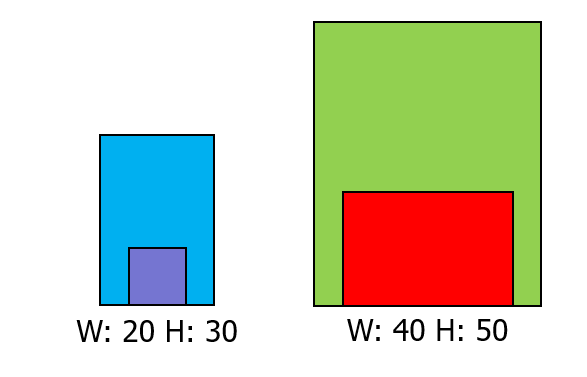
4

20 30 10 10 30 20 40 50

3

10 30 20 20 30 10

4

10 10 20 30 40 50 39 51

[Sample Output]

1

2

3

2

**解法：**

//由寬度大到小做檢查  
1. 先使用map<int, vector<int>>處理玩偶資料，把玩偶依據寬度，由小到大排好；同寬度的玩偶，則依據高度由小到大做排放。

2. 從寬度最大的玩偶 → 寬度最小的玩偶

//使用一個陣列紀錄當前有多少組Nested Dolls。

//先把寬度最大的所有玩偶push進陣列

//每一層寬度做完陣列都要由小到大做sorting

3. 同寬度下，由高度低 → 高度高對上述陣列(頭 → 尾)做檢查

4. 輸出陣列大小即為所求

**解法範例：**

8

(1 1) (3 2) (7 3) (7 5) (7 10) (10 1) (10 4) (10 7)

Width = 10 //長度代表有幾組Nested Doll，值為高度上限

|  |  |  |
| --- | --- | --- |
| 1 | 4 | 7 |

Width = 7

Height = 3

|  |  |  |
| --- | --- | --- |
| 1 | 4 | 7 |

Height = 5

|  |  |  |
| --- | --- | --- |
| 1 | 3 | 7 |

Height = 10 //都不可以放就先記錄起來，離開這個寬度前再放入陣列

|  |  |  |
| --- | --- | --- |
| 1 | 3 | 5 |

Width = 3

Height = 2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 3 | 5 | 10 |

Width = 1

Height = 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 5 | 10 |

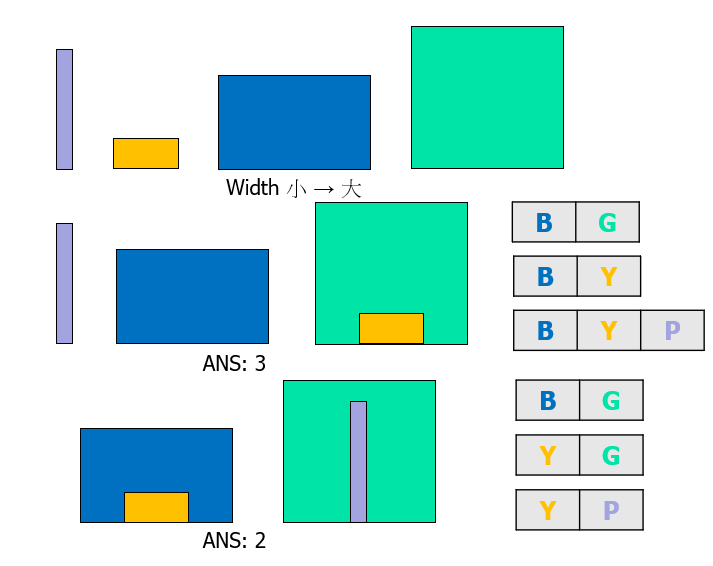
Result: 共4組Nested Dolls

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 5 | 10 |

**討論：**

1. 同寬度、高度的玩偶不能互塞(標記紅色處)

2. 同寬度下，不能對陣列從尾到頭做檢查。



**程式：**

#include <iostream>

#include <iterator>

#include <vector>

#include <algorithm>

#include <map>

using namespace std;

//pair sort costumized

bool mycomparation(int a, int b)

{

return a > b;

}

int main ()

{

int t; //number of test cases

cin >> t;

while(t-- && t>=0)

{

//initialized data

int m; //number of dolls

cin >> m;

map <int, vector<int> >doll;

int i, j; //var for loops

int width; //width of doll

int height; //height of doll

for(i=0; i<m; i++)

{

cin >> width >> height;

doll[width].push\_back(height);

}

for(map<int, vector<int> >::iterator it = doll.begin(); it != doll.end(); it++)

{

sort(it->second.begin(), it->second.end(), mycomparation);

}

vector<int> doll\_set;

for(map<int, vector<int> >::reverse\_iterator it = doll.rbegin(); it != doll.rend(); it++)

{

if(doll\_set.empty() == true)

{

for(i=it->second.size()-1; i>=0; i--)

{

doll\_set.push\_back(it->second[i]);

}

}

else

{

map<int, bool> used;

vector<int> unused;

int check\_1;

for(i=0; i<it->second.size(); i++)

{

check\_1 = 0;

for(j=0; j<doll\_set.size(); j++)

{

if(it->second[i] < doll\_set[j])

{

if(used.count(j) == 0)

{

check\_1 = 1;

doll\_set[j] = it->second[i];

used[j] = true;

break;

}

}

}

if(check\_1 == 0)

{

unused.push\_back(it->second[i]);

}

}

used.clear();

while(!unused.empty())

{

int temp;

temp = unused.back();

doll\_set.push\_back(temp);

unused.pop\_back();

}

sort(doll\_set.begin(), doll\_set.end());

}

}

cout << doll\_set.size() << endl;

}

}