Problem

The first two paragraphs (not counting this one) of this problem and "New Elements: Part 2" are identical. The probl ms can otherwise be solved independently; you do not need to read or solve one in order to read or solve the other.

Muriel is on the path to discovering two new elements that she has named Codium and Jamarium. She has not been ble to isolate them yet, but she wants to start investigating some important properties, like their atomic weights, by i direct means. Since Muriel is working with a single isotope of Codium and a single isotope of Jamarium, their atomi weights are strictly positive integers.

Muriel managed to create N different molecules, each of which contains one or more atoms of Codium and one or m re atoms of Jamarium, and no other elements. For each molecule, she knows how many atoms of each element are pr sent in it. The molecular weight of a molecule is the sum of the atomic weights of all the atoms it contains.

As a first step towards figuring out exact molecular weights for the molecules and atomic weights for the two eleme ts, Muriel wants to sort the molecules by strictly increasing molecular weight. To assess the difficulty of that task, sh wants to know how many orders are valid considering only the information she has right now. An ordering of the m lecules is considered valid if there exist values for the atomic weights of Codium and Jamarium such that the orderin is strictly increasing in molecular weight.

To give an example, we represent each molecule by the ordered pair of the number of atoms of Codium and Jamariu it contains. If Muriel has 3 molecules represented by (1, 1), (2, 1) and (1, 2), there are two possible orderings that ca be strictly increasing in molecular weight: (1, 1), (1, 2), (2, 1) and (1, 1), (2, 1), (1, 2). The first ordering is valid for ny assignment of atomic weights in which Codium is the heaviest of the two elements, and the second is valid for an assignment in which Jamarium is the heaviest. The only case remaining is when both Codium and Jamarium have t e same atomic weight, in which case (1, 2) and (2, 1) have the same molecular weight, so no strictly increasing order ng can be produced for that scenario.

Input

The first line of the input gives the number of test cases, T. T test cases follow. The first line of a test case contains a single integer N, the number of molecules. Each of the next N lines describes a different molecule with two integers i and Ji that represent the number of Codium and Jamarium atoms in the i-th molecule, respectively.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is th total number of valid orderings as defined above.

Limits

Time limit: 20 seconds per test set.
Memory limit: 1GB.
$1 \leq T \leq 100$.
$1 \leq Ci \leq 109$, for all i.
$1 \leq Ji \leq 109$, for all i.
$(Ci, Ji) \neq (Cj, Jj)$ for all $i \neq j$. (All molecules are different.)

Test set 1 (Visible)

$2 \leq N \leq 6$.

Test set 2 (Hidden)

$2 \leq N \leq 300$.

Sample

Input

Output

```
3
3
1 1
1 2
2 1
4
1 2
2 4
2 1
4 2
3
1 2
1 3
2 3
```

```
Case #1: 2
Case #2: 2
Case #3: 1
```

Sample Case #1 is explained in the statement.

In Sample Case #2, the two valid orderings are (1, 2), (2, 1), (2, 4), (4, 2) and (2, 1), (1, 2), (4, 2), (2, 4). Notice that t e ordering (1, 2), (2, 1), (4, 2), (2, 4) is invalid because if (1, 2) is strictly less heavy than (2, 1), then (2, 4), which is xactly twice as heavy as (1, 2), must be strictly less heavy than (4, 2), which is exactly twice as heavy as (2, 1).

Solution:

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>

using namespace std;
#define PB push_back
#define MP make_pair
#define LL long long
#define int LL
#define FOR(i,a,b) for(int i = (a); i <= (b); i++)
#define RE(i,n) FOR(i,1,n)
#define REP(i,n) FOR(i,0,(int)(n)-1)
#define R(i,n) REP(i,n)
#define VI vector<int>
#define PII pair<int,int>
#define LD long double
#define FI first
#define SE second
#define st FI
```

```
#define nd SE
#define ALL(x) (x).begin(), (x).end()
#define SZ(x) ((int)(x).size())

#define unordered_map __fast_unordered_map
template<class Key, class Value, class Hash = std::hash<Key>>
using unordered_map = __gnu_pbds::gp_hash_table<Key, Value, Hash>;

template<class C> void mini(C &a4, C b4) { a4 = min(a4, b4); }
template<class C> void maxi(C &a4, C b4) { a4 = max(a4, b4); }

template<class TH> void _dbg(const char *sdbg, TH h){ cerr<<sdbg<<'='<<h<<endl; }
template<class TH, class... TA> void _dbg(const char *sdbg, TH h, TA... a) {
  while(*sdbg!=',')cerr<<*sdbg++;
  cerr<<'='<<h<<','; _dbg(sdbg+1, a...);
}

template<class T> ostream &operator<<(ostream& os, vector<T> V) {
  os << "["; for (auto vv : V) os << vv << ","; return os << "]";
}
template<class L, class R> ostream &operator<<(ostream &os, pair<L,R> P) {
  return os << "(" << P.st << "," << P.nd << ")";
}

#ifdef LOCAL
#define debug(...) _dbg(#__VA_ARGS__, __VA_ARGS__)
#else
#define debug(...) (__VA_ARGS__)
#define cerr if(0)cout
#endif

struct Testcase {
  int test_idx_;
  Testcase(int tidx) : test_idx_(tidx) {}


  void Run() {
    int N;
    cin >> N;
    vector<PII> samples(N);

    for (auto &s : samples) { cin >> s.st >> s.nd; }

    set<PII> changes;

    for (int i = 0; i < N; ++i) {
      for (int j = 0; j < N; ++j) {
        PII diff{samples[i].st - samples[j].st, samples[i].nd - samples[j].nd};
        if ((LL)diff.st * diff.nd < 0) {
          const int g = abs(__gcd(diff.st, diff.nd));
          diff.st /= g;
          diff.nd /= g;
```

```
        if (diff.st < 0) { diff.st = -diff.st; diff.nd = -diff.nd; }
        changes.insert(diff);
      }
     }
    }

    cout << "Case #" << test_idx_ << ": " << SZ(changes) + 1 << "\n";
  }
};


int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cout << fixed << setprecision(11);
  cerr << fixed << setprecision(6);

  int T;
  cin >> T;
  for (int i = 1; i <= T; ++i) {
    Testcase(i).Run();
  }
}
```