USACO FEB08 Problem 'lines' Analysis

by Neal Wu

To solve this problem, we simply want to count the number of distinct slopes that occur among all pairs of the points. However, we have to be careful about two things:

-Slopes such as 1 / 2 and 2 / 4 need to be considered equal.

-We have to correctly handle slopes like "3 / 0."

To do this, we note that if we have two slopes y1 / x1 and y2 / x2, and we make sure that x1 and x2 are both positive, then the condition y1 / x1 = y2 / x2 is the same as the condition y1 * x2 = y2 * x1. (Similarly, y1 / x1 < y2 / x2 is the same as y1 * x2 < y2 * x1.)

Thus, we can compare slopes of lines by using this multiplication comparison, which correctly considers slopes like 1 / 2 and 2 / 4 as the same. To take care of slopes like "3 / 0" correctly, we note that all lines with slope in the form "y / 0" are parallel, so we count them only once (which is shown in the code below).

Finally, since there can be nearly 20,000 slopes to check, comparing every pair will not be fast enough. Thus, we need a method of counting distinct slopes quickly, and to do this we can efficiently sort all the slopes by the comparison above. After sorting, all pairs of equal slopes will be adjacent in our sorted list, so we can compare them easily.

The following is a sample solution:

```
#include <cstdio>
#include <algorithm>
using namespace std;

FILE *fin = fopen ("lines.in", "r");
FILE *fout = fopen ("lines.out", "w");

const int MAXN = 205;
const int MAXM = 20005;

struct slope
{
    int dx, dy;

    inline slope ()
    {
        dx = dy = 0;
    }

    inline slope (int y, int x)
    {
        dy = y, dx = x;
```

```
// take care of the "y / 0" case
        if (dx == 0)
            dy = 1;

// make sure dx is positive for our comparison to work properly
        if (dx < 0)
        {
            dx = -dx;
            dy = -dy;
        }
    }
};

// comparison of two slopes
inline bool operator < (const slope &left, const slope &right)
{
    return left.dy * right.dx < right.dy * left.dx;
}

int N, M, total;
int x [MAXN], y [MAXN];
slope lines [MAXM];

int main ()
{
    fscanf (fin, "%d", &N);

    for (int i = 0; i < N; i++)
        fscanf (fin, "%d %d", x + i, y + i);


    for (int i = 0; i < N; i++)
        for (int j = i + 1; j < N; j++)
            lines [M++] = slope (y [i] - y [j], x [i] - x [j]);

    sort (lines, lines + M);

// count distinct slopes
    total = 1;
    for (int i = 0; i < M; i++)
        if (lines [i].dy * lines [i + 1].dx != lines [i + 1].dy * lines
[i].dx)
            total++;


    fprintf (fout, "%d\n", total);

    return 0;
}
```

Maximiliano David Bustos adds:

As we only need the number of differents slopes we can use the set container, included in C++ STL. Set can add, remove and check the presence of particular element in O(log M), where M is

the count of objects in the set. While adding elements to set, the repeated elements are discarded. A count of the elements in the set, M, is returned in O(1).

The following is a sample solution:

```
#include <fstream>
#include <set>

using namespace std;

#define INF 99999
#define MAXN 20001

int main () {
   // Variables:
   int N;
   int i,j;
   float slope,x1,y1;
   set<float> s;
   int x[MAXN],y[MAXN];

   // Open files:
   ifstream fin("lines.in");
   ofstream fout("lines.out");

   // Read data:
   fin>>N;
   for(i=0;i<N;i++)
      fin>>x[i]>>y[i];

   // Try all pairs of points:
   for(i=0;i<N;i++)
      for(j=i+1;j<N;j++) {
         y1 = y[i] - y[j];
         x1 = x[i] - x[j];
         if(x1==0) slope = INF;
         else slope = y1/x1;
         s.insert(slope); // If the slope already exists in the set it is not
saved.
      }

   // Output:
   int ans = int(s.size()); // The number of elements saved is the number of
distinct slopes.
   fout<<ans<<endl;

   // Close files:
   fin.close();
   fout.close();
    return 0;
}
```