A Genetic Algorithm for the Partition Problem

H.S. de Hoop & N. Trigonis
H.S.de.Hoop@student.rug.nl & N.Trigonis@student.rug.nl

October 13, 2024

1 Problem description

Given a number of weighted values, find out how to distribute the values into parts as evenly as possible.

2 Problem analysis

We interpret the question as follows: Two line segments intersect if they have precisely one common point that is not an end-point of any of the segments. This interpretation is motivated by the following statements:

- Two line segments intersect if and only if A and B lay on different sides of the line segment trough P and Q. Moreover, P and Q lay on different sides of the line segment through A and B.
- All cases in which multiple points are on both line segments (i.e. overlapping segments) are categorized as not intersecting.

From the equation for a line $y = \frac{\Delta y}{\Delta x} \cdot x + c$ we conclude

$$y \cdot \Delta x - x \cdot \Delta y - c = 0$$

The advantage of the latter form is that lines parallel to the y-axis can be written in the same way as any other line. For the line trough A and B we find: $\Delta x = x_B - x_A$, $\Delta y = y_B - y_A$ and $c = y_A \cdot \Delta x - x_A \cdot \Delta y$. The line trough A and B splits the plane into two halves. Points on one side give a positive result and points on the other side a negative result for the following expression:

$$y \cdot \Delta x - x \cdot \Delta y - c \tag{1}$$

Thus, two points are located in different half planes if one yields a positive and the other a negative result, when we evaluate expression (1) for both points. We can simplify this conclusion:

two points are in different half planes if the product of the expressions (1) is negative.

3 Design

First the points A, B, P and Q are read from the standard input and stored (in the variables `xA`, `yA`, `xB`, `yB`, `xP`, `yP`, `xQ`, and `yQ`). Next, the parameters of the line through A and B are computed (`deltax`, `deltay`, and `c`). Using these, we determine whether P and Q are split up by the line trough A and B (stored in `PQdivided`).

After this computation, we perform the same for the other line segment through P and Q. Note that the variables `deltax`, `deltay` and `c` were used to compute `PQdivided`, and can now be

reused in the computation of `ABdivided`, which denotes whether the points A and B are split up by the line trough P and Q.

Finally the answer is computed from the variables `PQdivided` and `ABdivided`. It is presented to the user by printing it on the standard output.

4 Program code

intersect.c

```
#include <stdio.h>
#include <stdlib.h>

/* (C) Arnold Meijster & Doina Bucur, september 2015:
*/

int main(int argc, char *argv[]) {
    int PQdivided; /* Are P and Q divided through AB? */
    int ABdivided; /* Are A and B divided through PQ? */
    int xA, yA; /* coordinates of point A */
    int xB, yB; /* coordinates of point B */
    int xP, yP; /* coordinates of point P */
    int xQ, yQ; /* coordinates of point Q */
    int deltax, deltay, c; /* parameters of the equation */

    /* input coordinates */
    printf("Please enter the x- and y-coordinates, separated by a space.\n");
    printf("Point A: ");
    scanf("%d %d", &xA, &yA);

    printf("Point B: ");
    scanf("%d %d", &xB, &yB);

    printf("Point P: ");
    scanf("%d %d", &xP, &yP);

    printf("Point Q: ");
    scanf("%d %d", &xQ, &yQ);

    /* determine whether P and Q are separated by the line trough A and B */
    deltax = xB - xA;
    deltay = yB - yA;
    c = yA*deltax - xA*deltay;
    PQdivided = ((yP*deltax - xP*deltay - c)*(yQ*deltax - xQ*deltay - c) < 0 );
    /* determine whether A and B are sperated by the line trough P and Q */
    deltax = xQ - xP;
    deltay = yQ - yP;
    c = yP*deltax - xP*deltay;
    ABdivided = ((yA*deltax - xA*deltay - c)*(yB*deltax - xB*deltay - c) < 0 ) +
                ((yA*deltax - xA*deltay - c)*(yB*deltax - xB*deltay - c) < 0 );
    /* print result */
    if (PQdivided && ABdivided) {
        printf("The line segments intersect.\n");
    } else {
        printf("The line segments do not intersect.\n");
    }
    return 0;
}
```

5 Test results

- Input: (normal case with one common point)

```
0 0
9 9
4 9
8 1
```

Output:

```
The lines intersects each other.
```

- Input: (normal case without common point)

```
0 0
4 8
9 1
5 8
```

Output:

```
The lines do not intersects each other.
```

- Input: (P on line of AB)

```
0 0
9 9
5 5
9 1
```

Output:

```
The lines do not intersects each other.
```

- Input: (all points on one line)

```
0 0
7 7
5 5
9 9
```

Output:

```
The lines do not intersects each other.
```

- Input (lines with equal start-point and end-point)

```
5 5
5 5
8 2
1 9
```

Output:

```
The lines do not intersects each other.
```

6 Evaluation

Due to our relative simple definition of line segment intersection, the program was rather easy to design and write. For our definition of intersection (i.e. only one common point), the program produces correct results. If you would change the definition to at least one common point, further analysis would be necessary. Then the three different results of the product: positive, negative and zero have to be investigated. Which is a bit harder if both expressions return zero as a result. But luckily our definition is sufficient for this assignment.

We have chosen a geometric approach. A mathematical approach would have been possible as well. Then we would have computed the intersection point given the line equations. In this case you have to take care of multiple intersection points. Moreover, you also need to test whether the calculated intersection point is on the line segments. We think that the geometric approach is a lot easier.