2002 IC/CAD Contest

Problem 1: Rectilinear Polygon Resizing

Source: Avant!, Taiwan.

Dec 31 2002

I. Introduction

Design rules arise from limitations in the integrated circuit manufacturing process. They are imposed on the geometry of an integrated circuit layout in order to guarantee that the circuit can be fabricated with an acceptable yield. A popular method for checking design rules employs sequences of geometric operations. We can check minimum feature size on a layer by shrinking by half the minimum size for the layer and growing by the same value and find all missing areas. For example, let the minimum feature size to be checked be 2 units. We can find the area of an input layer A which violates this design rule by the following sequence of operations:

B ← **SHRINK**(A, 1)

C ← GROW(B, 1)

D ← A NOT C

Note that in the operation "B ← **SHRINK**(A, 1)", the left and right edges of the upper left rectangle of A coincide. Thus, the rectangle is dropped after the shrinking.

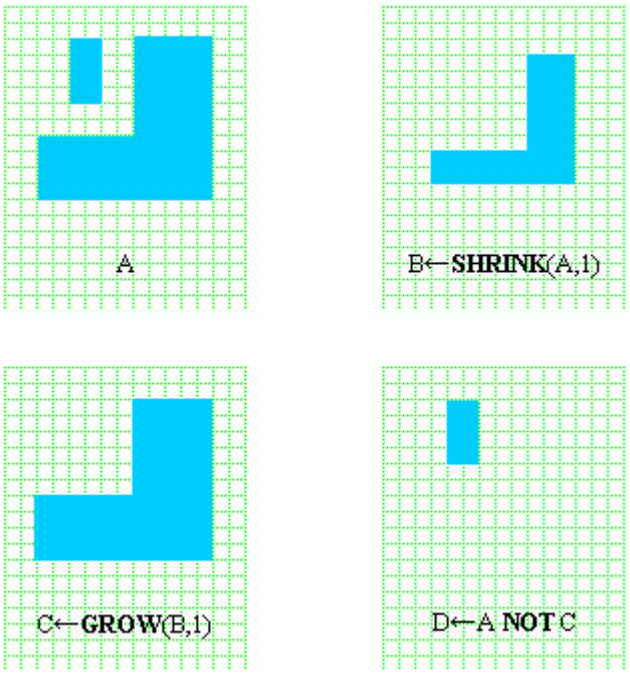


Figure 1.

Given a shape A, operation **GROW** (A, d) is defined as follows:

1. Bisect the angles associated with the vertices in the original shape A. For example, the angle associated with the lower left vertex of the shape "A" in Figure 2(a) is 270?. So the bisection results in 135?. In Figure 2(a), all bisections of the angles are shown as the dashed lines emitting from the shape A.
2. Following the angle bisectors (the dashed lines in the figure), move all edges of the shape "A" ***outward*** by a ***Euclidean distance*** d while maintaining the original connectivity. In Figure 2(a), the lower edge of the shape "A" is moved 2-unit down.

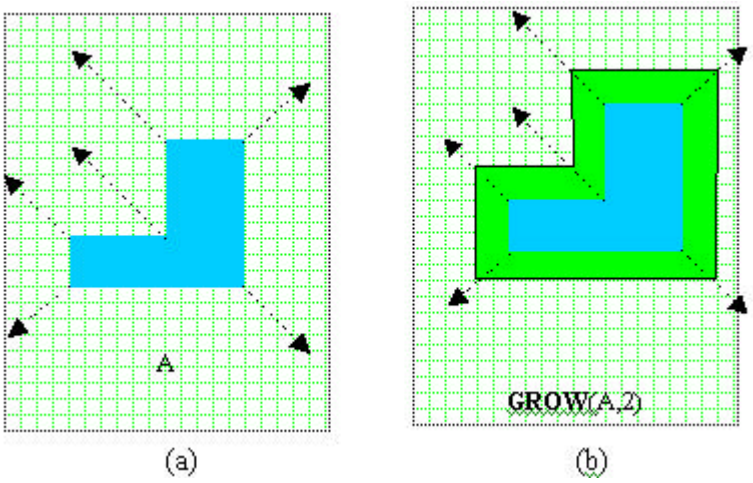


Figure 2 The original shape (a); and the shape after growing by 2 units (b).

3. Sometimes when moving edges by a distance d' (d' <d), the topology of the shape may be changed. Then we define the shape of **GROW**(A,d) by the shape of **GROW**(**GROW**(A,d'), d-d').

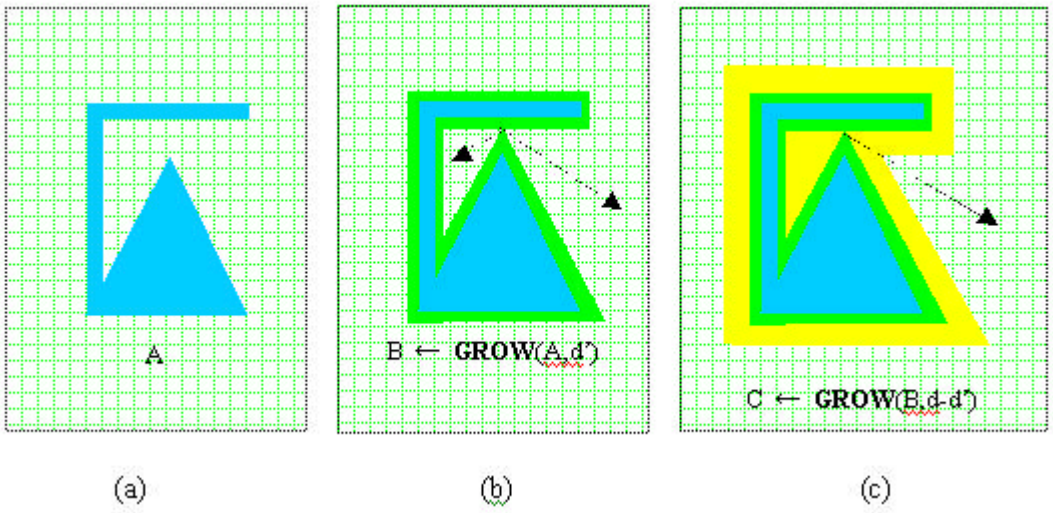


Figure 3

For example in Figure 3(a), the input shape A is a polygon without holes. While growing A by a distance d' , the upper vertex of the triangle shape in A bumps into the outer horizontal part of A. Thus, the resulting shaped B has a hole in it, i.e., the shape of A has been changed to a new shape B (B ← **GROW**(A,d'), Figure 3(b)). In this case, the growing should be continued with the new shape B. So two new angle bisectors, shown as the dashed lines in Figure 3(b), are formed. Then, the edges are moved according to the angle bisectors (C ← **GROW**(B,d-d'), Figure 3(c)). The hole in B might disappear when we continuously grow B. ***At any time the topology is changed, we grow it with the new shape until the total distance is enough.***

The operation **SHRINK** is similar to the **GROW** operation except that all edges are moved inward. The shrinking may result in two edges with distance less than one unit (see "B ← **SHRINK**(A, 1)" in Figure 1). In our cases, all area bounded by edges with less than one unit separation is removed from the layout.

To correctly and efficiently implement **GROW** and **SHRINK** operations for any given shape is not an easy task. In this problem, **only rectilinear simple polygons are considered as input shapes**. A **simple polygon** is a shape enclosed by a single

enclosed polygonal chain that does not intersect itself. There is no hole in a simple polygon. A **rectilinear simple polygon** is a simple polygon of which all edges are horizontal or vertical. All of the coordinators of points are integers. The growing or shrinking distance is also an **integer** and is measured either horizontally or vertically. Notice that when growing a simple polygon, the output shape may contain holes and when shrinking a simple polygon, the output may be several separated polygons.

II. Input/Output Format

A polygon is specified by the number of points following a list of pairs of x and y coordinators:

<polygon> ≡

n

x_1 y_1

x_2 y_2

x_3 y_3

•••

x_n y_n

Note that the list of points might be clockwise or counter-clockwise. Both represent the same polygon.

The input format is :

d_1

<polygon>$_1$

d_2

<polygon>$_2$

...

d_m

<polygon>$_m$

where d_i is a signed integer and if $d_i > 0$ then perform **GROW** operation on <polygon>$_i$ by distance d_i and if $d_i < 0$ then perform **SHRINK** operation by distance $|d_i|$.

Notice that when growing a simple polygon, the output shape might contain holes and when shrinking a simple polygon, the output might be several separated polygons.

The output of **GROW** or **SHRINK** a simple polygon is a polygon list consisting of a

list of simple polygons:

<polygon-list> ≡
k
<polygon>$_1$
<polygon>$_2$
...
<polygon>$_k$

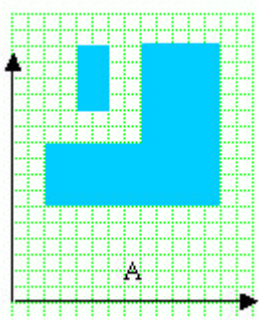
where k is the number of polygons in the polygon list. The order of the polygons in a polygon list makes no difference. Notice that when **SHRINK** a large distance for a given polygon, the output might be an empty polygon list, that is, k = 0.

The output is specified by a list of <polygon-list>:

<polygon-list>$_1$
<polygon-list>$_2$
...
<polygon-list>$_m$

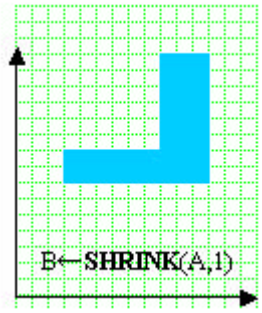
where <polygon-list>$_i$ is the output corresponding to the i-th input polygon.
Consider Figure 1 as an example. The input might be:

-1
4
6 12
6 16
4 16
4 12
-1
6
8 10
8 16
13 16
13 6
2 6
2 8



The output is:

```
0
1
6
3   7
12  7
12  15
9   15
9   9
3   9
```



Here are another examples:

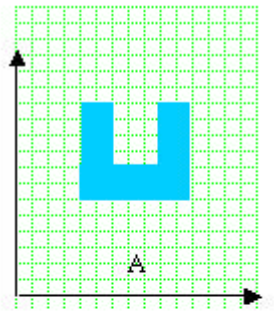
Input:

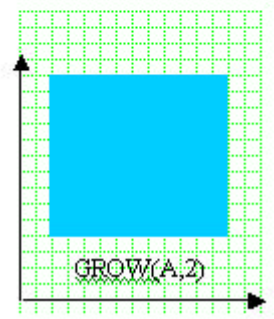
```
2
8
4   12
4   6
11  6
11  12
9   12
```

9 8
6 8
6 12

Output:

1
4
2 4
13 4
13 14
2 14





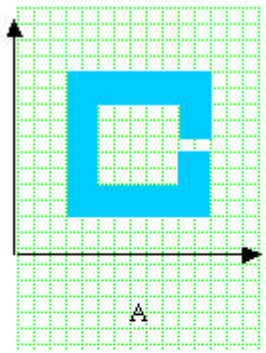
Input:

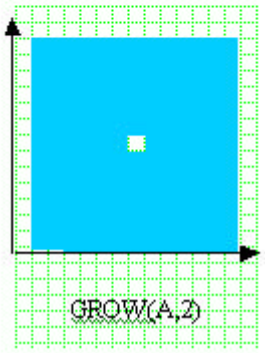
2
12
5 4
5 9
10 9
10 7
12 7
12 11
3 11

3 2
12 2
12 6
10 6
10 4

Output:

2
4
7 6
8 6
8 7
7 7
4
1 13
14 13
14 0
1 0





III. Languare and Platform

1. Language: C or C++
2. Platform: Sun Solaris

Please make sure your program can be compiled with gcc or g++ with optimization flag –O on Sun Solaris 5.6.

Usege:

<program> <input_file> <output_file>

IV. Evaluation

The score will be given based on correctness, time efficiency and memory requirement.

V. Questions

Please report any question regarding to this problem to cad@cs.nthu.edu.tw with the email subject "CAD Contest: Problem 1." Your question(s) will be answered in two weeks, and the Q&A's will be posted at the contest Web site.

VI. References

[1] M. de Berg, M. van Kreveld, M. Overmas and O. Schwarzkopf. *Computational Geometry, Algorithm and Applications,* 2nd ed., Springer-Verlag, Berlin,1998.

[2] B. Preas and M. Lorenzetti (Eds.) *Physical Design Automation of VLSI System,* The Benjamin/Cummings Publishing, 1988.