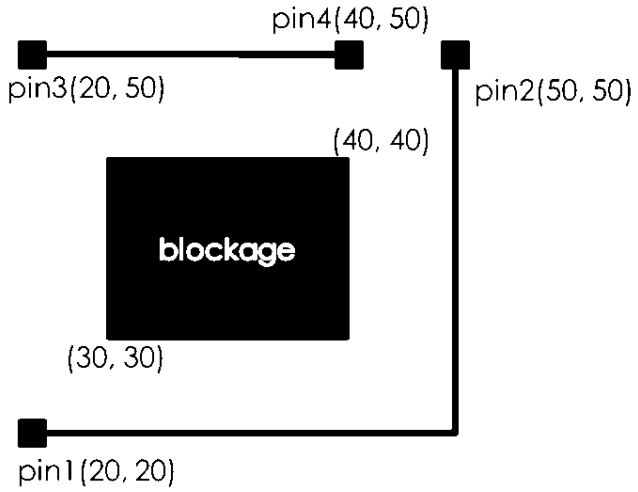
Algorithms

Programming Assignment #3

Net Routing

Introduction:

Routing problem is a critical issue in circuit design. It is a practical problem in the real world and helpful for your final project. You can apply your knowledge learned from the NTUEE algorithm class to the real-world problem. In this PA, you need to route wires to connect the pins. Given the locations of pins, locations of blockages, and netlist, you need to finish the routing that satisfies all requirements by using horizontal and vertical wires. Here is an example.



Locations of pins:

The first column is the ID of the pin. The second column is the x-coordinate of the pin location. The third column is the y-coordinate of the pin location.

```
1 20 20     # pin 1, location, x:20, y:20
2 50 50     # pin 2, location, x:50, y:50
3 20 50     # pin 3, location, x:20, y:50
4 40 50     # pin 4, location, x:40, y:50
```

Locations of blockages:

You cannot route wires across any blockage. The first column is the x-coordinate of the bottom-left corner of a blockage. The second column is y-coordinate of the bottom-left corner of a blockage. Similarly, the third/four columns are the x-coordinate and y-coordinate of the top-right corner, respectively.

```
30 30 40 40     # bottom-left x=30 y=30，top-right x=40 y=40
```

Netlist:

The first column is the ID of the net. The following columns are the pins that are required to be connected by the net. Note the number of pins may be greater than two.

```
1 1 2       # net 1, connect pin 1, 2
```

2 3 4 # net 2, connect pin 3, 4

For the above case, we can route a horizontal wire from (20, 20) to (50, 20) and a vertical wire from (50, 20) to (50, 50) to connect the first net. Next, we can route a horizontal wire from (20, 50) to (40, 50) to connect the second net. Therefore, the output is as follows:

Output:

The output file specifies the ID of the net first and then the locations of wires. The output file should be specified in increasing order of net ID. The first column is the x-coordinate of one side of the wire. The second column is y-coordinate of the same side of same wire. The third/fourth columns are the x-coordinate and y-coordinate of the other side of the same wire. The second line shows the x- y-coordinates of the second wire. Please note that there can be 1, 2, 3, or more wires that connect a net.

Net 1
20 20 50 20 # from (20, 20) to (50, 20)
50 20 50 50 # from (50, 20) to (50, 50)
Net 2
20 50 40 50 # from (20, 50) to (40, 50)

If you cannot route a net, you can report' FAIL' like this.

Net 1
FAIL
Net 2
20 50 40 50

Please implement an algorithm to complete the routing. Note that there are several feasible solutions to satisfy the requirements. You just only need to output one of them.

Assumptions

1. All coordinates are integers, from 0~1000
2. Wire width and height are zero so (20,20), (50,20) is a horizontal wire.
3. Maximum we have 500 pins.

Requirements

1. All pins in a single net need to be connected
2. No wire across or contact any blockage
3. No two wires can cross or overlap with each other.

Input/output Files:

For each test case, there are three input files (`*_pin.in, *_blockage.in, *_net.in`). The first file contains *p* lines that mean the IDs and locations of all pins. The second file contains *b* lines that indicate locations of all blockages. The third file contains *n* lines that show IDs and pins of the nets you need to build. Each location consists of x-coordinate and y-coordinate ($0 \le x < 1000$, $0 \le y < 1000$).

```
// case1_pin.in
1 20 20  // pin
2 50 50
3 20 50
4 40 50
```

```
// case1_blockage.in
30 30 40 40  // blockage
```

```
// case1_net.in
1 1 2    // net
2 3 4
```

In the output file (`*.out`), before you state the wires in a net, you should specify the net you are constructing. In other words, you should output "Net *x*" first, where *x* is the ID of the net. Then, output the locations of wires, each line contains the start point and the endpoint. The sample solution mentioned in the introduction is shown as follows:

```
Net 1
20 20 50 20
50 20 50 50
Net 2
20 50 40 50
```

Command line parameters:

In the command line, you are required to follow one of the following formats

```
./Routing    <input_file_name>    <output_file_name>
./Routing    <input_file_pin>    <input_file_block>    <input_file_net>
<output_file_name>
```

For example, the following command invokes the tool and run your program.

```
./bin/Routing ./inputs/case1.in ./outputs/case1.out
./bin/Routing ./inputs/case1_pin.in ./inputs/case1_blockage.in ./inputs/ca
se1_net.in ./outputs/case1.out
```

Requirements:

1. Your source code must be written in C or C++. The code must be executable on EDA union lab machines.

2. Your binary code should be compiled by the following commands under <student_id>_pa3 directory. The binary code is located in the bin directory.

```
make
cd bin
./Routing    <input_file_name>  <output_file_name>
./Routing    <input_file_pin>   <input_file_block>   <input_file_net>
<output_file_name>
```

3. In your report, you should do the following things,
   3.1 Describe your algorithm to complete the *Routing problem*.
   3.2 Please analyze the complexity of your algorithm in terms of the number of pins (*p*), the number of blockages (*b*) and the number of netlists (*n*)

Validation:

We will announce a program that check whether your routing satisfies all requirements after midterm. We encourage you to check the routing by yourself, so we will give extra credit if you do this. Please refer to the bonus part.

Submission:

Please submit a single **.tgz* file to CEIBA system. Your submission must contain:

1. <student_id>_pa3 directory contains your source code in *src* directory. By simply typing "make" can compile.
2. A report in the *doc* directory. <student_id>_pa3_report.doc. It's also ok for pdf file.
3. A README file that explains your files.
4. We will use our own test cases so do NOT include the input files.
5. In summary, you should have the following file in your directory,

   doc/<student_id>_pa3_report.doc
   lib/<all the lib files>
   src/<all your source code>
   bin/<your binary code>
   makefile
   README

The submission filename should be compressed in a single file <student_id>_pa3.tgz. (e.g. b90901000_pa3.tgz). You can use the following command to compress a whole directory:

```
tar -zcvf <filename>.tgz <dir>
```

You are required to run the checksubmitPA3 script to check if your .tgz submission file is correct. To use this script, simply type

```
./checkSubmitPA3.sh  b99901000_pa3.tgz
```

We have so many students in the class so we need automatic grading. Any mistake in the submission will result in cost 20% off your score. Please be very careful in your submission. `checkSubmitPA3.sh` will only check whether you have submit all the required files. You should validate your answer by yourself.

Grading:

60% correctness (including submission correctness)
20% file format and location
20% report

Bonus:

10% bonus will be given to writing a validation program to check the routing satisfies all requirement. Please describe how you check your routing clearly in your report.

NOTE:

Copying other source code can result in zero grade for all students involved.