National Taiwan University
Department of Electrical Engineering
Algorithms, Fall 2016

Handout #14
December 5, 2016
Yao-Wen Chang

Programming Assignment #3: Global Routing

(due 6pm, January 21, 2017 on-line;
a 10% bonus will be given for one submitted by 6pm, January 1, 2017)

Submission URL:

http://eda.ee.ntu.edu.tw/~lzw/alg16/submission/

Online Resources:

http://eda.ee.ntu.edu.tw/~lzw/alg16/resource/pa3/pa3.tar.gz

This programming assignment asks you to write a simplified global router that can route 2-pin nets (connection between two points). The problem description below is a simplified routing problem.

1. Input/output Specification

Input Format

The file format for the global routing problem is illustrated, with comments in italics (these comments will not be in actual input files). The 1st line gives the problem size in terms of the number of horizontal and vertical tiles. Each global routing tile (tile in short) has a *capacity* on its four boundaries to measure the available space, which is the maximum number of routing paths allowed to pass through the boundaries. The capacity value is given in the 2nd line. The 3rd line gives the number of nets, followed by line-by-line net descriptions, including the starting coordinate and the terminal coordinate. The input file format is as follows:

grid # # *//number of horizontal tiles, number of vertical tiles*
capacity # *//capacity of tile*
num net # *//number of nets*
net_id x_s y_s x_t y_t
...
//repeat for the appropriate number of nets

Output Format

All the routes in the output could run only either horizontally or vertically. For example, (18, 61)-(19, 62) is not acceptable, because it is diagonal. Remember that each route could be different either in the x or the y location only, and the difference must be 1. The output file format is as follows:

[net_id] [# of routes, k]
[$x_{1,1}$] [$y_{1,1}$] [$x_{1,2}$] [$y_{1,2}$]
[$x_{2,1}$] [$y_{2,1}$] [$x_{2,2}$] [$y_{2,2}$]
...
[$x_{(k-1),1}$] [$y_{(k-1),1}$] [$x_{k,2}$] [$y_{k,2}$]

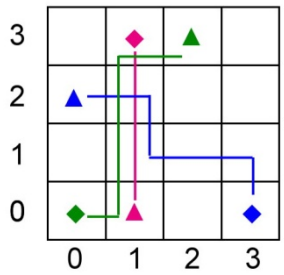
//repeat for the appropriate number of nets

Note that for a certain net, $x_{1,1}$, $y_{1,1}$, $x_{k,2}$, and $y_{k,2}$ must be the same as x_s, y_s, x_t, and y_t in the input file respectively. Also, for any *i*, $x_{i,2}$ and $y_{i,2}$ must be the same as $x_{(i+1),1}$ and $y_{(i+1),1}$ respectively.

2. Problem Statement

Given the problem size (the number of horizontal and vertical tiles), capacity, and a netlist, the global router routes all nets in the routing region. The main objective is to minimize total overflows. Here the overflow on a tile boundary is calculated as the amount of demand that exceeds the capacity, *i.e.*, overflow = max(0, demand - capacity). It is great if you can also minimize the total wirelength while minimizing the total overflows.

Sample case:



Sample input file:

```
grid 4 4
capacity 2
num net 3
0 2 3 0 0
1 0 2 3 0
2 1 0 1 3
```

Sample output file:

```
0 5
2 3 1 3
1 3 1 2
1 2 1 1
1 1 1 0
1 0 0 0
2 3
1 0 1 1
1 1 1 2
1 2 1 3
1 5
0 2 1 2
1 2 1 1
1 1 2 1
2 1 3 1
3 1 3 0
```

The total overflow is 1, which is caused by the boundary between tiles (1, 1) and (1, 2). (The

total wirelength is 13.)

3. Hints

You can first model the routing problem as a graph, where each node represents a tile and each edge denotes the tile boundary between tiles. The cost of an edge could be set to reflect the capacity usage (*e.g*, edge cost = demand/capacity). Then this problem can be solved by Dijkstra's shortest path algorithm. Note that different edge costs would result in different routing results; for example, you also can apply the edge cost as $2^{(\text{demand/capacity})}$-1. *You might want to try other cost metrics to see the effects.*

4. Required Files

You need to submit the following materials in a .tar.gz or .zip file: (1) Source codes (e.g. router.cpp); (2) Executable binary; (3) A text readme file describing how to compile and run your programs. The submission filename should be <student id>-<p3>.tar.gz or <student id>-<p3>.zip (e.g. b03901000-p3.zip). If you have a modified version, please add –v [version number] as a postfix to the filename and resubmit it to the submission website (*e.g.,* b03901000-p3-v2.zip, etc.).

5. Command-line Parameter

You have to name your executable "router" and add command-line parameters in your program to specify the input and output file name as the format:

```
[executable_file_name] [input_file_name] [output_file_name]
```

An example for running your command:

```
router gr5x5.in gr5x5.out
```

If your format is not the same as that given in the rule, you will receive a big penalty. By this format, your result must be written in the output file. Please DO NOT print the result on the screen (standard output).

6. Language/Platform

1. Language: C, C++, or Java.
2. Platform: Unix/Linux or Windows.

7. Evaluation

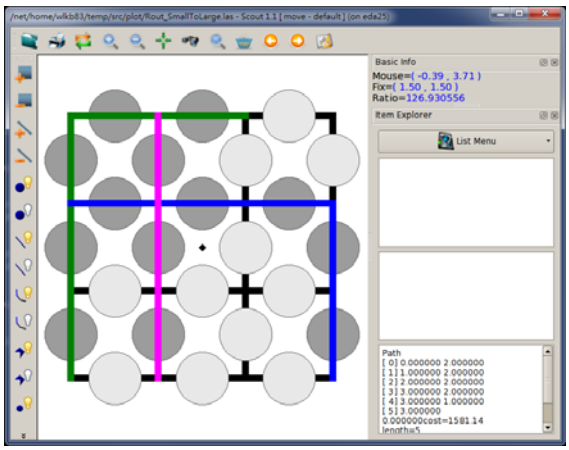
The individual baseline score per test case is determined by the correctness. A solution is correct if all nets are well-connected, i.e. no disconnection. On the other hand, the runtime is restricted to 2 hours for each test case which consists of at most 10,000 nets. A program fails a case if it is not efficient enough. There are more hidden test cases to evaluate your programs. Higher scores and bonuses will be given for high-quality programs. The quality is determined by the total overflows (tie is broken by the routed total wirelength).

8. Download

You are encouraged to use the following three tools for this programming assignment, all included in the file:
http://eda.ee.ntu.edu.tw/~lzw/alg16/resource/pa3/pa3.tar.gz

- ***Scout* (GUI for Display)**

  http://eda.ee.ntu.edu.tw/~lzw/alg16/resource/pa3/scout.tar.gz



A routing result is illustrated above by *Scout* for the sample case in Section 2. Notice that you are allowed to use *Scout* only for this assignment, due to the IP/copyright issue. Using this tool for any other application without advanced permission is subject to some legal issue!!

- ***AlgParser* (C++ API Parser)**

  http://eda.ee.ntu.edu.tw/~lzw/alg16/resource/pa3/pa3_parser.tar.gz

  To compile the sample program with ***AlgParser***, use command:

```
g++ -O2 main.cpp libparser.a -o parser
```

  To run the binary `parser`, use command:

```
./parser [input_file_name]
```

- ***Verify* (Verification Program)**

  http://eda.ee.ntu.edu.tw/~lzw/alg16/resource/pa3/pa3_verify.tar.gz

  To run the binary `verify_linux`, use command:

```
./verify_linux [test_case_file (*.in)] [your_output_file]
```