National Taiwan University of Science and Technology
Department of Electrical Engineering

Algorithm Design and Application, Fall 2020

Programming Assignment #2

Global Routing (due January 8, 2021 (Friday) on-line)

1. Problem Description

This programming assignment asks you to write a global router that can route 2-pin nets (connection between two points). The problem description below is a simplified routing problem. 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 the total overflows. Here the overflow on a tile boundary is calculated as the amount of demand that excesses the capacity, *i.e.*, overflow = max(0, demand-capacity).

2. Input

The file format for the global routing is illustrated, with comments in italics (these 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 passing through boundaries. The capacity value is given by the 2nd line. The 3rd line gives the number of nets and following indicate each net, including starting position and terminal position. 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

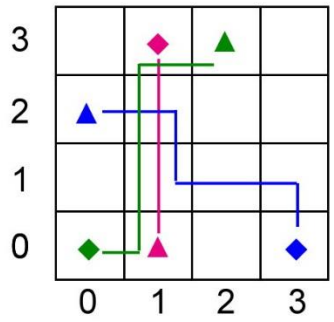
3. Output

All the routes in the output could only be horizontal lines and vertical lines. 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 y location only, and the difference must be 1.** The output file format is as follows:

[net_id] [# of routes, k]
[x_{11}] [y_{11}] [x_{12}] [y_{12}]
[x_{21}] [y_{21}] [x_{22}] [y_{22}]
...
[$x_{(k-1)1}$] [$y_{(k-1)1}$] [x_{k2}] [y_{k2}]
//repeat for the appropriate number of nets

Note that for a certain net, x_{11}, y_{11}, x_{k2} and y_{k2} must be the same as x_s, y_s, x_t and y_t in the input file respectively. Also, for any i, x_{i2} and y_{i2} must be the same as $x_{(i+1)1}$ and $y_{(i+1)1}$ respectively.

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.)

4. 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 $2^{(\text{demand/capacity})}-1$.

5. Language/Platform

(a) Language: C or C++.

(b) Platform: Unix/Linux or Windows.

6. Command-line Parameter

In order to test your program, you are asked to add the following command-line parameters to your program (e.g., routing.exe 5x5.in 5x5.out):

[executable file name] [input file name] [output file name]

7. Submission

You need to submit the following materials in a .tgz or a .zip file by following the naming rules highlighted in red (e.g., **m10907400-p2.tgz**) at the course website by the deadline: (1) A directory name src/ containing your source codes (**m10907400-p2.cpp**), (2) A directory name bin/ containing your executable binaries (**gr.exe**), (3) A makefile name makefile or Makefile that produces an executable binary from your source codes by simply typing "make": the binary should be generated under the directory bin/, and (4) a text readme file (**readme.txt**) stating how to build and use your programs. Please check these items before your submission.

8. Grading Policy

This programming assignment will be graded based on (1) the correctness (a solution is correct if all nets are well-connected, i.e. no disconnection), (2) solution quality (The quality is determined by the total overflows, and tie is broken by the total routing wirelength), (3) running time (the runtime is restricted in 1 hours for each case), and (4) required submission files with correct file names.

8. Online Resources

Sample input files (*.in) and sample readme.txt can be found at the course website.