Assignment 1 Report

Dingqing Yang 38800141

2020 Janurary

1 Discussion on Aspects of the Marking Criteria

- *Basic Lee-Moore*: The Lee-Moore shortest path algorithm is implemented in `shortestPath` method in `Router` class. The score map is store in `label` member in the class. Neighbour expansion and backtrack are implemented in `expand` and `backTrack` method. The result of basic Lee-Moore is tested based on inspecting the status of the `label` member.
- *A* Algorithm*: I did not attempt this part.
- *Multiple Sinks*: This is achieved by changing `shortestPath` method's input argument from a single source to a list of sources. The list of source point are in fact the result of routed segements in the same wire. Before neighbour expansion, all source pin are assigned a score of 1. One good test case for this is the wavy benchnark.
- *Graphics*: I use python tkinter package. GUI only support display the final result.
- *Code Quality*: I implement a `Router` class that contains methods that work for GUI and the routing algorithm. Each method in the class is documented.
- *Initiative*: In order to achieve the maximum number of routed segments, I choose to iteratively route and reorder nets. The best routing result are selected among all attempts. I investigate how to order nets in a way that maximizes the number of segments routed. I find that the following nets should be route with high priority:
  - Eazy-to-route nets
  - Previously failed nets

  Easy-to-route nets are connected with high priority because the greedy nature of the problem. To determine if a net is easy-to-route, I use the average L1 distance among all pairs within a net. Previously failed nets should increase priority because we want to improve the routing bottleneck.

Benchmarks	**Number of Segments Connected**
impossible	3
impossible2	3
kuma	5
misty	5
oswald	1
rusty	4
stanley	5
stdcell	18
sydney	3
temp	15
wavy	7

Table 1: Connected Segments for all provided benchmarks using Lee-Moore

2 Table of Routed Segments

Table 1 shows number of connected segments for all benchmarks using Lee-Moore shortest path algorithm. I assume that the definition of a segment is a source-target pair. Result in this table can be reproduced by running `python test_all.py`

Most of them should achieve the optimal number of connection possible. I am not able to achieve optimal result for oswald because the order of backtracking direction is fixed (e.g. always attempting to go left before attempting to go right). This may get fixed if some heuristic can be applied. One possibility is to favour stick on boundary if the pin is on boundary already.

3 Plot for *stanley* and *stdcell*

Figure 1 and figure 2 show the final routing results for *stanley* and *stdcell* benchmarks. Both of them successfully route all connections.

4 How to run it

Examples:

- To launch GUI and show final result for *stdcell.infile*, run `python routing.py --infile benchmarks/st`
- Test all benchmarks without launching GUI, run `python test_all.py`

See README for more.

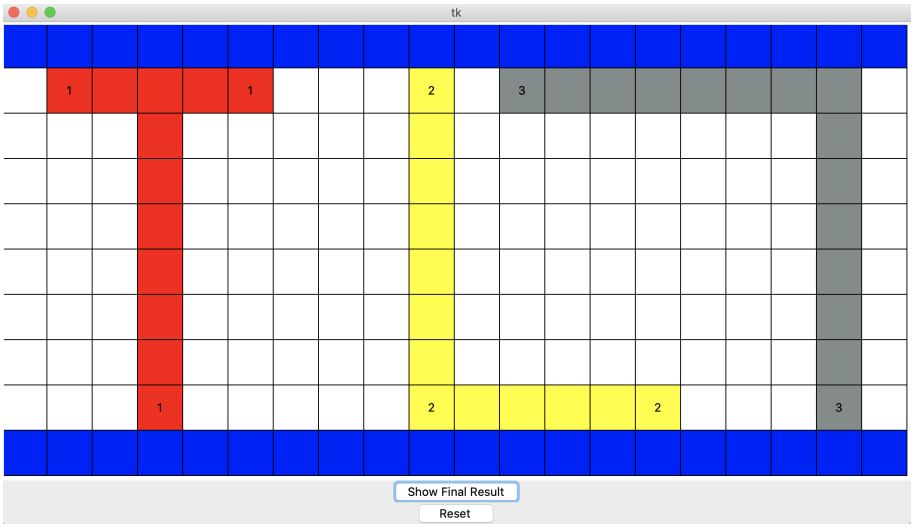

Figure 1: Stanley Result

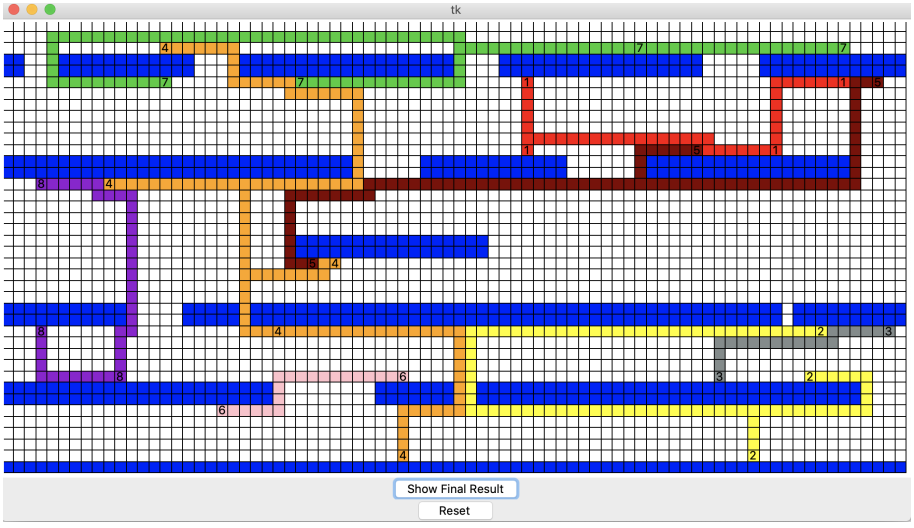

Figure 2: Stdcell Result