Stephen Chambers
CS858
April 18th, 2016

Assignment 11 Writeup

1. **Briefly list any parts of your program which are not fully working. Include transcripts showing the successes or failures. Is there anything else that we should know when evaluating your implementation work?**
   My program works correctly. Here is a transcript of the harness agreeing.

   [smx227@agate a11]$ ./flow-harness --size 5
   flow magnitude matches with a value of 11
   done validating capacity constraints
   [smx227@agate a11]$ ./flow-harness --size 10
   flow magnitude matches with a value of 44
   done validating capacity constraints
   [smx227@agate a11]$ ./flow-harness --size 14
   flow magnitude matches with a value of 58
   done validating capacity constraints
   [smx227@agate a11]$ ./flow-harness --size 40
   flow magnitude matches with a value of 207
   done validating capacity constraints
   [smx227@agate a11]$ ./flow-harness --size 100
   flow magnitude matches with a value of 503
   done validating capacity constraints

2. **Exercise 26.1–6 in CLRS.**
   Consider this problem as a directed graph, where each vertex is a corner in the town, and each edge connects the corners. Let the source vertex be Professor Adam's house and let the sink vertex be the school. Let each edge weight be 1, signifying that only one student can cross this edge. If the max flow is >=2, the students can find paths to walk on without walking on the same edge.

3. **Exercise 26.2–6 in CLRS. Don't forget to prove that your construction is correct!**
   To solve this problem as a single source single sink max flow problem, assuming we have a graph with the given constraints, add a 'super source' with infinite capacity that has an edge to all of the sources, and add a 'super sink' with infinite capacity that has an edge to all of the sinks. The solution is provably correct because the sum of the super source to all vertexes in the graph is Pi, and the sum of all vertexes of the graph to the super sink is Qi. If this were not the case, there may be a vertex that was left out of the maximum flow, causing a proof by contradiction.

4. **Part a of problem 26–1 in CLRS.**
   To reduce a problem with edge and vertex capacity constraints into a normal flow problem of a comparable size, we must use the important property that the graph is *undirected*.

   so for a particular vertex *v*, we would have two new vertices *vin* and *vout.* An edge (*vin*, *vout*) can then be introduced between *vin* and *vout* with the capacity of the original vertex before the split.

   Now we have a graph that can be solved as a normal max flow problem, but with edge and vertex capacity constraints.

5. **(Those in 858 only) Part b of problem 26–1 in CLRS.**
   First, have a 'super source' have an edge between each *m* starting point, and have a 'super sink' have an edge between each boundary point. Then, for each edge (u,v), change each undirected edge in the graph to two directed edges (u,v) with capacity 1 and (v,u) with capacity 1. Use edmonds-carp to determine the max flow of the after the changes made above. If the max flow is equal to *m*, there exists an escape path. If not, there does not exist an escape path. Changing the original graph takes O(V) time, but this is dominated by the running time of Edmonds-Carp.

   The time complexity of all Ford-Fulkerson based algorithms with integer weights is O(Ef*). The maximum flow for this graph is equal to the number of boundary points in the grid, because the max flow possible is if each *m* point when into each boundary point. Therefore, the maximum flow is bounded by *n*. Additionally, the number of edges is bounded by n^2. Therefore, the time complexity for this algorithm is n^3.

6. **What suggestions do you have for improving this assignment in the future?**
   Graphviz is software that visualizes graphs given a specific format, and it was invaluable in debugging my residual graphs. I'm quite familiar with python, and I am willing to include it in the harness as an option for students in future classes. Here's an example graph. You can color paths in the graph, and it may be helpful for visualizing flows.