CS 456: Advanced Algorithms

Programming Assignment #03

Assigned Date : Tuesday, November 18, 2014
Due Date : Thursday, December 04, 2014 @ 12:29:59 p.m.

Objectives

Your third (and final) programming assignment is to **implement Ford-Fulkerson algorithm** to solve the **maximum edge-disjoint paths problem** in a flow network. Note that, by considering unit capacities for each edge $(u, v) \in E$ of a flow network, the maximum flow problem morphs into a maximum disjoint path problem; the maximum flow in such a network defines the maximum number of edge-disjoint (*no two paths may have common edges, but may contain common vertices*) paths from source $s \in V$ to sink $t \in V$. However, this assignment counts only the disjoint paths in the **core network** – the network resulting from excluding the source s and the sink t from the original flow network. Hence, the objectives of this assignment are:

1. To implement Ford-Fulkerson max-flow algorithm;
2. To verify (using empirical evidence) the max-flow min-cut theorem; and
3. To find the maximum number of edge-disjoint paths from $s \rightarrow t$ for a given arbitrary flow network.

You are expected to properly demonstrate and justify that you were able to achieve these objectives through documented work listed in your project report.

Instructions

- This is an individual assignment. **Do your own work**.
- **Start early!!**
- **Take backups of your code often!!**.
- Read all instructions provided here thoroughly and carefully.
- The report part of your solution must be produced using a word processor. Any figures, graphs, plots, etc., should also be produced using appropriate computer applications. Graphs/plots should be properly labeled.
- You may use **any programming language** of your choice. However, you should make sure that your code compiles and runs on a typical Linux machine.
- Your report must be in **PDF** format.
- Follow a good coding standard. Use the CS department programming styling guide found here https://www.cs.siue.edu/programming-style-guide.
- You are required to make your submission through Moodle. A dropbox will be opened within a 24 hour window of the deadline and will continue to be open for an additional 24 hours. There is

no extra credit for submitting early but there will be a 15% penalty for being late. Please carefully read the Deliverables section below for more details.

- Total points: [**100 points**]

I/O Specifications

Your program should be capable of reading an input file that describes a graph using the following format:

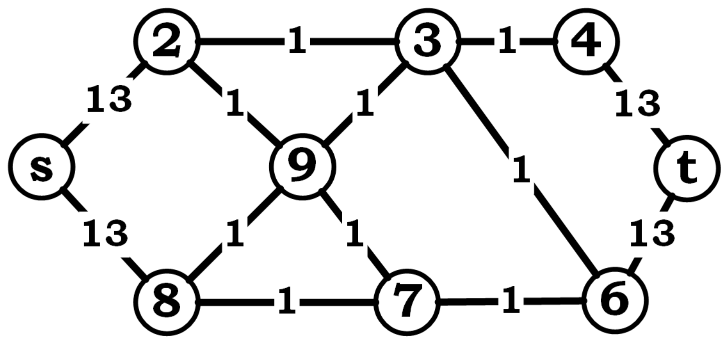
```
<source-vertex> <destination-vertex> <capacity>
```

While it is not mandatory, you may consider having the first line of your input file to list the total number of vertices and edges of your graph, if such a listing helps you create your data structures. You are not to assume any particular ordering of the edge listings, but may assume that the edge capacities of the core network to be integral. More precisely, edge capacity $c(u,v)$ is defined as follows:

$$c(u,v) = \begin{cases} 1 & \forall(u,v) \in E, \quad u,v \in V - \{s,t\} \\ 1 & \forall(s,t) \in E \\ |E| & \text{otherwise} \end{cases}$$

Any <u>direct</u> edges between the source s and t are considered a core network edge with capacity 1.

Your program should use this input file to calculate the minimum number of edge-disjoint paths from the first level of the BFS tree for the given graph and generate an output file that lists the total number of paths found, followed by each path in the following format: $s-->u-->v-->x-->t$ per line. Furthermore, your output file should list the time that it took your program to find the total number of paths.



Input.dat

```
9 13
1 2 13
2 3 1
3 4 1
4 5 13
1 8 13
3 9 1
7 8 1
7 9 1
8 9 1
2 9 1
5 6 13
6 7 1
3 6 1
```

Output.dat

```
Time   : x seconds
Paths  : 3
         s → 2 → 3 → 4 → t
         s → 2 → 9 → 3 → 6 → t
         s → 8 → 7 → 6 → t
```

As an example, consider the following undirected graph and its corresponding edge list `Input.dat` and `Output.dat` files.

The instructor will independently verify the programmatical correctness of your submitted solution using his own test file that has a reasonably large flow network. Thus, you are also not to assume any particular size limitation of the graph your program can handle.

Deliverables

The due date of this assignment is **Thursday, December 04, 2014 @ 12:29:59 p.m.** A dropbox will be opened for submission on Moodle before the due date. A complete solution comprises of:

- A report **in PDF format** that includes the followings:
  - **Motivation and Background :** Use this section to explain succinctly your approach to achieving the objectives listed above [**10 points**]
  - **Description :** A pseudocode listing of your program. If you are using classes, make sure to also include an appropriate UML class interaction diagram [**5 points**]
  - **Testing Plan :** In particular, what you consider to be the true runtime complexity of the algorithm and how you plan to separate that from auxiliary tasks such as input processing and output file generation. You must properly justify any assumptions you make. [**10 points**]
  - **Results and Analysis :**
    - **Theoretical Results**: A description of the expected asymptotic behavior of your program. You may (and probably are advised) use your pseudocode to aid this explanation [**15 points**]
    - **Empirical Results**: Observations from your experiments. You should probably repeat each experiment several times (5-10) to eliminate any statistical errors, and list the outcomes of each run in tabular format. Also, you are expected to produce output for at least three problems – a smaller size problem ($E \leq 100$), a medium size problem ($100 \leq E \leq 1000$), and a large size problem ($E > 1000$) . [**15 points**]
  - **Analysis :** Show sufficient evidence to show that your implementation achieves the (or bare minimum mimics) the expected theoretical behavior. You are required to provide a comparative comparative plot between theoretical expectation and empirical outcome vs. input (in this case E), and to properly explain your observations. [**20 points**]
  - **Conclusions :** Describe your observations and conclusions of your experiment including any problems encountered, limitations, and/or any key insights you've discovered. [**10 points**]
  - **Instructions :** How to compile and run your program [**5 points**]
  - **Program listing** [**5 points**]
    - Good programming structure (headers, variable names, code re-use, functional decomposition, object-oriented design, and comments)
  - References (if any)
- A compressed tarball of the directory containing your source codes. Do not include executables, test input or sample outputs in this tarball. I will generate them using my own test input files. To create a compressed tarball of the directory source, use the following command: `tar -zcvf name111-pr3.tar.gz source/`. Obviously, change the name to your last name and 111 to the last three digits of your SIUE ID [**5 points**].

Finding large datasets to experiment will require you to either research what's out there or generate them using your own code. If you do use resources other than your own, make sure to properly cite them in your report and give due credit to the original authors. By absolutely no means, you are allowed to plagiarize code. DO NOT use code you find online.

Here are couple of resources to look for large datasets. You may have to parse them to the input specification described above. Use your scripting skills (or write an appropriate parser). If you use an outside resource to do the parsing for you, make sure to cite the source.

- SNAP Dataset found at http://snap.stanford.edu/data/

- Pajek Dataset found at http://vlado.fmf.uni-lj.si/pub/networks/data/

MS requirement [20 points]

List and explain at least one more application of maximum flow.