CSC190: Computer Algorithms and Data Structures
Lab 7
Assigned: Mar 23, 2015; Due: Mar 30, 2015 @ 10:00 a.m.

1 Objectives

In this lab, you will implement all interface functions required to create a graph representation of a flow network and perform operations on this graph. All functions implemented in this lab can be used to fulfill the requirements of Part 1 of Assignment 3. Please refer to Assignment 3's instruction manual for more information on flow networks.

You will download content in the `Lab7` folder which contains two sub-folders (`code` and `expOutput`) into your ECF workspace. Folder `code` contains a skeleton of function implementations and declarations. Your task is to expand these functions appropriately in `adjMatrix.c` and `lab7.c` as required for implementation. `main.c` evokes all the functions you will have implemented and is similar to the file that will be used to test your implementations. Use `main.c` file to test all your implementations. Folder `expOutput` contains outputs expected for the supplied `main.c` file. Note that we will **NOT** use the same `main.c` file for grading your lab. Do **NOT** change the names of these files or functions.

2 Grading

It is **IMPORTANT** that you follow all instructions provided in this lab very closely. Otherwise, you will lose a *significant* amount of marks as this lab is auto-marked and relies heavily on you accurately following the provided instructions. Following is the mark composition for this lab (total of 20 points):

- Successful compilation of all program files with no memory leaks i.e. the following command results in no errors (2 points):
  ```
  gcc adjMatrix.c lab7.c main.c -o run
  valgrind --quiet --leak-check=full --track-origins=yes ./run
  ```
- Output exactly matches expected output (10 points)
- Code content (8 points)

Sample expected outputs are provided in folder `expOutput`. We will test your program with a set of completely different data files. Late submissions will **NOT** be accepted.

Interface Functions for Flow Network Graph Representation

You will implement interface functions in `adjMatrix.c` and `lab7.c` files for performing operations on a flow network graph. For this lab, you will represent a flow graph via an **adjacency matrix**. One structure definition provided in the `lab7.h` file that is to be used for the adjacency matrix definition is `struct Edge`. This structure has the following members:

- `int flow;`
- `int flowCap;`

`struct Edge` represents an edge and stores the current flow in that edge and the maximum capacity of that edge in the `flow` and `flowCap` members respectively. The adjacency matrix is an n by n matrix in which each element is of type `struct Edge`. The i, j entry of the adjacency matrix represents the edge connecting v_i and v_j. Flow on this edge is $f_{i,j}$ and the capacity of this edge is $c_{i,j}$. These metrics are stored in the `flow` and `flowCap` members. There are three interface functions you will implement first that will initialize, insert elements into and delete an adjacency matrix:

- `struct Edge ** initAdjMatrix();`

  - In this function, a 2-D matrix of type `struct Edge **` should be dynamically created
  - Since every matrix entry is of type `struct Edge`, the `flow` and `flowCap` members of each entry should be initialized to 0
  - The double pointer pointing to the dynamically allocated matrix should be returned by the function

- `void insertAdjMatrix(struct Edge ** aM, int vi, int vj, int flow, int flowCap);`
  - Suppose the flow network contains an edge (v_i, v_j) with a flow of `flow` and edge capacity of `flowCap`
  - This function should access the entry `aM[i][j]` corresponding to the edge (v_i, v_j) and set the members of this entry in accordance to the parameters `flow` and `flowCap` passed to this function

- `void printAdjMat(struct Edge ** aM);`
  - This function prints the flow and capacity of an edge if that edge has a capacity greater than 0 (please refer to the files in the `expOutput` folder for details on formatting)

- `void deleteAdjMatrix(struct Edge ** aM);`
  - In this function, the dynamically allocated adjacency matrix `aM` should be freed

Next, a set of functions that initialize and perform operations on a flow network are to be implemented:

- `struct flowNetwork * initFlowNetwork();`
  - This function will dynamically allocate a `struct flowNetwork` variable and initialize its members `adjMatrix`, `visitedNodes` and `parent`
  - `adjMatrix` is the graph representation of the flow network and is to be initialized using the `initAdjMatrix` function
  - `visitedNodes` is an array that keeps track of vertices that are visited in the graph and all elements are to be initialized to 0
  - `parent` is an array that is to be used by the path-finding algorithm and all elements are to be initialized to -1

- `void deleteFlowNetwork(struct flowNetwork * fN);`
  - All dynamically allocated members in `fN` are to be freed in this function

This lab will be tested via the following commands:

- `./run`
- `valgrind --quiet --leak-check=full --track-origins=yes ./run`

Outputs from these tests must match the content of `lab7.txt` which is the result of the parameters passed to function calls in `main.c`.

3 Code Submission

You can submit via git or the `submitcsc190s` command on your ECF machine (no bonus points for submissions either way). Ensure that you submit through only one venue.

3.1 Submission through `Git`

Once you have completed this lab, you will submit your work by:

- Log onto your ECF account
- Browse into the directory you had cloned in Lab 0 (i.e. `cd ~/UTORID/`)
- Create a folder named `Lab7` (i.e. `mkdir Lab7`) in that cloned directory
- Ensure that your code compiles in the ECF environment
- Copy all your completed code (`adjMatrix.c` and `lab7.c`) into the Lab7 folder
- Browse into the `~/UTORID/` directory
- Add all files in the Lab7 folder (i.e. `git add *`)
- Commit all files that have been modified in the Lab 7 folder (i.e. `git commit -m "adding lab files"`)
- Push all changes committed to the `git` server (i.e. `git push origin master`)

3.2 Submission through `submitcsc190s`

- Log onto your ECF account
- Ensure that your completed code compiles
- Browse into the directory containing your completed code (`adjMatrix.c` and `lab7.c`)
- Submit by issuing the command:
  `submitcsc190s 10 adjMatrix.c lab7.c`

3.3 Checklist

ENSURE that your work satisfies the following checklist:

- You submit before the deadline
- All files and functions retain the same original names
- Your code compiles without error in the ECF environment (if it does not compile then your maximum grade will be 3/20)
- Do not resubmit any files in Lab 7 after the deadline (otherwise we will consider your work to be a late submission)