Assignment 11: Network Flow
CS 758/858, Fall 2024
Due at **11:30pm on Mon Nov 11**

Implementation

The skeleton code on the course web page is the start of a program for computing maximum flows in networks. Complete it by implementing any variation of Ford-Fulkerson that you wish. (We recommend Edmonds-Karp.)

Networks are given on standard input. The first lines gives the number of nodes, the second gives the number of edges, and the remaining lines give the edges, one per line. Edges are specified by the start vertex, end vertex, and capacity. An example:

```
4
6
0 1 1
1 2 3
2 3 2
3 1 2
2 1 3
0 3 2
```

Your program must write the flow to standard output as a sequence of edges, one per line, followed by the value of the flow. Each edge is specified by the start vertex, end vertex, and amount of flow. An example:

```
0 3 2
0 1 1
3 1 2
3
```

Node 0 is always the source and node 1 is always the sink.

You will probably want to eliminate antiparallel edges. This is fine, but you will need to make sure your program prints out something that corresponds to the original graph. This means that if you elect to add nodes to the graph, if your final flow goes through nodes you added, you will need to eliminate the extra nodes. You are free to, and probably should, add things that you feel are necessary to the structs defined in the skeleton code.

A complete example:

```
cmo66@zelinka:~/cs858/flow$ cat chris.flow
7
8
0 2 5
0 3 7
3 4 7
2 6 10
4 6 7
4 5 6
5 1 6
6 1 7
cmo66@zelinka:~/cs858/flow$ ./flow < chris.flow
0 3 7
0 2 5
2 6 5
3 4 7
4 5 5
4 6 2
5 1 5
6 1 7
12
```

Testing

We provide `flow-check`, which generates a random network, calls your code, and verifies that the flow you return is feasible. For example:

```
./flow-check --binary ./flow --ref ./flow-reference --size 100 --seed 21
```

Changing the seed and size will change the graph the harness generates. You can also specify a network from a file:

```
./flow-check --graph simple.network --ref ./flow-reference --binary ./flow
```

For your amusement, we also provide `segment`, which performs image segmentation using your flow program. Some example input images can be found in `/home/cs758/data/asn11` on `agate`. To see the UI, you will need to 'forward' the harness program's graphics commands to an 'X server' on your local machine. Some instructions on doing this are available at `https://kb.iu.edu/d/bdnt` (on linux, `ssh -X agate.cs.unh.edu` should work).

To run the harness:

```
segment -in path/to/input/image -sf path/to/your/max_flow/executable
```

Select 5 foreground pixels and then 5 background pixels by clicking on the image. The selected pixel locations will be displayed in the stdout. Then click again on the image to start the max-flow algorithm. If it ran successfully, it will display the foreground and background images along with the original.

Currently there are two methods implemented to determine similarity between pixels: boundary penalty and Euclidean distance between rgb values. By default, the harness uses boundary penalty, but Euclidean distance can be selected using the `-gc 2` flag. For boundary penalty, two additional parameters may be chosen via the `-ca` and `-cb` flags. The default values seem to produce ok results but feel free to experiment, as in:

```
segment -in path/to/input/image -sf path/to/your/max_flow/executable -gc 1 -ca 200.0
-cb 30.0
```

Full usage info for the harness is available with the `-h` flag. Note that the harness won't like your debug messages.

Written Problems

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?
2. Exercise 24.1–6 in CLRS.
3. Part a of problem 24–1 in CLRS.
4. (Those in 858 only) Part b of problem 24–1 in CLRS.
5. Exercise 29.2–1 in CLRS.
6. Exercise 29.2–7 in CLRS.
7. What suggestions do you have for improving this assignment in the future?

Submission

Electronically submit your work using the script on agate (eg, `~cs758/scripts/sub758 11-undergrad your-asn11-dir`).

In addition to correctness, your work will be evaluated on clarity and efficiency. Tentative breakdown:

7 flow implementation (6 for grads)

3 written problems (4 for grads)