CPSC 420+500 Problem Set 3 7 Feb 2017
W. Evans Due: ~~15~~ 17 Feb 2017 at 21.00 (9pm)

Submission Instructions

Handin your solutions using `handin`. You can write your solutions by hand and scan the pages or take pictures of them with your phone; or use a word processing package to typeset your solutions. Whatever you do, you should produce **pdf** files **assign3Q1.pdf**, **assign3Q2.pdf**, etc. containing your solutions for Question 1, Question 2, etc.

To handin: Copy your solution files to the directory `~/cs420/a3` in your home directory on a CS undergraduate machine. (You may have to create this directory using `mkdir ~/cs420/a3`.) Then run `handin cs420 a3` from your home directory.

If you do not have a CS department account, you can email your assign1.pdf to me `will@cs.ubc.ca`.

Late submissions are not accepted.

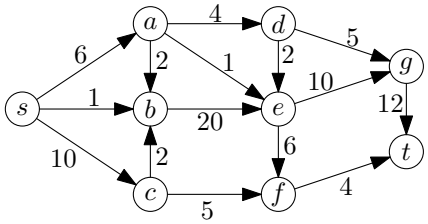
Grading Policy

We will grade a subset of these questions of size at least two. It's a good idea to do all of the questions because (1) you don't know which ones we'll grade and (2) answering these questions is good practice for the exams. We will email feedback to the email address associated with the account you used to handin the assignment.

Questions

Try to answer these on your own but if you work with someone or use an outside source you must acknowledge them in your write-up.

1. (Problem 7.10 Algorithms by Dasgupta, Papadimitriou, Vazirani) For the following network, with edge capacities as shown, find the maximum flow from s to t, along with a matching cut.



2. (From Problem 23-12 Algorithms Course Notes by Erickson) The GreedyFlow algorithm is a simplification of the generic Ford-Fulkerson augmenting path algorithm that doesn't use a residual graph. Instead, after finding an augmenting path from s to t, it just reduces the capacity of every edge on the path by the capacity of the smallest capacity edge on the path, removing edges with zero capacity.

GreedyFlow(G, c, s, t):
    for every edge e in G
        $f(e) = 0$

```
while there is a path from s to t
    π = an arbitrary path from s to t
    b = minimum capacity of any edge in π
    for every edge e in π
        f(e) = f(e) + b
        if c(e) = b then remove e from G
        else c(e) = c(e) − b
return f
```

Show that this algorithm does *not* always compute a maximum flow.

3. Let a and b be two vertices in a directed graph G. Give an algorithm that calculates the number of *vertex disjoint* paths from a to b. A set of paths from a to b is vertex disjoint if no two paths in the set share a common vertex, other than a and b. Explain why your solution works.

   Hint: Formulate the problem as a network flow. Suppose we add unit capacities to all the edges in G and make a the source and b the sink. Why isn't the max flow in this flow network the answer? How should we construct a flow network from G so that the max flow is the answer?

4. (Problem 7-22 Algorithm Design by Kleinberg and Tardos) Let M be an $n \times n$ matrix with each entry equal to either 0 or 1. Let m_{ij} denote the entry in row i and column j. A *diagonal entry* is one of the form m_{ii} for some i.

   Swapping rows i and j of the matrix M means swapping m_{ik} with m_{jk} for all $k = 1..n$. A similar definition holds for swapping columns.

   M is *rearrangeable* if by swapping rows and/or swapping columns of M one can make all diagonal entries be 1.

   (a) Give an example of a matrix M that is not rearrangeable even though every column and every row contains at least one entry that equals 1.

   (b) Describe an efficient algorithm (running in polynomial time) that determines if M is rearrangeable.

5. (Problem 24-4 Algorithms Course Notes by Erickson) A *cycle cover* of a given directed graph $G = (V, E)$ is a set of vertex-disjoint cycles that cover all the vertices. Describe and analyze an efficient algorithm to find a cycle cover for a given graph, or correctly report that no cycle cover exists. *[Hint: Use bipartite matching!]*