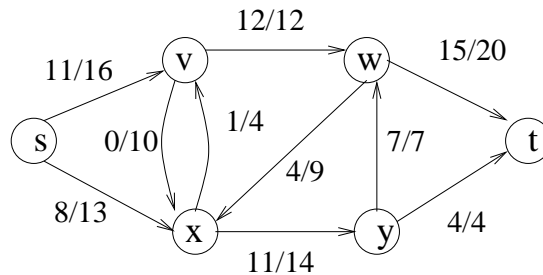


Algorithms and Data Structures 2011/12

Week 9 tutorial sheet (Tues 15th - Fri 18th Nov)

Below are a list of *suggested* exercises. You should also see the tutorial as a resource to get answers to questions you have, don't feel compelled to stick to the sheet.

- Consider the flow network below (for each arc of the network, the first number is the current flow along that arc, and the second number is the capacity of the arc). We write f for the flow function.



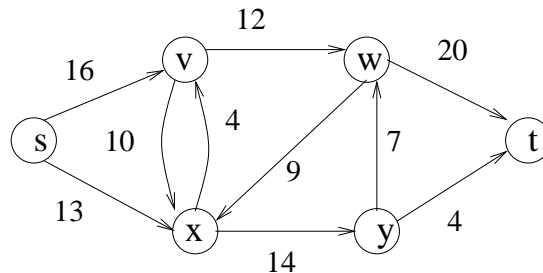
Two questions:

- Find a pair of subsets $X, Y \subseteq V$ such that $f(X, Y) = -f(V - X, Y)$.
- Find a different pair of subsets $X, Y \subseteq V$ such that $f(X, Y) \neq -f(V - X, Y)$.

Hint: This does not have as much to do with this particular network as you might initially think.

This is Ex. 26.1-5 of [CLRS] (ed 2). Ex. 27.1-5 of [CLR].

- Execute the Ford-Fulkerson algorithm (*using the Edmonds-Karp heuristic*) on the Network below:



(notice that this is the same network as in question 1, except in this case, there is no flow constructed yet).

This is Ex. 26.2-2 [CLRS] (ed 2). Similar to Ex 26.2-1 of (ed 3). Ex. 27.2-2 of [CLR].

3. This question considers the scenario of *dynamically-evolving* capacities in a network. We start with a fixed network $\mathcal{N} = (\mathcal{G} = (\mathbf{V}, \mathbf{E}), \mathbf{c}, \mathbf{s}, \mathbf{t})$, and compute a maximum flow f for \mathcal{N} . However at any time in the future we may receive notification that the capacity of a particular arc in the network is being increased by 1 (or decreased by 1). Two questions:
- Suppose that the capacity of a single edge $(\mathbf{u}, \mathbf{v}) \in \mathbf{E}$ is increased by 1. Give a $O(|\mathbf{V}| + |\mathbf{E}|)$ -time algorithm which updates f to obtain a max flow f' for the updated network.
 - Suppose that the capacity of a single edge $(\mathbf{u}, \mathbf{v}) \in \mathbf{E}$ is decreased by 1. Give a $O(|\mathbf{V}| + |\mathbf{E}|)$ -time algorithm which updates f to obtain a max flow f' for the updated network.

This is Prob 26-4 of [CLRS] (eds 2 & 3). Prob 27-4 of [CLR].

4. A well-known problem in graph theory is the problem of computing a *maximum matching* in a *bipartite graph* \mathcal{G} . Give an algorithm which shows how to solve this problem in terms of the network flow problem.

Definitions:

A (undirected) graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is *bipartite* if we have $\mathbf{V} = \mathbf{L} \cup \mathbf{R}$ for two disjoint sets \mathbf{L}, \mathbf{R} , such that for every edge $\mathbf{e} = (\mathbf{u}, \mathbf{v})$ exactly one of the vertices \mathbf{u}, \mathbf{v} lies in \mathbf{L} , and the other in \mathbf{R} .

A *matching* in an (undirected) graph \mathbf{G} is a collection \mathbf{M} of edges, $\mathbf{M} \subseteq \mathbf{E}$, such that for every vertex $\mathbf{v} \in \mathbf{V}$, \mathbf{v} belongs to *at most one* edge of \mathbf{M} .

A *maximum matching* is a matching of maximum cardinality (for a particular graph).

5. If you have any issues understanding Coursework 2, you can discuss these a bit with your tutor.

Mary Cryan