University of Toronto at Scarborough
CSCC73 - Algorithm Design and Analysis

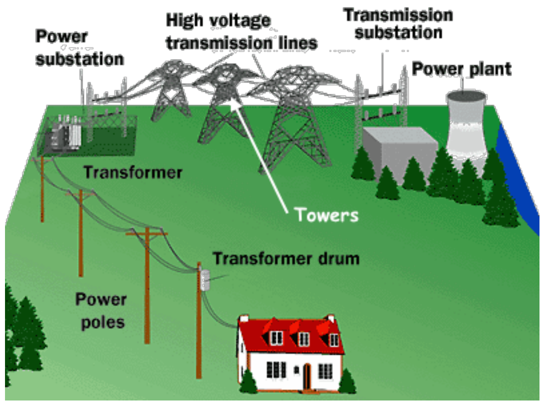
Assignment #6: Network Flow

Due: November 15, 2018 at 11.59pm This exercise is worth 5% of your final grade.

Warning: Your electronic submission on MarkUs affirms that this exercise is your own work and no one else's, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism in CSCC73. Late assignments will not be accepted. If you are working with a partner your partners' name must be listed on your assignment and you must sign up as a "group" on MarkUs. Recall you must not consult **any outside sources except your partner, textbook, TAs and instructor**.

1. (10 marks) After a large winter storm, your local power company's network is down. They need to diagnose the extent of a failure in the power grid to get it and their customers back on the grid.

   The network is designed to carry power from the power plant which is designated as a source node s to a main power station (denoted t) with many power substations along the way. The transmission lines are carried by towers.



   We will model the network as a directed graph $G = (V, E)$, in which the capacity of each edge is 1 and in which each node is a tower or power substation and lies on at least one path from s to t.

   When everything is running smoothly in the grid, the maximum s, t-flow in G has value k. Unfortunately, some lines between towers or substations have gone down in the storm so that there is now no path from s to t using the remaining (surviving) edges. The assumption is that the storm only destroyed k edges, the minimum number needed to separate s from t (i.e., the size of a minimum s, t-cut); and we'll assume they're correct in believing this.

   It is very costly to send out a work crew to check on every tower or substation to assess the lines between them (edges). Fortunately, in anticipation of natural disasters, they have set up a monitoring tool at the power station (ie, node s), which has the following behaviour. If you issue the command $pulse(v)$, for a given node v, it will tell you whether there is currently a path from s to v. (So, $pulse(t)$ reports that no path currently exists; on the other hand, $pulse(s)$ always reports a path from s to itself.) To speed repairs, they'd like to locations of the downed lines using this monitoring tool to determine where to send their work crews. In addition as the grid is huge, they need to efficiently use the $pulse$ command.

   The problem: Given the original network, provide an algorithm that issues a sequence of $pulse$ commands to various nodes in the network and then reports the *full* set of nodes that are not currently reachable from s. This could be done by sending a pulse to every node in the network, but they would like to do it using many fewer pulses (given the assumption that only k edges have been deleted). In issuing this sequence, the algorithm is allowed to decide which node to pulse next based on the outcome of earlier $pulse$ operations.

Give an algorithm that accomplishes this task using only $\mathcal{O}(k \log n)$ pulses.

2. (10 marks) Suppose you're looking at a flow network G with source s and sink t, and you want to express the intuition that some nodes are clearly on the "*source side*" of the main bottlenecks; some nodes are clearly on the "*sink side*" of the main bottlenecks; and some nodes are in the middle. However, G can have many minimum cuts, so we have to be careful in how we make this idea precise.

   - We say a node v is *upstream* if, for all minimum s, t-cuts (A, B), we have $v \in A$–that is, v lies on the source side of every minimum cut.
   - We say a node v is *downstream* if, for all minimum s, t-cuts (A, B), we have $v \in B$–that is, v lies on the sink side of every minimum cut.
   - We say a node is *central* if it is neither *upstream* nor *downstream*; *i.e.*, there is at least one minimum s, t-cut (A, B) for which $v \in A$, and at least one minimum cut (A^*, B^*) for which $v \in B^*$.

   Give an algorithm that takes a flow network G and classifies each of its nodes as either upstream, downstream, or central. The running time of your algorithm should be within a constant factor of the time required to compute a *single* maximum flow. You should carefully explain why your algorithm works.