CO 602: Fundamentals of Optimization
Fall 2010
Problem Set 7
S. Vavasis

Handed out: 2010-Nov-12.

Due: 2010-Nov-19, in lecture.

1. Implement Prim's algorithm for min-weight spanning tree using a priority queue. The running time should be $O(m \log n)$, where m is the number of edges and n the number of nodes.

   Here is a suggested approach. First, transform the graph into a digraph by replacing each edge with two arcs. Then call `makeoutarcs` and follow a strategy very much like Dijkstra's algorithm. Use the `priorityQueue` data structure. The main difference from Dijkstra's and Prim's algorithm is the formula for computing and updating keys. An additional issue is that your program needs to safeguard against reinserting a node into the heap after it has already been added to N_T. The program should set return argument treeEdges to be empty if the graph is disconnected.

   Hand in a listing of your code and its output on the testcases sevennode.mat and eightnode.mat posted on the course webpage. Your program should have the following header.

```
function [cost,treeEdges] = prim(nnode, edgelist, dist)
% [cost,treeEdges] = prim(nnode, edgelist, dist)
% Use Prim's algorithm to find the min cost spanning tree of a graph.
% Input arguments are as follows:
%    nnode = number of nodes
%    edgelist = m-by-2 array; each row is an edge (contains two distinct
%          indices between 1 and nnode)
%    dist = m-by-1 array of nonnegative numbers; contains edge costs.
% Output arguments are as follows:
%    cost = cost of min-cost spanning tree.
%    treeEdges = (n-1)-by-1 array containing the edges (integers between
%         1 and m) of the min-cost spanning tree.  If the algorithm
%         returns with length(treeEdges) == 0, this means that the
%         input graph was not connected.
```

2. (a) Write the capacitated min-cost flow problem in s.e.f. using the standard technique of inserting slack variables.

   (b) Let the formulation in part (a) be written min $\mathbf{c}^T\mathbf{x}$ s.t. $\hat{A}\mathbf{x} = \hat{\mathbf{b}}$, $\mathbf{x} \geq \mathbf{0}$ where $\mathbf{x}$ includes both flow variables and slacks. Show that the coefficient matrix constructed

in (a) is totally unimodular. [Hint: One approach is to use induction on l to prove a stronger claim that matrices of the form $[A, 0; V_l, I_l]$ are T.U.M., where V_l stands for the first l rows of the $n \times n$ identity matrix while I_l stands for the $l \times l$ identity matrix.]

3. (a) Formulate the minimum graph bisection problem as an integer linear programming problem. The *minimum graph bisection problem* is: given a graph $G = (N, E)$ such that $|N|$ is even, find the subset of nodes S that minimizes the number of edges $e = \{i, j\}$ such that $i \in S$ and $j \in N - S$ subject to the constraint that $|S| = |N|/2$. Your formulation should have one variable per node and one per edge. Be sure to explain the interpretation of your variables.

   (b) Consider the relaxation of your ILP problem in which you drop the integrality constraints to obtain an LP. Show by means of a very small example that the optimizer for the LP relaxation is not in general a solution to the minimum graph bisection problem.

4. (a) The *max-cut problem* takes as input a directed graph $G = (N, E)$ and seeks a subset of the nodes S such that the number of edges $e = \{i, j\}$ such that $i \in S$, $j \in N - S$ is maximized (no other constraints). Formulate this problem as an integer linear programming problem using the following framework.

   The formulation has one 0-1 variable x_i for each node $i \in N$ that indicates membership in S. It also has one 0-1 variable w_{ij} for each ordered pair of nodes $(i, j) \in N \times N$. It also has the constraint $x_i \oplus x_j = w_{ij}$ for all $(i, j) \in N \times n$. Here '$\oplus$' denotes XOR and has the following formula: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. Note: you need to figure out how to express the equation $x_i \oplus x_j = w_{ij}$ using only linear equations and inequalities because the problem you formulate must be ILP. Actually, it suffices to express merely $w_{ij} \leq x_i \oplus x_j$, since the problem is maximization.

   The edges of the graph do not enter into the constraints; the edges play a role only in the objective function. The objective function depends only on the w_{ij}'s (not the x_i's). Note: please insert also the symmetrization equations $w_{ij} = w_{ji}$ as additional linear constraints.

   (b) Consider the so-called "triangle inequalities" which are of two forms: $w_{jk} \leq w_{ij} + w_{ik}$ for all $(i, j, k) \in N^3$ and $w_{jk} + w_{ij} + w_{ik} \leq 2$ for all $(i, j, k) \in N^3$. Show that these inequalities hold for all solutions to the original max-cut problem instance. On the other hand, show that they form valid cuts in the sense that for a particular example of max-cut (not very large), there is a feasible noninteger solutions for the LP relaxation of your ILP formulation that does not satisfy one or more of these inequalities.