CS 670 (Spring 2011) — Midterm (Takehome)

Unless you are specifically asked to reprove a known fact, you can use all facts from class, homeworks and the textbook without proof. There are four questions on this exam. The only legitimate sources in solving this exam are (1) your textbook and course notes, (2) any handouts provided by us, (3) discussions with the TA or instructor. You cannot discuss the problems with anyone else (whether in the class or not), nor can you look for solutions in other textbooks, on the Internet, or in other sources. The exam is due in class by the end of class on Thursday, 03/24. If you submit after that, your exam will automatically be graded as 0.

G O O D   L U C K

(1) [10 points]
You will analyze another algorithm for computing a Minimum Spanning Tree. As before, $G = (V, E)$ is an undirected graph with $n = |V|$ nodes, $m = |E|$ edges, and positive (and distinct) edge costs c_e. The new algorithm is the following:

Algorithm 1 Minimum Spanning Tree

```
Start with T = ∅.
while n > 1 do
  Let S = ∅.
  for each vertex v do
    Let e_v be the cheapest edge incident on v.
    Include e_v in S.
  Contract all connected components. That is, replace each connected component C in S with just one
  new vertex u that inherits all edges of all v ∈ C (except self-loops, which are deleted).
  Add all of S to T.
  Use the contracted graph for the next iteration of the loop.
```

Prove that this algorithm finds a Minimum Spanning Tree, and takes time $O(m \log n)$.

(2) [15 points]
Throughout, $G = (X \cup Y, E)$ is a bipartite graph. X and Y may (a priori) contain different numbers of nodes.

(a) Prove that if each node has degree exactly d, then G has a perfect matching.

(b) Prove that if each node has degree exactly d, then there are d disjoint perfect matchings $M_1, M_2, \ldots, M_d$ such that $E = \bigcup_{i=1}^{d} M_i$.

(c) Prove that if each node has degree at most d, then there are d disjoint matchings $M_1, M_2, \ldots, M_d$ (not necessarily perfect) such that $E = \bigcup_{i=1}^{d} M_i$.

(Hint: If you use earlier parts of this problem to prove later ones (even if you cannot prove the earlier parts yourself), you still get partial credit.)

(3) [10 points]
You are given a tree G of maximum degree 3 in which each node v has a non-negative value b_v. You are also given a number $k \leq n$. Your goal is to compute a subtree T (i.e., connected subset) of G, containing at most k nodes, of maximum total value $\sum_{v \in T} b_v$. Give a polynomial-time algorithm for this problem.

(Hint: You may first want to solve the problem when you are given a particular root node $r \in V$ that must be included in T. Then, using this version, the given problem should be pretty easy to solve.)

(4) [10 points]

Suppose that we change the previous problem a little bit. You are still given a tree G of maximum degree 3, and each node v still has a non-negative value b_v. Instead of a target number k of nodes, we now have costs $c_e \geq 0$ on the edges. The goal is now again to find a subtree T of maximum total value, but this time, the constraint is that the edges together must not cost more than a given target value C. Prove that this version of the problem is NP-complete.