CS 180, Fall 2015 Homework 7

The following homework is due on Wednesday, November 25th at the beginning of lecture.

When submitting your homework, please include your name at the top of each page. If you submit multiple pages, please staple them together. Late submissions are not accepted.

1. (a) In a graph that possibly has negative edge weights but no negative cycles, prove that there is a node whose shortest path from the source consists of a single edge.

   (b) Prove that a graph has a negative cycle if and only if in Bellman-Ford algorithm the n-th round of the algorithm changes the shortest distance of some vertices.

2. Minimum cut of a graph is the minimum over all pairs of vertices of the minimum cut for the pair. Give an algorithm that finds the minimum cut of a directed strongly connected graph. Your algorithm should run in $O(m^2n^2)$ time.

3. In class we learned that the minimum number of edges that need to be removed to disconnect a pair s and t of vertices equals the maximum number of edge disjoint paths between s and t. Can we make a similar statement about vertices? Specifically, is the following statement true or false: the minimum number of vertices that need to be removed to disconnect s and t equals the maximum number of vertex disjoint paths between s and t. Either prove the correctness or give a counter-example disproving the statement.

4. Some of your friends have recently graduated and started a small company, which they are currently running out of their parents' garages in Santa Clara. They are in the process of porting all their software from old system to a new, revved-up system; and they are facing the following problem. They have a collection of n software applications, $\{1, 2, ..., n\}$, running on their old system; and they would like to port some of these to the new system. If they move application i to the new system, they expect a net benefit of $b_i \geq 0$. The different software applications interact with one another; if applications i and j have extensive interaction, then the company will incur an expense if they move one of i or j to the new system but not both; let's denote this expense by $x_{ij} \geq 0$. So, if the situation were really this simple, your friends would just port all n applications, achieving a total benefit of $\sum_{i=1}^{n} b_i$. Unfortunately, there is a problem ...

   Due to small but fundamental incompatibilities between the two systems, there is no way to port application 1 to new system; it will have to remain on the old system. Nevertheless, it might still pay off to port some of the other applications, accruing the associated benefit and incurring the expense of the interaction between applications on different systems. So this is the question they pose to you: which of the remaining applications, if any, should be moved? Give a polynomial-time algorithm to find a set $S \subseteq 2, 3, \ldots, n$ for which the sum of the benefits minus the expenses of moving the applications in S to the new system is maximized.