CPSC 455/555 // ECON 425/563, Fall 2011
Solution Set for Exam 1

Some of these answers are considerably longer and more detailed than answers needed to be to earn full credit. The additional information is provided so that anyone who answered a question incorrectly can read a full explanation of what the question was getting at.

Question 1:

(a) As we argued in class, any Nash equilibrium (NE) of this game must be connected. Thus, if $G = (V, E)$ were a NE that is not a tree, it would have to contain a cycle. Let $s = (s_1, \ldots, s_n)$ be a strategy vector that results in G, where $n = |V(G)|$. Let $x_0 - -x_1 - - \cdots - -x_{k-1} - -x_0$ be a cycle in G, and choose the numbering of these nodes in the cycle so that there is an x_i whose strategy in s includes the purchase of edge $e = x_i - -x_{i+1 \bmod k}$. We claim that x_i could lower its total cost by not purchasing e, thus lowering the edge-purchasing component of its total cost by α. We need to show that not purchasing e increases the shortest-path component of x_i's total cost by less than α. Let $y \neq x_i$ be any other node in V. If there is a shortest path from x_i to y in G that does not use the edge e, then the part of x_i's shortest-path cost that's contributed by its distance from y does not increase as a result of x_i's not purchasing e. So, assume that all shortest paths from x_i to y in G are of the form eP and that P has length p. Then there is a path $x_i - -x_{i-1 \bmod k} - - \cdots - -x_{i+1 \bmod k} - -P$ to y in $(V, E - \{e\})$ that "goes the other way around the cycle to $x_{i+1 \bmod k}$" before setting off on P and has length $k - 1 + p$; note that this may or may not be a shortest path (and, in fact, may or may not be a simple path), but its existence proves that a shortest path from x_i to y in $(V, E - \{e\})$ has length at most $k - 1 + p$. Thus, not purchasing e will increase x_i's total cost by at most $k - 1$ for each such y. Now, $k \leq n$, and there are at most $n - 1$ such y's. Thus, not purchasing e increases the shortest-path component of x_i's total cost by at most $(n-1)^2 < n^2 < \alpha$. This means that strategy vector s is not stable and, because s was chosen arbitrarily, that G is not a NE of the local connection game.

(b) Because this instance is a tree in which the s_i's are all at the root and each leaf contains a single t_i, there is a unique path from s_i to t_i, for $1 \leq i \leq k = 4$. This set of paths is the unique NE of the game. Therefore, both the Price of Anarchy and the Price of Stability are 1.

Question 2:

(a) Yes, it will converge, because this instance obeys the Gao-Rexford constraints. No, the stable routing tree on which it converges is not optimal. It converges on the set of routes $\{1 - -d, 2 - -d, \ldots, n - -d, n+1 - -n - -d, n+2 - -n+1 - -n - -d\}$, in which the sum of the valuations is $2n+1$. In the set of routes $\{1 - -d, 2 - -3, \ldots, n - -d, n+1 - -1 - -d, n+2 - -n+1 - -1 - -d\}$, which is the optimal tree, the sum of the valuations is $n^2 + n + 1$. The latter tree will not be chosen, however, because AS $n+1$ will choose route $n+1 - -n - -d$, export it to AS $n+2$, and, once in a stable tree, never export $n+1 - -1 - -d$ to AS $n+2$ (thus depriving AS $n+2$ of the opportunity to choose $n+2 - -n+1 - -1 - -d$).

(b) According to the Gao-Rexford "scoping" constraints, AS 1 will export its peer route $1 - -d$ to its customer AS 2, but it will not export this route to its provider AS 4. Thus, AS 4 will not be given the opportunity to choose route $4 - -1 - -d$.

Note that this question said that the "route-export policies of source ASes 1, 2, 3, and 4 satisfy the Gao-Rexford scoping constraints." It did *not* say that the ASes follow the Gao-Rexford

preference constraints. Partial credit was given for an answer that conveyed knowledge of the Gao-Rexford preference constraints, but that was not the point of this question.

Question 3:

(a) MinWork is a VCG mechanism and therefore, by Theorem 9.17, is strategyproof.

Full credit was given to anyone who simply stated that MinWork is a VCG mechanism, but a proof of this fact is given here.

We must first show that MinWork always produces an allocation that maximizes the sum of the agents' valuations. In any feasible allocation Z, each task is assigned to exactly one agent and thus contributes exactly one additive term to the corresponding sum of the agents' valuations of Z. In any allocation produced by MinWork, this additive term is as large as possible, because it is the negative of the smallest reported completion time for the task; thus the sum of these additive terms, which is the overall sum of the agents' valuations, is also maximized by MinWork. Next, we must show that the MinWork payment functions are of the form required by VCG. Indeed, this is the case, because

$$\begin{aligned} p^i(A^1, \ldots, A^n) &= \sum_{z_j \in Z^i} \min_{i' \neq i} a_j^{i'} \\ &= \left(\sum_{z_j} \min_{i' \neq i} a_j^{i'} \right) - \left(\sum_{z_j \notin Z^i} \min_{i' \neq i} a_j^{i'} \right) \\ &= \left(\sum_{j=1}^{k} \min_{i' \neq i} a_j^{i'} \right) - \sum_{i' \neq i} \left(\sum_{z_j \in Z^{i'}} a_j^{i'} \right) \\ &= \left(\sum_{j=1}^{k} \min_{i' \neq i} a_j^{i'} \right) + \sum_{i' \neq i} v^{i'}(A^{i'}, Z) \end{aligned}$$

and $\sum_{j=1}^{k} \min_{i' \neq i} a_j^{i'}$ has the required form $h^{-i}(A^{-i})$ for a VCG payment rule; that is, it does *not* depend on A^i.

MinWork is also clearly polynomial-time computable. Let W be the maximum time needed to add, subtract, or compare numbers in $(A^1, \ldots, A^n)$; note that W is polynomial in the size of this numerical input. Then, in time $O(Wkn)$, MinWork can scan the input $(A^1, \ldots, A^n)$ and correctly assign each z_j to an agent with smallest a_j^i while keeping track of the value $\min_{i' \neq i} a_j^{i'}$. The payments $p^i(A^1, \ldots, A^n) = \sum_{z_j \in Z^i} \min_{i' \neq i} a_j^{i'}$ can then be computed in time $O(Wkn)$.

(b) For a given instance $z_1, \ldots, z_k, T^1, \ldots, T^n$, let Z_{MW} be an allocation produced by MinWork and Z_{opt} be an optimal allocation. For any allocation Z, let $\mathit{TotalTime}(Z)$ be the total amount of time that all agents spend executing tasks assigned to them by Z, *i.e.*, $\mathit{TotalTime}(Z) = \sum_{1 \leq i \leq n} \sum_{z_j \in Z^i} t_j^i$. Because MinWork is strategyproof and assigns each task to an agent who can execute it in the minimum reported time, we know that $\mathit{TotalTime}(Z_{MW}) \leq \mathit{TotalTime}(Z_{opt})$. By definition, $\mathit{Makespan}(Z_{MW}) \leq \mathit{TotalTime}(Z_{MW})$, because, in the worst case for this comparison, all tasks would be assigned to one agent. Similarly, $\mathit{TotalTime}(Z_{opt}) \leq n \cdot \mathit{Makespan}(Z_{opt})$, because, in any allocation, at least one of the n agents has to spend at least $1/n$ of the total time spent by all agents. Thus,

$$\mathit{Makespan}(Z_{MW}) \leq \mathit{TotalTime}(Z_{MW}) \leq \mathit{TotalTime}(Z_{opt}) \leq n \cdot \mathit{Makespan}(Z_{opt}),$$

and $Makespan(Z_{MW}) \leq n \cdot Makespan(Z_{opt})$ is exactly what it means for MinWork to produce an n-approximately optimal allocation.
(c) The minimum-makespan problem is NP-hard. Thus, if P $\neq$ NP, there is no polynomial-time algorithm that solves it exactly; *a fortiori*, there are currently no known algorithmic techniques that point the way toward such an algorithm. Strategyproofness is a red herring here; even if we knew that all agents were acting truthfully, we still could not compute an optimal allocation exactly in polynomial time.

Question 4:
(a) If $S = \{s_1, \ldots, s_m\}$, and $p_1 \geq p_2 \geq \cdots \geq p_m$ is a sequence of prices such that the bidder's value of S is $v(S) = \sum_{1 \leq j \leq m} p_j$, then $v()$ can be expressed as

$$\text{OR}_{j=1}^{m} \left((\{s_1\}, p_j) \text{ XOR } (\{s_2\}, p_j) \text{ XOR } \cdots \text{ XOR } (\{s_m\}, p_j) \right),$$

the size of which (*i.e.*, the number of atoms in which) is m^2. Definition 11.18 tells us that, if the bidder receives bundle $T = \{s_{i_1}, \ldots, s_{i_k}\}$, he will assign it value $\sum_{1 \leq j \leq k} p_j$ by, *e.g.*, using the $(\{s_{i_1}\}, p_1)$ atom of the first XOR clause in his bid to value s_{i_1} at p_1, the $(\{s_{i_2}\}, p_2)$ atom of the second XOR clause in his bid to value s_{i_2} at p_2, *etc.*
(b) This statement is clearly true for $m = 1$; so assume that $m > 1$. Recall that all valuation functions in Chapter 11 satisfy $v(\emptyset) = 0$ and are monotonic, *i.e.*, $v(S_1) \leq v(S_2)$ for all $S_1 \subseteq S_2$. We claim that an XOR bid that represents a symmetric, downward-sloping valuation function in which the price sequence is strictly decreasing and greater than 0 must contain an atom (T, p) for each nonempty subset T of S, of which there are $2^m - 1$. Suppose, by way of contradiction, that T is a nonempty subset of S such that the XOR bid does not contain an atom (T, p). By Definition 11.18 and monotonicity, the bidder would then have to assign to T the value $v(T')$ for some T' that is a proper subset of T and for which his XOR bid contains an atom (T', p'); however, the definition of "symmetric, downward-sloping" and the fact that $p_1 > p_2 > \cdots > p_m > 0$ together imply that $v(T)$ must be strictly greater than $v(T')$ for any such T'.

Question 5:
(a) See Definitions 14.4 and 14.5. This solution concept is weaker than dominant-strategy equilibrium (DSE). In a DSE, each agent i would have to be able to maximize his utility by using s_i^* regardless of the strategies of the other agents, not only regardless of their types.
(b) See inequalities 28.9 and 28.10 and surrounding text in Section 28.3.2. As discussed in class, Chapter 28 is vague about whether the p_i are total prices or prices per click; full credit was given for either if the rest of the definition provided is correct.
(c) The definition is given in Sections 1.3.3 and 1.3.4. Matching Pennies, which is Example 1.7, is given in Section 1.1.4.