COS 423	Theory of Algorithms	Spring 2018

Problem Set 7

This assignment is due Wednesday, May 2 at 11pm via electronic submission. Collaboration is permitted, according to the rules specified in the syllabus.

Read CHAPTER 7 in *Algorithm Design*.

1. Given an undirected graph $G = (V, E)$, a subset of nodes $S \subseteq V$ is *node cover* if for each edge $(v, w) \in E$: either $v \in S$, or $w \in S$, or both.

   (a) Given a *bipartite* graph, describe how to find a node cover with the fewest nodes by solving a minimum st-cut problem.

   Hint: use the same flow network that we used to find a max cardinality bipartite matching.

   (b) Conclude that, in any bipartite graph, the maximum cardinality of any matching equals the minimum number of nodes in any node cover.

2. Consider an n-by-n statistical table produced by the U.S. Census Bureau. Each table entry a_{ij} is a non-negative integer. Let r_i denote the sum of the entries in row i and let c_j denote the sum of the entries in column j. The U.S. Census Bureau discloses all of the row and column sums along with a subset S of the table entries, but suppresses the remaining entries in order to safeguard privileged information. Nevertheless, it might be possible to deduce the exact value of one or more of the undisclosed table entries from the disclosed information. We say that a table entry (i, j) is *unprotected* if there is only one possible value for a_{ij} that is consistent with the disclosed information.

   For example, the following table has 11 suppressed entries (designated with a $\star$), of which 3 are unprotected: $a_{23} = 0, a_{53} = 3$, and $a_{54} = 2$.

	5	3	9	7	2
8	$\star$	$\star$	4	0	1
6	$\star$	$\star$	$\star$	4	0
3	0	0	2	$\star$	$\star$
1	0	0	0	$\star$	$\star$
8	3	0	$\star$	$\star$	0

   Design an algorithm to identify the locations and values of all unprotected elements. Your algorithm should take $O(n^4)$ time and use $O(n^2)$ space.

   Note: only a minor deduction for $O(n^6)$ time.

3. Given a digraph and two distinguished nodes s and t, the *edge-disjoint paths problem* is to find a maximum number of edge-disjoint paths from s to t. In lecture, we solved this problem by formulating it as a maximum flow problem in a *unit-capacity network*: a flow network in which the capacity of every edge is 1. Prove that Dinitz' algorithm solves the maximum flow problem in unit-capacity networks in $\mathcal{O}\left(m^{3/2}\right)$ time.