HW 8 (due Wednesday, at noon, October 31, 2018)

CS 473: Algorithms, Fall 2018 Version: **1.2**

Submission guidelines and policies as in homework 1.

1 (100 PTS.) Cry me a matching.

1.A. (20 PTS.) Let $\mathsf{G} = (L \cup R, \mathsf{E})$ be a bipartite graph where $|L| = |R| = n$. Furthermore, assume that there is a ***full matching*** M in G (i.e., every vertex of G appears in a matching edge of M). Prove, that for any subset $X \subseteq L$, we have that $|\mathrm{N}(X)| \geq |X|$, where

$$\mathrm{N}(X) = \{y \mid x \in X \text{ and } xy \in \mathsf{E}\}$$

is the ***neighborhood*** of X.

1.B. (20 PTS.) Let $\mathsf{G} = (L \cup R, \mathsf{E})$ be a bipartite graph where $|L| = |R| = n$. Let M be a matching, and let $u \in L$ be a vertex that is not in M. Let R' be the set of all the vertices in R reachable via alternating paths from u, and let L' be all the vertices of L that are reachable from u via an alternating path (note that $u \in L'$). Assume that there is no augmenting path starting at u. Prove that $R' = \mathrm{N}(L')$ and $|L'| > |R'|$ (note, that $\mathrm{N}(L') \subseteq R$).

1.C. (20 PTS.) For a number $d > 0$, a graph is ***d-uniform*** if every vertex has exactly d of edges incident to it. Let $\mathsf{G} = (L \cup R, \mathsf{E})$ Be a d-uniform graph. Prove that for every set $X \subseteq L$ we have that $|\mathrm{N}(X)| \geq |X|$.

1.D. (40 PTS.) Using the above, and only the above, prove that a bipartite d-uniform graph has a full matching.

2 (100 PTS.) Like a rolling matching.

You can use the above question in solving this question.

2.A. (30 PTS.) Prove that the edges of a k-uniform bipartite graph over n vertices (i.e., a graph where every vertex has degree k) can be assigned numbers $\{1, \ldots, k\}$, such that all adjacent edges to a vertex have different numbers. Describe an efficient algorithm, as fast as possible, for computing this numbering. What is the running time of your algorithm as a function of n and k.

2.B. (70 PTS.) Given a bipartite graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, the task is compute disjoint cycles that cover all the vertices of G if possible (or return that this is not possible). Put somewhat differently, we want to find a maximum subset of edges such that as many vertices as possible are adjacent to exactly two edges in this collection.

Observe that we seen in class how to solve the problem for the case that every vertex has to be adjacent to one such edge. Describe in detail how to modify the (slow) matching algorithm, seen in class, so that it works in this case. What are the augmenting paths in this case? What is the running time of your algorithm.

Prove the correctness of your algorithm.

(You can not use network flow to solve this problem.)

To clarify – the algorithm just needs to output a cover by cycles if it exists and it covers *all* vertices.

3 (100 PTS.) Vertex cover for half the edges.

3.A. (40 PTS.) Describe how to compute the smallest vertex cover for a given bipartite graph $\mathsf{G} = (L \cup R, \mathsf{E})$ with n vertices and m edges (hint: consider the maximum matching in this graph). What is the running time of your algorithm? Prove the correctness of your algorithm.

(Hint: Lookup Kőnig theorem and its proof.)

3.B. (60 PTS.) You are given a graph G (not necessarily bipartite), that has a vertex cover of size k. Describe a polynomial time algorithm that computes a set of k vertices, such that these k vertices cover at least $m/2$ edges of G, where m is the number of edges in G.