CMU 15-251 **Fall 2018**

Midterm 3
Practice Problems

1. Let T be a tree which contains no vertex with degree 2. Prove that the number of leaves in T must be at least the number of non-leaf vertices.

2. Suppose G is a graph with n vertices and $n-2$ edges. Show that at least one of the following must be true:

   (i) G has an isolated vertex (i.e. a vertex of degree 0);

   (ii) G contains two connected components each of which is a tree with at least 2 vertices.

3. Consider a table with 2 rows and $n-1$ columns. The first row holds the numbers $1, 2, \ldots, n-1$; the second row holds arbitrary numbers between 1 and n (repetitions allowed). Construct an undirected graph G on n vertices labeled $\{1, 2, \ldots, n\}$ by connecting the two nodes in each column of the table (so each column corresponds to an edge).

   (a) Show by an example that this graph is not always a tree.

   (b) Show that if the graph G is connected, then it is a tree.

   (c) Prove that a connected component of G that has exactly k vertices has at most k edges.

   (d) Deduce that every connected component of G has at most one cycle.

4. Let $G = (X, Y, E)$ be a bipartite graph on $2n$ vertices with $|X| = |Y| = n$. Suppose that all the vertices in G have degree at least $n/2$. Prove that G has a perfect matching.

5. Let G be a graph in which every vertex has degree 4. Show that G contains two cycles that do not share any edges.

6. Suppose we are given n CMU students and m companies. Each company c has some number σ_c of slots, and we assume that the total number of students is larger than the total number of slots (i.e., $\sum_{c=1}^{m} \sigma_c < n$). Each student ranks the m companies in order of preference, and each company ranks the n students. The goal is to find an assignment of students to slots (one student per slot) that is stable in the sense that there is no unstable student-company pair. An unmatched student-company pair (s, c) is considered unstable (with respect to an assignment of students to slots) if and only if s would prefer c to her current situation (namely, she is either unmatched or matched to a company she likes less than c), and c would prefer s over one of the students assigned to c. There are two main differences between this problem and the regular Stable Matching Problem: (i) there are more students than slots, and (ii) each company may have more than one slot.

   Show that there is always a stable assignment of students to companies, and give an efficient algorithm to find one.

   Hint: Let the students propose to companies, and let each company maintain a waitlist of provisionally accepted students. In the proof of correctness, use the following version of the Improvement Lemma: For each company, after its waitlist becomes full, its least favorite student on the waitlist can only improve over time.

7. A language $L \subseteq \{0,1\}^*$ is called *unary* if it does not contain any word with a 0 symbol in it. Show that any unary language has polynomial-size circuit complexity.

8. Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be binary strings in $\{0,1\}^n$. We define

$$\text{“}x \leq y\text{”} \quad \text{to mean} \quad x_i \leq y_i \text{ for all } 1 \leq i \leq n.$$

Also, a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is called “monotone” if it satisfies the following property:

$$\text{for all inputs } x, y \in \{0,1\}^n, \quad x \leq y \implies f(x) \leq f(y).$$

Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone function. Prove that f can be computed by a circuit C that does not contain any NOT gates. (So we are only allowed to use the input gates, constant gates, AND gates and OR gates.)

9. An st-path in a graph G is just a path from vertex s to vertex t. Two st-paths are *vertex disjoint* if they don't share any vertices other than s and t. Menger's theorem says that the maximum number of vertex disjoint st-paths in a graph G is equal to the minimum number of vertices whose removal disconnects s and t. Show that the following problems are both in NP (you can use Menger's theorem for free):

$$X = \{\langle G, s, t, k\rangle : \text{there is a set of } k \text{ vertex disjoint } st\text{-paths in } G\}.$$

$$\overline{X} = \{\langle G, s, t, k\rangle : \text{there is no set of } k \text{ vertex disjoint } st\text{-paths in } G\}.$$

10. Consider the following decision problem: Given a graph $G = (V, E)$, does it contain a clique of size at least $|V|/2$? Recall that a clique is a subset of the vertices such that every pair of vertices in the subset is connected with an edge.

    Show that this problem is NP-complete.

11. For a polynomial with integer coefficients in several variables, an integral root is an assignment of integers to the variables so that the polynomial evaluates to 0. For example, the polynomial $2x_1^2x_2 - x_1^3x_2^2$ has an integral root $x_1 = 2$ and $x_2 = 1$, whereas the polynomial $x_1^2 + x_2^2 + 1$ has no integral root. Consider the computational problem of determining, given a polynomial with integer coefficients, whether it has an integral root. We can capture this problem by the language:

$$\begin{aligned}\text{ROOT} = \{\langle p\rangle \mid\ & p \text{ is a polynomial in several variables with integer coefficients} \\ & \text{that has an integral root}\}.\end{aligned}$$

In his famous address at the International Congress of Mathematicians in Paris in 1900, David Hilbert posed 23 mathematical problems as challenges for the 20th century, the 10'th problem of which, re-stated in the language of computability theory, was to give a decider for ROOT. As mentioned in lecture, we now know, however, that ROOT is in fact undecidable, so no algorithm exists to tell if a polynomial in several variables has an integral root.

In this problem you are to prove a different property of ROOT, namely, that ROOT is NP-hard.

Hint: How can you enforce a potential integral root to only take values in $\{0,1\}$? One possible reduction is from 3SAT (though, you may reduce from any of CIRCUIT-SAT, 3SAT, 3COL, CLIQUE, or even some new intermediate problem that you prove to be NP-complete).

12. NAE-4SAT refers to the following problem: The instance is n Boolean variables $x_1, \dots, x_n$, along with a list of "constraints". Each constraint is a set of exactly four "literals", where recall that a literal is either x_i or $\neg x_i$. Given a truth assignment to the n variables, we say that the constraint is satisfied if the four literals in it are not all true and not all false. We say that the whole instance is satisfied if all constraints in the instance are satisfied. Finally, NAE-4SAT is the language of all satisfiable instances. Show that NAE-4SAT is NP-complete. Hint: Reduce from 3SAT. Introduce exactly one extra variable.

13. In the "Betweenness" problem, an adversary has an ordered list L of n letters. He tells you the names of the letters in some arbitrary order (not necessarily in L's order), as well as a list B of some "betweenness" facts. Each fact in B is a statement of the form "x is between $\{y, z\}$ in L"; here x, y, z are letters, and the statement means that the list looks like either $\dots y \dots x \dots z \dots$ or $\dots z \dots x \dots y \dots$. (The $\dots$ can stand for any number of letters.)

    Turns out that given the information B, it's hard to recover L; specifically, it's NP-hard even to recover some other list L' that is consistent with all the betweenness facts B. Thus we may turn to approximation algorithms. Your task is to find an ordered list L' of the letters so that *at least half* of the betweenness facts in B are satisfied.

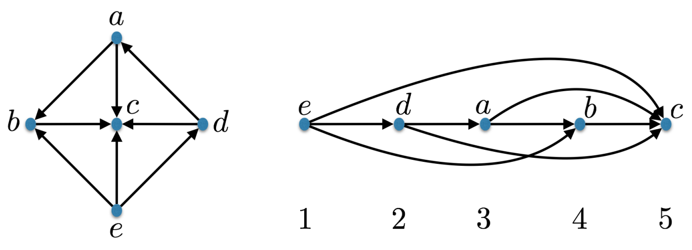
    Argue that the following algorithm solves the task in polynomial time:

    (a) Identify a "free" letter v; this means a letter v such that there is no fact in B of the form "v is between…".
    (b) Delete v and all facts involving v from B, forming B''.
    (c) Recursively solve B'', getting list L''.
    (d) Output either the list $[v, L'']$ or $[L'', v]$ (i.e., put v either at the beginning or the end), which ever is better in terms of satisfying facts in B.

    Some things to consider in your analysis: Why is there always a "free" letter (hint: remember that L exists)? Why is the overall algorithm polynomial time? What is the recursive base case? Why does the algorithm ultimately satisfy at least half of the betweenness facts?

14. Consider the following approximation algorithm to find minimum vertex cover in a connected graph G: run the DFS algorithm starting from some arbitrary vertex in the graph, and then output the set S consisting of the non-leaf vertices of the DFS tree. Show that this algorithm is a 2-approximation algorithm for MIN-VERTEX-COVER. Hint: Show that G has a matching of size at least $|S|/2$.

15. A *topological order* of an n-vertex directed graph $G = (V, A)$ is a bijection $f : V \to \{1, 2, \dots, n\}$ such that if $(u, v) \in A$, then $f(u) < f(v)$. Below is an example:



Here, $f(e) = 1$, $f(d) = 2$, $f(a) = 3$, $f(b) = 4$, and $f(c) = 5$.

(a) Prove that if a directed graph has a topological order, then it is acyclic.

(b) A *subgraph* H of G is any graph whose set of vertices and edges are a subset of the set of vertices and edges of G respectively. Consider the following problem: Given a directed graph $G = (V, A)$, find an acyclic subgraph H of G that contains as many directed edges as possible. Provide a polynomial-time $\frac{1}{2}$-approximation algorithm for this problem.