Homework 6
Due Date: April 19, 2018 at 11:59 PM on Blackboard.

Question 1
An independent set of a graph \( G = (V, E) \) is a subset \( V' \subseteq V \) of vertices such that each edge in \( E \) is incident on at most one vertex in \( V' \). The independent-set problem is to find a maximum-size independent set in \( G \). Although the independent-set decision problem is NP-complete, certain special cases are polynomial-time solvable.

a. Give an efficient algorithm to solve the independent-set problem when each vertex in \( G \) has degree 2. Analyze the running time, and prove that your algorithm works correctly.

b. Give an efficient algorithm to solve the independent-set problem when \( G \) is bipartite. Analyze the running time, and prove that your algorithm works correctly.

Question 2
Consider the Traveling-Salesman problem in which the distance function is symmetric, i.e. \( d(u, v) = d(v, u) \) for all \( u, v \in V \), and it satisfies the triangle inequality. We will also assume that the distance function is positive, i.e., \( d(v, u) \geq 0 \) for all \( u, v \in V \). Show an algorithm that produces a 2-approximation to an optimal-length hamiltonian tour in this graph. (Hint 1: It may be easier to first construct low-cost Hamiltonian Walk which is not necessarily simple (may repeat nodes), and prune this walk to yield a simple hamiltonian tour. Hint 2: You may want to look at MST algorithms for inspiration.).

Question 3
The Geography on Graphs Game is defined as follows: Given a directed graph \( G = (V, E) \) and a start node \( s \), two players alternate turns by following, if possible, an edge out of the current node to an unvisited node. The player who loses is the first one who cannot move to a node that hasnt been visited earlier. In other words, a player loses if it is his turn, the game is currently at node \( v \) and for each directed edges of the form \( (v, w) \in E \) the node \( w \) has already been visited.

1. Prove that the geography on graphs game is PSPACE-Complete

2. Suppose that the graph \( G \) has no directed cycles. Show that the Geography on a Graph decision problem is solvable in polynomial time.
**Question 4**

The difficulty in 3-SAT comes from the fact that there are $2^n$ possible assignments to the input variables $x_1, x_2, \ldots, x_n$, and there's no apparent way to search this space in polynomial time. This intuitive picture, however, might create the misleading impression that the fastest algorithms for 3-SAT actually require time $2^n$. In fact, though it’s somewhat counterintuitive when you first hear it, there are algorithms for 3-SAT that run in significantly less than $2^n$ time in the worst case; in other words, they determine whether there’s a satisfying assignment in less time than it would take to enumerate all possible settings of the variables.

Here well develop one such algorithm, which solves instances of 3-SAT in $O(p(n) \cdot (\sqrt{3})^n)$ time for some polynomial $p(n)$. Note that the main term in this running time is $(\sqrt{3})^n$, which is bounded by $1.74^n$.

(a) For a truth assignment $\Phi$ for the variables $x_1, x_2, \ldots, x_n$, we use $\Phi(x_i)$ to denote the value assigned by $\Phi$ to $x_i$. (This can be either 0 or 1.) If $\Phi$ and $\Phi'$ are each truth assignments, we define the distance between $\Phi$ and $\Phi'$ to be the number of variables $x_i$ for which they assign different values, and we denote this distance by $d(\Phi, \Phi')$. In other words, $d(\Phi, \Phi') = |\{i : \Phi(x_i) \neq \Phi'(x_i)\}|$.

A basic building block for our algorithm will be the ability to answer the following kind of question: Given a truth assignment $\Phi$ and a distance $d$, we’d like to know whether there exists a satisfying assignment $\Phi'$ such that the distance from $\Phi$ to $\Phi'$ is at most $d$. Consider the following algorithm, $\text{Explore}(\Phi, d)$, that attempts to answer this question.

**Algorithm 1** $\text{Explore}(\Phi, d)$

1: if $\Phi$ is a satisfying assignment then
2:     return “yes”
3: else if if $d = 0$ then
4:     return “no”
5: else
6:     Let $C_i$ be a clause that is not satisfied by $\Phi$ (i.e., all three terms in $C_i$ evaluate to false)
7:     Let $\Phi_1$ denote the assignment obtained from $\Phi$ by taking the variable that occurs in the first term of clause $C_i$ and inverting its assigned value
8:     Define $\Phi_2$ and $\Phi_3$ analogously in terms of the second and third terms of the clause $C_i$
9:     Recursively invoke: $\text{Explore}(\Phi_1, d - 1), \text{Explore}(\Phi_2, d - 1), \text{Explore}(\Phi_3, d - 1)$
10: if any of these three calls return “yes” then
11:     return “yes”
12: else
13:     return “no”
14: end if
15: end if

Prove that $\text{Explore}(\Phi, d)$ returns “yes” if and only if there exists a satisfying assignment $\Phi'$ such that the distance from $\Phi$ to $\Phi'$ is at most $d$. Also, give an analysis of the running time of $\text{Explore}(\Phi, d)$ as a function of $n$ and $d$. 

2
(b) Clearly any two assignments $\Phi$ and $\Phi'$ have distance at most $n$ from each other, so one way to solve the given instance of 3-SAT would be to pick an arbitrary starting assignment $\Phi$ and then run $Explore(\Phi, n)$. However, this will not give us the running time we want.

Instead, we will need to make several calls to $Explore$, from different starting points $\Phi$, and search each time out to more limited distances. Describe how to do this in such a way that you can solve the instance of 3-SAT in a running time of only $O(p(n) \cdot (\sqrt{3})^n)$.

**Bonus (10 points)**

The **Hitting Set** problem is similar to the **Set Cover** problem. An instance is given by a list $S_1, \ldots, S_n \subseteq \{1, \ldots, m\}$ of $n$ subsets of a universe $U$ of size $|U| = m$. In the **Set Cover** problem we want to find a minimum size subset of sets $S \subseteq \{1, \ldots, n\}$ which covers the universe $U$ i.e., $\bigcup_{i \in S} S_i = U$. In the **Hitting Set** problem we are given an absolute upper bound $k$ on the number of sets in $S$ and we are asked to maximize the number of elements covered by $S$ i.e., $|\bigcup_{i \in S} S_i|$. The search version of the **Hitting Set** problem is **NP-Hard** (you dont need to prove this).

1. Let $V^* = \bigcup_{i \in S^*} S_i$ denote the items covered by the optimal solution the **Hitting Set** problem. Given a set $P \subseteq \{1, \ldots, n\}$ define $Uncovered(V^*, P) = V^* \setminus (\bigcup_{i \in S^*} S_i)$ and let $n_P = |Uncovered(V^*, P)|$ denote the number of elements in $V$ that are still uncovered by a **Hitting Set** solution $P$. Show that for any subset $P \subseteq \{1, \ldots, n\}$ there exists some $j \notin P$ s.t. the set $S_j$ covers at least $n_P/k$ of the remaining items in $Uncovered(V^*, P)$.

2. Develop a greedy algorithm that always returns a hitting set solution which always covers at least $(1 - 1/e)|V^*|$ where $|V^*|$ is the number of items covered in the optimal solution. (**Hint:** You may use without proof the fact that $(1 - 1/k)^k <= e^{-1}$ for all integers $k, x > 0$.)