Instructions: Everyone needs to submit their own write-up. If you work together with other students, indicate their names on your write-up.

The possible grades for each (sub)problem are Check-, Check and Check+, corresponding to 0, 1 and 2 points respectively. For this problem set the maximum number of points is 22.

All graphs in this problem set are undirected, simple and unweighted.

Problem 1: 3-Paths are easier to find than Triangles.

Let $P_3$ be the path on 3 nodes. We say that an undirected graph $G = (V,E)$ contains an induced $P_3$ if there exist $u,v,w \in V$ such that $(u,v), (v,w) \in E$ but $(u,w) \notin E$.

Design an $O(m+n)$ time algorithm that finds an induced $P_3$ in any $m$-edge, $n$-node graph $G$, or determines that $G$ does not contain an induced $P_3$.

Problem 2: Four-cycle in sparse graphs.

In class we showed that a 4-cycle (if one exists) in an $n$-node graph can be found in $O(n^2)$ time using a lookup table approach. In the following two subproblems, assume that you have a data structure that can initialize a look-up table in constant time and can write and read from the look-up table in constant time.

(E.g. a hash table can accomplish this, at the cost of randomization and expected running time, but here you can assume a deterministic data structure. One way to do this is to be given an $n \times n$ matrix that is initialized at all zeros, where you don’t have to pay for the initialization.)

(a) Using a lookup table approach, give an algorithm that can find a 4-cycle in an $m$-edge $n$-node graph in $O(m\sqrt{n})$ time. (Hint: use the high degree-low degree technique)

(b) Using a lookup table approach, give an algorithm for finding a 4-cycle running in $O(m^{4/3})$ time; the algorithm can be randomized, working with high probability. (Hint: Consider the following iterative procedure: While the graph contains a node $v$ of degree $< 400m^{1/3}$, remove $v$. Repeat until every vertex in the remaining graph has degree at least $400m^{1/3}$, or the remaining graph is empty. If the remaining graph is nonempty, show that it has at least $200 \cdot 3 \cdot (n')^{3/2}$ edges, where $n'$ is its number of vertices, so that you can quickly return a 4-cycle. What do you do if the remaining graph is empty?)

Problem 3: Reducing $H$ to Clique.

Let $k \geq 3$ be a constant integer. Let $H$ be any $k$ node graph. Suppose there is an $O(n^c)$ time algorithm that can determine if an $n$-node graph contains a $k$-clique. Given this, give an $O(n^c)$ time algorithm that can determine if an $n$-node graph contains an induced copy of $H$.

Hint: Given an $n$-node $G$ in which you want to detect an induced copy of $H$, create a new $k$-partite graph $G'$ on $kn$ nodes that contains a $k$-clique if and only if $G$ contains an induced copy of $H$.

Problem 4: A useful lemma.

Recall that a polynomial over variables $x_1, \ldots, x_n$ is multilinear over $\mathbb{Z}_m$ if it is of the form $P(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i$, where the coefficients $c_S \in \{0, \ldots, m-1\}$ are elements of $\mathbb{Z}_m$ for every choice of $S \subseteq [n]$. The degree of a multilinear polynomial is the largest size of $S$ such that $c_S \neq 0$.

Here you will prove the following statement: Let $m \geq 2$ be an integer. Let $P(x_1, \ldots, x_n)$ be a non-zero multilinear polynomial over $\mathbb{Z}_m$ of degree $d$. Then
\[ Pr_{(a_1, \ldots, a_n) \in \{0,1\}^n}[P(a_1, \ldots, a_n) \neq 0 \mod m] \geq 1/2^d. \]

To prove the above statement prove the following:
(a) For any \(Q(x_1, \ldots, x_n)\) that is a nonzero multilinear polynomial over \(\mathbb{Z}_m\), there exist some \(a_1, \ldots, a_n \in \{0,1\}\) so that \(Q(a_1, \ldots, a_n) \neq 0 \mod m\).

Hint: Use induction on the number of variables.
(b) Recall that \(P(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i\), with \(c_S \in \mathbb{Z}_m\) and that \(P\) has degree \(d\).

Consider any one of the nonzero monomials of \(P\) of degree \(d\); say it is \(c_S \prod_{i \in S} x_i\) with \(|S| = d\). Rename the variables so that in the variables in \(S\) are \(x_1, \ldots, x_d\).

Now consider any setting of the variables not in \(S\), \(x_j = \alpha_j \in \mathbb{Z}_m\) for \(x_j \notin S\). Let \(Q\) be the polynomial that is \(P\) with the values \(x_j = \alpha_j \in \mathbb{Z}_m\) for \(x_j \notin S\) plugged in. \(Q\) is a multilinear polynomial only over the variables \(x_1, \ldots, x_d\) (the variables in \(S\)):

\[ Q(x_1, \ldots, x_d) = \sum_{T \subseteq S} c'_T \prod_{i \in T} x_i, \]

for some \(c'_T \in \mathbb{Z}_m\).

Argue that \(Q(x_1, \ldots, x_d) \neq 0 \mod m\), and that \(Q\) is a nonzero polynomial over \(\mathbb{Z}_m\).

(c) Use (a) and (b) to conclude that the probability that \(P\) evaluates to nonzero mod \(m\) on a random assignment in \(\{0,1\}^n\) is at least \(1/2^d\).

**Problem 5: 6-cycle detection**

In this problem we will give an \(O(n^2)\) time algorithm for detecting whether a graph contains a 6-cycle, and returns a 6-cycle if there exists one. We will actually prove a stronger result. The algorithm will run a BFS-like procedure rooted at every vertex \(v\). If \(v\) is in a 6-cycle, the algorithm should return a 6-cycle (not necessarily containing \(v\)) in \(O(n)\) time. For convenience, we will use \(L_i\) to denote the set of vertices at distance \(i\) from \(v\). We use \(e(L_i)\) to denote the number of edges whose two endpoints are both in \(L_i\) and use \(e(L_i, L_{i+1})\) to denote the number of edges whose two endpoints are in \(L_i\) and \(L_{i+1}\) respectively. (Recall that in a BFS the edges are all contained in \(\cup_i(e(L_i) \cup e(L_i, L_{i+1}))\)).

(a) Show that there exists a constant \(C\) so that if \(H\) is a connected graph with \(k\) vertices and at least \(Ck\) edges, then from any vertex \(u \in V(H)\), there exists a simple path on 4 edges from \(u\) to some other vertex \(x\).

For the following subproblems, the input to the algorithms will be a vertex \(v\), and for each \(i\), \(L_i\) is the \(i\)th layer of the BFS out of \(v\).

(b) Give an \(O(n)\) time algorithm that given a starting vertex \(v\), reports a 6-cycle if \(e(L_1) \geq C|L_1|\). Similarly, give an \(O(n)\) time algorithm that given a starting vertex \(v\) for which \(e(L_1) < C|L_1|\), reports a 6-cycle if \(e(L_1, L_2) \geq C(|L_1| + |L_2|)\).

(c) Give an \(O(n)\) time algorithm that given a starting vertex \(v\) for which \(e(L_1) < C|L_1|\) and \(e(L_1, L_2) < C(|L_1| + |L_2|)\), reports a 6-cycle if \(e(L_2) \geq C|L_2|\). Similarly, give an \(O(n)\) time algorithm that given a starting vertex \(v\) for which \(e(L_1) < C|L_1|\), \(e(L_1, L_2) < C(|L_1| + |L_2|)\) and \(e(L_2) < C|L_2|\), reports a 6-cycle if \(e(L_2, L_3) \geq C(|L_2| + |L_3|)\).

(d) Give an \(O(n \log n)\) time algorithm that given \(v\) that lies on a 6-cycle and for which \(e(L_1) < C|L_1|\), \(e(L_1, L_2) < C(|L_1| + |L_2|)\), \(e(L_2) < C|L_2|\), and \(e(L_2, L_3) < C(|L_2| + |L_3|)\), with high probability returns a 6-cycle (not necessarily containing \(v\)).

Using the previous parts, conclude that there is an \(O(n^2 \log n)\) time algorithm that with high probability returns a 6-cycle if the graph has a 6-cycle, and if the graph does not have a 6-cycle always correctly returns that there is no 6-cycle.