You may work with a partner on this problem set. You need only turn in one solution. The multi-part problem has some challenging parts. Please start early.

Question 1. *Given an undirected, bipartite graph $G = (V, E)$ where $V = L \cup R$ and all edges have exactly one endpoint in L, let M be a matching in G. We say that a simple path P in G is an* augmenting path *with respect to M if it starts at an unmatched vertex in L, ends at an unmatched vertex in R, and its edges belong alternately to $E \setminus M$ and M. (This definition of an augmenting path mimics the idea of an augmenting path a flow network corresponding to the bipartite graph.) In this problem, we treat a path as a sequence of edges, rather than as a sequence of vertices. A shortest augmenting path with respect to a matching M is an augmenting path with a minimum number of edges.*

Given two sets A and B, the symmetric difference *$A \oplus B$ is defined as $(A \setminus B) \cup (B \setminus A)$, that is, the elements that are in exactly one of the two sets.*

(a) Show that if M is a matching and P is an augmenting path with respect to M, then the symmetric difference $M \oplus P$ is a matching and $|M \oplus P| = |M| + 1$. Show that if $P_1, P_2, \ldots, P_k$ are vertex-disjoint augmenting paths with respect to M, then the symmetric difference $M \oplus (P_1 \cup P_2 \cup \cdots \cup P_k)$ is a matching with cardinality $|M| + k$.

The general structure of our algorithm is the following:

Algorithm 1 Hopcroft-Karp(G)

Require: A bipartite graph G

1: $M \leftarrow \emptyset$
2: **repeat**
3: $\mathcal{P} \leftarrow \{P_1, P_2, \ldots, P_k\}$ be a *maximal* set of vertex-disjoint shortest augmenting paths with respect to M
4: $M \leftarrow M \oplus \{P_1 \cup P_2 \cup \cdots \cup P_k\}$
5: **until** $\mathcal{P} = \emptyset$
6: **return** M

Note that a maximal *set is one that cannot be extended, so a* maximal *set of vertex-disjoint shortest augmenting paths with respect to M is one in which there are no more augmenting paths which can be added to M and keep it vertex-disjoint. The remainder of this problem asks you to analyze the number of iterations in the algorithm (that is, the number of iterations in the* **repeat** *loop) and to describe an implementation of line 3.*

(b) Given two matchings M and M^* in G, show that every vertex in the graph $G' = (V, M \oplus M^*)$ has degree at most 2. Conclude that G' is a disjoint union of simple paths or cycles. Argue that edges in each such simple path or cycle belong alternately to M or M^*. Prove that if $|M| \leq |M^*|$, then $M \oplus M^*$ contains at least $|M^*| - |M|$ vertex-disjoint augmenting paths with respect to M.

Let l be the length of a shortest augmenting path with respect to a matching M, and let $P_1, P_2, \ldots, P_k$ be a maximal set of vertex-disjoint augmenting paths of length l with respect to M. Let $M' = M \oplus (P_1 \cup P_2 \cup \cdots \cup P_k)$, and suppose that P is a shortest augmenting path with respect to M'.

(c) Show that if P is vertex-disjoint from $P_1, P_2, \ldots, P_k$, then P has more than l edges.

(d) Now suppose that P is not vertex-disjoint from $P_1, P_2, \ldots, P_k$. Let A be the set of edges $(M \oplus M') \oplus P$. Show that $A = (P_1 \cup P_2 \cup \cdots \cup P_k) \oplus P$ and that $|A| \geq (k+1)l$. Finally, show that P has more than l edges by arguing that P must not only share a node with $P_1, P_2, \ldots, P_k$, but, in fact, an edge.

(e) Prove that if a shortest augmenting path for M has length l, the size of the maximum matching is at most $|M| + |V|/(l+1)$.

(f) Show that the number of **repeat** *loop iterations in the algorithm is at most $2\sqrt{n}$. (*Hint: *By how much can M grow after iteration number $\sqrt{n}$?)*

(g) Give an algorithm that runs in $O(m)$ time to find a maximal set of vertex-disjoint shortest augmenting paths $P_1, P_2, \ldots, P_k$ for a given matching M. As a hint, think about a breadth-first search that labels each node as either EVEN or ODD depending on when it is encountered. Conclude that the total running time of HOPCROFT-KARP *is $O(\sqrt{n}m)$.*

Question 2 (extra-credit)**.** *Implement Hopcraft-Karp and show empirically that it runs in $O(\sqrt{n}m)$ time.*