Assignment 1

Deadline:

1. Prove the König theorem: Let G be bipartite, then cardinality of maximum matching = cardinality of minimum vertex cover.

2. Consider the algorithm **Negative-Dijkstra** for computing shortest paths through graphs with negative edge weights (but without negative cycles) Note that **Negative-Dijkstra** shifts all edge weights to be non-negative(by shifting all edge weights by the smallest original value) and runs in $O(m + \log n)$ time.

Algorithm 1 Algorithm 1: Negative-Dijkstra(G,s)

$w^* \leftarrow$ minimum edge weight in G;
for $e \in E(G)$ **do**
    $w'(e) \leftarrow w(e) - w^*$
end for
$T \leftarrow$**Dijkstra**(G', s);
return weights of T in the original G;

Prove or Disprove: **Negative-Dijkstra** computes single-source shortest paths correctly in graphs with negative edge weights. To prove the algorithm correct, show that for all $u \in V$ the shortest $s - u$ path in the original graph is in T. To disprove, exhibit a graph with negative edges, with no negative cycles where **Negative-Dijkstra** outputs the wrong "shortest" paths, and explain why the algorithm fails.

3. Consider a weighted, directed graph G with n vertices and m edges that have integer weights. A graph walk is a sequence of not-necessarily-distinct vertices $v_1, v_2, \ldots, v_k$ such that each pair of consecutive vertices v_i, v_{i+1} are connected by an edge. This is similar to a path, except a walk can have repeated vertices and edges. The length of a walk in a weighted graph is the sum of the weights of the edges in the walk. Let s, t be given vertices in the graph, and L be a positive integer. We are interested counting the number of walks from s to t of length exactly L.

- Assume all the edge weights are positive. Describe an algorithm that computes the number of graph walks from s to t of length exactly L in $O((n + m)L)$ time. Prove the correctness and analyze the running time
- Now assume all the edge weights are non-negative(but they can be 0), but there are no cycles consisting entirely of zero-weight edges. That is, for any cycle in the graph, at least one edge has a positive weight.
  Describe an algorithm that computes the number of graph walks from s to t of length exactly L in $O((n + m)L)$ time. Prove correctness and analyze running time.

4. The diameter of a connected, undirected graph $G = (V, E)$ is the length (in number of edges) of the longest shortest path between two nodes. Show that is the diameter of a graph is d then there is some set $S \subseteq V$ with $|S| \leq n/(d-1)$ such that removing the vertices in S from the graph would break it into several disconnected pieces.

5. Let G be a n vertices graph. Show that if every vertex in G has degree at least $n/2$, then G contains a Hamiltonian path.

6. Show how to find a minimal cut of a graph (not only the cost of minimum cut, but also the set of edges in the cut).

7. Let $G(V, E)$ be a connected undirected graph with a weight $w(e) > 0$ for each edge $e \in E$. For any path $P_{u,v} =< u, v_1, v_2, \ldots, v_r, v >$ between two vertices u and v in G,let $\beta(P_{u,v})$ denote the maximum weight of an edge in $P_{u,v}$. We refer to $\beta(P_{u,v})$ as the **bottleneck weight** of $P_{u,v}$. Define

$$\beta^*(u, v) = \min\{\beta(P_{u,v}) : P_{u,v} \text{ is a path between } u \text{ and } v\}.$$

Give a polynomial algorithm to find $\beta^*(u, v)$ for each pair of vertices u and v in V and a proof of the correctness of the algorithm.

8. Let $G = (V, E)$ be a directed graph. Give a linear-time algorithm that given G, a node $s \in V$ and an integer k decides whether there is a walk in G starting at s that visits at least k distinct nodes.

9. Minimum Bottleneck Spanning Tree: Given a connected graph G with positive edge costs, find a spanning tree that minimizes the most expensive edge.