COMP 6651: Worksheet 5

Graph Algorithms

Let $G = (V, E)$ be an undirected graph. Some terminology and notation:

- The number of vertices $|V|$ is typically denoted by n, and the number of edges $|E|$ is typically denoted by m.
- A subset $S \subseteq V$ is an **independent set** in G if no pair of vertices in S is adjacent. The **independence number** of G, denoted by $\alpha(G)$, is the maximum size of its independent set.
- A subset $S \subseteq V$ is a **clique** in G if every pair of vertices in S is adjacent. The **clique number** of G, denoted by $\omega(G)$, is the maximum size of its clique.
- A subset $S \subseteq V$ is a **vertex cover** in G if every edge $e \in E$ includes at least one vertex in S, i.e., $S \cap e \neq \emptyset$. In other words S "hits" every edge. The **covering number**, denoted by $\tau(G)$, is the minimum size of its vertex cover.
- A function $c : V \to \{1, 2, \dots, k\}$ is a **valid k-coloring** of G if $c(u) \neq c(v)$ for every edge $\{u, v\} \in E$. In other words, every vertex receives a color such that no edge is monochromatic. The **coloring number**, denoted by $\chi(G)$, is the minimum value k for which there exists a **valid k-coloring** of G.
- G is **bipartite** if it is 2-colorable. In other words, the set of vertices V can be partitioned into two blocks such that every edge $e \in E$ has one endpoint in each block.
- A subset $M \subseteq E$ is a **matching** if for every pair of edges $e_1, e_2 \in M$ we have $e_1 \cap e_2$. In other words, edges in the matching do not share any vertices. The **matching number**, denoted by $\mu(G)$, is the maximum size of a matching in G.
- A **complement graph**, denoted by $\overline{G}$, has vertex set V and edge set $\overline{E} = \binom{V}{2} \setminus E$. In other words, every pair of vertices flips their adjacency status: $\{u, v\}$ is adjacent in G if and only if $\{u, v\}$ is non-adjacent in $\overline{G}$.
- An **Eulerian tour** is a cycle that is allowed to pass through each vertex multiple times, but must use each edge exactly once.
- An **Eulerian path** is a path that uses each edge exactly once.
- The graph $G = (V, E)$ is **r-regular** if the degree of every vertex $v \in V$ is r, i.e., $deg(v) = r$.
- The graph $H = (W, F)$ is a **subgraph** of G if $W \subseteq V$ and $F \subseteq E$.
- $H = (W, F)$ is a **spanning subgraph** of G if it is a subgraph and $W = V$.
- $H = (W, F)$ is an **induced subgraph** of G if it is a subgraph and for all $x, y \in W$ we have that x and y are adjacent in G if and only if they are adjacent in H.
- C_n denotes a **cycle** on n vertices.
- K_n denotes a **complete graph** on n vertices, i.e., $V = \{1, 2, \dots, n\}$ and $E = \binom{V}{2}$.

1. Prove that $\sum_{v \in V} deg(v) = 2m$.
2. Prove that $\omega(G) = \alpha(\overline{G})$.

3. Prove that $\alpha(C_n) = \lfloor n/2 \rfloor$ for $n \geq 3$.
4. Prove that $\omega(C_3) = 3$ and $\omega(C_n) = 2$ for $n \geq 4$.
5. Prove that $\alpha(G) + \tau(G) = n$.
6. Prove that $\tau(G) \leq 2\mu(G)$.
7. Prove that $\alpha(G) \cdot \chi(G) \geq n$.
8. Prove that $\alpha(G) \geq \frac{n}{d_{\max}+1}$, where $d_{\max} = \max\{deg(v) \mid v \in V\}$ is the maximum degree. Find an independent set of this size in linear time.
9. Prove that $\mu(G) \leq \tau(G)$.
10. Prove that G is bipartite if and only if G contains no odd cycles.
11. Count (a) the number of induced subgraphs of G; (b) the number of spanning subgraphs of G. Both answers should be very simple expressions in terms of n and m.
12. Prove that if a vertex $v \in V$ has odd degree in G then there exists another vertex w, also of odd degree, such that v and w are connected by a path.
13. Let u and v be two opposite corners of the $k \times \ell$ grid graph. Count the number of shortest paths between u and v.
14. Prove that if G is connected then $m \geq n - 1$.
15. Prove that if G has no cycles then $m \leq n - 1$.
16. Prove that if G has maximum degree d then G is $(d + 1)$-colorable. *Hint:* give a simple greedy coloring.
17. Prove that if G is a DAG then it has at most $\binom{n}{2}$ edges. For every n, find a DAG that has exactly $\binom{n}{2}$ edges.
18. Prove that a directed graph G admits a topological order if and only if G has no cycles.
19. Create an arbitrary directed graph on 10 vertices. Execute DFS on this graph. Write down discovery and finishing times of all vertices. Classify each edge as a tree, cross, forward or back edge.
20. Give a linear time algorithm to decide if a given undirected graph is bipartite.
21. Give a linear time algorithm to decide if given an undirected graph G and an edge e there exists a cycle containing e.
22. Design an efficient dynamic programming algorithm to compute the length of a longest path in a DAG.
23. You are given a tree $T = (V, E)$ (in adjacency lists format), along with a designated root $r \in V$. You should preprocess the tree so that you can answer queries of the type "is u an ancestor of v?" in constant time. Your preprocessing should take linear time. Explain how to do this.
24. Prove that in any connected undirected graph $G = (V, E)$ there is a vertex v whose removal leaves G connected.

25. Find a strongly connected digraph $G = (V, E)$ such that for every $v \in V$ removal of v leaves a digraph that is not strongly connected.

26. You are given a vertex-weighted tree $G = (V, E)$ with a weight function $w : V \to \mathbb{R}$ and a designated root node $r \in V$. Define array $Z[1..n]$ such that

$$Z[i] = \text{ the maximum } w(u) \text{ value over all nodes } u \text{ in the subtree rooted at } i.$$

Give a linear-time algorithm to calculate all entries in the $Z[1..n]$ array.

27. Give a linear-time algorithm to find an odd-length cycle in a digraph.

28. Give an efficient algorithm that given a DAG $G = (V, E)$ and two vertices s and t, outputs the number of different directed paths from s to t.

29. You are given a vertex-weighted digraph $G = (V, E)$ with a weight function $w : V \to \mathbb{R}$. Define the following array:
$$Z[u] = \min\{w(v) \mid v \text{ is reachable from } u\}.$$
   (a) Design a linear-time algorithm to compute all entries in the $Z[]$ array if G is acyclic.
   (b) Design a linear-time algorithm to compute all entries in the $Z[]$ array for general G.

30. Show that an undirected graph is Eulerian if and only if all its vertices have an even degree.

31. Give an "if and only if" characterization (similar to the previous exercise) of undirected graphs that have Eulerian paths. Prove the characterization.

32. Give an "if and only if" characterization (similar to the previous exercise) of directed graphs that have Eulerian tours. Prove the characterization.

33. Show that if an undirected graph has n vertices and k connected components then the number of edges is at least $n - k$.

34. Suppose that you have computed an MST and shortest paths from the source s for some weighted undirected graph $G = (V, E)$ with edge-weights $w : E \to \mathbb{R}$. Suppose each edge weight is increased by 1, i.e., $w'(e) = w(e) + 1$.
   (a) Does the MST change? Give an example where it changes or prove that it cannot change.
   (b) Do the shortest paths change? Give an example where they change or prove that they cannot change.

35. Prove that if all edge weights are distinct in a weighted undirected graph then it has a unique MST.

36. Design an algorithm to compute a maximum weight spanning tree.

37. Given a weighted undirected graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$ and an edge $e \in E$, the goal is to compute a minimum spanning tree subject to it containing e. Design an efficient algorithm for this task.

38. Consider a directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$ and the source vertex $s \in V$. Suppose that the only negative-weight edges are those leaving s. Can Dijkstra's algorithm started at s fail on such input? Either give an example on which Dijkstra fails or prove that it cannot fail.

39. You are given a strongly connected graph $G = (V, E)$ with positive weight edges along with a special vertex $v \in V$. Give an efficient algorithm to find all pairs shortest paths subject to these paths necessarily passing through v.

40. Given an undirected graph $G = (V, E)$ with positive edge weights $w : E \to \mathbb{R}_{>0}$ and the source vertex $s \in V$, the goal is to decide for each vertex $u \in V$ whether shortest path from s to u is unique or not. In other words, you should compute the entire array $Z[]$ where $Z[u] = true$ if and only if the shortest path from s to u is unique. Design an efficient algorithm for this problem.