Problem 1. Let \( G = (V, E) \) be a directed graph. Suppose that we perform BFS starting from a source vertex \( s \), and obtain a BFS-tree \( T \). For any vertex \( v \in V \), denote by \( l(v) \) the level of \( v \) in the BFS-tree. Prove that BFS enqueues the vertices \( v \) of \( V \) in non-descending order of \( l(v) \).

Solution. Take any vertices \( u, v \) such that \( l(u) > l(v) \). Let \( v_1, v_2, \ldots, v_{l(v)} \) be the vertices on the path from the root to \( v \) in \( T \); note that \( v_1 = s \) and \( v_{l(v)} = v \). Let \( u_1, u_2, \ldots, u_{l(v)} \) be the last \( l(v) \) vertices on the path from the root to \( u \) in \( T \); note that \( u_1 \neq s \) and \( u_{l(v)} = u \). It thus follows that \( v_1 \) is en-queued before \( u_1 \). Remember that BFS en-queues \( v_2 \) when de-queuing \( v_1 \), and similarly, enqueues \( u_2 \) when de-queuing \( u_1 \). By the FIFO property of the queue, we know that \( v_2 \) is en-queued before \( u_2 \). By the same reasoning, \( v_3 \) is en-queued before \( u_3 \), \( v_4 \) before \( u_4 \), etc. This means that \( v \) is before \( u \).

Problem 2. Let \( G = (V, E) \) be a directed graph. Suppose that we perform BFS starting from a source vertex \( s \), and obtain a BFS-tree \( T \). For any vertex \( v \in V \), prove that the path from \( s \) to \( v \) in \( T \) is a shortest path from \( s \) to \( v \) in \( G \).

Solution. We will instead prove the following claim: all the vertices with shortest path distance \( l \) from \( s \) are at level \( l \) (recall that the root is at level 0). This will establish the conclusion in Problem 3 because the path from \( s \) to a level-\( l \) node \( v \) in \( T \) has length \( l \).

We will prove the claim by induction on \( l \). The base case where \( l = 0 \) is obviously true.

Assuming that the claim holds for all \( l \leq k - 1 \) \( (k \geq 1) \), next we prove that the claim is also true for \( l = k \). Let \( v \) be a vertex with shortest path distance \( k \) from \( s \). Consider all the shortest paths from \( s \) to \( v \). From every such shortest path, take the vertex immediately before \( v \) (i.e., the predecessor of \( v \) in that path), and put that vertex into a set. Let \( S \) be the set of all such “predecessors of \( v \)” collected. Let \( u \) be the vertex in \( S \) that is the earliest one entering the queue. We know that the shortest path distance from \( s \) to \( u \) is \( k - 1 \). It thus follows from the inductive assumption that \( u \) is at level \( k - 1 \) of \( T \).

Consider the moment when \( u \) is removed from the queue in BFS. We will argue that the color of \( v \) must be white. Hence, BFS makes \( v \) a child of \( u \), thus making \( v \) at level \( k \).

Suppose for contradiction that the color of \( v \) is gray or black. This means that \( v \) has been put into the queue when another vertex \( u' \) was de-queued earlier. From the conclusion of Problem 2 and the definition of \( u \), we know that \( l(u') < l(u) \). From the inductive assumption, this means that the shortest path distance of \( u' \) from \( s \) that is less than \( k - 1 \), implying that the shortest path distance from \( s \) to \( v \) is less than \( k \), thus giving a contradiction.

Problem 3. Let \( G = (V, E) \) be an undirected graph. We will denote an edge between vertices \( u, v \) as \( \{u, v\} \). Next, we define the single source shortest path (SSSP) problem on \( G \). Define a path from \( s \) to \( t \) as a sequence of edges \( \{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_t, v_{t+1}\} \), where \( t \geq 1 \), \( v_1 = s \), and \( v_{t+1} = t \). The length of the path equals \( t \). Then, the SSSP problem gives a source vertex \( s \), and asks to find shortest paths from \( s \) to all the other vertices in \( G \). Adapt BFS to solve this problem in \( O(|V| + |E|) \) time. Once again, you need to produce a BFS tree where, for each vertex \( v \in V \), the path from the root to \( v \) gives a shortest path from \( s \) to \( v \).
**Solution.** Same as BFS, except that when a vertex $v$ is de-queued, we inspect all its neighbors (as opposed to its out-neighbors as in the directed version).

**Problem 4 (Connected Components).** Let $G = (V, E)$ be an undirected graph. A connected component (CC) of $G$ includes a set $S \subseteq V$ of vertices such that

- For any vertices $u, v \in S$, there is a path from $u$ to $v$, and a path from $v$ to $u$.
- (Maximality) It is not possible to add any vertex into $S$ while still ensuring the previous property.

For example, in the above graph, $\{a, b, c, d, e\}$ is a CC, but $\{a, b, c, d\}$ is not, and neither is $\{g, f, e\}$.

Prove: Let $S_1, S_2$ be two CCs. Then, they must be disjoint, i.e., $S_1 \cap S_2 = \emptyset$.

**Solution.** Suppose that a vertex $v$ is in $S_1 \cap S_2$. Then, for any vertex $u_1 \in S_1$ and $u_2 \in S_2$, we know:

- There is a path from $u_1$ to $u_2$ by way of $v$.
- There is a path from $u_2$ to $u_1$ by way of $v$.

This violates the fact that $S_1$ and $S_2$ must be maximal.

**Problem 5.** Let $G = (V, E)$ be an undirected graph. Describe an algorithm to divide $V$ into a set of CCs. For example, in the example of Problem 5, your algorithm should return 3 CCs: $\{a, b, c, d, e\}$, $\{g, f\}$, and $\{h, i, j\}$.

**Solution.** Run BFS starting from an arbitrary vertex in $V$. All the vertices in the BFS-tree constitute the first CC. Then, start another BFS from an arbitrary vertex that is still white. All the vertices in this BFS-tree constitute another CC. Repeat this until $V$ has no more white vertices.