Exercise 1. *Suppose that we create a spanning tree T using a BFS starting at some vertex v in a graph G. For example, let v be the root, and when a vertex w is marked in line 11 of the algorith, we record its predecessor u and add the edge wu to T. Show that the distance $d(v, w)$ in T is equal to the distance $d(v, w)$ in G, for any $w \in V$. Note that distances between other pairs (two vertices not involving v) is not necessarily preserved.*

Solution. We will assume that G is connected. (When G is disconnected, vertices in other components are recorded as distance ∞, so the claim holds.) We claim that vertices are explored in order of distance, and that $d_T(v, w) = d_G(v, w)$ for all $w \in V(G)$.

We proceed by induction on the distance from v. Clearly, this is true for all vertices at distance 0 and distance 1. Suppose this holds for all vertices at distance k from v and consider a w such that $d_G(v, w) = k+1$. Then w has neigbor(s) u in G such that $d_G(v, u) = d_T(v, u) = k$ by the inductive hypothesis. Also by the inductive hypothesis, these were explored before any neighbors of w at distance greater than k (and w does not have neighbors at distance less than k). Therefore, we let u' be the first neighbor of w at distance k from v that is explored by the BFS, u' is then also the first neighbor of w overall explored, and therefore recorded as the predecessor of w, giving us $d_G(v, w) = d_T(v, w) = k + 1$.

Conversely, after all vertices at distances $0, \ldots, k$ has been explored, the queue contains exactly the set of vertices at distance $k + 1$, so they are next before any vertices of greater distance. This completes the two parts of the claim.

Exercise 2. (Erickson 5.8) *Let G be a connected graph, and let T be a depth-first spanning tree of G rooted at some node v. Prove that if T is also a breadth-first spanning tree of G rooted at v, then $G = T$.*

Solution. Suppose that $G \neq T$. Then $E(G) \setminus E(T) \neq \emptyset$. Let $xy \in E(G) \setminus E(T)$, and WLOG suppose that x was explored before y. We claim that T is a DFS spanning tree if and only if x is a predecessor of y in the tree (i.e. x lies on the unique path from y to v). Since $xy \notin E(T)$, y was added to the stack after x, and therefore its parent in T is either x or some vertex succeeding x (in the subtree rooted at x). If T is a BFS and x is explored first, then either y is already in the tree (but not yet explored) or it is added as a child of x. Since $xy \in E(G) \setminus E(T)$ it must be the former, which means that when x is explored, y is already in the tree, and therefore cannot be in the subtree rooted at x.

Exercise 3. *Let $V = \{0, 1\}^d$. In other words, V is the set of all binary strings of length d. The d-dimensional hypercube Q_d is the graph on V such that two vertices in V share an edge if and only if the strings differ in exactly one bit. Show that the hypercube Q_d is a bipartite graph, for $d = 1, 2, \ldots$*

Solution. We make two sets V_0 and V_1, where a vertex is in V_0 if its binary string has an even number of 1s, and V_1 is it has an odd number of 1s. Clearly, if two strings differ in exactly one bit, their number of 1s have opposite parities. Therefore, this gives a bipartition of the graph.

Exercise 4. *Suppose instead that we try Kruskal in the reverse direction. Start with $T = G$, and while there are edges in T in cycles, delete such an edge of greatest weight. Does this also result in a minimum spanning tree?*

Solution. Yes, this works with a very similar proof.

Exercise 5. *Do Dijkstra's and/or Kruskal's algorithm work if we allow negative weights?*

Solution. Let e be the edge of minimum weight, and suppose this is negative. In every step of Kruskal's algorithm and its proof, we can check that the steps still hold if we add $w(e) + 1$ to all edge weights, which simulates a graph with only positive weights.

Dijkstra is different, because we have to treat our weights as distances and assume the triangle inequality in our proof. For example, even at the start we make the assumption that the distance from the starting vertex v to itself is 0. With negative weights, this falls apart. If v has a negative-weight edge to a neighbor w, we could create an arbitrarily "short" path by walking back and forth across vw.