Discrete Mathematics Exercise 17

Qiu Yihang, 2020/12/2

1. ***Proof:***

Suppose the starting point of DFS is r, **i.e.** the root of the spanning tree is r.

v, w must be starting points of two backwards step respectively. Let v be the vertex which is the starting point of the earlier backwards step.

Suppose it is the k-th step.

At this step, there are four cases:

CASE 01. w is the root. Obviously, w is included in the unique simple path from r to u in T, **i.e.** w is u's ancestor in T or $w = u = r$.

CASE 02. w appears as the termination of exactly one forward step but as no starting point of any backwards step in the first k steps. Then w must be included in the unique simple path from r to v.

Since u is v's ancestor, **i.e.** u is included in the unique simple path from r to v, either **(i)** $u = w$ **(ii)** u is included in the unique simple path from r to w **(iii)** w is included in the unique simple path from r to u.

In other words, either **(i)** $u = w$, **(ii)** w is u's descendant in T or **(iii)** w is u's ancestor in T.

CASE 03. w appears as the termination of exactly one forward step and as the starting point of exactly one backwards step in the first k steps. Impossible. (Contradict to the supposition)

CASE 04. w is still not visited. Impossible. (if this is the case, since there exists an edge from v to w, the k-th step should be a forward step)

In conclusion, either **(i)** $u = w$, **(ii)** w is u's descendant in T, or **(iii)** w is u's ancestor in T.

QED

2. ***Proof:***

Suppose the root of T is r.

We know every vertex in V is the termination of exactly one forward step and in DFS process, a forward step visits a yet unvisited vertex, **i.e.** the forward step to a vertex v is the first time the DFS process visits v $(v \neq r)$.

Suppose the forward step whose termination is v is the k-th step.

When u is v's ancestor in T, u is definitely included in the unique simple path from r to v.

Therefore, either $u = r$ or u appears as the termination of exactly one forward step in the first k-th steps (the first $(k-1)$-th steps in fact, since $u \neq v$).

Either way, u is firstly visited before the k-th step, **i.e.** the first time visiting v.

In other words, the first time visiting u happens before the first time visiting v in the DFS process.

QED

3. ***Proof:***

Suppose the tree generated by DFS process is T, with a root r.

Def. We say "u is a self-descendant of v" **iff.** "either $u = v$ or u is a descendant of v."

Since there exist a forward move from u to v and another **distinct** forward move from u to w, v and w are self-descendants of u's two **distinct** children v^* and w^* respectively. $(v^* \neq w^*)$

CASE 01. The simple path from v to w only contains tree edges.

Then the path is also the unique simple path from v to w in T.

There exists a simple path from v to u: $v, x_1, x_2, \dots, x_k, v^*, u$ and a simple path from u to w: $u, w^*, y_1, y_2, \dots, y_s, w$.

Since $v^* \neq w^*$, the two paths have no shared edges. Thus, $v, x_1, x_2, \dots, x_k, v^*, u, w^*, y_1, y_2, \dots, y_s, w$ is the unique simple path from v to w, passing through u.

CASE 02. The simple path from v to w contains back edges.

Since a back edge only connects x with its ancestor, its descendant or itself in T, we know there are no edges connecting a self-descendant of v^* with a self-descendant of w^*.

Therefore, the simple path from v to w must pass through at least one common ancestor of v and w in T. Otherwise, exists a back edge connects a x to a vertex which is not itself nor its ancestor nor its descendant.

Since the common ancestor of v and w is either u or its ancestors, the simple path from v to w must pass through u or u's ancestors.

In conclusion, the simple path from v to w either passes through u or passes through at least one of u's ancestors in the tree generated by the DFS process.

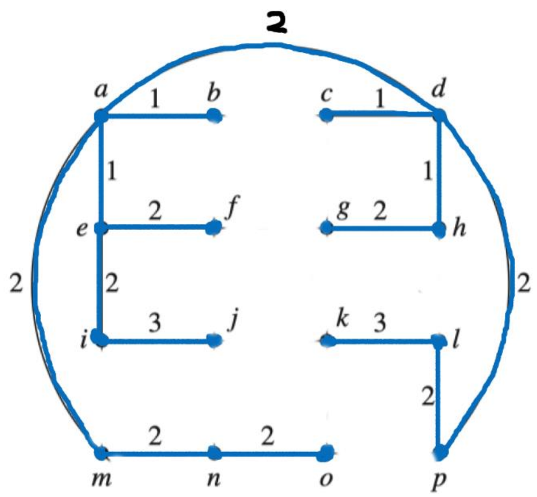
QED

4. ***Solution:***

Pick f as the starting point.

1) ef is added. 2) ae is added. 3) ab is added. 4) ad is added.
5) cd is added. 6) dh is added. 7) am is added. 8) gh is added.
9) ei is added. 10) dp is added. 11) pl is added. 12) mn is added.
13) no is added. 14) ij is added. 15) kl is added.

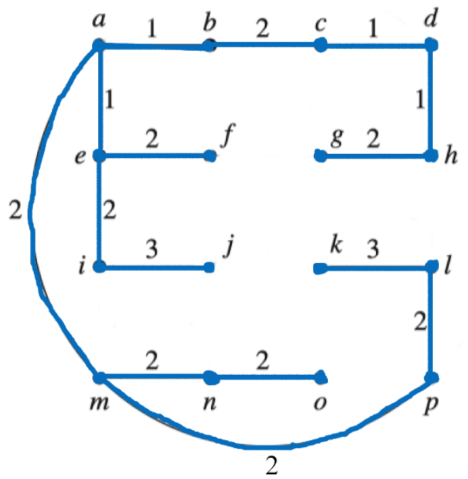
A minimum spanning tree generated using *Prim's Algorithm* is as follows:



5. ***Solution:***

1) ab is added. 2) ae is added. 3) cd is added. 4) dh is added.
5) ef is added. 6) gh is added. 7) ei is added. 8) bc is added.
9) mn is added. 10) no is added. 11) lp is added. 12) am is added.
13) mp is added. 14) ij is added. 15) kl is added.

A minimum spanning tree generated using *Kruskal's Algorithm* is as follows:



6. ***Proof:***

Proof by Contradiction.

Def. $\textbf{Sum}(T)$ = the sum of all weights in the tree T, $\textbf{Max}(T)$ = the maximum weight of all the edges in T, $w(e)$ = the weight of the edge e.

Suppose the connected weighed graph is $G = (V, E)$.

Assume there exist two minimum spanning trees T_1, T_2 in G. Let $T_1 = (V, E_1), T_2 = (V, E_2)$. Then $\textbf{Sum}(T_1) = \textbf{Sum}(T_2)$.

Since the weights of edges in G are all different, we know $\textbf{Max}(T_1) \neq \textbf{Max}(T_2)$.

Suppose $\textbf{Max}(T_1) > \textbf{Max}(T_2)$. Let e_1 be the edge with the maximum weight in T_1, e_2 be the edge with the maximum weight in T_2. Then $w(e_1) > w(e_2)$.

Let $G_1 = (V, E')$, where $E' = E_1 \backslash \{e_1\}$.

Since $T_1 = (V, E_1)$ is a tree, we know there are two connected components c_1, c_2 in G_1. (if less than two, there exists a circuit in T_1; if more than two, T_1 is not connected)

We can always find an edge e^* in T_2 connecting a vertex in c_1 and a vertex in c_2. Otherwise, vertices in c_1 and vertices in c_2 are not connected in T_2, **i.e.** T_2 is not connected. **Contradiction.**

Let $T^* = (V, E^*)$, where $E^* = E' \cup \{e^*\} = E_1 \backslash \{e_1\} \cup \{e^*\}$. Then T^* is connected.

Since T_1 is a tree, G_1 has no circuit. Then T^* has no circuit. (otherwise, G_1 is connected.)

Therefore, T^* is a tree and is therefore a spanning tree of G.

Meanwhile, $\textbf{Sum}(T^*) = \textbf{Sum}(T_1) - w(e_1) + w(e^*) < \textbf{Sum}(T_1) - w(e_1) + w(e_2) < \textbf{Sum}(T_1) = \textbf{Sum}(T_2)$, **i.e.** T_1, T_2 are not minimum spanning trees of G. **Contradiction.**

Therefore, there is a unique minimum spanning tree in a connected weighted graph if the weights of the edges are all different.

QED