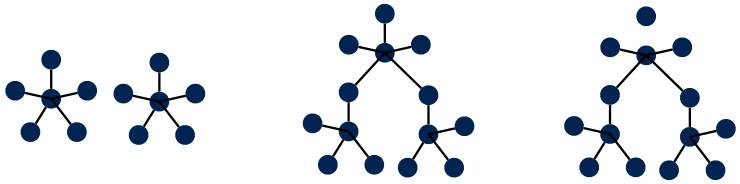
Solution to Homework 2

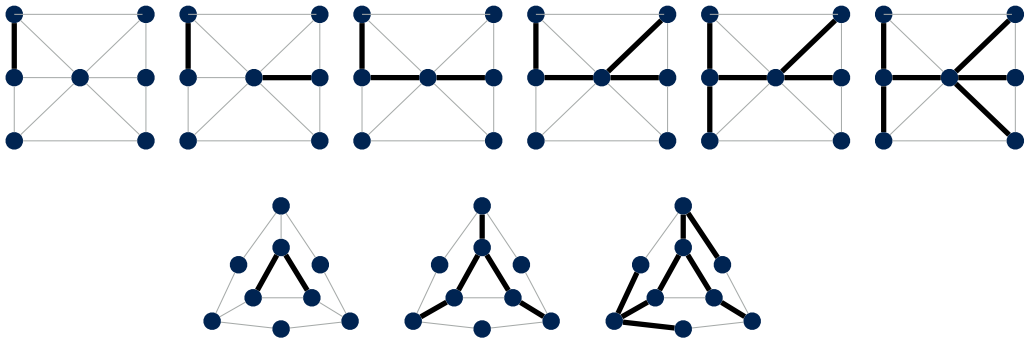
Yanheng Wang 517021910537

Problem 1. Since for any forest it holds that $m \leq n - 1$, the requirements in (a) and (c) cannot be fulfilled. The illustrations for (b)(d)(e) are shown in the figure below.



Problem 2. *Proof.* Let's designate an arbitrary vertex as the root of T. We can thus define the depth for each vertex, as usual. Then we take v to be the deepest vertex of T – it is of course a leaf. Let p be its parent. Since $\deg(p) \geq 3$ where p's parent and v each takes up one degree, we conclude that p has an at least one more child $v' \neq v$. But v' must be a leaf as well; otherwise v' will be deeper than v, violating our requirement. □

Problem 3. The following figure shows the procedure of Kruskal's algorithm; some steps are combined into one.



Both MSTs are unique in their corresponding graphs. The following lemma is perhaps the most uniform way to show this.

Definition 1. Given a weighted graph G and a constant $c \in \mathbb{N}$, we define $G_c := (V, E)$ where $V := V(G)$ and $E := \{e \in E(G) \mid w(e) \leq c\}$.

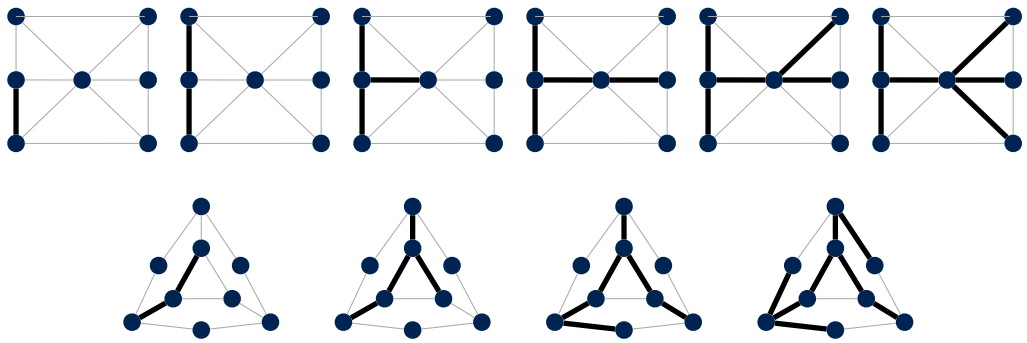
Lemma. Suppose T is a minimum spanning tree of G, and $c \in \mathbb{N}$. Then T_c and G_c have exactly the same components.

Proof. Since T is a subgraph of G, it is obvious that every component in T_c is contained by some component in G_c.

Now we prove the other direction, i.e. every component in G_c is contained by some component in T_c. Suppose, for the sake of contradiction, that $\exists u, v \in G_c$ connected in G_c yet disconnected in T_c. Without loss of generality, we assume that u and v are adjacent. Since $\{u, v\} \in G_c$, we know $w(u, v) \leq c$. Now we let X and Y be components in T_c where u and v reside, respectively. Why aren't X and Y connected in T_c? The only reason is that T connects X and Y via edge(s) with weight $> c$! Interchanging that connection with $\{u, v\}$ will gives us a spanning tree with strictly smaller weight, contradicting the optimality of T. □

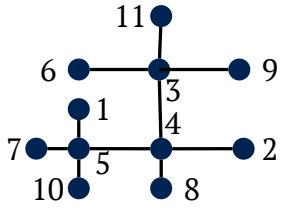
As a direct consequence, given $c \in \mathbb{N}$ and any two MSTs T, T' of G, the components of T_c and T'_c are the same. Using this and by lifting c from 0 all the way up, it is easy to show that the MSTs in the problem are unique.

Problem 4. We start working from the bottom-left vertex. The process is demonstrated below.



Problem 5. (a) $2, 2, 1, 3, 1, 4, 4, 1, 5$. (b) $5, 8, 4, 3, 3, 3, 9$.

Problem 6.



Problem 7. Suppose $e = \{u, v\}$. Fix a labelling f such that $f(u) = n - 1$ and $f(v) = n$; that is, u and v occupy the two largest labels. For any spanning tree T on K_n, we denote its Prüfer code as $\sharp T$. Then we have the following claim:

Claim. $e \in T \iff \sharp T$ ends with $n - 1$ or n.

Proof. If $e \in T$, then neither u nor v will be removed in the procedure we construct $\sharp T$. (Because u and v are adjacent and have the largest labels, they cannot be removed unless

both of them become leaves.) It follows immediately that the final vertex removed was a neighbour of u or v. Therefore, $\sharp T$ ends with $f(u) = n-1$ or $f(v) = n$.

Conversely, if $e \notin T$, then we find the unique path, P, from u to v. Clearly $|P| \geq 3$. Now we analyse the procedure of generating $\sharp T$. Before we could touch on P, we must first remove all vertices outside P because the path P is "guarded" by two largest "sentinels". After that, the only possible move would be deleting u, and continue all the way until there are two vertices left. In other words, we must work from the u-side and proceed to the v-side in order. Therefore, $\sharp T$ ends with neither $n-1$ nor n. □

But the number of spanning trees on $K_n - e$ is exactly the number of spanning trees on K_n which does *not* contain e. Hence, that number equals $n^{n-3} \cdot (n-2)$, i.e. the number of Prüfer codes that does *not* end with $n-1$ or n.

Problem 8. *Proof.* Note that the total count of labelled spanning trees on n vertices equals the count of spanning trees on K_n. Applying Kirchhoff's Matrix Tree Theorem, we obtain

$$\# = \det \begin{pmatrix} n-1 & -1 & \cdots & -1 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ -1 & \cdots & -1 & n-1 \end{pmatrix}_{(n-1)\times(n-1)} =: N_n$$

To compute the value of N_n, we subtract the second row from the first row. Now the first row becomes $(n, -n, 0, \cdots, 0)$. Then we extract the factor n from it, yielding

$$N_n = n \cdot \det \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & n-1 & -1 & \cdots & -1 \\ -1 & -1 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -1 \\ -1 & -1 & \cdots & -1 & n-1 \end{pmatrix}$$

Expanding the first row gives $N_n = n \cdot (N_{n-1} + 0)$. Combining with the boundary case that $N_2 = 1$, we have $\# = N_n = \prod_{i=2}^{n-1} n = n^{n-2}$. □

Problem 9. The statement is true.

Proof. We know G is bipartite iff it contains no odd cycle. Next we show that G contains odd cycle iff there are two adjacent vertices equidistant to some other vertex.

($\Rightarrow$) Let C be a *shortest* odd cycle in G. We claim that $\forall u, v \in C$, there's always a shortest path from u to v that goes along C. Otherwise, the shortest path, P, would break C into two smaller cycles, one being even and the other being odd, contradicting our assumption that C is minimal.

Now we take an arbitrary $w \in C$. Starting from w, we walk clockwise for $(|C|-1)/2$ steps and arrive at u; walk counterclockwise for $(|C|-1)/2$ and reach v. It is clear that u and v are adjacent. In addition, by the claim above, $d(u,w) = d(v,w) = (|C|-1)/2$.

($\Leftarrow$) Suppose $\exists u, v, w \in V : \{u, v\} \in E \ \wedge \ d(u,w) = d(v,w) =: d$. Then we may find an odd cycle $u \rightsquigarrow w \rightsquigarrow v \to u$ (whose length is $2d+1$). □