CS499 Homework 7 (First Draft)

Intersteller

Exercise 7.1

(1) Since e is in the minimum spanning tree, we split the minimum spanning tree into two components by deleting e. Let the vertices in the two components consist S and $V \backslash S$ respectively. Since there is no circle in a tree, obviously e is the only edge which is good and cross this cut, which means no edge from X crosses this cut.

(2) Suppose e is not the minimum weight edge crossing this cut, assume there is an edge e' which has less weight and crosses this cut. e' can replace e and consists a spanning tree with less weight. This means e is not in the minimum spanning tree, which means e is not good, which contradicts the condition.

Exercises 7.4

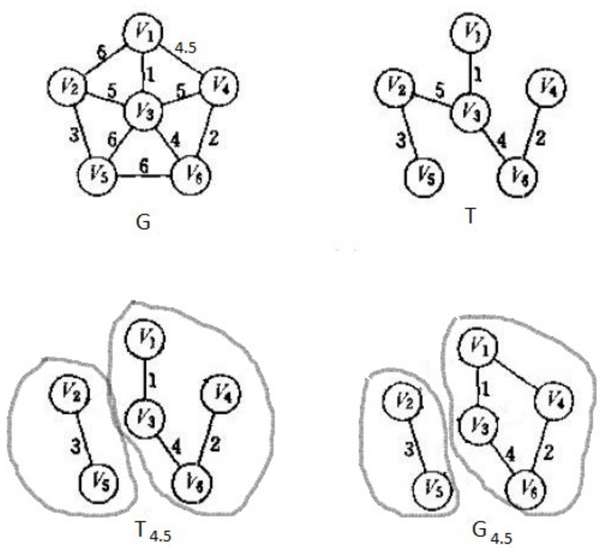

Figure 1:

Exercises 7.5 Obviously, if two vertices are connected in T_c, they are connected in G_c, since T_c is in G_c.
Suppose u,v are connected in G_c, but not connected in T_c. Let two connected components in T_c contain u and v respectively be A and B. Let e be an edge in G_c that connect A and B. Using defination, $w(e) \leq c$. Since A and B are not connected in T_c, there must be an edge e' in T that connects A and B, and $w(e') > c$. So, e'¿e. Obviously T which contains e' is not the minimum spanning tree, since e' can be replaced by e with less weight. This contradicts the condition. So, if two vertices are connected in G_c, they are connected in T_c.

Exercise 7.8

As the picture shows , for $\forall c, m_c(T) = m_c(T\prime)$.

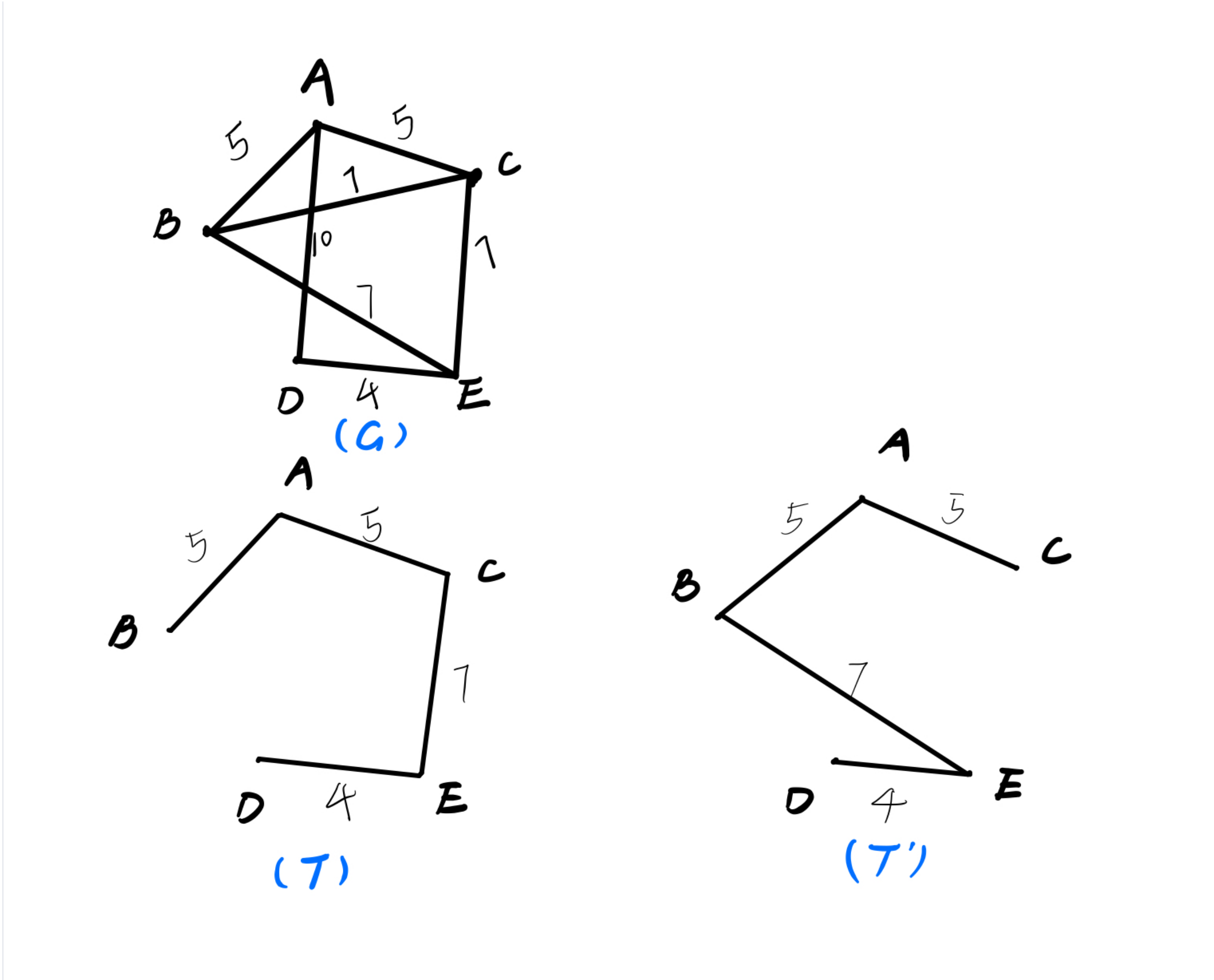

Figure 2:

Exercise 7.9

Sort by the weight of $T's$ edges and $T\prime s$ edges , we have $(a1, a2, a3, \cdots, a_{n-1})$, $(b1, b2, b3, \cdots, b_{n-1})$. Suppose $a_i \neq b_i$, $\forall k < i, a_k = b_k$ and $w(a_i) \geq w(b_i)$, there are two situations:

(1) edge b_i exists in the T , then we can find $j(j > i)$ and $a_j = b_i$. Because $w(b_i) = w(a_j) \geq w(a_i) \geq w(b_i), w(a_i) = w(b_i) = w(a_j)$.So we can exchange a_i and a_j and new sequence is still ordered . $T's$ and $T\prime s$ i position is the same edge.

(2) edge b_i doesn't exist in the T , then we add b_i to T to form a cycle . Because T is a minimum spanning tree , w(edge in the cycle)$\leq w(b_i)$. And we can find $a_j(j > i$ and a_j doesn't exist in the $T\prime$ and a_j in the cycle) . Because $w(b_i) \geq w(a_j) \geq w(a_i) \geq w(b_i), w(b_i) = w(a_i) = w(a_j)$. So we can change a_j with b_i . Turn to the situation (1).

So we know the ordered edge weight list of any two minimum spanning trees is the same.

Obviously, $m_c(T) = m_c(T\prime)$.

Exercise 7.10

Suppose there are two minimum spanning tree , sort by the weight of $T's$ edges and $T\prime's$ edges , we have$(a1, a2, a3, \cdots, a_{n-1})$, $(b1, b2, b3, \cdots, b_{n-1})$.$\exists i, a_i \neq b_i$,based on the 7.9,the ordered edge weight list of any two minimum spanning trees is the same,so $w(a_i) = w(b_i)$.But no two edges of G have the same weight , so there is contradiction . So G has exactly one minimum spanning tree!

Exercise 7.11

A function with a core of size 1 forms a rooted tree (the element in core is the root). There are n^{n-2} trees we can form. For each tree we can choose any one of n nodes to be the root, so there are totally $n \cdot n^{n-2} = n^{n-1}$ different rooted trees, which means there are n^{n-1} such functions.

Exercise 7.12

A function with a core of size 2 forms a tree whose head and but are connected. There are n^{n-2} trees we can form. For each tree we can choose any one of $n-1$ edges to be the edge connecting the head and the but. Since the head's order number is smaller than but's,once we choose an edge, the head and but are fixed.So there are totally $(n-1) \cdot n^{n-2} = (n-1) \cdot n^{n-2}$ different rooted trees, which means there are $(n-1) \cdot n^{n-2}$ such functions.

Exercise 7.13

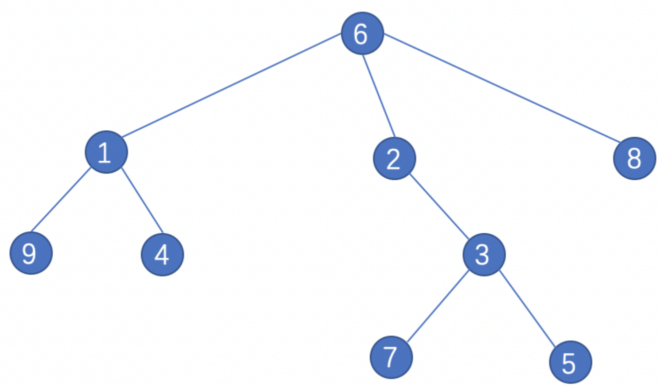

Figure 3:

Exercise 7.14

The degree of vertex i is equal to appearance times of i in **p** plus one.

The nodes that don't appear in **p** are the leaves of T.

Exercise 7.15

1.

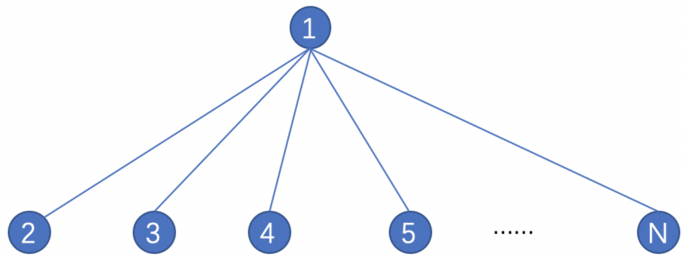

Figure 4:

2.

Figure 5:

3.

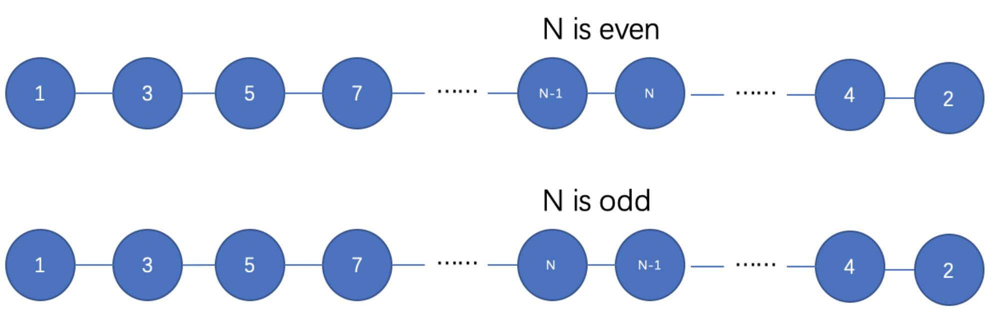

Figure 6:

4.

Figure 7:

5.

Figure 8:

6.

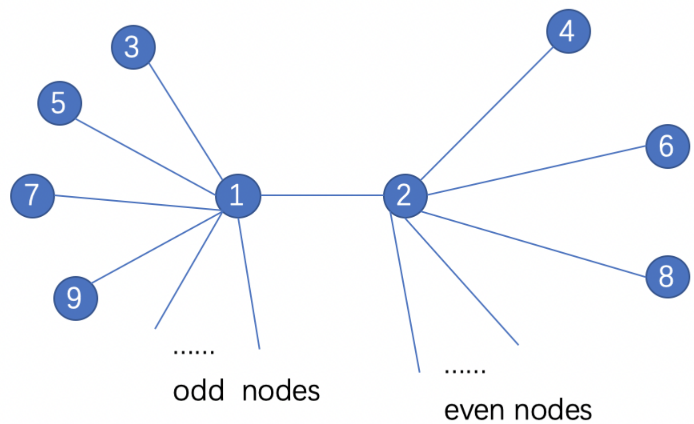

Figure 9:

Exercise 7.16

```
Decode(V,C1C2……Ck):
{
        if n>=3:  l:=min V\{C1,……, Ck}
                  T’:=decode(V\{l},C2C3……Ck)
                  T:=T’ plus the edge from C1 to l
                  return T.
        else if n=2: return the unique tree on V;
}
procedure_tree(n):
{
        for (i=1;i<=n-2;i++) Ci:=randomInt(n-2);
        return decode(V, C1C2……Ck);
} |
```

Figure 10:

Exercise 7.17

Pr[u is a leaf in T] $= (\frac{n-1}{n})^{n-2}$

E[number of leaves] $= n \times (1 - \frac{1}{n})^n \times (\frac{n}{n-1})^2$

As $n \to \infty$, $(1 - \frac{1}{n})^n \to \frac{1}{e}$, $(\frac{n}{n-1})^2 \to 1$

Thus E[number of leaves] $= \frac{n}{e}$

Exercise 7.18

u has degree 2 means u appear one time in code.

$\Pr[u \text{ has degree } 2] = (n-2) \times \frac{1}{n} \times (\frac{n-1}{n})^{n-3} = \frac{(n-2)(n-1)^{n-3}}{n^{n-2}}$

Question:

1. In Exercise 7.11 & 7.12, how can we compute the number of functions with a core of size k? $(1 \leq k \leq n)$ **2.** In the video lecture about ordering, we think a lot on the last problem. We feel that there is no infinite antichain and every infinite subset contain an infinite chain out of intuition. We want to know how to analyse these problems.