Mathematical Foundations of Computer Science

CS 499, Shanghai Jiaotong University, Dominik Scheder

8 Spanning Trees

- Homework assignment published on Monday, 2018-04-16
- Submit questions and first solution by Sunday, 2018-04-22, 12:00, by email to me and the TAs.
- Submit your final solution by Sunday, 2018-04-29.

8.1 Minimum Spanning Trees

Throughout this assignment, let $G = (V, E)$ be a connected graph and $w : E \to \mathbb{R}^+$ be a weight function.

Exercise 8.1. Prove the inverse of the cut lemma: If X is good, $e \notin X$, and $X \cup e$ is good, then there is a cut $S, V \setminus S$ such that (i) no edge from X crosses this cut and (ii) e is a minimum weight edge of G crossing this cut.

Definition 8.2. *For $c \in \mathbb{R}$ and a weighted graph $G = (V, E)$, let $G_c := (V, \{e \in E \mid w(e) \leq c\})$. That is, G_c is the subgraph of G consisting of all edges of weight at most c.*

Lemma 8.3. *Let T be a minimum spanning tree of G, and let $c \in \mathbb{R}$. Then T_c and G_c have exactly the same connected components. (That is, two vertices $u, v \in V$ are connected in T_c if and only if they are connected in G_c).*

Exercise 8.4. Illustrate Lemma **??** with an example!

Exercise 8.5. Prove the lemma.

Definition 8.6. *For a weighted graph G, let $m_c(G) := |\{e \in E(G) \mid w(e) \leq c\}|$, i.e., the number of edges of weight at most c (so G_c has $m_c(G)$ edges).*

Lemma 8.7. *Let T, T' be two minimum spanning trees of G. Then $m_c(T) = m_c(T')$.*

Exercise 8.8. Illustrate Lemma **??** with an example!

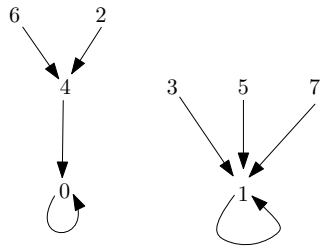
Exercise 8.9. Prove the lemma.

Exercise 8.10. Suppose no two edges of G have the same weight. Show that G has exactly one minimum spanning tree!

8.2 Counting Special Functions

In the video lecture, we have seen a connection between functions $f : V \to V$ and trees on V. We used this to learn something about the number of such trees. Here, we will go in the reverse direction: the connection will actually teach us a bit about the number of functions with a special structure.

Let V be a set of size n. We have learned that there are n^n functions $f : V \to V$. For such a function we can draw an "arrow diagram" by simply drawing an arrow from x to $f(x)$ for every V. For example, let $V = \{0, \ldots, 7\}$ and $f(x) := x^2 \mod 8$. The arrow diagram of f looks as follows:



The *core* of a function is the set of elements lying on cycles in such a diagram. For example, the core of the above function is $\{0, 1\}$. Formally, the core of f is the set

$$\{x \in V \mid \exists k \geq 1 f^{(k)}(x) = x\}$$

where $f^{(k)}(x) = f(f(\dots f(x)\dots))$, i.e., the function f applied k times iteratively to x.

Exercise 8.11. Of the n^n functions from V to V, how many have a core of size 1? Give an explicit formula in terms of n.

Exercise 8.12. How many have a core of size 2 that consists of two 1-cycles? By this we mean that $\text{core}(f) = \{x, y\}$ with $f(x) = x$ and $f(y) = y$.

Hint. For the previous two exercises, you need to use the link between functions $f : [n] \to [n]$ and vertebrates (T, h, b) from the video lecture.

8.3 Counting Trees with Prüfer Codes

In the video lecture, we have seen Cayley's formula, stating that there are exactly n^{n-2} trees on the vertex set $[n]$. We showed a proof using *vertebrates*. For this homework, read Section 7.4 of the textbook, titled "A proof using the Prüfer code".

Exercise 8.13. Let $V = \{1, \dots, 9\}$ and consider the code $(1, 3, 3, 2, 6, 6, 1)$. Reconstruct a tree from this code. That is, find a tree on V whose Prüfer code is $(1, 3, 3, 2, 6, 6, 1)$.

Exercise 8.14. Let $\mathbf{p} = (p_1, p_2, \dots, p_{n-2})$ be the Prüfer code of some tree T on $[n]$. Find a way to quickly determine the degree of vertex i only by looking at $\mathbf{p}$ and not actually constructing the tree T. In particular, by looking at $\mathbf{p}$, what are the leaves of T?

Exercise 8.15. Describe which tree on $V = [n]$ has the

1. Prüfer code $(1, 1, \dots, 1)$.
2. Prüfer code $(1, 2, 3, \dots, n-2)$.
3. Prüfer code $(3, 4, 5, \dots, n)$.
4. Prüfer code $(n, n-1, n-2, \dots, 4, 3)$.
5. Prüfer code $(n-2, n-3, \dots, 2, 1)$.
6. Prüfer code $(1, 2, 1, 2, \dots, 1, 2)$ (assuming n is even).

Justify and explain your answers.

The next two exercises use a bit of probability theory. Suppose we want to sample a random tree on $[n]$. That is, we want to write a little procedure (say in Java) that uses randomness and outputs a tree T on $[n]$, where each of the n^{n-2} trees has the same probability of appearing.

Exercise 8.16. Sketch how one could write such a procedure. Don't actually write program code, just describe it informally. You can assume you have access to a random generator `randomInt(`n`)` that returns a function in $\{1, \ldots, n\}$ as well as `randomReal()` that returns a random real number from the interval $[0, 1]$.

Clearly, a tree T on $[n]$ has at least 2 and at most $n - 1$ leaves. But how many leaves does it have on average? For this, we could use your tree sampler from the previous exercise, run it 1000 times and compute the average. However, it would be much nicer to have a closed formula.

Exercise 8.17. Fix some vertex $u \in [n]$. If we choose a tree T on $[n]$ uniformly at random, what is the probability that u is a leaf? What is the expected number of leaves of T?

Exercise 8.18. For a fixed vertex u, what is the probability that u has degree 2?