MAT344 Lecture 13

2019/July/2

1 Announcements

1. Midterm

2 This week

This week, we are talking about

1. Prüfer codes
2. Includion-Exclusion

3 Recap

Last time we talked about

1. Euler's theorem for graphs
2. Homeomorphic graphs
3. Labeled trees and Prüfer codes

4 Midterm Question 7

Question 7 on the Midterm had a very low average, so let's look at the question and the solution.

Exercise 4.1 (Midterm Question 7)**.** *Fix positive integers n and k. Find the number of k-tuples $(S_1, S_2, \ldots, S_k)$ of subsets S_i of $[n] = \{1, 2, \ldots, n\}$ subject to each of the following conditions **separately**, that is, the three parts are independent problems.*

(a) $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_k$.

(b) The S_i are pairwise disjoint (i.e. $S_i \cap S_j = \emptyset$ for $i \neq j$).

(c) $S_1 \cap S_2 \cap \cdots \cap S_k = \emptyset$.

This is a counting problem ("Find the number of ..."), but what are the objects we are trying to count? If you are having trouble understanding something, it is often helpful to try to reduce the complexity by specifying some or all of the variables. The question has two variables, n and k. Let's consider the case where $k = 3$ and $n = 4$.

Exercise 4.2 (Midterm Question 7 with $k = 3, n = 4$)**.** *Find the number of* triples (S_1, S_2, S_3) *of subsets of $\{1, 2, 3, 4\}$ subject to each of the following conditions **separately**, that is, the three parts are independent problems.*

(a) $S_1 \subseteq S_2 \subseteq S_3$.

(b) The S_i are pairwise disjoint (i.e. $S_1 \cap S_2 = S_1 \cap S_3 = S_2 \cap S_3 = \emptyset$).

(c) $S_1 \cap S_2 \cap S_3 = \emptyset$.

Another simplification that can help you get started when a question is about counting things that satisfy certain conditions is to relax the conditions. For example, if we want to count triples (S_1, S_2, S_3) of subsets of $\{1, 2, 3, 4\}$ *with no restrictions*, then we can choose S_1, S_2, S_3 independently of each other, and each S_i can be chosen in 2^4 ways (the number of subsets of a set with 4 elements). So in this case, the answer is $\left(2^4\right)^3$. Let's consider each of the cases separately.

(a) Let us try some examples that satisfy the condition $S_1 \subseteq S_2 \subseteq S_3$. For example, $\emptyset \subseteq \{3\} \subseteq \{1, 3\}$ works. So does $\{1\} \subseteq \{1, 2\} \subseteq \{1, 2, 3, 4\}$. Maybe at this point it is still unclear how we could count all of these triples of subsets.

Let's focus on just one one the elements of $\{1, 2, 3, 4\}$, namely 2. Let's say we put the number 2 into the set S_1. Then to satisfy the condition, we *must* put it in S_2 and S_3 as well. What if we put 2 in S_2? We have to put it in S_3, but it may or may not be in S_1. Let's give a complete list of the options for the number 2. It may appear in

- None of the subsets.
- In S_3, but not in S_1 or S_2.
- In S_2 and S_3, but not in S_1.
- In all 3 of the subsets.

So we have 4 options for the number 2. Similarly we have 4 options for all the other numbers, so in total, we have 4^4 such subsets.

(b) Let's start again by writing down some examples. $S_1 = \emptyset, S_2 = \{1, 4\}, S_3 = \{3\}$ satisfies the condition. Here we might notice that if we put a number (say, 2) into one of the sets, we can't put it in any other. Of course, we might not put it in any of the subsets. So again we find that we have 4 options for every number, for a total of 4^4 many subsets.

(c) At first it may seem like this is the same as the previous part, but this is not true. The triple $S_1 = \{1, 2\}, S_2 = \{2, 3\}, S_3 = \{1, 3\}$ satisfies this condition, but has subsets that have nonempty pairwise intersections. Again, let's start by figuring out where we can put the number 2. It can be in

- None of the subsets.
- Any one subset (in just S_1, just S_2, or just S_3).
- Any two subsets (in (S_1 and S_2), (S_1 and S_3), or (S_2 and S_3)).

It can not be in all three subsets, since that would violate the condition. So there are 7 options for 2, and the same number for all other numbers for a total of 7^4 options.

Exercise 4.3. *Solve the original Question 7 for general k and n.*

5 Prüfer codes

Let's recall the algorithm that produces a code word from a tree.

We define Prüfer(T) as follows

1. If T is the unique labeled tree on 2 vertices, return the empty string.
2. Else, let v be the leaf of T with the smallest label, and let u be its unique neighbor. Let i be the label of u. Return $(i, \text{Prüfer}(T - v))$.

Let's recall the example we did, consider the labeled tree T on figure 1.
Let v be the vertex with label 2 (it is the leaf with the smallest label). It is adjacent to 6, so

$$\text{Prüfer}(T) = (6, \text{Prüfer}(T - v).$$

Exercise 5.1. *Find the Prüfer code of T.*

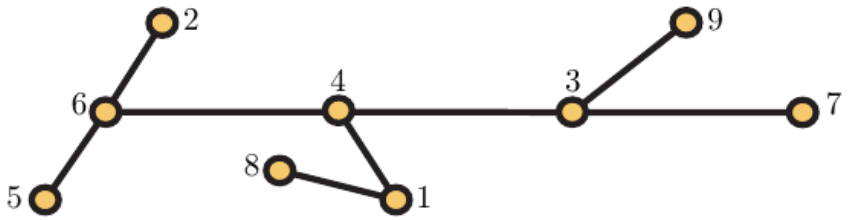


Figure 1: The graph T

Solution:

- The next smallest leaf is labeled 5, also adjacent to 6, so so far our code is 66.
- The next smallest leaf is labeled 6 (note that this just became a leaf), it is adjacent to 4, so our code so far is 664.
- Then 7 is deleted, and our code is 6643.
- Then 8 is deleted, our code is 66431.
- Then 1 is deleted, our code is 664314.
- Then 4 is deleted, our code is 6643143, and the remaining tree has 2 vertices.

So

$$\text{Prüfer}(T) = 6643143.$$

Now we will define an algorithm that produces a labeled tree on $n+2$ vertices from a Prüfer code of length n.

First let's think what labels are included in the string. Notice that leaves of T never appear in the code, but every non-leaf vertex will appear when a leaf next to it is removed. Also, any non-leaf vertex has another vertex attached to it. So the labels that do not appear in the code are precisely the labels of the leaves of the final tree.

We will recursively reconstruct the tree T from the code, starting with the vertex set (the independent graph on $[n+2]$ vertices). We list the labels of the vertices (they form the set $[n+2]$). From the code and the list, we can tell what the label of the smallest-labeled leaf of T is. Call this label i The first entry of the code tells us where to attach i. Then we remove i from the possible label set (since nothing else can attach to it), erase the first letter of the code, and repeat the procedure recursively. Note that now we are in effect reconstructing the tree $T \setminus \{i\}$. Once we get to the empty word and two vertices, we just add an edge between the two vertices still in our list.

Exercise 5.2. *Construct a labeled tree from the Prüfer code 75531.*

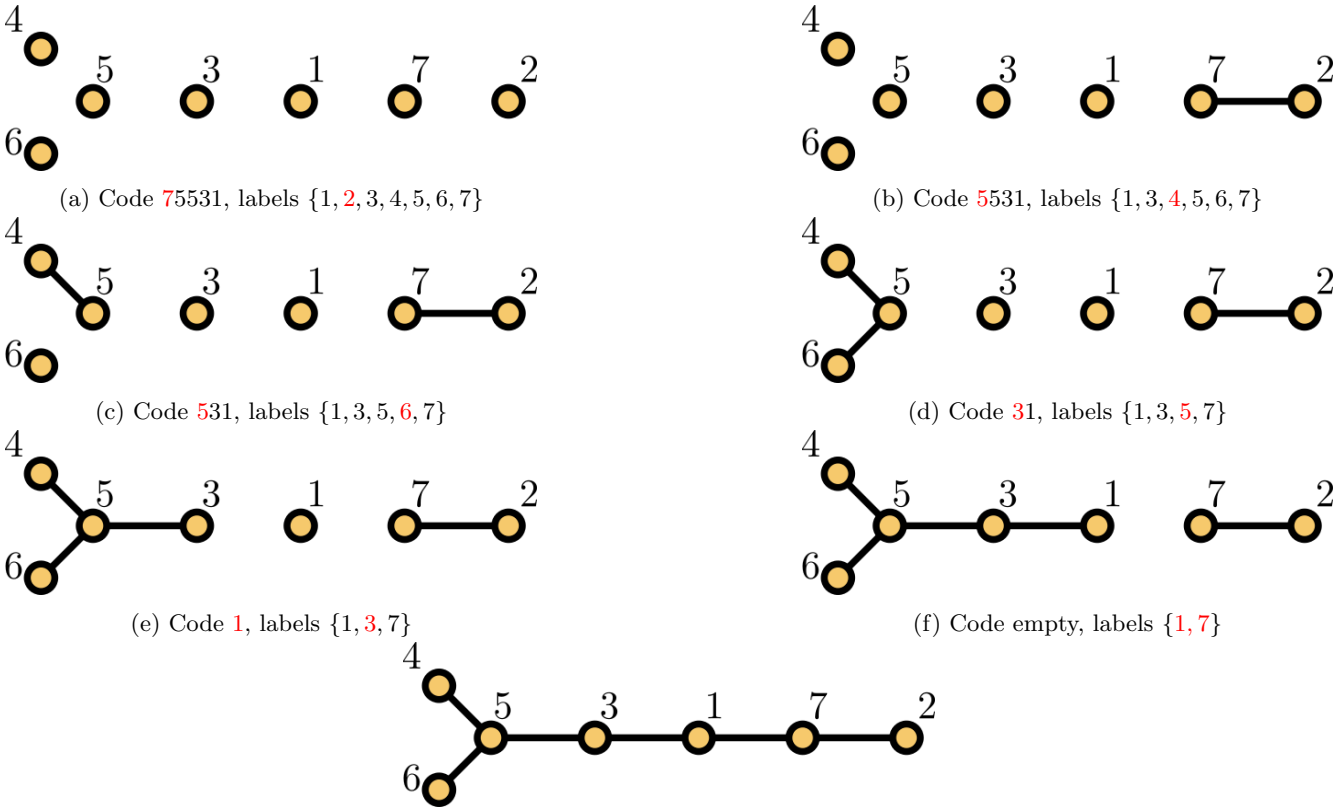
Solution: Let's record the state of the algorithm in a table

Prüfer code	Label set	Edge added
75531	$\{1,2,3,4,5,6,7\}$	2-7
5531	$\{1,3,4,5,6,7\}$	4-5
531	$\{1,3,5,6,7\}$	6-5
31	$\{1,3,5,7\}$	5-3
1	$\{1,3,7\}$	3-1
	$\{1,7\}$	1-7

and this results in the labeled tree in figure 2

Exercise 5.3. *Check that our algorithm reconstructs the tree on Figure 1 from the Prüfer code 6643143.*

Exercise 5.4. *Prove by induction that the result of this procedure uniquely defines a labeled tree on $[n]$ vertices, and that if we start from a tree T with Prüfer code Prüfer(T), then the algorithm recovers T from the code.*



(a) Code 75531, labels $\{1, 2, 3, 4, 5, 6, 7\}$

(b) Code 5531, labels $\{1, 3, 4, 5, 6, 7\}$

(c) Code 531, labels $\{1, 3, 5, 6, 7\}$

(d) Code 31, labels $\{1, 3, 5, 7\}$

(e) Code 1, labels $\{1, 3, 7\}$

(f) Code empty, labels $\{1, 7\}$

(g) Final step

Figure 2: Reconstructing the labeled tree from the Prüfer code 75531

References

[Gui18] David Guichard. *Combinatorics and Graph Theory.* Open access, 2018. Available at https://www.whitman.edu/mathematics/cgt_online/book/.

[KT17] Mitchel T. Keller and William T. Trotter. *Applied Combinatorics.* Open access, 2017. Available at http://www.rellek.net/appcomb/.