CO 330, LECTURE 34 SUMMARY

FALL 2017

Summary

Today we talked about the Prüfer[1] encoding. This is a classic combinatorial algorithm. It provides a bijective way to prove the obsevation we made a few weeks ago that the number of labelled (unrooted) trees on n vertices is n^{n-2}. It also lets us rank and unrank these trees.

Let $\mathcal{L}$ be the class of labelled unrooted trees. Here is the algorithm very briefly and at a high level.

```
def Pruefer (high level version)
  input a tree in L_n
  for i from 1 to n-2
    find the leaf with largest label
    L(i) = label of the neighbour of the leaf
    remove the leaf and its edge from the tree
  return L
```

To make this more precise, we will represent our trees as edge sets, and fill in the details.

```
def Pruefer (detailed version)
  input: E, n (E is the edge set of a tree in L_n)

  d = (0,..., 0) (length n)
  for {x,y} in E
    d(x) = d(x)+1
    d(y) = d(y)+1

  for i from 1 to n-2
    x = n
    while d(x) != 1
       x = x-1
    y = n
    while {x,y} not in E
       y = y-1

    L(i) = y
    d(x) = d(x)-1
    d(y) = d(y)-1
```

[1]Incidentally, if you have a German or German-origin word with an umlaut in it, like on the u in Prüfer, and for whatever reason you can't put the umlaut in (say you're working in plain ascii for some reason), then you should put an extra e after the vowel that should have had the umlaut. You can see an example in the code listing for the algorithm.

```
    E = E-{x,y}

  return L
```

We tried the algorithm on an example and you noticed that each vertex x appears $\deg(x)-1$ times in the output list. You can find other examples in the notes linked below.

Proposition 1. *Let $t \in \mathcal{L}_n$ and let E be the edges of t. Then x appears $\deg(x)-1$ times in* `Pruefer(E,n)`.

Proof. Let $L =$ `Pruefer(E,n)`.

In each iteration of the main for loop we remove an edge incident to a leaf and put the other incident vertex in L. This other vertex was not itself a leaf because if it were then either the tree at that stage would have only one edge or would be disconnected; the first of these is impossible because the algorithm ends before that and the second because removing a leaf can never disconnect a tree. Furthermore, when a vertex is entered into L it's degree is decreased by 1. Therefore the maximum number of times a vertex v can appear in L is $\deg(v)-1$.

Also

$$\begin{aligned} n-2 = |L| &\geq \sum_{v \in V(t)} (\deg(v)-1) \\ &= \sum_{v \in V(t)} \deg(v) - |V(t)| \\ &= 2|E(t)| - |V(t)| \\ &= |E(t)| - 1 = n-2 \end{aligned}$$

so all steps must have been equalities and the result follows. □

Here is the inverse map

```
def inversePruefer
  input L, n  (L a list of length n-2 of elements of {1,2,...,n}

  L(n-1) = 1
  d = (1,...,1) (n times)
  for i from 1 to n-2
    d(L(i)) = d(L(i)) + 1

  for i from 1 to n-1
    x = n
    while d(x) != 1
      x = x-1
    y = L(i)
    d(x) = d(x)-1
    d(y) = d(y)-1
    E = E cup {{x,y}}

  return E
```

Part of the reason to give the more detailed version of the Prüfer algorithm is that it makes the inverse clearer: the code is directly parallel except in reverse. There are only two other things to note. First, by the proposition both `Pruefer` and `inversePruefer` build the same d. Secondly, the edge remaining after `Pruefer` must be of the form $\{1, x\}$ as otherwise it would have been taken earlier. Thus if we ran the for loop once more we would get $L(n-1) = 1$. The beginning of `inversePruefer`. The rest of the two algorithms are directly inverses of each other.

That's all we need to conclude the things promised at the begining, but we ran out of time, so we'll do those things (and an example of `inversePruefer`) on Monday.

References

You can find this material in these lecture notes:
`http://people.math.sfu.ca/~kyeats/teaching/math343/10-343.pdf`
Again, for more look at the book "Combinatorial Algorithms" by Donald Kreher and Douglas Stinson.