Math 343, Lecture 10

① The Prüfer correspondence

Consider a combinatorial class $\mathcal{C}$ which has

A labelling of $c \in \mathcal{C}_n$ is

eg A labelled rooted tree is

subeg

Two labelled combinatorial objects are the same if

There is a whole theory of labelled combinatorial classes, constructions; exponential generating functions etc.

If you're interested you can read the supplemental notes

For the Prüfer correspondence we just need the following labelled trees

Let $\mathcal{U}$ be

eg

let $\mathcal{L}$ be

eg

The Prüfer correspondence is an algorithm which takes an element $t \in \mathcal{L}$ and returns a list of integers of length $|t|-2$.

This procedure defines a bijection and so we can also give an InvPrüfer algorithm to reverse the process.

How should we represent an element of $\mathcal{L}$
(in a computer / for the algorithm)

Algorithm Prüfer

Input: E, n , E the edge set of a $t \in \mathcal{L}_n$

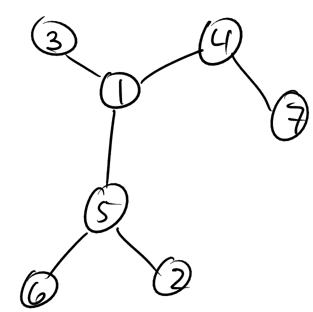
$d = \underbrace{(0, \ldots, 0)}_{n}$

for $\{x, y\}$ in E

  $d(x) = d(x) + 1$

  $d(y) = d(y) + 1$

eg $t =$



```
for i from 1 to n-2
        x = n
        while d(x) ≠ 1
                x = x-1
        y = n
        while {x,y} ∉ E
                y = y-1
        L(i) = y
        d(x) = d(x) - 1
        d(y) = d(y) - 1
        E = E \ {x,y}
output L
```

Can we see a patten in how many times each vertex label appears in the list

Prop Let $t \in \mathcal{T}_n$ and E the edges of t
then x appears times in $\text{Prüfer}(E, n)$

pf.

But $\sum_{v \in t} (\deg(v) - 1)$

Here is the inverse map

Algorithm InvPrüfer

input L, n — L a list of length $n-2$ of elements of $\{1, \dots, n\}$

$L(n-1) = 1$

for i from 1 to n

  $d(i) = 1$

for i from 1 to $n-2$

  $d(L(i)) = d(L(i)) + 1$

for i from 1 to $n-1$

  $x = n$

  while $d(x) \neq 1$

    $x = x - 1$

  $y = L(i)$

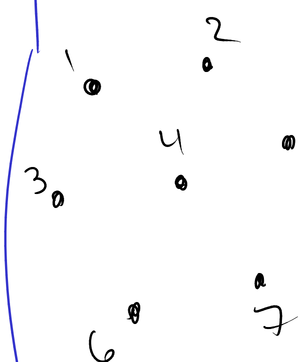
  $d(x) = d(x) - 1$

  $d(y) = d(y) - 1$

  $E = E \cup \{\{x, y\}\}$

return E

eg $L = (4, 5, 1, 1, 5, 1)$, $n = 7$



How do we know these are inverses of each other?

In view of the proposition

Consider the rest of the calculation:

```
for i from 1 to n-2
    x = n
    while d(x) ≠ 1
        x = x-1
    y = n
    while {x,y} ∉ E
        y = y-1
    L(i) = y
    d(x) = d(x) - 1
    d(y) = d(y) - 1
    E = E ∖ {x,y}
output L
```

```
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 ∪ {{x,y}}
return E
```

The only remaining issue is the last edge

One consequence of this is

Prop $|\mathcal{L}_n| =$

proof

Now say we lexicographically order trees by their lists under the Prüfer correspondence. Then we have

Algorithm RankTree

input E, n $\qquad$ E edge set of a tree $t \in \mathcal{T}_n$

$L = \text{Prüfer}(E, n)$

return

Algorithm UnrankTree

input r, n

return

eg

What are the runtimes of these algorithms?