Math 343, Lecture 16

(1) Revolving door successor

Some examples to look at

$$R_2(5) = (\{1,2\}, \{2,3\}, \{1,3\}, \{3,4\}, \{2,4\}, \{1,4\}, \{4,5\}, \{3,5\}, \{2,5\}, \{1,5\})$$

$$R_3(5) = (\{1,2,3\}, \{1,3,4\}, \{2,3,4\}, \{1,2,4\}, \{1,4,5\}, \{2,4,5\}, \{3,4,5\}, \{1,3,5\}, \{2,3,5\}, \{1,2,5\})$$

$$R_4(5) = (\{1,2,3,4\}, \{1,2,4,5\}, \{2,3,4,5\}, \{1,3,4,5\}, \{1,2,3,5\})$$

What is the successor function?

Algorithm Successor RevolvingDoor

input S, k, n S a k-subset of n

$j = 1$

if $k \neq j \bmod 2$

else $e = j$th smallest element of S

return S

To begin understanding how this algorithm works lets consider how each part affects j

```
if k ≠ j mod 2
    if j = 1
        decrease the smallest element of S
    else if j = 2
        increase the smallest element of S
    else
        replace j-2 by j in S

else
    e = jth smallest element of S
    if e+1 ∉ S
        if j = 1
            increase the smallest element of S
        else if j = k and e = n
            return no successor
        else
            replace j-1 by e+1 in S
    else
        replace e+1 by j in S

return S
```

proof the algorithm works

If $n \notin S$ then

If $n \in S$ then first note that the only way to obtain $j = k$ is

In all other cases $n \in S$ is not touched so let T be S after the algorithm. let

$$\tilde{S} = S \setminus \{n\}, \quad \tilde{T} = T \setminus \{n\}$$

consider it case by case.

If S is in A with first element > 2

If S is in A with first element = 2

If S is in B

If S is in C

If S is in D

If S is in E

If S is in F

If S is in F

eg lets do a few step in $R_{5,3}$

$\{1,2,3\}$

```
j = 1
while j ∈ S
    j = j+1
if k ≠ j mod 2
    if j = 1
A       decrease the smallest element of S
    else if j = 2
B       increase the smallest element of S
    else
C       replace j-2 by j in S

else
    e = jth smallest element of S
    if e+1 ∉ S
        if j = 1
D           increase the smallest element of S
        else if j = k and e = n
            return no successor
        else
E           replace j-1 by e+1 in S
    else
F       replace e+1 by j in S

return S
```

2 Minimal change ordering of permutations

Def A permutation of $\{1, \dots, n\}$ is a bijection

$$\sigma : \{1, \dots, n\} \longrightarrow \{1, \dots, n\}$$

We can represent a permutation by the list of its values

$$(\sigma(1), \sigma(2), \dots \sigma(n))$$

eg

eg

There was a homework question / midterm question on lexicographic successor for these.

What should minimal change mean for permutations?

Def We say two permutations σ and τ differ by a transposition if

σ and τ differ by an adjacent transposition

In fact there's much more structure here

Def A minimal change ordering on the set of permutations of $\{1, \ldots, n\}$ is

Here is an example of such an ordering on permutations called Trotter-Johnson ordering.

Again lets describe it recursively

first examples

$T(1) =$

$T(3) =$

$T(2) =$

I

$T(4) =$

precisely

Def $T(1) = (\ (1)\ )$.

Given a permutation σ of $\{1, \ldots, n-1\}$ say the j^{th} insertion slot is after $\sigma(j)$ for $1 \le j < n$ and before $\sigma(1)$ for $j = 0$

$T(n)_i$ is

with n inserted in the

insertion slot

lets check this works correctly for $n=3$.

What about rank, unrank, and successor.
Recursively they are not too hard

lets consider rank

•

•

$\rightarrow$

Algorithm RecursiveRankTrotterJohnson

input L, n L a permutation of n written as a list of values

if $n = 1$ return 0

$k = 1$

while

if r even

else

eg $n = 4$ $L = (3, 4, 2, 1)$

Now how to write this non recursively?

we need to do the analogous thing for every value, not just n

Algorithm RankTrotterJohnson

input L, n   L a permutation of n written as a list of values

r = 0

for j

while

if r even

else

return r

same eg $n=4$ $L=(3,4,2,1)$

How fast are these?

We can unrank similarly.

What about successor.

We need to

Can we

See your next homework.

③ Next time

Generating trees.