Math 343, Lecture 8

① Generation and Random generation

Given a combinatorial class $\mathcal{C}$, here are two things I might want to do

Generate all elements of $\mathcal{C}_n$

Our specifications tell us

next

fifth

Generate a random element of $\mathcal{C}_n$

So some things we would like to do are

Generate all elements of $\mathcal{C}_n$

successor

ranking

unranking

Generate a random element of $\mathcal{C}_n$

Def Let S be a finite set

a rank function is

the corresponding unrank function is

so

Let $\mathcal{C}$ be a combinatorial class.

If we have an efficient unranking algorithm for $\mathcal{C}_n$ then we can efficiently

If we have both efficient ranking and unranking on $\mathcal{C}_n$ then we can efficiently

What exactly do we mean by a random element?

Def Let $\mathcal{C}$ be a combinatorial class

An algorithm which takes as input a nonnegative integer n and returns as output a element $c \in \mathcal{C}_n$ such that

uniform generation

② Lexicographic ranking and unranking of binary strings / subsets

Note that binary strings are simple enough that we have a direct algorithm for uniform random generation namely:

Say irand(a,b) picks a random integer in $[a, b]$

Then

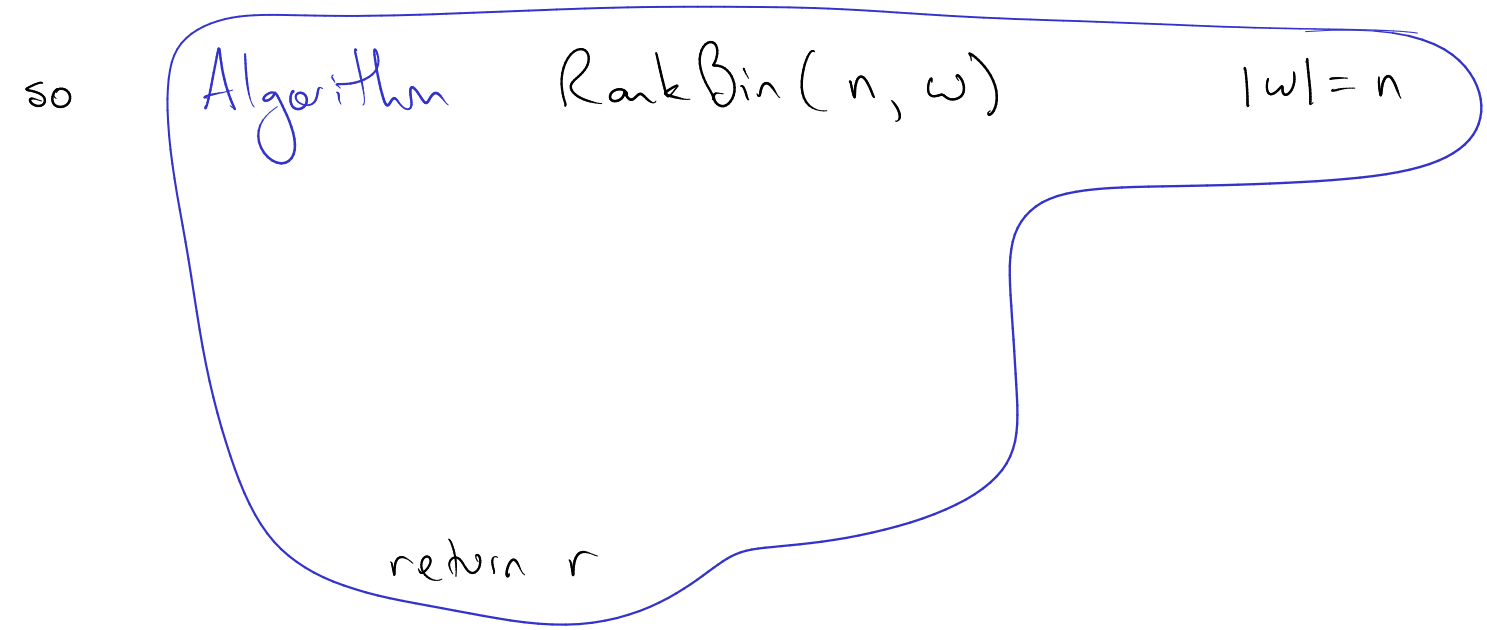
```
Algorithm RandBin(n)        n integer n≥0

return
```

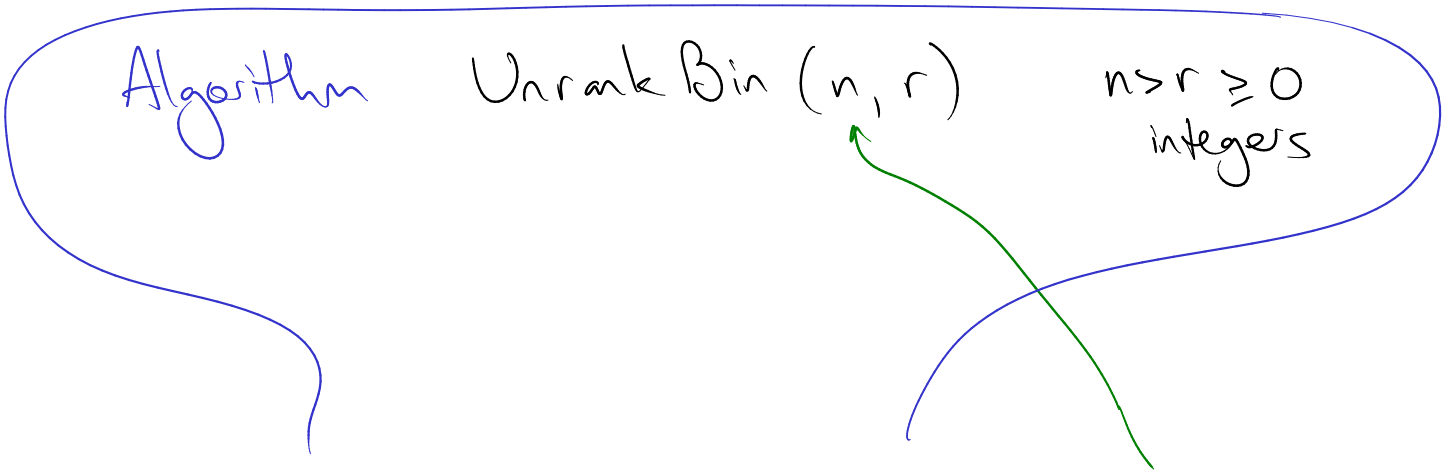
This algorithm is

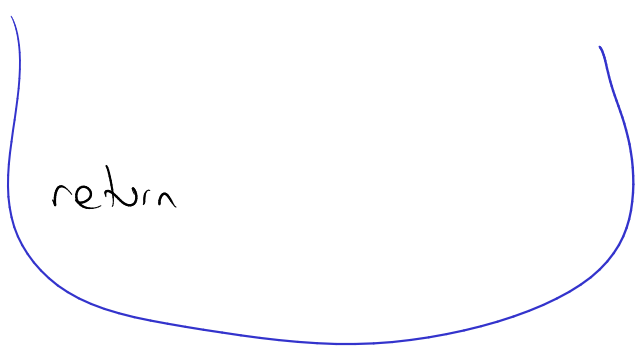
There are also nice ranking and unranking functions let the binary word represent a binary numbe

Ne

so







These are

Note binary strings of length n are in bijection with subsets of $\{1, 2, \ldots, n\}$

eg

So

Note It is reasonable to view this ordering as a (somewhat trivial) example of lexicographic ordering for the following reason:

View

Lexicographic order says

If the words are not the same length

This is

and also

eg

Returning to thinking of binary strings as subsets, a natural restriction we could be interested in would be subsets of a particular size

eg subsets of size 3 of $\{1,2,3,4,5\}$

We can view

Using this ordering we can make a successor function and ranking and unranking functions

Algorithm Successorksubset (L, k, n)

$L_{next} = L$

$i = k$

while

if $i = 0$

return no successor (L was the last k-subset)

else

We are interested in k-subsets of $\{1, 2, \ldots, n\}$

L is the current k-subset as a list in increasing order

To rank and unrank we need

Proposition Let $\{n_1, \dots, n_k\} \subseteq \{1, 2, \dots, n\}$ with $n_1 < n_2 < \dots < n_k$. Write $L = (n_1, n_2, \dots, n_k)$ then

$$\operatorname{rank}(L) = \sum_{i=1}^{k} \sum_{j=n_{i-1}+1}^{n_i - 1} \binom{n-j}{k-i}$$

with the convention $n_0 = 0$

proof there are k-subsets beginning with 1

So Algorithm RankSubset (L, k, n) (same conditions on the input as before)

$r = 0$

$L(0) = 0$

for

if

return

Algorithm UnrankkSubset (r, k, n) $k \leq n$ $r < \binom{n}{k}$

$j = 1$

for

while $\binom{n-j}{k-i} \leq r$

return

What are the runtimes?

Sucessor is

Unrank is

So is rank

We can actually do better and get a rank function which is $O(k)$

Here's how $\rightsquigarrow$

corank (L)

Proposition Let $\{n_1, \dots, n_k\} \subseteq \{1, 2, \dots, n\}$ with $n_1 < n_2 < \dots < n_k$. Write $L = (n_k, n_{k-1}, \dots, n_1)$ then

$$\operatorname{corank}(L) = \sum_{i=1}^{k} \binom{n_{k-i+1} - 1}{k+1-i}$$

proof There are lists with all elements $< n_k$

Further

Proposition let $\{n_1, \dots, n_k\} \subseteq \{1, 2, \dots, n\}$

with $n_1 < n_2 < \dots < n_k$.

let $L = (n_1, n_2, \dots, n_k)$

let $\tilde{L} = (n+1-n_1, n+1-n_2, \dots, n+1-n_k)$

then

proof

So we have

Algorithm Rank2ksubset(L, n, k) L in increasing order

r = 0

for

return

eg Successor k Subset $((1,2,5), 5, 3)$

```
Lnext = L
i = k
while (i ≥ 1) and (L(i) = n - k + i)
    i = i - 1
if i = 0
    return no successor
else
    for j from i to k
        Lnext(j) = L(i) + 1 + j - i
    return Lnext
```

eg Rank2 k subset $((1,2,5), 5, 3)$

```
r = 0
for i from 1 to k
    r = r + ( n - L(i) )
            ( k + 1 - i )
return (n choose k) - 1 - r
```

eg Unrank ksubset $(3, 5, 3)$ ($r = 3$, $n = 5$, $k = 3$)

```
j = 1
for i from 1 to k
    while (n-j choose k-i) ≤ r
        r = r - (n-j choose k-i)
        j = j + 1
    T(i) = j
    j = j + 1
return T
```

Back on binary strings we've just been working with

How do we specify this class

But

③ Next time

More classes we can give a lexicographic order to.