Math 343, Lecture 11

① Limitations of ranking and unranking for random generation

One thing we can use ranking and unranking for is random generation.

Recall we are interested in uniform random generation that is

We assume we can generate random numbers uniformly in $(0,1)$

Say rand()

So I have random generate by

note

But there's another problem

Consider the class $\mathcal{B}$ of binary strings with no consecutive zeros.

How do I uniformly randomly generate an element of $\mathcal{B}_n$?

Ideas

Problems

(2) Recursive random generation

We have the information of a combinatorial specification. How can we use it for random generation? The first answer is recursive random generation.

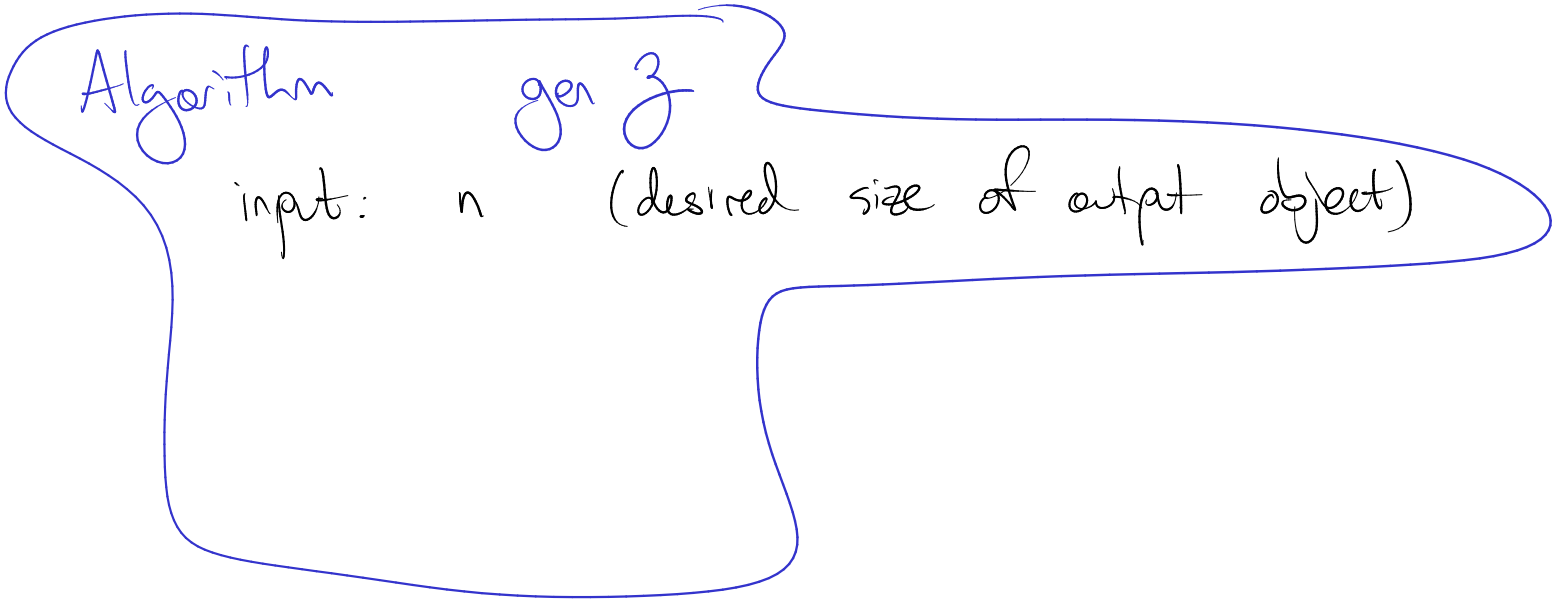
Lets start simply:

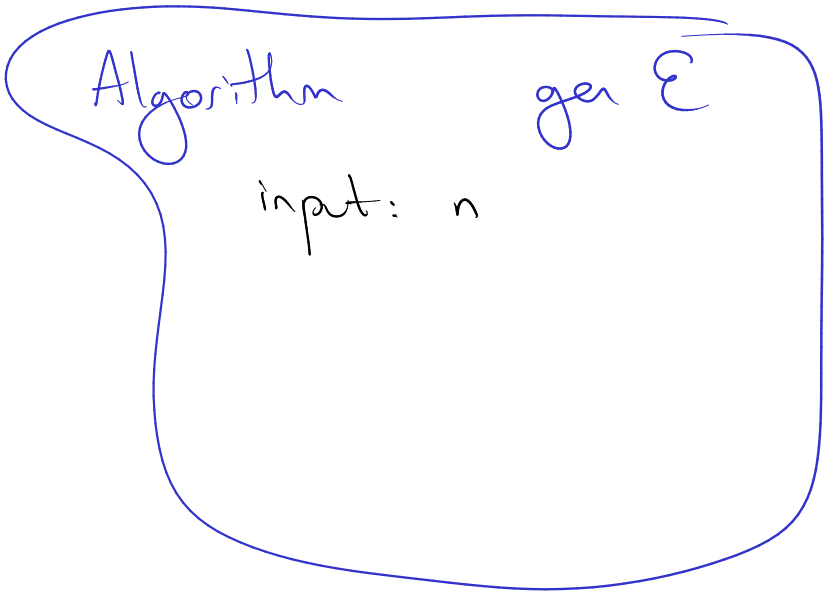
Suppose $\mathcal{C}$ is a combinatorial class with a specification in terms of $+, \times, \mathcal{E}, \mathcal{Z}$

(eg

Can view this specification as a tree

now


Algorithm gen Z
input: n (desired size of output object)


Algorithm gen E
input: n

Suppose $A + B + C$ how do we pick a random element of size n?

We just need to know

Algorithm Gen $A = B + C$

Input: n

$x = rand()$

if

Note that we're cheating slightly

For some classes

Otherwise

Now suppose $A = B \times C$ and suppose we want to generate a random element of A_n

The probability that

Algorithm Gen $A = B \times C$

Input: n

$x = rand()$

$k = 0$

$s = \frac{b_0 c_n}{a_n}$

while

Return

Suppose we have a specification Φ with just one equation. Write this equation as a tree and then run the appropriate one of the above on the root of the tree, the recursive calls will then run on the lower vertices of the tree

For our particular example we can collect the all into one function.

Let's see it in practice

The other class of binary trees is in the notes

③ Next time

Boltzmann Samplers