CO 330, LECTURE 33 SUMMARY

FALL 2017

Summary

Today we talked about random generation by rank and unrank.

Let $\mathcal{A}$ be a combinatorial class. A *rank* function is a bijection from $\mathcal{A}_n$ to $\{1, 2, \ldots, a_n\}$. An *unrank* function is the reverse bijection. If you uniformly generate an element of $\{1, 2, \ldots, a_n\}$ and then unrank you have uniformly generated an element of size n of $\mathcal{A}$. This idea doesn't have the same generality as the random generation algorithms we've seen so far but in its specificity it can be better in particular examples.

One important class of rank/unrank functions are ones coming from a lexicographic order. Any class made of lists can be ordered lexicographically provided the set of elements for the lists have a total order. Specifically, order first by the first element, then break ties by the second element, then break remaining ties by the thir element, etc; finally, nothing comes before something. This is the usual dictonary order.

In a somewhat trivial way you can view binary strings as lists of bits and order them lexicographically. If we choose to take 0 before 1 and start at the left then ordering all binary strings of length n this way the rank is just the number obtained by interpreting the binary string as a binary expansion, and unrank is just finding the binary representation of a number. There are also other ways to do a lexicographic order, say take 1 before 0, or start on the right; you tried some of these in class.

What about subsets? You can use the bijection with binary strings where $b_1 b_2 \ldots b_n \mapsto S$ with $b_i = 1 \Leftrightarrow i \in S$, and then just use the rank/unrank for binary stings.

What about k-subsets? We can view $\{n_1, n_2, \ldots, n_k\} \subseteq \{1, 2, \ldots, n\}$ as a list $(n_1, n_2, \ldots, n_k)$ with $n_1 < n_2 < \ldots n_k$. We can order these lists lexicographically. This will end up being the same as the lexicographic order on the associated binary strings where we take 1 before 0. With this notation there is a reasonable formula for the rank:

$$\operatorname{rank}((n_1, n_2, \ldots, n_k)) = \sum_{i=1}^{k} \sum_{j=n_{i-1}+1}^{n_i - 1} \binom{n-j}{k-i}$$

This formula can be proved as follows. Let $L = (n_1, n_2, \ldots, n_k)$. There are $\binom{n-j}{k-i}$ k-subsets whose list begins with $n_1, n_2, \ldots, n_{i-1}, \ell$ for $n_{i-1} < \ell < n_i$ and all these lists come before L in lexicographic order. Summing all these possibilities gives the result.

The formula gives the rank function. For unrank just pull terms off the formula as follows:

```
def unrank_k_subset(r, n, k)   (with k <= n, r < binom(n,k))
  j = 1
  for i from 1 to k
    while binom(n-j, k-i) <= r
      r = r - binom(n-j, k-i)
      j = j+1
    L(i)=j
    j = j+1
  return L
```

References

You can find this material in these lecture notes:
`http://people.math.sfu.ca/~kyeats/teaching/math343/8-343.pdf`
If you want to know more (and many other interesting and important) combinatorial algorithms, look at the book “Combinatorial Algorithms” by Donald Kreher and Douglas Stinson.