Math 206B Lecture 23 Notes

Daniel Raban

March 4, 2019

1 Complexity Aspects of Young Tableau

1.1 Generating random Young tableau

Given λ, we want to generated a random $A \in \mathrm{SYT}(\lambda)$. We have two algorithms for doing this:

1. NPS algorithm: the sort of 2-dimensional bubblesort on tableaux
2. GNW algorithm: Place n according to a hook walk, update λ to be $\lambda \setminus \{n\}$, and repeat.

Which one is faster?

Example 1.1. Let $\lambda = (n)$, which will produce a 1-row tableau. Then NPS will take $\mathbb{E}[\mathrm{inv}(\sigma)] = \Theta(n^2)$ steps. Alternatively, GNW will take $\Theta(n \log(n))$ steps.

Example 1.2. Let λ be the shape of a $k \times k$ square tableau, $\lambda = (k^k)$. Here, $n = k^2$. Then GNW will run in $\Theta(n \log(n))$ time. What about NPS? It goes column by column, and each column takes $\Theta(k^2)$ time. So does NPS run in $\Theta(k^3) = \Theta(n^{3/2})$ time? We really want to look at the average complexity. It turns out that the average complexity is $\Theta(k^3) = \Theta(n^{3/2})$ anyway.

1.2 Generating random partitions

Given n, we want to uniformly generate a random partition $|\lambda| = n$. Here is a (not too) slow algorithm:

Compute $p_k(n)$, the number of partitions λ of n such that $\lambda_1 = k$ parts. Then $p_k(n) = p_1(n-k) + p_n(n-k) + \cdots + p_k(n-k)$ and $p_1(n) = 1$, so we can solve the recurrence relation. This takes $\Theta(n^{3.5})$ steps.

What is a smarter algorithm? We know

$$\sum_{n=0}^{\infty} p(n) t^n = \prod_{i=1}^{\infty} \frac{1}{1-t^i}$$

Write this as the following:

$$= \sum_{\lambda \in \mathcal{P}} t^{|\lambda|}.$$

Now think of this probabilistically. Let $Z_i + 1 \sim \text{Geom}(1 - t^i)$. Then let $\lambda = 1^{Z_1} 2^{Z_2} \cdots$. Now

$$\mathbb{P}(\lambda) = \mathbb{P}(Z_1 = m_1(\lambda), Z_2 = m_2(\lambda), \dots) = \prod_{i=1}^{\infty} t^{i m_i}(1 - t^i) = t^{\sum i m_i} \prod_{i=1}^{n} (1 - t^i).$$

If we generate λ like this, then $|\lambda| = n \iff Z_1 + \cdots + 2Z_2 + \cdots = n$. So we can generate all these and output λ. So we get

$$\mathbb{P}(|\lambda| = n) = p(n) t^n \prod_{i=1}^{n} (1 - t^i) \sim e^{c\sqrt{n}} t^n \prod_{i=1}^{n} (1 - t^i).$$

Optimize over all possible values of t. Then the number of steps becomes $\Theta(n^{3/4})$ or so. This idea is called **Boltzmann sampling**.

1.3 Generating random 3-dimensional partitions

Given n, we want a random 3-dimensional (called solid/plane) partition A. We proved[1] the generating function

$$\sum_{A \in \mathcal{PP}} t^{|A|} = \sum_{n=0}^{\infty} \mathcal{PP}(n) t^n = \prod_{i=1}^{\infty} \frac{1}{(1 - t^i)^i}.$$

We can do the same type of sampling as before. Let $Z_{i,j} \sim \text{Geom}(1 - t^i)$, where $1 \leq j \leq i$. Place the $Z_{i,j}$ in a matrix:

$$\begin{matrix} Z_{1,1} & Z_{2,1} & Z_{3,1} & \cdots \\ Z_{2,2} & Z_{3,2} & \cdots & \\ Z_{3,3} & \cdots & & \end{matrix}$$

Flip the matrix across the (non-main) diagonal and apply the Hilman-Grassl algorithm.

1.4 Generating random skew shapes

Suppose we have the skew shape $\lambda \setminus \mu$. How do we generate a random $A \in \text{SYT}(\lambda \setminus \mu)$? The Jacobi-Trudy identity gives us

$$s_\lambda = \det([h_{\lambda_i - i + j}]).$$

[1] Maybe the word 'proved' should be taken lightly.

Then we get

$$f^{\lambda} = n! \det\left(\left[\frac{1}{(\lambda_i - i + j)!}\right]\right)$$

We can calculate the probability of putting n into a corner using

$$\mathbb{P}(n \text{ in corner}) = \frac{f^{\lambda \setminus \{n\}}}{f^{\lambda}}$$

Here, we use a generalized version of the Jacobi-Trudy identity:

Theorem 1.1 (Jacobi-Trudy)**.** *Let*

$$s_{\lambda\setminus\mu} := \sum_{A \in \mathrm{SSYT}(\lambda\setminus\mu)} t^{|A|}.$$

Then

$$s_{\lambda\setminus\mu} = \det([h_{\lambda_i - \mu_i - i + j}])$$