1

Contents

2 Combinatorial and Algorithmic Applications

2.1 Trees

Types of trees I

- A *free tree* is a connected graph with no cycles.
- A *rooted tree* is a free tree with a distinguished node called the *root*.
- An *ordered tree* is a rooted tree where the order of subtrees is important; recursively, a root connected to a sequence of ordered trees.
- A *m-ary tree* is an ordered tree where every node has 0 or m children.
- A *labelled tree* is a tree with n nodes such that each node is labelled by an element of $[n]$ and all labels are distinct.

Synonyms in literature: plane = ordered; oriented = rooted.

Some trees in analysis of algorithms

- *Binary search tree*: a binary tree with each internal node having a *key*, such that the key of each node n is $\leq$ all keys in R_n and $\geq$ all keys in L_n. Applications: database for comparable data, model for quicksort.
- *Heap-ordered tree*: a binary tree such that the key of each node is $\geq$ the key of anything in its subtree. Applications: priority queue.
- *Trie*: an m-ary tree where each external node may contain data; children of leaves must be nonempty. Applications: database for string data, model for radix exchange sort, leader election in distributed computing.
- *Rooted tree*: a root connected to a set of rooted trees. Used in the union-find problem (for example in Kruskal's algorithm for minimum spanning tree).

Tree attributes

- The *size* is the number of (external, internal, or just plain) nodes.
- The *depth* of a node in a rooted tree is the distance to the root.
- The maximum depth is the *height*. The sum of all depths of internal (external) nodes is the *internal (external) path length*.

Path length in binary trees (uniform model)

- The bivariate generating function $F(z,u)$ enumerating binary trees by number of nodes and internal path length satisfies the equation
$$F(z,u) = 1 + zF(zu,u)^2.$$
- The mean and variance are given by a standard computation. Note that
$$F_u(z,u) = 2zF(zu,u)[F_u(zu,u) + zF_z(zu,u)]$$
and so $F_u(z,1) = 2zF(z,1)[F_u(z,1) + zF_z(z,1)]$. Thus
$$\mu_n := \frac{[z^n]\frac{zF_z(z,1)}{1-2zF(z,1)}}{[z^n]F(z,1)}$$
- The mean μ_n is asymptotic to $\sqrt{\pi}n^{3/2}$, so the mean level of a node is of order $\sqrt{n}$. The variance is also of order $n^{3/2}$.

Path length in binary search trees

- Suppose we insert n distinct keys into an initially empty BST. The uniform distribution on permutations of size n induces the *quicksort distribution* on BSTs of size n.
- The internal path length equals the construction cost of a binary search tree of size n; dividing by n gives the expected cost of a successful search.
- Let $F(z,u) = \sum \frac{z^{|\pi|}}{|\pi|!}u^{\ell(\pi)}$ be the BGF of BSTs by size and internal path length. Note that $[z^n]F(z,u)$ is the PGF of internal path length on BSTs with n nodes.

  Then
$$F_z(z,u) = F(zu,u)^2 \qquad F(0,u) = 1.$$
- Moments of the distribution are easily obtained as for the uniform model. The mean is $\sim 2n\log n$ and variance is in $\Theta(n^2)$. Quite a different shape.

2.2 Strings

String basics

- Let $\mathcal{A}$ be a finite set called the *alphabet.* A *string* over $\mathcal{A}$ is a finite sequence of elements of $\mathcal{A}$. The set of all strings over $\mathcal{A}$ is written $\mathcal{A}^*$.
- A subset of $\mathcal{A}^*$ is a *language.*
- If $\mathcal{A} = \{0, 1\}$ the strings are called *bitstrings.*
- Basic algorithmic questions: *string matching* (find a pattern in a given string); search for a word in a dictionary; compress a string. Many applications in computational biology, computer security, etc.

Hidden pattern occurrences

- The set of occurrences of the *subsequence* (hidden pattern) $-a_1-a_2-\cdots-a_k-$ in a string of length n corresponds to $\mathcal{A}^*a_1\mathcal{A}^*a_2\mathcal{A}^*a_3\ldots\mathcal{A}^*a_k\mathcal{A}^*$.
- The counting OGF of $\mathcal{A}^*$ is $1/(1-mz)$ so the OGF for all pattern occurrences is $P(z) = z^k/(1-mz)^{k+1}$ where $m = |\mathcal{A}|$.
- The expected number of occurrences in a random "word" of length n is $[z^n]P(z)/(m^n) = m^{-k}\binom{n}{k}$.
- The OGF for total occurrences of the *substring* $a_1a_2\ldots a_k$ is $z^k/(1-mz)^2$ and a similar analysis applies.
- Note relevance to various conspiracy theories. *We should expect a sufficiently long random text to contain any given hidden message.*

Pattern avoidance

- We have already counted total *occurrences* of a given substring or pattern. Now we want to count *number of words* a_n not containing a given pattern (a harder problem).
- A nice trick: let T be the position of the end of the first occurrence of the pattern, X_n the event that the first n bits of a random bitstring do not contain the pattern. Then $S(z) = \sum_{n\geq 0} a_n z^n$ implies that

$$S(1/2) = \sum_{n\geq 0} a_n/2^n = \sum_{n\geq 0} \Pr(X_n) = \sum_{n\geq 0} \Pr(T > n) = E[T].$$

Pattern avoidance - simple example

- Given substring $\sigma = 00 \cdots 0$ of length k, let $S(z)$ be the counting OGF for bitstrings without σ as substring.
- Recursion/symbolic method gives
$$S(z) = \left(\sum_{i<k} z^i\right)(1 + zS(z))$$
so
$$S(z) = \frac{1 + z + z^2 + \cdots + z^{k-1}}{1 - z - z^2 - \cdots - z^k} = \frac{1 - z^k}{1 - 2z + z^{k+1}}.$$
- Asymptotics: $a_n \approx C\rho^{-n}$ where ρ is smallest modulus root of denominator.
- Note that $\rho = 1/2 + \rho^{k+1}/2$ and $0 < \rho < 1$. Thus $1/2 < \rho < 1/2 + 1/2^k$, etc, and we can compute ρ quickly by iteration.
- Note $S(1/2) = 2^{k+1} - 2$.

Substring patterns - autocorrelation polynomial

- Consider an arbitrary binary string $\sigma = \sigma_0\sigma_1 \cdots \sigma_{k-1}$ of length k.
- For $0 \le j \le 1$, shift σ right j places. Define $c_j = 1$ if the overlap matches the tail $\sigma^{(j)}$ of σ, $c_j = 0$ otherwise. The *autocorrelation polynomial* is $c(z) = \sum_j c_j z^j$.
- Let $\mathcal{S}$, (resp. $\mathcal{T}$) be the set of bitstrings not containing p (resp. containing it once at the end). Then
$$\mathcal{S} \cup \mathcal{T} \cong \{\epsilon\} \cup \mathcal{S} \times \{0, 1\}$$
$$\mathcal{S} \times \{\sigma\} \cong \mathcal{T} \times \cup_{\{j:c_j \neq 0\}} \sigma^{(j)}$$
and the symbolic method gives $S(z) + T(z) = 1 + 2zS(z)$ and $S(z)z^k = T(z)c(z)$.
- Thus
$$S(z) = \frac{c(z)}{z^k + (1 - 2z)c(z)}.$$
- Note $[z^n]S(z/2) = \Pr(X_n)$. Looking at the poles of $S(z/2)$ we see that $\Pr(X_n) \to 1$ exponentially fast as $n \to \infty$: *every fixed substring is contained with high probability in a sufficiently long string.*

Regular languages

- Rational GFs always arise from the *transfer matrix method*.
- Special case: the counting GF of an unambiguous regular language is rational (Chomsky-Schützenberger, 1963).
- Recall that every regular language can be defined by an unambiguous regular expression.
- Thus if we construct a combinatorial class iteratively using only disjoint union, cartesian product, and sequence, the counting GF is rational.

Regular expression example

- Consider language (over alphabet $\{a, b\}$) defined by $(bb \mid a(bb)^*aa \mid a(bb)^*(ab \mid ba)(bb)^*(ab \mid ba))^*$ (number of b's is even, number of a's divisible by 3).
- The symbolic method gives
$$S(z) = \frac{(1-z^2)^2}{1-3z^2-z^3+3z^4-3z^5+z^6}.$$
Hence $a_n \approx CA^n$, $A \cong 1.7998$.
- Need to check that the expression is unambiguous.

Patterns in strings: summary

- The generating function for any regular expression is a rational function and can be computed algorithmically.
- In certain special cases more efficient methods (such as autocorrelation polynomial) exist.
- We have really only considered the case where all letters appear independently and with equal probability. In real applications, letter probabilities vary and letters are not independent. The methods above can be extended to cope with this.

2.3 Tries

Tries

- Each binary tree corresponds to a set of binary strings (0 encodes left branch, 1 encodes right branch, string is given by labels on path to external node). This set of strings is *prefix-free*.
- Conversely a finite prefix-free set of strings corresponds to a unique binary tree, a *full trie*.

- More generally, we may stop branching as soon as the strings are all distinguished. This gives a *trie*, a binary tree such that all children of leaves are nonempty. Each string is stored in an external node but not all external nodes have strings. Can be described by symbolic method.
- A *Patricia trie* saves space, by collapsing one-way branches to a single node.
- Relevant parameters: number of internal nodes I_n; external path length L_n; height H_n.

Trie recurrences

We assume that a trie is built from n infinite random bitstrings. Each bit of each string is independently either 0 or 1.

- $$L_n = n + \frac{1}{2^n} \sum_k \binom{n}{k} \left(L_k + L_{n-k}\right),$$
  L the mean external path length.
- $$I_n = 1 + \frac{1}{2^n} \sum_k \binom{n}{k} \left(I_k + I_{n-k}\right),$$
  I the number of internal nodes.
- Let $L(z) = \sum_n L_n z^n/n!$, etc. Then
  $$\begin{aligned} L(z) &= 2L(z/2)e^{z/2} + ze^z - z \\ I(z) &= 2I(z/2)e^{z/2} + e^z - z - 1. \end{aligned}$$

Solving the trie recurrences, I

- If $\phi(z) = 2e^{z/2}\phi(z/2) + a(z)$, then by iteration we obtain
  $$\phi(z) = \sum_{j \geq 0} 2^j e^{z(1-2^{-j})} a(2^{-j}z).$$
- Thus we obtain
  $$\begin{aligned} L_n &= n \sum_{j \geq 0} \left(1 - \left(1 - 2^{-j}\right)^{n-1}\right) \\ I_n &= \sum_{j \geq 0} 2^j \left[1 - \left(1 - 2^{-j}\right)^n - \frac{n}{2^j}\left(1 - 2^{-j}\right)^{n-1}\right]. \end{aligned}$$
- How to derive an asymptotic approximation? See Flaj-Sedg p211, p402 for elementary arguments. Answers: $L_n \sim n \lg n, I_n \sim n/\lg 2$. More precise answers are obtained by complex methods (*Mellin transform*).

Solving the trie recurrences, II

- Define $\hat{\phi}(z) = e^{-z}\phi(z)$, etc (this is the *Poisson transform*). Then we have
$$\hat{\phi}(z) = 2\hat{\phi}(z/2) + \hat{a}(z).$$
- Iteration yields
$$\hat{\phi}(z) = \sum_{j \geq 0} 2^j \hat{a}(2^{-j} z).$$
- This gives, on inverting the transform,
$$L_n = \sum_{k \geq 2} (-1)^k \binom{n}{k} \frac{k 2^{k-1}}{2^{k-1} - 1}.$$
- Asymptotics for such alternating sums can be obtained by *Rice's method.*

Summary: tries

- A useful data structure for dictionary and pattern matching. Also a mathematical model for many algorithms.
- Asymptotically optimal ($\lg n$) expected search cost.
- Space wastage: about 44% extra nodes ($1/\lg 2 - 1$).
- Recurrences under the infinite random bitstring model yield GF equations that are tricky. Solution involves infinite sums of functions.
- Explicit formulae for solutions are infinite sums. Mellin transforms or Rice's integrals give precise asymptotics; elementary methods can also be used.