[CS1961: Lecture 7] Algorithmic Zeta Transform and Möbius Transform, Fast Set Convolution

Instructor: Chihao Zhang;
Scribed by Shuangcheng Liu, Zidong Yang, Zhuoheng Zhang, Yuchen He

1 Inclusion-Exclusion Principle Revisit

Let $\mathcal{B}$ be a set of bad sets $\{B_1, B_2, \ldots, B_m\}$ where each B_i is a subset of the universe U. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_t\} \subseteq \mathcal{B}$. Let

$$N_{\geqslant}(\mathcal{S}) \triangleq \{x \in U | \forall i \in [t], x \in S_i\}.$$

be the set of numbers belonging to all sets in $\mathcal{S}$. Let

$$N_{=}(\mathcal{S}) \triangleq \{x \in U | \forall i \in [t], x \in S_i \wedge \forall B \in \mathcal{B} \setminus \mathcal{S}, x \notin B\}$$

be the set of numbers belonging to all sets in $\mathcal{S}$ and *not* in any set in $\mathcal{B} \setminus \mathcal{S}$.

It is clear that $N_{\geqslant}(\mathcal{S}) = \sum_{T:\mathcal{S}\subseteq T} N_{=}(T)$. The principle of inclusion-exclusion (PIE) states that

$$N_{=}(\mathcal{S}) = \sum_{T:\mathcal{S}\subseteq T} (-1)^{|T\setminus\mathcal{S}|} N_{\geqslant}(T).$$

If we view $N_{\geqslant}$ and $N_{=}$ as vectors over $2^{\mathcal{B}}$, we can describe the relationship as $N_{\geqslant} = Z \cdot N_{=}$ where $Z \subseteq \{0,1\}^{2^{\mathcal{B}} \times 2^{\mathcal{B}}}$ is the *zeta matrix* satisfying

$$Z(I, J) = \begin{cases} 1, & I \subseteq J; \\ 0, & \text{otherwise.} \end{cases}$$

The inverse of Z is called the *Möbius matrix* satisfying

$$Z^{-1}(I, J) = \begin{cases} (-1)^{J\setminus I}, & I \subseteq J; \\ 0, & \text{otherwise.} \end{cases}$$

Using these noatations, the inclusion-exclusion principle can be described as $N_{=} = Z^{-1} \cdot N_{\geqslant}$.

These identities can be generalized to a poset $(P, \leqslant)$. Let $f\colon P \to \mathbb{R}$. The zeta transform is defined as

$$(\zeta f)(x) \triangleq \sum_{y\,:\,y \geqslant x} f(y)$$

where ζ is the zeta function satisfying

$$\zeta(x, y) = \begin{cases} 1, & x \leqslant y; \\ 0, & \text{otherwise.} \end{cases}$$

and the Möbius transform is defined as

$$(\mu f)(x) \triangleq \sum_{y\,:\,y \geqslant x} \mu(x,y) f(y)$$

where μ is the Möbius function satisfying

$$\mu(x,y) = \begin{cases} 0, & x \not\leqslant y; \\ 1, & x = y; \\ -\sum_{z\,:\,x \leqslant z < y} \mu(x,z), & x < y. \end{cases}$$

Note that ζ and μ are natural generalizations of Z and Z^{-1} respectively. Consider the poset $\left(2^{[n]}, \subseteq\right)$. For $I, J \subseteq 2^{[n]}$, if $I = J$ or $I \not\subseteq J$, it is obvious that $\mu(I,J) = Z^{-1}(I,J)$. If $I \subsetneq J$, letting $m = |J \setminus I|$, we have

$$\begin{aligned} \mu(I,J) &= -\sum_{X\,:\,I \subseteq X \subsetneq J} \mu(I,X) = -\sum_{X\,:\,I \subseteq X \subsetneq J} (-1)^{|X \setminus I|} \\ &= -\sum_{Y \subsetneq J \setminus I} (-1)^{|Y|} = -\sum_{i=0}^{m} \binom{m}{i} (-1)^i + \binom{m}{m} (-1)^m = (-1)^{|J \setminus I|} \end{aligned}$$

where the second equation follows from induction. This shows that PIE is a special case of Möbius transform.

1.1 Application of PIE

PIE is extensively used in counting problems. Given a matrix A, the determinant is defined as $\texttt{Det}(A) = \sum_{\sigma \in S_n} (-1)^{\|\sigma\|} \prod_{i=1}^{n} A_{i,\sigma(i)}$ where S_n is the set of all permutations and $\|\sigma\|$ is the number of inversion pairs in σ. It can be efficiently calculated in polynomial time.

The permanent of A is defined as $\texttt{Perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)}$. Calculating the permanent is an important counting problem in computer science. Its value has combinatorial meaning. Given a bipartite graph G with n vertices on each side, let A be the matrix describing the connectivity of G in the way that $A_{i,j} = 1$ if there is an edge between the i-th vertex on the left side and the j-th vertex on the right side and $A_{i,j} = 0$ otherwise. The permanent of A is exactly the number of perfect matchings in G.

However, to calculate the permanent is $\#\mathbf{P}$-hard[1]. Note that if $\mathbf{NP} \neq \mathbf{P}$, then SAT has no $n^{O(1)}$ algorithms where n is the number of variables in the given formula. On the other hand, we don't even know any $2^{o(n)}$ algorithms for SAT so far. This leads to a stronger hypothesis, the exponential time hypothesis (ETH), which states that SAT has no $2^{o(n)}$ algorithms. A slightly weaker hypothesis, #ETH, is to conjecture that #SAT has no $2^{o(n)}$ algorithms. Based on #ETH, Rado Curticapean proved the following theorem

[1] $\#\mathbf{P}$ is the class of problems that can be defined as counting the number of accepting paths of a polynomial-time nondeterministic Turing machine. We call a problem $\#\mathbf{P}$-hard if every other problem in $\#\mathbf{P}$ can reduce to it via a polynomial-time Turing reduction.

Theorem 1 $\texttt{Perm}(A)$ *has no* $2^{o(n)}$ *algorithms assuming #ETH.*

This means that if #ETH is true, the best algorithm we can find to calculate $\texttt{Perm}(A)$ needs at least $2^{\Omega(n)}$ time.

Now we come to find good algorithms to compute $\texttt{Perm}(A)$. The first try is to calculate $\texttt{Perm}(A)$ by definition. However, this takes $O(n!) = O\left(2^{n \log n}\right)$ time. The Ryser's formula below gives an equivalent form of $\texttt{Perm}(A)$. The formula can be evaluated by enumerating all $I \subseteq [n]$ in $2^{O(n)}$ time, which is optimal assuming #ETH.

Theorem 2 (Ryser's formula) $\texttt{Perm}(A) = \sum_{I \subseteq [n]} (-1)^{|I|} \prod_{i=1}^{n} \left(\sum_{j \in [n] \setminus I} A_{i,j} \right)$.

Proof. Let U be the set of all mappings $\sigma \colon [n] \to [n]$ satisfying for all $i \in [n], A_{i,\sigma(i)} = 1$. Let $B_i = \{f \in U \mid f \text{ has no preimage for } i\}$ for $i \in [n]$. Then by the PIE, the number of perfect matchings in a bipartite graph corresponding to A is $N_{=}(\varnothing) = \sum_{S \subseteq [n]} (-1)^{|S|} N_{\geqslant}(S) = \sum_{S \subseteq [n]} (-1)^{|S|} \prod_{i=1}^{n} \left(\sum_{j \in [n] \setminus S} A_{i,j} \right)$. [2] □

[2] Here we use the set S to represent $\{B_i \mid i \in S\}$ for brevity.

2 Algorithmic Zeta Transform and Möbius Transform

2.1 Algorithmic Zeta Transform

Consider the poset $\left(P = 2^{[n]}, \subseteq\right)$ and a function $f \colon P \to \mathbb{R}$. The zeta transform of f is $\zeta f \in \mathbb{R}^{2^n}$. How can we calculate ζf for all of its 2^n inputs efficiently?

A simple idea is to use brute-force. That is, we compute $(\zeta f)(X)$ by definition for each of the 2^n X's respectively and this takes at most 2^n time for each X. More specifically, there are $2^{n-|X|}$ such Y's that $Y \geqslant X$. Therefore, the calculation of $\sum_{Y \colon Y \geqslant X} f(Y)$ needs $2^{n-|X|}$ operations. The total time cost is $\sum_{X \subseteq [n]} 2^{n-|X|} = \sum_{i=0}^{n} \binom{n}{i} 2^i = 3^n$.

We can use dynamic programming to accelerate the computation. View each set X as a binary vector $x = (x_1, x_2, \ldots, x_n)$ where $x_i = \mathbf{1}[i \in X]$. Let

$$G(x) = (\zeta f)(X) = \sum_{y \in \{0,1\}^n} \mathbf{1}[y_1 \geqslant x_1, y_2 \geqslant x_2, \cdots, y_n \geqslant x_n] f(y).$$

For $j \in \{0\} \cup [n]$,

$$G_j(x) \triangleq \sum_{y \in \{0,1\}^j} \mathbf{1}[y_1 \geqslant x_1, y_2 \geqslant x_2, \cdots, y_j \geqslant x_j] f(y_1, y_2, \ldots, y_j, x_{j+1}, x_{j+2}, x_n).$$

Therefore we have $G_n(x) = G(x)$ and $G_0(x) = f(x)$ by definition. Note that

$$G_j(x_1, x_2, \ldots, x_n) = \begin{cases} G_{j-1}(x_1, x_2, \ldots, x_n), & x_j = 1 \\ G_{j-1}(x_1, x_2, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n) & \\ \quad + G_{j-1}(x_1, x_2, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n) & x_j = 0 \end{cases}.$$

We can calculate each $G_j(x)$ from the sub-result $G_{i-1}(x)$ in $O(1)$ time and recursively get $G(x)$ in polynomial time. The total time cost of dynamic programming is $2^n \cdot \text{poly}(n)$.

Möbius transform can also be calculated efficiently in a similar way.

2.2 Fast Set Convolution

Consider the problem of q-coloring. That is, given a graph $G = (V, E)$, color each vertex with one of the q colors while guaranteeing no adjacent vertices are monochromatic. The brute-force method to count the number of proper coloring takes $\Omega(q^n)$ time.

One can use dynamic programming to to obtain an $O(3^n)$ algorithm. In the following, we describe a beautiful algorithm based on zeta transformation to count the number for G and all its induced subgraphs in $O(2^n)$ time simultaneously.

Note that a proper coloring corresponds to a feasible partition dividing V into q independent sets. Define the indicator of independent sets $S\colon 2^V \to \{0,1\}$ as $S(X) = \mathbf{1}[X \text{ is an independent set}]$. Then the number of proper colorings in $G[X]$ is[3]

$$\left(\underbrace{S * S * S * \cdots * S}_{q \text{ terms}}\right)[X].$$

[3] The convolution of two functions $f, g\colon 2^{[n]} \to \mathbb{R}$ is defined as $(f * g)(Z) \triangleq \sum_{X \subseteq Z} f(X) g(Z \setminus X)$.

Counting the proper coloring ways can be reduced to calculating the set convolution.

We introduce an auxiliary operation $*_c$ called *cover convolution*:

$$(f *_c g)(Z) = \sum_{X \cup Y \subseteq Z} f(X) g(Y).$$

Evaluation the convolution of two functions can be reduced to calculating cover convolution on functions with *restricted size*. For every $i \in \mathbb{N}$, let $f_i(X) = \mathbf{1}[|X| = i] f(X)$ be the restriction of function f to inputs with size i. Then

$$(f * g)(Z) = \sum_{X \cup Y = Z, X \cap Y = \varnothing} f(X) g(Y) = \sum_{i=0}^{n} (f_i *_c g_{n-i})(Z).$$

Therefore, we can compute $(f * g)(Z)$ in polynomial-time if $(f_i *_c g_{n-i})(Z)$ is known for every i. Consider the poset $\left(2^{[n]}, \leqslant\right)$ where $X \leqslant Y$ iff $Y \subseteq X$[4]. The following theorem is an analogue to the Fourier transform for the convolution of two functions.

[4] To make our discussion more clean, we reverse the definition of $\leqslant$ here!

Theorem 3 $\zeta(f *_c g)(Z) = \zeta f(Z) \cdot \zeta g(Z)$.

Proof.

$$\begin{aligned}
\zeta(f *_c g)(Z) &= \sum_{Y \subseteq Z} (f *_c g)(Y) = \sum_{Y \subseteq Z} \sum_{A \cup B = Y} f(A) \cdot g(B) \\
&= \sum_{A \cup B \subseteq Y} f(A) \cdot g(B) = \sum_{A \subseteq Z} f(A) \cdot \left(\sum_{B \subseteq Z} g(B)\right) \\
&= \zeta f(Z) \cdot \zeta g(Z).
\end{aligned}$$

□

Equipped with Theorem 3, we can get $\zeta(f *_c g)(Z)$ in $2^n \cdot \mathrm{poly}(n)$ time by calculating the zeta transform of f and g respectively. Then we use another $2^n \cdot \mathrm{poly}(n)$ time to calculate the Möbius transform of $\zeta(f *_c g)$ and compute $(f * g)$ sequentially.

This fast set convolution method takes $2^{O(n)}$ time to counting proper q-coloring ways. Note that there are no $2^{o(n)}$ algorithms for q-coloring problem assuming #ETH. This indicates that fast set convolution algorithm is optimal if #ETH holds.