Resumo para Seminário de MO829

Lucas P. Melo
RA 153629

01 de Dezembro de 2016

Este texto é um resumo das seções 10.1 e 10.2 do livro-texto [1] da disciplina de Tópicos em Teoria da Computação (MO829). Essas seções tratam de alguns métodos algébricos para a solução de problemas de decisão.

1 Princípio da Inclusão-Exclusão

O princípio da inclusão-exclusão é dado pelo seguinte lema:

Lema 1. *Dado um conjunto finito U e uma família $A_1, A_2, \ldots, A_n$ de subconjuntos de U,*

$$\left|\bigcup_{i\in[n]} A_i\right| = \sum_{\emptyset\neq X\subseteq[n]} (-1)^{|X|+1} \left|\bigcap_{i\in X} A_i\right|$$

Demonstração. Para verificar esse resultado, basta contar quanto cada elemento $e \in \bigcup_{i\in[n]} A_i$ contribui na expressão do lado direito da igualdade. Suponha que e ocorra nos conjuntos $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$, assim e ocorre somente numa interseção $\bigcap_{i\in X} A_i$ quando $X \subseteq \{i_1, i_2, \ldots, i_k\}$. Portanto, a contribuição de e se torna:

$$\sum_{\emptyset\neq X\subseteq\{i_1,i_2,\ldots,i_k\}} (-1)^{|X|+1} = \sum_{i=1}^{k} \binom{k}{i} (-1)^{i+1} = (-1)((-1+1)^k - 1) = 1$$

conforme desejado. □

Um lema equivalente que nos será mais útil para resolver alguns problemas é:

Lema 2. *Dado um conjunto finito U e uma família $A_1, A_2, \ldots, A_n$ de subconjuntos de U,*

$$\bigcap_{i\in[n]} A_i = \sum_{X\subseteq[n]} (-1)^{|X|} \left|\bigcap_{i\in X} U \setminus A_i\right|.$$

Demonstração. Exercício. □

1.1 Ciclo Hamiltoniano

Seja G um grafo. Um *passeio* de G é uma sequência $v_1e_1v_2e_2\ldots e_\ell v_{\ell+1}$ onde $v_1, v_2, \ldots, v_{\ell+1} \in V(G)$ e $e_1, e_2, \ldots, e_\ell \in E(G)$ e, para $i = 1, 2, \ldots, \ell$, os vértices v_i e v_{i+1} são incidentes em e_i. Note que pode haver repetição de vértices em passeios. O comprimento de um passeio é dado pelo número de arestas pertencentes a ele e o passeio é chamado de *passeio fechado* quando o último vértice da sequência é igual ao primeiro.

Desejamos determinar se existe ciclo Hamiltoniano em G. Faça U ser o conjunto de todos os passeios fechados de comprimento n e, para $v \in V(G)$, faça A_v ser todo passeio fechado de comprimento n que contenha o vértice v. Note que os elementos de $\bigcap_{v\in V(G)} A_v$ são todos os ciclos Hamiltonianos de G, que podemos contar pelo Lema 2 se, para cada $X \subseteq V(G)$, pudermos computar $\left|\bigcap_{i\in X} U \setminus A_i\right|$.

Assim, dado $X \subseteq V(G)$, computaremos $\left|\bigcap_{i\in X} U \setminus A_i\right|$ através da matriz de adjacências M do grafo $G' = G - X$. É possível provar por indução que o elemento da i-ésima linha e j-ésima coluna da matriz M^n é o número de passeios de tamanho n em G' que começam no vértice i e terminam no vértice j. Como precisamos contar o número de passeios fechados, basta somar os valores na diagonal dessa matriz.

Como os elementos de M^n são no máximo n^n, os valores intermediários no algoritmo requerem somente $O(n \log n)$ bits para serem representados e, portanto, o algoritmo possui complexidade de tempo de $2^n n^{O(1)}$ e $O(n^2)$ de espaço.

1.2 Árvore de Steiner

Dado um grafo G, um conjunto de terminais $K \subseteq V(G)$ e um comprimento $\ell \in \mathbb{N}$, desejamos determinar se existe subárvore T de G tal que $K \subseteq V(T)$

e $|E(T)| \leq \ell$.

Para isso, iremos fazer algo similar ao problema do ciclo Hamiltoniano, mas, em vez de contarmos passeios, iremos contar os chamados *passeios com ramificações* (tradução livre do nome original *branching walks*).

Um passeio com ramificações B é uma tupla (H, h), onde H é uma árvore não-rotulada, enraizada e com ordenação de filhos e $h : V(H) \to V(G)$ é um homomorfismo de H para G (um homomorfismo é uma função que respeita a propriedade de que se $xy \in E(H)$ então $h(x)h(y) \in E(G)$). Dizemos que $V(B) = \{h(x) : x \in V(H)\}$, o tamanho de B é o número de arestas de H e que B parte de vértice s quando $s = h(r)$ onde r é raiz de H.

É relativamente simples mostrar que, dado $s \in K$, G possui árvore de Steiner de comprimento máximo ℓ que engloba todos os terminais sse existe passeio com ramificações $B = (H, h)$ partindo de s tal que $K \subseteq V(B)$ e $|E(H)| = \ell$. Assim, basta contarmos o número de passeios com ramificações com essas características.

Fixe um $s \in K$. Faça U ser o conjunto de passeios com ramificação em G de comprimento ℓ que partem de s e defina $A_v = \{B \in U : v \in V(B)\}$. Desejamos contar o número de elementos de $\bigcap_{v \in K} A_v$ pelo Lema 2.

Para isso, para cada $X \subseteq K$, desejamos encontrar $\left|\bigcap_{v \in X} U \setminus A_v\right|$ quando $s \notin X$ (quando $s \in X$, consideramos que a expressão é 0 pois os passeios com ramificações precisam partir de s). Faça $G' = G - X$, e, para $v \in V(G')$ e $j \in \{0, 1, \ldots, \ell\}$, defina $b_j(v)$ como o número de passeios com ramificações (em G') partindo de v. Podemos computá-los de maneira recursiva:

$$b_j(v) = \begin{cases} 1 & \text{se } j = 0 \\ \sum_{t \in N_{G'}(v)} \sum_{j_1 + j_2 = j-1} b_{j_1}(v) b_{j_2}(t) & \text{caso contrário.} \end{cases}$$

Note que $\left|\bigcap_{v \in X} U \setminus A_v\right| = b_\ell(s)$.

Como precisamos de $O(\ell \log n)$ bits para representar os resultados intermediários, temos um algoritmo de tempo $2^{|K|} n^{O(1)}$ e espaço polinomial para o problema de decisão.

1.3 k-Coloração

Dado grafo G, uma k-coloração é uma função $\chi : V(G) \to [k]$ tal que, para cada aresta $uv \in E(G)$, $\chi(u) \neq \chi(v)$. G é dito k-colorível sse existe k-coloração para G. Além disso, existe k-coloração sse existem conjuntos

$I_1, I_2, \ldots, I_k \subseteq V(G)$ tais que $\bigcup_{i\in[k]} I_i = V(G)$ e, para $i = 1, 2, \ldots, k$, conjunto I_i é independente (um conjunto $X \subseteq V(G)$ é considerado *independente* sse não há $u, v \in X$ tais que $uv \in E(G)$).

Esse último fato nos permite usar o Lema 2 para decidirmos se um determinado grafo G é k-colorível. Faça U ser o conjunto de k-tuplas de conjuntos independentes em G e, para cada $v \in V(G)$ e faça

$$A_v = \left\{ (I_1, I_2, \ldots, I_k) \in U : v \in \bigcup_{i\in[k]} I_i \right\}.$$

Novamente queremos computar $\left|\bigcap_{v\in V(G)} A_v\right|$ utilizando o Lema 2 e, assim, precisamos computar $\left|\bigcap_{v\in X} U \setminus A_v\right|$ para cada $X \subseteq V(G)$.

Para cada $Y \subseteq V(G)$, definamos $s(Y)$ como o número de conjuntos independentes no grafo induzido $G[Y]$. Podemos precomputar a função s com 2^n operações aritméticas usando o fato que $s(\emptyset) = 1$ e a seguinte recursão (fixando vértice arbitrário $y \in Y$):

$$s(Y) = s(Y \setminus \{y\}) + s(Y \setminus N[y]),$$

onde $N[y]$ é a vizinhança fechada de y.

Dessa maneira, obtemos que

$$\left|\bigcap_{v\in X} U \setminus A_v\right| = s(V(G) \setminus X)^k.$$

Ao total, o algoritmo possui complexidade de tempo e espaço de $2^n n^{O(1)}$ pois os valores intermediários atingidos pelo algoritmo requerem $O(nk)$ bits para serem representados.

Para obter espaço polinomial, podemos computar $s(V(G) \setminus X)$ separadamente para cada subconjunto $X \subseteq V(G)$ em tempo $2^{|V(G)|-|X|} n^{O(1)}$ com um algoritmo de força-bruta de espaço polinomial, resultando em um procedimento de tempo:

$$\sum_{X\subseteq V(G)} 2^{|V(G)|-|X|} n^{O(1)} = \sum_{\ell=0}^{n} \binom{n}{\ell} 2^{n-\ell} n^{O(1)} = 3^n n^{O(1)}$$

O exercício 3.17 do livro-texto permite computar $s(V(G) \setminus X)$ em tempo $1,381^{n-|X|} n^{O(1)}$ e o algoritmo conhecido de menor complexidade faz o mesmo

cálculo em tempo $1,238^{n-|X|}n^{O(1)}$, resultando, respectivamente em algoritmos de k-coloração de complexidades $2,381^n n^{O(1)}$ e $2,238^n n^{O(1)}$ respectivamente e espaço polinomial.

2 Transformadas Zeta e Möbius

Seja o anel $(\mathbb{Z}, +, \cdot)$ (que pode ser substituído por qualquer outro anel) e um conjunto finito V qualquer, dada uma função $f : 2^V \to \mathbb{Z}$, definimos:

1. Transformada Zeta: $(\zeta f)(X) = \sum_{Y \subseteq X} f(Y)$;
2. Transformada Möbius: $(\mu f)(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} f(Y)$;
3. Negação-ímpar: $(\sigma f)(X) = (-1)^{|X|} f(X)$.

Podemos observar que $\zeta = \sigma\mu\sigma$ e $\mu = \sigma\zeta\sigma$. Além disso, $(\mu\sigma f)(X) = (\sigma\mu f)(X) = f(X)$ pelo Teorema 10.11 do livro-texto, ou seja, o operador inverso da transformada de Möbius é a transformada Zeta e vice-versa.

Dado grafo G, seja U definido conforme a Seção 1.3 que trata do problema da k-coloracão. Defina uma função $f : 2^{V(G)} \to \mathbb{Z}$ onde

$$f(X) = \left| \left\{ (I_1, I_2, \ldots, I_k) \in U : \bigcup_{i \in [k]} I_i = X \right\} \right|.$$

Essa função nos permitiria determinar se há k-coloração para cada subgrafo induzido de G pois, para $X \subseteq V(G)$, $G[X]$ possui k-coloração sse $f(X) \neq 0$ conforme observado na Seção 1.3.

Note, no entanto, que ζf é mais fácil determinar que f pois, para cada $X \subseteq V(G)$, $(\zeta f)(X)$ é exatamente $s(X)^k$ onde $s(X)$ é o número de conjuntos independentes do subgrafo induzido $G[X]$.

Assim, poderíamos precomputar ζf com a recorrência definida na Seção 1.3 em tempo $2^n n^{O(1)}$ e então usar a transformada de Möbius para obter a função f original. Se usásemos a definição da transformada de Möbius de maneira ingênua, obteríamos um algoritmo de tempo:

$$\sum_{X \subseteq V(G)} 2^{|X|} n^{O(1)} = \sum_{\ell=0}^{n} \binom{n}{\ell} 2^\ell n^{O(1)} = 3^n n^{O(1)}$$

No entanto, é possível computar a transformada com $O(2^n n)$ operações aritméticas, obtendo um algoritmo de tempo $2^n n^{O(1)}$, com o seguinte teorema:

Teorema 1 (Transformada rápida Zeta e de Möbius)**.** *Dada função $f : 2^{[n]} \to \mathbb{Z}$ como entrada, podemos computar os 2^n valores de ζf e μf com $O(2^n n)$ operações aritméticas.*

Demonstração. Primeiro encontraremos o algoritmo para a transformada Zeta. Reinterprete f como uma função que recebe n bits que determinam o subconjunto considerado, i.e. defina $f(x_1, x_2, \ldots, x_n) = f(X)$, onde x_i é um bit 0 ou 1 (para $i \in [n]$) e $X = \{i \in [n] : x_i = 1\}$.

Defina $\zeta_j(x_1, \ldots, x_n)$ (para $j \in \{0, 1, \ldots, n\}$) como

$$\sum_{y_1, y_2, \ldots, y_j \in \{0,1\}} [y_1 \leq x_1, y_2 \leq x_2, \ldots, y_j \leq x_j] f(y_1, y_2, \ldots, y_j, x_{j+1}, \ldots, x_n),$$

onde, dada uma fórmula booleana F, $[F]$ é 1 quando a expressão possui valor verdadeiro e 0 caso contrário.

Obtemos a seguinte recorrência:

$$\zeta_j(x_1, \ldots, x_n) = \begin{cases} \zeta_{j-1}(x_1, \ldots, x_n) & \text{quando } x_j = 0, \\ \zeta_{j-1}(\ldots, x_{j-1}, 0, x_{j+1}) + \zeta_{j-1}(\ldots, x_{j-1}, 1, x_{j+1}, \ldots) & \text{caso contrário.} \end{cases}$$

Que pode ser computada com $O(2^n n)$ operações aritméticas para todos as possíveis entradas $x_1, \ldots, x_n \in \{0, 1\}$ e $j \in \{0, 1, \ldots, n\}$. Note que $\zeta f = \zeta_n$.

Para obter μf, basta usar o fato de que $\mu = \sigma \zeta \sigma$. □

3 Exercício

Como exercício designado para esse seminário, selecionamos a prova do Lema 2.

Referências

[1] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.