Enumeração de Subconjuntos de um Conjunto

Kaio Augusto de Camargo
Departamento de Informática
Universidade Federal do Paraná - UFPR
Curitiba, PR, Brasil
Email: kac14@inf.ufpr.br

***Resumo*—Esse trabalho visa apresentar dois problemas referentes a enumeração de subconjuntos de um conjunto — mais especificamente, enumerar o próximo subconjunto e enumerar todos os subconjuntos. Serão apresentadas soluções para esses problemas, bem como suas respectivas provas de corretude.**

1. Definições dos problemas

1.1. Enumeração do próximo subconjunto

Dado dois conjuntos $C = \{x \in \mathbb{Z} \mid 0 > x \geq n\}$ e $K \subset C$, queremos encontrar um terceiro conjunto $X \subset C$ tal que $|X| = |K|$ e $X \neq K$. A relação entre X e K deve ser tal que possamos encontrar o próximo subconjunto de X, o próximo subconjunto do próximo conjunto de X, e assim *ad infinitum*, até que eventualmente voltemos ao ponto de partida: Isto é, onde o próximo subconjunto seja o nosso C original.

Chamemos de S o conjunto de todos os subconjuntos gerados por um algoritmo que resolva este problema. Para ser realmente uma solução, esse conjunto S gerado por um algoritmo deve obedecer duas regras simples:

$$s \subset C, \forall s \in S \tag{1}$$

$$\{\bigcup_{i=1}^{|K|} x_i \mid x_i \in C\} \in S \tag{2}$$

1.2. Enumeração de todos os subconjuntos

Similar ao item acima, porém ao invés de nos restringirmos apenas ao próximo conjunto, uma solução para este problema deve gerar o conjunto S com todos os subconjuntos de uma vez. Obviamente, um algoritmo que acha o próximo subconjunto também resolveria esse problema, porém podemos ter algum ganho de performance se gerarmos todos os subconjuntos de uma só vez.

2. Enumeração de próximo subconjunto

2.1. Proposta de solução

Proporemos o algoritmo $S(|C|, K)$ para resolver o problema, onde $|C|$ é o tamanho do conjunto original C e K é o subconjunto K ao qual se deseja encontrar o próximo. Por fins práticos (de indexação), definiremos que K é uma tupla ao invés de um conjunto.

$$S(|C|, K) = \begin{cases} (1), \text{ se } |K| = 1 \text{ e } K_1 > |C| \\ (K_1, K_2, \ldots, K_{|C|} + 1), \text{ se } K_n \neq |C| - 1 \\ (x_i \mid 0 > i \geq |C|, x_i = \begin{cases} S(|C| - 1, K - K_n)_i \\ \text{, se } i < |C| \\ x_{i-1} + 1 \end{cases} \\ \text{caso contrário.} \end{cases}$$

2.2. Prova de corretude

A primeira base da recursão basicamente é um artifato para que o próximo subconjunto do último subconjunto (em ordem lexografica) volte a ser o primeiro algoritmo em ordem lexografica.

A segunda base da recursão é o caso onde calcular o próximo subconjunto em ordem lexográfica é trivial, somando 1 ao último elemento do conjunto.

O passo normal da recursão se dá quando descobrir o próximo subconjunto em ordem lexográfica não é trivial, pois o último elemento do subconjunto atual é o último elemento do conjunto. Nesse caso, dividimos o problema em um subproblema menor: Encontrar o próximo subconjunto em ordem lexográfica do subconjunto atual, porém retirando o último elemento desse mesmo conjunto. Depois que encontrarmos a solução desse subproblema, o último elemento do novo subconjunto deve ser o último elemento da solução do subproblema acrescido de 1.

3. Enumeração de todos os subconjuntos

3.1. Proposta de solução

Proporemos o algoritmo $T(|C|, |K|, S)$, onde $|C|$ é o tamnho do conjunto original C, $|K|$ é o tamanho dos subconjuntos desejados e S representa os elementos que já estão no conjunto em uma determinada iteração.

$$T(|C|, |K|, S) = \begin{cases} \{\}, \text{ se } |K| > |C| \\ \\ S, \text{ se } |K| = 0 \\ \\ T(|C| - 1, |K|, S) \cup \\ \\ T(|C| - 1, |K| - 1, S \cup |C|) \end{cases}$$

3.2. Prova de corretude

Os dois casos bases da recorrência são triviais: O primeiro se dá quando o subconjunto a ser gerado precisa de mais elementos do que estão sobrando no conjunto original, ou seja, não haverão subconjuntos gerados em qualquer iteração a partir daquele ponto, e, portanto, o algoritmo retorna um conjunto vazio para a iteração anterior. O segundo se dá quando o algoritmo incluiu |K| elementos em S, ou seja, quando o algoritmo tem um subconjunto em S, e logicamente apenas retorna o dito subconjunto. Por fim, o passo normal da recursão se dá por unir os subconjuntos que incluem o último elemento do conjunto atual e os que não incluem esse mesmo elemento, garantindo que todos os subconjuntos serão gerados.