

CI1238 - Otimização

Aula 09 - Algoritmos de Enumeração (Parte 1)

Professor Murilo V. G. da Silva

Departamento de Informática
Universidade Federal do Paraná

2026 / Primeiro Semestre

Contextualizando:

Problemas Computacionais e Algoritmos

Conceitos Preliminares

Introdução à Problemas de Otimização

Parte 1: Programação Linear

1.1 Modelagem por Programação Linear

1.2 Algoritmo Simplex

1.3 Programação Linear Inteira

Parte 2: Técnicas combinatórias em algoritmos

2.1 **Enumeração** (assunto de hoje)

2.2 Backtracking

2.3 Branch-and-bound

2.4 Programação dinâmica

2.5 Algoritmos gulosos

2.6 Algoritmos de Aproximação

Algumas considerações preliminares

Técnicas combinatórias, algoritmos combinatórios:

- ▶ Técnicas para resolução de problemas: programação dinâmica, divisão e conquista, algoritmos gulosos, etc
 - ▶ Técnicas baseadas em estruturas combinatórias: conjuntos (discretos), partições, permutações, grafos, etc
- Atenção:** mas não existem parâmetros reais (pesos, custos)? → Sim, mas as ideias chaves e a “maneira de pensar” são combinatórias

Programação linear:

- ▶ Também é uma técnica (bem geral) para resolução de problemas
 - ▶ Pensada em termos de números e funções reais, em espaço \mathbb{R}^n .
- Atenção:** mas na prática não é impossível trabalhar com reais em computadores? → Sim, mas este não o argumento central aqui. O Simplex é um algoritmo que lida com equações de números reais. A representação finita de reais em computadores é uma questão separada.
- Atenção:** o Simplex não é apenas manipulação de tableaux? → Sim, mas programação linear (e mesmo o Simplex) é mais naturalmente entendida em termos de reais.

Considerações acima: não são nosso foco, são apenas para vocês pensarem um pouco.

- ▶ Nosso foco neste curso: técnicas para projetos de algoritmos

Técnicas combinatórias em algoritmos

Mais considerações preliminares:

- ▶ Tópico da aula de hoje: [algoritmos de enumeração](#)
 - ▶ Algoritmos de Busca/Otimização podem usar enumeração. (normalmente quando o objetivo é percorrer todo espaço de busca)
 - ▶ Algoritmos de Enumeração também são interessantes por si mesmos
 - ▶ Livro para este tópico: [KS99] – **Combinatorial Algorithms: Generation, Enumeration, and Search**

Mais uma última questão preliminar!

Atentar para vocabulário:

- ▶ **Contagem**: *determinar* quantas estruturas de um certo tipo existem.

Exemplo: Dado conjunto S de tamanho n , queremos saber **quantos** subconjuntos $S' \subseteq S$ de tamanho k existem? **resp:** $\binom{n}{k}$.

No livro: enumeration.

- ▶ **Enumeração**: *listar/gerar/enumerar* estruturas combinatórias.

Exemplo: Dado conjunto S de tamanho n , **listar cada um** dos subconjuntos $S' \subseteq S$ tal que $|S'| = k$.

No livro: generation.

Algoritmos para enumeração

Seja A um conjunto de objetos combinatórios que queremos enumerar.

- ▶ Seja $|A| = N$.
- ▶ Para **enumerar** objetos, normalmente precisamos estabelecer em qual **ordem** estes objetos vão ser enumerados.
- ▶ Ou seja, **associado** um algoritmo de enumeração, temos uma função **$rank()$**
 - uma função que associa a um objeto, a sua posição na enumeração.
 - esta função pode ou não pode ser explicitamente definida

A função **$rank()$** tem o seguinte domínio e imagem:

- ▶ $rank : A \rightarrow \{0, \dots, N - 1\}$

Domínio e imagem de funções relacionadas com a função **$rank()$** :

- ▶ **$unrank$** : $\{0, \dots, N - 1\} \rightarrow A$
- ▶ **$successor$** : $A \rightarrow A$ (mais no próximo slide)

Algoritmos para enumeração

- ▶ Dada uma função $rank : A \rightarrow \{0, \dots, N - 1\}$, temos:

Para $unrank()$, a ideia é que $\forall a \in A$ e $\forall i \in \{0, \dots, N - 1\}$, temos:

$$rank(a) = i \Leftrightarrow unrank(i) = a$$

Para $sucessor()$, queremos:

$$sucessor(a) = b \Leftrightarrow rank(b) = rank(a) + 1$$

Definindo $sucessor()$ em termos de $rank()$ e $unrank()$:

$$sucessor(a) = \begin{cases} unrank(rank(a) + 1), & \text{se } rank(a) < N - 1, \\ \text{indefinida,} & \text{se } rank(a) = N - 1. \end{cases}$$

- ▶ Em alguns contextos é conveniente fazer o inverso: definimos primeiro $sucessor()$ e depois definimos $rank()$ e $unrank()$ a partir disso.
- ▶ Às vezes podemos definir separadamente para cada uma das três funções

Neste curso: veremos dois algoritmos de enumeração:

- (1) Algoritmo em que $rank()$ é dado pela **ordenação lexicográfica** de A
- (2) Algoritmo em que $rank()$ é dado por uma **ordenação de mudança mínima**

Ranking definido por Ordem Lexicográfica

Qual é o conjunto A que queremos enumerar?

- ▶ Precisamos saber concretamente quem é A (e.g., todas cliques de um grafo)
- ▶ Mas queremos estudar algoritmos “suficientemente” gerais.

Nosso objetivo: Dado $S = \{1, \dots, n\}$, queremos enumerar todos os subconjuntos de S .

- ▶ Ou seja, o conjunto a ser enumerado é $A = \mathcal{P}(S)$
- ▶ Algoritmos de enumeração lexicográfica: Além de enumeração de todos os subconjuntos de um dado conjunto, também pode ser usado em enumeração de todo subconjunto de tamanho k , toda partição, toda permutação, etc... (cada caso tem especificidades)
- ▶ Algoritmo que veremos é aplicável em enumeração de cliques, conjuntos independentes, todo emparelhamento, etc
- ▶ Note: Como $N = |A|$ e $n = |S|$, vamos enumerar N elementos, sendo $N = 2^n$

Enumeração em Ordem Lexicográfica

Do slide anterior: Dado $S = \{1, \dots, n\}$, queremos enumerar $A = \mathcal{P}(S)$

Dado $T \in \mathcal{P}(S)$, o vetor característico de T é o vetor binário $\chi(T) = [x_{n-1}, \dots, x_1, x_0]$

$$\text{tal que } x_i = \begin{cases} 1 & \text{se } n-i \in T, \\ 0 & \text{se } n-i \notin T \end{cases} \quad i \in [0..n-1].$$

Simplificação da notação: às vezes usaremos χ_T ou invés de $\chi(T)$

Exemplo: Seja $S = \{1, 2, 3\}$ e $T = \{2, 3\}$, qual o valor $\chi(T)$?

► Sendo $\chi(T) = [x_2, x_1, x_0]$, vamos determinar os valores de x_2, x_1, x_0 :

$$i = 2: \quad 3 - 2 = 1 \notin T \therefore x_2 = 0$$

$$i = 1: \quad 3 - 1 = 2 \in T \therefore x_1 = 1$$

$$i = 0: \quad 3 - 0 = 3 \in T \therefore x_0 = 1 \quad \text{Portanto } \chi(T) = 011$$

Outro exemplo: Se $T = \{1\}$, então $\chi(T) = 100$

Dado $T \in \mathcal{P}(S)$, definimos $\text{rank}(T)$ como o inteiro cuja representação binária é $\chi(T)$.

$$\text{rank}(T) = \sum_{i=0}^{n-1} x_i \cdot 2^i$$

Enumeração em Ordem Lexicográfica

Vamos considerar o caso $n = 3$ (ou seja, $S = \{1, 2, 3\}$)

T	$\chi(T) = [x_2, x_1, x_0]$	$\text{rank}(T)$
\emptyset	$[0, 0, 0]$	0
$\{3\}$	$[0, 0, 1]$	1
$\{2\}$	$[0, 1, 0]$	2
$\{2, 3\}$	$[0, 1, 1]$	3
$\{1\}$	$[1, 0, 0]$	4
$\{1, 3\}$	$[1, 0, 1]$	5
$\{1, 2\}$	$[1, 1, 0]$	6
$\{1, 2, 3\}$	$[1, 1, 1]$	7

Enumeração em Ordem Lexicográfica

Algoritmos para $\text{rank}(T)$ e $\text{unrank}(r)$

No livro [KS99], chamados de $\text{SUBSETLEXRANK}()$ e $\text{SUBSETLEXUNRANK}()$

Algorithm 2.1: $\text{SUBSETLEXRANK}(n, T)$

```
 $r \leftarrow 0$   
for  $i \leftarrow 1$  to  $n$   
  do  $\begin{cases} \text{if } i \in T \\ \text{then } r \leftarrow r + 2^{n-i} \end{cases}$   
return  $(r)$ 
```

Algorithm 2.2: $\text{SUBSETLEXUNRANK}(n, r)$

```
 $T \leftarrow \emptyset$   
for  $i \leftarrow n$  downto  $1$   
  do  $\begin{cases} \text{if } r \bmod 2 = 1 \\ \text{then } T \leftarrow T \cup \{i\} \\ r \leftarrow \lfloor \frac{r}{2} \rfloor \end{cases}$   
return  $(T)$ 
```

Exercício para casa: Para $n = 3$, execute os algoritmos $\text{rank}(\{2, 3\})$ e $\text{unrank}(4)$.

Enumeração em Ordem Lexicográfica

Algorithm 2.1: SUBSETLEXRANK (n, T)

```
 $r \leftarrow 0$   
for  $i \leftarrow 1$  to  $n$   
  do  $\left\{ \begin{array}{l} \text{if } i \in T \\ \text{then } r \leftarrow r + 2^{n-i} \end{array} \right.$   
return ( $r$ )
```

Soma calculada na execução do algoritmo para $n = 8$ e $T = \{1, 3, 4, 6\}$:

$$\begin{aligned} \text{rank}(T) &= 2^7 + 2^5 + 2^4 + 2^2 \\ &= 128 + 32 + 16 + 4 \\ &= 180. \end{aligned}$$

Enumeração em Ordem Lexicográfica

Algorithm 2.2: SUBSETLEXUNRANK (n, r)

```
 $T \leftarrow \emptyset$   
for  $i \leftarrow n$  downto 1  
  do  $\begin{cases} \text{if } r \bmod 2 = 1 \\ \text{then } T \leftarrow T \cup \{i\} \\ r \leftarrow \lfloor \frac{r}{2} \rfloor \end{cases}$   
return ( $T$ )
```

Construção de T para $n = 8$ e $r = 180$:

i	r	$r \bmod 2$	T
8	180	0	\emptyset
7	90	0	\emptyset
6	45	1	{6}
5	22	0	{6}
4	11	1	{4, 6}
3	5	1	{3, 4, 6}
2	2	0	{3, 4, 6}
1	1	1	{1, 3, 4, 6}.

Enumeração em Ordem Lexicográfica

Seja $S = \{1, 2, \dots, n\}$. Sendo $N = 2^n$, segue o algoritmo de enumeração para $\mathcal{P}(S)$:

Algoritmo de enumeração:

```
for  $i = 0$  to  $N - 1$  do  
  | Print(unrank( $i$ ));  
end
```

Ok, mas se tivermos um conjunto S' diferente de $S = \{1, 2, \dots, n\}$?

- ▶ Obtenha uma bijeção $S \leftrightarrow S'$ (qualquer uma serve)