BCM0505-15 - Processamento da Informação

Feb 10, 2020

Laboratório 04 – 10/03

Permutações

Uma **permutação** para um conjunto finito $A = \{a_0, a_1, \ldots, a_{n-1}\}$ é uma função bijetora $\sigma : A \to A$ que mapeia elementos distintos de A a elementos distintos de A. O mapeamento estabelecido por σ pode ser interpretado como substituições entre os elementos de A. Informalmente, permutações são rearranjos ou embaralhamentos dos elementos de A.

É conveniente indexar os elementos de A — como fizemos — e listá-los como uma sequência, com a_i na posição i, para $0 \leq i < n$:

$$\langle a_0, a_1, \ldots, a_{n-1} \rangle =: \mathsf{id}.$$

Definindo $[\![n]\!] := \{0, 1, \ldots, n-1\}$, podemos abusar da notação e reinterpretar σ como uma função bijetora $\sigma : [\![n]\!] \to [\![n]\!]$ que mapeia índices a índices. A sequência id acima é denominada permutação **identidade,** pois

$$\sigma(A) := \langle a_{\sigma(0)}, a_{\sigma(1)}, \ldots, a_{\sigma(n-1)} \rangle = \langle a_0, a_1, \ldots, a_{n-1} \rangle,$$

com $\sigma(i) = i$.

A título de ilustração:

- uma permutação $\pi : [\![n]\!] \to [\![n]\!]$ para A definida por $\pi(i) = n - i - 1$ equivale a

  $$\pi(A) := \langle a_{\pi(0)}, a_{\pi(1)}, \ldots, a_{\pi(n-1)} \rangle = \langle a_{n-1}, a_{n-2}, \ldots, a_0 \rangle,$$

  que é a permutação **reversa** à id;
- uma permutação $\tau : [\![n]\!] \to [\![n]\!]$ para A definida por $\tau(i) = (i+1) \mod n$ equivale a

  $$\tau(A) := \langle a_{\tau(0)}, \ldots, a_{\tau(n-2)}, a_{\tau(n-1)} \rangle = \langle a_1, \ldots, a_{n-1}, a_0 \rangle,$$

  que é a permutação **cíclica à esquerda** de ordem 1 à id;
- uma permutação $\rho : [\![n]\!] \to [\![n]\!]$ para A definida por $\rho(i) = (i-2) \mod n$ equivale a

  $$\rho(A) := \langle a_{\rho(0)}, a_{\rho(1)}, a_{\rho(2)}, \ldots, a_{\rho(n-1)} \rangle = \langle a_{n-2}, a_{n-1}, a_0, \ldots, a_{n-3} \rangle,$$

  que é a permutação **cíclica à direita** de ordem 2 à id;
- uma permutação $\theta : [\![n]\!] \to [\![n]\!]$ para A definida por

  $$\theta(i) = \begin{cases} i/2 & \text{se } i \text{ é par,} \\ \lfloor n/2 \rfloor + \lfloor i/2 \rfloor & \text{se } i \text{ é ímpar,} \end{cases}$$

  equivale a

  $$\theta(A) := \langle a_{\theta(0)}, a_{\theta(1)}, a_{\theta(2)}, \ldots, a_{\theta(n-1)} \rangle = \langle a_0, a_2, a_4, \ldots, a_1, a_3, a_5, \ldots \rangle,$$

  uma permutação em que os elementos de índice par em id ocorrem antes dos elementos de índice ímpar, ambos os grupos de índices de forma crescente.

Não é difícil perceber que a composição de permutações é uma permutação, que a permutação identidade (id) age como elemento neutro da composição, e que para cada permutação σ existe uma única permutação **inversa** σ^{-1} tal que $\sigma \circ \sigma^{-1} = \sigma^{-1} \circ \sigma = \mathsf{id}$.

Representação em Python

Podemos utilizar listas para representar permutações. Armazenamos o i-ésimo elemento da sequência descrevendo a permutação na i-ésima posição da lista. Por exemplo, a permutação identidade $\mathsf{id} = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ pode ser representada por uma lista

$$A[0 \,..\, n-1] = [a_0, a_1, \ldots, a_{n-1}].$$

A permutação reversa $\pi(\mathsf{id}) = \langle a_{n-1}, a_{n-2}, \ldots, a_0 \rangle$ pode ser representada por outra lista

$$B[0 \,..\, n-1] = [a_{n-1}, a_{n-2}, \ldots, a_0].$$

Observe que a orderm dos elementos na lista corresponde à ordem dos elementos na sequência — como esperado — e não à indexação original dos elementos. Isto é, $B[0]$ é igual a a_{n-1} e não a a_0.

Em certas circunstâncias, é conveniente representarmos uma permutação por uma lista $C[1\,..\,n]$ em vez de $C[0\,..\,n-1]$. Nestes casos, $C[1\,..\,n]$ deve ser interpretada como uma lista $C[0\,..\,n]$ com $n+1$ elementos em que ignoramos a posição 0.

Ordem entre permutações

Quando os elementos do conjunto A são comparáveis, podemos estabelecer ordens entre diferentes permutações para A.

Dadas duas permutações $B[0\,..\,n-1]$ e $C[0\,..\,n-1]$ para $A = \{0, 1, \ldots, n-1\}$, dizemos que B é *lexicograficamente menor* que C, indicado por $B < C$, se existe um índice $j \in \{0, 1, \ldots, n-1\}$ tal que

- $B[i] = C[i]$ para $0 \leq i < j$, e
- $B[j] < C[j]$.

Por exemplo, $[1, 2, 3, 4] < [1, 3, 2, 4]$; $[1, 2, 3, 4] < [1, 3, 4, 2]$; e $[1, 3, 2, 4] < [1, 3, 4, 2]$.

Exercícios

- **(01)** *Enumeração I.*
  Escreva uma função que devolve em uma lista todas as $4!$ permutações para $\{1, 2, 3, 4\}$, em ordem lexicográfica. Isto é, sua função deve devolver:

  $$\big[[1, 2, 3, 4], [1, 2, 4, 3], [1, 3, 2, 4], [1, 3, 4, 2], \ldots\big].$$

  Dica: utilize as funções `append()` e `extend()` para inserir elementos ao final de uma lista.

- **(02)** *Enumeração II.*
  Agora, escreva uma função que devolve em uma lista todas as $5!$ permutações para $\{1, 2, 3, 4, 5\}$.

Observe que as soluções dos exercícios (01) e (02), baseadas em laços aninhados, podem ser estendidas para qualquer $n \geq 1$ fixo — via incremento de código. Tais soluções são *não uniformes*, já que são específicas para cada valor de n. No exercício (12), vamos revisitar este problema de enumeração e desenvolver um algoritmo *uniforme*, que funcione para qualquer valor inteiro de $n \geq 1$.

- **(03)** *Pontos fixos e desarranjos.*
  Seja $A[1\,..\,n]$ uma permutação para $\{1, 2, \ldots, n\}$. Um *ponto fixo* de A é um índice j tal que $A[j] = j$.
  Escreva uma função que recebe A e devolve uma lista com os pontos fixos de A.

  Nota: uma permutação que não possui pontos fixos é chamada *desarranjo*.

- **(04)** *Permutações cíclicas I.*
  Uma permutação $A[0\,..\,n-1]$ de $\langle 0, 1, \ldots, n-1\rangle$ é *cíclica* se existe um índice j tal que

  $$A[0\,..\,j-1] = \langle n-j, n-j+1, \ldots, n-1\rangle \quad \text{e} \quad A[j\,..\,n-1] = \langle 0, 1, \ldots, n-j-1\rangle.$$

  Por exemplo, $\langle 0, 1, 2, 3, 4\rangle$ e $\langle 3, 4, 0, 1, 2\rangle$ são cíclicas; $\langle 0, 3, 4, 1, 2\rangle$ e $\langle 4, 3, 2, 1, 0\rangle$ não são.

  Escreva uma função que recebe A e determina se ela é uma permutação cíclica de $\langle 0, 1 \ldots, n-1\rangle$. Em caso afirmativo, devolva um tal índice j; em caso contrário, devolva $j = n$.

- **(05)** *Permutações cíclicas II.*
  O exercício anterior é um caso particular deste. Dadas duas listas $A[0\,..\,n-1]$ e $B[0\,..\,n-1]$, dizemos que B é uma *permutação cíclica* de A se existe um índice j tal que

  $$A[0\,..\,j-1] = B[n-j\,..\,n-1] \quad \text{e} \quad A[j\,..\,n-1] = B[0\,..\,n-j-1].$$

  Se B é uma permutação cíclica de A, claramente A é uma permutação cíclica de B. Escreva uma função que recebe A e B e devolve um tal índice j se A e B são permutações cíclicas, ou devolve $j = n$ em caso contrário.

- **(06)** *Permutações adjacentes.*
  Duas permutações $A[0\,..\,n-1]$ e $B[0\,..\,n-1]$ para $\{0, 1, \ldots, n-1\}$ são *adjacentes* se existe um índice j tal que $k = (j+1) \mod n$ e

  $$A[j] = B[k], \quad A[k] = B[j] \quad \text{e} \quad A[i] = B[i] \text{ para } i \in \{1, 2, \ldots, n\} \setminus \{j, k\}.$$

  Em outras palavras, A e B diferem somente nas posições j e k. Escreva uma função que recebe A e B e determina se elas são adjacentes.

- **(07)** *Permutações inversas I.*
  Duas permutações $A[1\,..\,n]$ e $B[1\,..\,n]$ para $\{1, 2, \ldots, n\}$ são *inversas* se $A[B[i]] = B[A[i]] = i$ para todo índice i. Em outras palavras, se a composição de A com B ou de B com A resulta na permutação identidade. Escreva uma função que recebe A e B e determina se elas são inversas.

- **(08)** *Permutações inversas II.*
  Escreva uma função que recebe uma permutação $A[1\,..\,n]$ para $\{1, 2, \ldots, n\}$ e devolve **a** permutação inversa a A.
  Nota: veja exercício anterior para definição.

- **(09)** *Permutações alternantes.*
  Uma permutação $A[1\,..\,n]$ é *alternante* se

$$A[1] < A[2] > A[3] < A[4] > \cdots \lesseqgtr A[n] \qquad \text{ou} \qquad A[1] > A[2] < A[3] > A[4] < \cdots \gtreqless A[n].$$

  Escreva uma função que recebe A e determina se ela é alternante.

- **(10)** *Verificação de permutações.*
  Escreva uma função que recebe uma lista de `int`s $A[1\,..\,n]$ e determina se A é uma permutação para $\{1, 2, \ldots, n\}$.
  Dica: utilize uma lista auxiliar.

- **(11)** *Inversões em permutações.*
  Seja $A[1\,..\,n]$ uma permutação para $\{1, 2, \ldots, n\}$. Dois índices $i < j$ são uma *inversão* em A se $A[i] > A[j]$.
  Escreva uma função que recebe A e devolve o número de inversões em A.

- **(12)** *Enumeração III.* *
  Escreva uma função que recebe um inteiro $n \geq 1$ e imprime todas as $n!$ permutações de $\{1, 2, \ldots, n\}$ em ordem lexicográfica.

- **(13)** *Caminho do cavalo.*
  Suponha dado um tabuleiro de xadrez $n \times n$, com $n \geq 1$, e cujas casas são numeradas por

$$(1,1), (1,2), \ldots, (1,n), (2,1), (2,2), \ldots, (2,n), (3,1), (3,2), \ldots, (n,n). \qquad (*)$$

  Determine se é possível a um *cavalo de jogo de xadrez* partir da posição $(1,1)$, visitar todas as demais casas do tabuleiro uma única vez em $n^2 - 1$ passos válidos. Por exemplo, para um tabuleiro 5×5 uma solução é

	1	2	3	4	5
1	1	6	15	10	21
2	14	9	20	5	16
3	19	2	7	22	11
4	8	13	24	17	4
5	25	18	3	12	23

  em que um número k na casa (i,j) representa que ela é a k-ésima casa visitada ao longo do caminho.
  Dica: represente o tabuleiro em uma lista, como sugerido por $(*)$; para cada permutação de $\{1, 2, \ldots, n^2\}$, verifique se ela representa um caminho válido.

- **(14)** *Permutações aleatórias uniformes.*
  Escreva uma função que devolve uma permutação aleatória de $\{0, 1, \ldots, n-1\}$.
  Dica: utilize as funções `random.seed()` e `random.randrange()` do módulo `random`.

Aritanan Gruber

Assistant Professor

"See, if y'all haven't the same feeling for *this*, I really don't give a damn. If you ain't feeling it, then dammit *this* ain't for you!"
(desconheço a autoria; agradeço a indicação)