Soluções para a Prova 1
Processamento da Informação – 2020.1

Universidade Federal do ABC

Aritanan Gruber

aritanan.gruber@ufabc.edu.br

http://professor.ufabc.edu.br/∼aritanan.gruber

Duração de 4h contíguas, escolhidas entre as 19h de 26/06/2020 até as 23h de 26/06/2020
Submissão via Tidia em pdf único; se não conseguir acesso, envie o pdf para meu email acima

Questões

1. (1) *Sequência de Newman-Conway.* Considere a recorrência $g(\cdot)$, definida abaixo para inteiros $n \geq 1$ como:

$$g(n) = \begin{cases} 1 & \text{se } n \in \{1, 2\}, \\ g\big(g(n-1)\big) + g\big(n - g(n-1)\big) & \text{se } n \geq 3. \end{cases}$$

Escreva uma função que recebe n e devolve $g(n)$.

Solução:

Assertiva: $n \geq 1$ inteiro. A recorrência fornece de maneira direta um código puramente recursivo: fácil, mas lento.

```
def g (n: int) -> int:
  if n in (1,2): return 1
  return g (g (n-1)) + g (n - g(n-1))
```

Não é difícil, no entanto, construir uma versão iterativa. A diferença desta sequência de Conway para a de Fibonacci (feita em sala) é que a última requer somente memória para os dois termos anteriores imediatos (logo, pode ser feita com duas variáveis adicionais; veja código desenvolvido à época); já a primeira, requer o termo imediatamente anterior e outros dois termos mais antigos, que variam à medida que a série progride. Logo, precisamos armazenar os termos gerados em uma lista.

```
def g (n: int) -> int:
  C = [0,1,1] # C[0] não é usado, serve apenas para evitar subtrações em índices
  for i in range (3, n+1):
    C.append (C [C[i-1]] + C[i - C[i-1]])
  return C[-1]
```

Qualquer uma das formas acima, recursiva ou iterativa, é válida como resposta. □

2. (6 × 1) *Simetrias em matrizes.* Uma matriz quadrada $A \in \mathbb{R}^{n\times n}$ é **$\mathscr{C}$-simétrica** se suas entradas são simétricas com relação ao centro de A. Exemplos de matrizes $\mathscr{C}$-simétricas com $n \in \{1, 2, 3, 4, 5\}$ são

$$\begin{pmatrix} a \end{pmatrix}, \quad \begin{pmatrix} a & b \\ b & a \end{pmatrix}, \quad \begin{pmatrix} a & b & c \\ d & e & d \\ c & b & a \end{pmatrix}, \quad \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ h & g & f & e \\ d & c & b & a \end{pmatrix}, \quad \begin{pmatrix} a & b & c & d & e \\ f & g & h & i & j \\ k & l & m & l & k \\ j & i & h & g & f \\ e & d & c & b & a \end{pmatrix},$$

em que $\{a, b, \ldots, m\}$ são valores reais, não necessariamente distintos.

(a) Forneça um predicado que defina matrizes $\mathscr{C}$-simétricas, determinando $k = k(n, i, j)$ e $\ell = \ell(n, i, j)$ tais que

$$A[i][j] == A[k][\ell] \qquad \text{para} \qquad 0 \leq i, j < n$$

quando A é $\mathscr{C}$-simétrica.

Solução: Temos que $k = n - 1 - i$ e $\ell = n - 1 - j$. □

(b) Com base no predicado do item anterior, escreva uma função que recebe uma matriz quadrada $A \in \mathbb{R}^{n\times n}$ e decide se A é $\mathscr{C}$-simétrica.

Solução:

```
def is_C_symmetric (A: [[float]]) -> bool:
  n = len (A); m = (n+1) // 2
  for i in range (n):
    for j in range (m):
      if A[i][j] != A[n-1-i][n-1-j]:
        return False
  return True
```

Claramente, não é preciso varrer todas as entradas de A: se o teste funcionar para $0 \le j \le \lceil n/2 \rceil$, temos que A é $\mathscr{C}$-simétrica. (Por que?) □

Uma matriz quadrada $T \in \mathbb{R}^{n\times n}$ é **$\mathscr{D}$-constante** se as entradas em cada diagonal de sentido principal são iguais. Exemplos de matruzes $\mathscr{D}$-constantes com $n \in \{1, 2, 3, 4, 5\}$ são

$$\begin{pmatrix} a \end{pmatrix}, \quad \begin{pmatrix} a & b \\ c & a \end{pmatrix}, \quad \begin{pmatrix} a & b & c \\ d & a & b \\ e & d & a \end{pmatrix}, \quad \begin{pmatrix} a & b & c & d \\ e & a & b & c \\ f & e & a & b \\ g & f & e & a \end{pmatrix}, \quad \begin{pmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{pmatrix},$$

em que $\{a, b, \dots, i\}$ são valores reais, não necessariamente distintos.

(c) Escreva uma função que recebe uma matriz quadrada $T \in \mathbb{R}^{n\times n}$ e decide se T é $\mathscr{D}$-constante.

Solução:

```
def is_D_constant (T: [[float]]) -> bool:
  n = len (T)
  for k in range (-n+2, n-1):
    l, u = (0, n-k) if k >= 0 else (-k, n)
    x = T[l][l+k]
    for i in range (l+1, u):
      if T[i][i+k] != x:
        return False
  return True
```

Claramente, os cantos inferior esquerdo e superior direito não precisam ser verificados; as demais posições são acessadas de acordo com a diagonal em que residem. □

(d) Dada uma matriz $\mathscr{D}$-constante T, que propriedade adicional T deve possuir para que T também seja $\mathscr{C}$-simétrica? Dica: claramente, a resposta procurada não é "$\mathscr{C}$-simétrica."

Solução: Se T, que é $\mathscr{D}$-constante, também for **simétrica** (i.é, $T = T^\top$), então T será $\mathscr{C}$-simétrica. A prova desta afirmação fica como exercício. Observe que a recíproca não é verdadeira. □

Para dois vetores $x[0..n-1]$ e $y[0..n-1]$, defina a matriz $\mathscr{D}$-constante $T(y) \in \mathbb{R}^{n\times(2n-1)}$ como

$$T(y) = \begin{pmatrix} y_0 & y_1 & y_2 & y_3 & y_4 & \cdots & y_{n-1} & 0 & 0 & \cdots & 0 \\ 0 & y_0 & y_1 & y_2 & y_3 & \cdots & y_{n-2} & y_{n-1} & 0 & \cdots & 0 \\ 0 & 0 & y_0 & y_1 & y_2 & \cdots & y_{n-3} & y_{n-2} & y_{n-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & y_0 & \cdots & y_{n-3} & y_{n-2} & y_{n-1} \end{pmatrix}$$

e considere o produto $z = x^\top T(y)$. Claramente, z é um vetor com $2n-1$ coordenadas.

(e) Escreva uma função que recebe $x[0..n-1]$ e $y[0..n-1]$, determina $T(y)$ e devolve $z = x^\top T(y)$ como acima.

Solução:

```
def convolute (x: [float], y: [float]) -> [float]:
  n = len (x); assert (n == len (y))
  T = [[(y[j-i] if i <= j < n+i else 0) for j in range (2*n-1)] for i in range (n)]
  return [sum (x[i]*T[i][j] for i in range (n)) for j in range (2*n-1)]
```

(f) O produto $x^\top T(y)$ equivale a qual operação entre x e y? Justifique.

Solução: Como indicado pelo nome escolhido para a função do item anterior, z é um vetor de **convoluções** entre x e y. Especificamente, $z = x \circledast y$, em que

$$z[k] = \sum_{i+j=k} x[i] \cdot y[j] \qquad \text{para} \qquad 0 \le k < 2n-1,$$

com $0 \le i, j < n$. De fato, para $0 \le k < n$,

$$z[k] = \big(x^\top T(y)\big)[k] = (x_0, x_1, \ldots, x_{n-1}) \cdot (y_k, \ldots, y_0, 0, \ldots, 0)^\top = \sum_{i=0}^{k} x_i y_{k-i}.$$

E para $n \le k < 2n-1$,

$$z[k] = \big(x^\top T(y)\big)[k] = (x_0, x_1, \ldots, x_{n-1}) \cdot (0, \ldots, 0, y_{n-1}, \ldots, y_{k-n+1})^\top = \sum_{i=k-n+1}^{n-1} x_i y_{k-i}.$$

(Compare as somas acima com o Exercício 1 da Aula A05.) □

3. $(1+2)$ *Permutações camaradas.* Três permutações $p[1..n]$, $q[1..n]$ e $r[1..n]$ para o conjunto $S_n := \{1, 2, \ldots, n\}$, com $n \ge 1$ inteiro, são **camaradas** se

$$p \equiv q \circ r \circ q^{-1},$$

em que a operação $q \circ r := q(r)$ denota composição das permutações q e r, e q^{-1} é a permutação inversa à permutação q. Por exemplo, se $n = 9$, as permutações

$$p = [2, 1, 4, 5, 3, 6, 8, 7, 9], \qquad q = [3, 5, 1, 6, 2, 4, 8, 9, 7], \qquad \text{e} \qquad r = [6, 1, 5, 4, 3, 2, 7, 9, 8]$$

são camaradas.

(a) Escreva uma função que recebe as três permutações p, q e r e decide se elas são camaradas.

Solução:
Suponha que p, q e r sejam camaradas. Não é difícil perceber que a estrutura cíclica de p e r é a mesma. Vamos ilustrar isso no exemplo fornecido. Temos que $p = [2, 1, 4, 5, 3, 6, 8, 7, 9]$ pode ser escrita em forma cíclica como

$$p = (1, 2)(3, 4, 5)(6)(7, 8)(9),$$

pois $p[1] = 2$ e $p[2] = 1$; $p[3] = 4$, $p[4] = 5$ e $p[5] = 3$; $p[6] = 6$; $p[7] = 8$ e $p[8] = 7$; $p[9] = 9$. O mesmo pode ser feito com r:

$$r = (1, 6, 2)(3, 5)(4)(7)(8, 9).$$

Observe que ambas possuem dois 1-ciclos, dois 2-ciclos e um 3-ciclo, em que um n-ciclo é um ciclo com n elementos. Elas podem então ser pareadas de diversas formas (quantas?). Uma delas é:

$$p = (3, 4, 5)(1, 2)(7, 8)(6)(9),$$
$$r = (1, 6, 2)(3, 5)(8, 9)(4)(7).$$

Levando $3 \to 1$ e $1 \to 3$, $4 \to 6$ e $6 \to 4$, $5 \to 2$ e $2 \to 5$, $7 \to 8$, $8 \to 9$ e $9 \to 7$, obtemos

$$q = (1, 3)(4, 6)(2, 5)(7, 8, 9)$$

que resulta em $q = [3, 5, 1, 6, 2, 4, 8, 9, 7]$. Invertendo q, obtemos $q^{-1} = [3, 5, 1, 6, 2, 4, 9, 7, 8]$ ou

$$q^{-1} = (1, 3)(4, 6)(2, 5)(7, 9, 8).$$

pois a inversão de 1- ou 2-ciclos é trivial (são os próprios) e a de ciclos maiores é uma simples reversão. Terminar de escrever...

```
# recebe uma permutação s[1..n] e devolve a permutação s^{-1}[1..n] inversa a s.
def invert (s: [int]) -> [int]:
  t = [0] * (m := len (s)) # m = n+1
  for i in range (1, m):
    t[s[i]] = i
  return t

# terminar o código...
```

Duas permutações $p[1..n]$ e $r[1..n]$ para S_n são **camaráveis** se existe uma terceira permutação $q[1..n]$ para S_n tal que p, q e r são camaradas como acima, isto é, se

$$p \equiv q \circ r \circ q^{-1}.$$

(b) Escreva uma função que recebe as permutações p e r e decide se elas são camaráveis; em caso afirmativo, devolva uma permutação q que as tornam camaradas. Nota: q, quando existe, raramente é única.

Solução:

Por definição, duas permutações p e r para S_n são **conjugadas**[1] se existe uma permutação q para S_n tal que $p = q \circ r \circ q^{-1}$. Não é difícil perceber que uma mesma estrutura cíclica é uma condição necessária para p e r serem conjugadas. Examinando um pouco o problema, vemos que tal condição é também suficiente. Apresentamos o argumento a seguir. Terminar de escrever...

[1] Utilizamos o termo *camaráveis* no enunciado para evitar cópias da internet.