Soluções para a Lista 2
Processamento da Informação – 2020.1

Universidade Federal do ABC

Aritanan Gruber

aritanan.gruber@ufabc.edu.br

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

Entregue todos os exercícios via Tidia-4 até às 01/06/2020 às 19h00.

1. *Mistério.* Sem usar o computador, diga qual o conteúdo da lista A após a execução do código abaixo.

```
A = [98 - i for i in range (99)]
for i in range (99):
  A[i] = A[A[i]]
```

Solução:

Após o código, temos que:

$$A = [0, 1, 2, \ldots, 48, 49, 48, \ldots, 2, 1, 0].$$

2. *Soma positivos.* Escreva uma função que calcula a soma dos elementos positivos de uma lista $A[0..n-1]$ de inteiros. O problema faz sentido quando $n = 0$? Quanto deve valer a soma nesse caso?

Solução:

```
def sum_positives (A: [int]) -> int:
  s = 0
  for a in A:
    if a > 0: s += a
  return s
```

No caso em que $n = 0$, o valor devolvido deve ser 0, o elemento neutro da operação de adição. Outra versão, mais sucinta:

```
def sum_positives (A: [int]) -> int:
  return sum (a for a in A if a > 0)
```

3. *Conjuntos em listas.* Podemos representar conjuntos de inteiros através de listas. Para um conjunto de elementos $\{a_0, a_1, \ldots, a_{n-1}\}$, todos naturalmente distintos, a forma mais simples é via declaração de uma lista A tal que $A[i] = a_i$. Escreva funções que recebem listas (representando conjuntos) A e B com $n \geq 0$ e $m \geq 0$ elementos, respectivamente, e devolva:

   (a) $A \cup B$, a união de A com B;

   (b) $A \cap B$, a intersecção de A com B;

   (c) $A \setminus B$, a diferença de A com B;

   (d) $A \triangle B$, a diferença simétrica de A com B, em que

$$A \triangle B := (A \cup B) \setminus (A \cap B) = (A \setminus B) \cup (B \setminus A).$$

Solução:

Como $|A| + |B| = |A \cap B| + |A \cup B|$, vamos resolver os itens (a) e (b) em uma mesma função (é o que faz sentido). Todos os códigos consomem $O(mn)$ no pior caso (muito ruim na prática).

```python
# Recebe dois conjuntos A[0..m-1] e B[0..n-1].
# Devolve (C = A interseção B, D = A união B).
def inter_union (A: [int], B: [int]) -> ([int], [int]):
  C, D = [a for a in A], []
  for b in B:
    if b in A: D.append (b)
    else     : C.append (b)
  return (C, D)

# Recebe dois conjuntos A[0..m-1] e B[0..n-1].
# Devolve A diferença B.
def difference (A: [int], B: [int]) -> [int]:
  return [a for a in A if a not in B]

# Recebe dois conjuntos A[0..m-1] e B[0..n-1].
# Devolve A diferença simétrica B.
def symmetric_difference (A: [int], B: [int]) -> [int]:
  C, D = inter_union (A, B)
  return difference (D, C)
```

4. *Distância τ de Kendall.* Seja A um conjunto com $n \geq 0$ valores inteiros e sejam $x[0..n-1]$ e $y[0..n-1]$ duas permutações para A. Definimos a distância de Kendall entre x e y como

$$\tau(x,y) := \big|\big\{(x[i],x[j]) : i<j\big\} \setminus \big\{(y[i],y[j]) : i<j\big\}\big|.$$

Em outras palavras, $\tau(x,y)$ é igual ao número de pares de elementos em A que ocorrem em diferentes ordens relativas em x e y. Por exemplo, se $A = \{1,2,3\}$, $x = [3,2,1]$ e $y = [1,3,2]$, temos que

$$X := \big\{(x[i],x[j]) : i<j\big\} = \big\{(3,2),(3,1),(2,1)\big\}$$
$$Y := \big\{(y[i],y[j]) : i<j\big\} = \big\{(1,3),(1,2),(3,2)\big\}$$

e, portanto, que

$$\tau(x,y) = \big|X \setminus Y\big| = \big|\big\{(3,1),(2,1)\big\}\big| = 2.$$

(a) Escreva uma função (eficiente) que recebe x e y e calcula $\tau(x,y)$.

Solução:

```python
def kendall_distance (x, y):
  n, pX, pY = len (x), [], []

  for i in range (0, n-1):
    for j in range (i, n):
      pX.append ((x[i],x[j]))
      pY.append ((y[i],y[j]))
  pX.sort()
  pY.sort()

  i = j = tau = 0
  while i < len (pX) and j < len (pY):
    if   pX[i] < pY[j]: i, tau = i+1, tau+1
    elif pX[i] > pY[j]: j += 1
    else:
      i, j = i+1, j+1
  return tau + len (pX) - i
```

(b) Mostre que a distância de Kendall é simétrica, mostrando que $\tau(x, y) = \tau(y, x)$.

Solução:
Temos que um elemento $(a, b) \in X \cap Y$ se e somente se a e b tem a mesma ordem relativa em x e y, isto é, a ocorre antes de b tanto em x quanto em y. Como $|X| = |Y|$, claramente é o caso de que

$$\tau(x, y) = |X \setminus Y| = |X| - |X \cap Y| = |Y| - |X \cap Y| = |Y \setminus X| = \tau(y, x).$$

5. *Contando asteriscos.* Considere o seguinte algoritmo recursivo, cujo argumento n é um inteiro.

```
def Asterisco (n: int):
  if n > 0:
    Asterisco (n-1)
    for i in range (n):
      print ("*", end = "")
    Asterisco (n-1)
```

(a) Para um dado valor de $n \geq 0$, quantos asteriscos são impressos pela chamada `Asterisco (n)`? Tente especificar a sua resposta como uma fórmula fechada (em função de n).

Solução:
Seja $A(n)$ o número de asteriscos impressos por uma chamada a `Asterisco(n)`. Inspecionando o código acima, temos que

$$A(n) = \begin{cases} 0 & \text{se } n = 0, \\ 2A(n-1) + n & \text{em caso contrário.} \end{cases}$$

Supondo $n > 0$ e grande o suficiente, e iterando a recorrência acima, obtemos

$$\begin{aligned} A(n) &= 2A(n-1) + n \\ &= 2(2A(n-2) + (n-1)) + n = 2^2 A(n-2) + 2(n-1) + (n+0) \\ &= 2^2(2A(n-3) + (n-2)) + 2(n-1) + (n+0) = 2^3 A(n-3) + 2^2(n-2) + 2^1(n-1) + 2^0(n-0) \\ &\vdots \\ &= 2^k A(n-k) + \sum_{j=0}^{k-1} 2^j (n-j). \end{aligned}$$

Quando $n - k = 0$, $A(n-k) = A(0) = 0$. Logo, temos que

$$A(n) = \sum_{j=0}^{n-1} 2^j (n-j).$$

Abrindo o somatório, vem

$$A(n) = n \sum_{j=0}^{n-1} 2^j - \sum_{j=0}^{n-1} j 2^j.$$

O primeiro somatório do lado direito é uma progressão geométrica finita de razão 2, portanto igual a $2^n - 1$. Já o segundo somatório, utilizando a variável x no lugar da constante 2 (lembre-se, generalização!), é dado por

$$\sum_{j=0}^{n-1} j x^j = x \frac{d}{dx} \left(\sum_{j=0}^{n-1} x^j \right) = x \frac{d}{dx} \left(\frac{x^n - 1}{x - 1} \right) = \frac{(n-1)x^n - nx^{n-1} + 1}{(x+1)^2}$$

Substituindo-se $x = 2$ acima, obtemos

$$A(n) = n(2^n - 1) - (n-1)2^{n+1} + n2^n - 2 = 2^{n+1} - n - 2.$$

(b) Escreva uma função iterativa (utilizando laços) que realiza a mesma tarefa que `Asterisco (n)`.

Solução:

```
def Estrelas (n: int):
  print ("*" * (2**(n+1) - n - 2))
```

6. *Exponenciação de matrizes.*

(a) Escreva uma função que recebe duas matrizes $A, B \in \mathbb{R}^{n\times n}$ – isto é, duas matrizes reais com n linhas e n colunas – e devolva o produto $C = A * B$. Lembre-se que

$$C_{i,j} = \sum_{k=0}^{n-1} A_{i,k} * B_{k,j} \qquad \text{para todos } 0 \le i, j < n.$$

Solução:

```
def matrix_prod (A: [[float]], B: [[float]]) -> [[float]]:
  n = len (A)
  C = [[0 for j in range (n)] for i in range (n)]
  for i in range (n):
    for j in range (n):
      for k in range (n):
        C[i][j] += A[i][k] * B[k][j]
  return C
```

(b) Utilizando a função do item anterior, escreva uma outra função que recebe uma matriz $A \in \mathbb{R}^{n\times n}$ e um inteiro $m > 1$ e devolva uma aproximação da exponencial de A, denotada por e^A, e definida como

$$e^A = I + A + \frac{A^2}{2} + \frac{A^3}{3!} + \cdots + \frac{A^k}{k!} + \cdots$$

que inclua os m primeiros termos da série acima. Lembre-se que $A^0 = I$, em que I é a matriz identidade de dimensões $n \times n$.

Solução:

```
def matrix_sum (A: [[float]], B: [[float]]) -> [[float]]:
  return [[A[i][j]+B[i][j] for j in range (len (A))] for i in range (len (A))]

def matrix_scale (A: [[float]], x: float):
  return [[x*A[i][j] for j in range (len (A))] for i in range (len (A))]

def matrix_exp (A: [[float]], m: int) -> [[float]]:
  n, d = len (A), 1
  eA = Ai = [[(1 if i==j else 0) for j in range (n)] for i in range (n)]

  for i in range (1, m+1):
    d *= i
    Ai = matrix_prod (Ai, A)
    eA = matrix_sum (eA, matrix_scale (Ai, 1/d))
  return eA
```

7. *Matrizes estocásticas.* Uma matriz $M \in [0,1]^{m \times n}$ é *estocástica* se a soma dos elementos de cada linha de M é igual a 1. M é *duplamente estocástica* se M e $M^\top$ (a transposta de M) são estocásticas. Por exemplo, a matriz abaixo é estocástica, mas não é duplamente estocástica.

$$\begin{pmatrix} 0.386 & 0.147 & 0.202 & 0.062 & 0.140 & 0.047 & 0.016 \\ 0.107 & 0.267 & 0.227 & 0.120 & 0.207 & 0.052 & 0.020 \\ 0.035 & 0.101 & 0.188 & 0.191 & 0.357 & 0.067 & 0.061 \\ 0.021 & 0.039 & 0.112 & 0.212 & 0.431 & 0.124 & 0.061 \\ 0.009 & 0.024 & 0.075 & 0.123 & 0.473 & 0.171 & 0.125 \\ 0.000 & 0.103 & 0.041 & 0.088 & 0.391 & 0.312 & 0.155 \\ 0.000 & 0.008 & 0.036 & 0.083 & 0.364 & 0.235 & 0.274 \end{pmatrix}$$

(a) Escreva uma função que recebe uma matriz M e decide se M é estocástica.

(b) Escreva uma outra função que recebe uma matriz M e decide se M é duplamente estocástica.

Solução: Vamos fornecer uma só função, que devolve 2 se M é duplamente estocástica, 1 se é simplesmente estocástica, ou 0 em caso contrário.

```
def is_stochastic (M: [[float]]) -> int:
  m, n = len (M), len (M[0])

  for i in range (m):
    s = 0
    for j in range(n): s += M[i][j]
    if s != 1: return 0

  for j in range (n):
    s = 0
    for i in range(m): s += M[i][j]
    if s != 1: return 1

  return 2
```