# MC458 - Projeto e Análise de Algoritmos I
# Prova Individual - 03/09/2018

**Instruções:**

1. Comece pelas questões que você tem mais certeza de saber fazer.
2. Não perca tempo com detalhes menores. Foque no que é relevante para resolver a questão e passe logo para a próxima.
3. Se sobrar tempo, você pode acrescentar detalhes às questões já resolvidas.

**Questão 1** (valor 2,5) Analise o algoritmos a seguir e dê a ordem de crescimento do número de operações básicas para o pior caso e o melhor caso em função de $n$, o tamanho dos vetores:

**Algorithm 1** Subtract binary vector $B$ from binary vector $A$ yielding binary vector $C$ and *carry*. All vectors have the same size and are indexed from 1. Least significant bit is at index 1.

```
1:  procedure SUBTRACT(A, B, C, carry)
2:      carry ← 0
3:      n ← len(A)
4:      for i ← 1 to n do
5:          if A[i] < B[i] then
6:              C[i] ← 1
7:              j ← i
8:              while j ≤ n and A[i] = 0 do
9:                  A[i] ← 1
10:                 j ← j + 1
11:             if j ≤ n then
12:                 A[j] = 0
13:             else
14:                 carry ← 1
15:         else
16:             C[i] ← A[i] − B[i]
```

**Questão 2** (Valor 2,5) Ordene as funções a seguir como $f_1, f_2, f_3, f_4, f_5$ de forma que $f_i = O(f_{i+1})$ para $i = 1, 2, 3, 4$. Mostre **todas** as ordens que satisfaçam a esta condição.

$$(\sqrt{3})^{\lg n}, \quad n^{4/\lg n}, \quad n2^n, \quad 3^n, \quad n/10.$$

Nota: lg significa logaritmo na base 2.

**Questão 3** (Valor 2,5) Calcule o valor da somatória infinita abaixo:

$$\sum_{i=1}^{\infty} \frac{2i-1}{4^i} = \frac{1}{4} + \frac{3}{16} + \frac{5}{64} + \frac{7}{256} + \ldots$$

**Questão 4** (Valor 2,5) Resolva a recorrência abaixo:

$$\begin{aligned} T(n) &= T(3n/7) + T(4n/7) + 3n \\ T(1) &= 1 \end{aligned}$$

Boa sorte!

# Soluções nas próximas páginas

**Questão 1** (valor 2,5)

Há uma falha no enunciado da questão. Veja o que deveria ser em vermelho nas linhas 8 e 9:

**Algorithm 2** Subtract binary vector $B$ from binary vector $A$ yielding binary vector $C$ and *carry*. All vectors have the same size and are indexed from 1. Least significant bit is at index 1.

```
 1: procedure SUBTRACT(A, B, C, carry)
 2:     carry ← 0
 3:     n ← len(A)
 4:     for i ← 1 to n do
 5:         if A[i] < B[i] then
 6:             C[i] ← 1
 7:             j ← i
 8:             while j ≤ n and A[j] = 0 do
 9:                 A[j] ← 1
10:                 j ← j + 1
11:             if j ≤ n then
12:                 A[j] = 0
13:             else
14:                 carry ← 1
15:         else
16:             C[i] ← A[i] − B[i]
```

Felizmente, este erro não afeta a análise do algoritmo. Porém, o instrutor pode ter corrigido algumas provas errado por causa disso. Se sua nota nesta questão não é o que você esperava, marque com o instrutor para ver a correção e discutir o assunto.

Infelizmente, o algoritmo original não calcula corretamente a subtração $A - B$. Mas a análise abaixo se mantém.

Melhor caso: $O(n)$.

Pior caso: $O(n)$. Apesar de as linhas 9 e 10 estarem dentro de dois loops, a soma total das vezes em que são executadas ainda é $O(n)$. Na versão errada que foi divulgada na prova, o loop **while** poderia até ser trocado por um **if**, já que logo de cara faz-se $A[i] \leftarrow 1$. Na versão correta acima, o loop **while** só é executado quando $A[i] = 0$, mas, quando isto ocorre, os próximos $A[j]$ são jogados para 1 até que apareça o próximo $A[j] = 0$. Desta forma,

os comandos das linhas 9 e 10 só são executados no máximo uma vez por índice.

**Questão 2** (valor 2,5)

A ordem correta é:

$$n^{4/\lg n},\ \ (\sqrt{3})^{\lg n},\ \ n/10,\ \ n2^n,\ \ 3^n.$$

Aplicando lg a todas elas é possível ver porque:

$$4,\ \ 0.8\lg n,\ \ \lg n - 3.3,\ \ n + \lg n,\ \ 1.6n,$$

onde usamos as aproximações: $0.8 = \lg\sqrt{3}$, $3.3 = \lg 10$, e $1.6 = \lg 3$. Se você não tem calculadora, o importante é saber que:

$$2 < 3 < 4 \implies 1 < \lg 3 < 2 \implies 0.5 < (\lg 3)/2 < 1$$

$\lg 10$ é uma constante

**Questão 3** (valor 2,5)

O resultado é $5/9 = 0.5555....$

Os alunos resolveram de várias formas, algumas delas listadas a seguir:

- Pelo método da dica dada no teste T2:

$$\begin{aligned}
\sum_{i=1}^{\infty} \frac{2i-1}{4^i} &= \frac{1}{4} + \frac{3}{16} + \frac{5}{64} + \frac{7}{256} + \dots \\
&= \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \dots \\
&\quad + \frac{2}{16} + \frac{2}{64} + \frac{2}{256} + \dots \\
&\quad + \frac{2}{64} + \frac{2}{256} + \dots \\
&\quad + \frac{2}{256} + \dots
\end{aligned}$$

  Cada linha dá uma PG, que pode ser somada facilmente. A primeira linha dá 1/3. A segunda linha dá 1/6, já que é 2/4 da linha número 1. Além disso, os resultados de cada linha, a partir da segunda, formam também uma PG, já que cada linha é a anterior multiplicada por 1/4.

Então temos:

$$\sum_{i=1}^{\infty} \frac{2i-1}{4^i} = \frac{1}{3} + \sum_{i=0}^{\infty} \frac{1}{6} \left(\frac{1}{4}\right)^i = \frac{1}{3} + \frac{1}{6}\frac{4}{3} = \frac{1}{3} + \frac{2}{9} = \frac{5}{9}.$$

- Buscando uma função $f(s)$ tal que $\Delta f(s) = (2s-1)/4^s$. Uma resposta é tentar $f(s) = (As + B)/4^s$ para certas constantes $A$ e $B$. Para descobrir $A$ e $B$ calculamos $\Delta f(0)$ e $\Delta f(-1)$, que sabemos que devem ser iguais aos valores de $(2s-1)/4^s$ calculados em $s = 0$ e $s = -1$, respectivamente. Vejamos:

$$\Delta f(0) = f(1) - f(0) = \frac{A+B}{4} - B = -1,$$

$$\Delta f(-1) = f(0) - f(-1) = B - 4(-A+B) = -12.$$

Resolvendo, dá $A = -8/3$ e $B = 4/9$. Uma verificação direta comprova que $\Delta f(s) = (2s-1)/4^s$ para estes valores de $A$ e $B$. Então temos

$$\sum_{i=1}^{\infty} \frac{2i-1}{4^i} = f(\infty) - f(1) = -f(1),$$

pois $f(\infty) = 0$. Calculando $f(1)$ obtemos

$$f(1) = \frac{A+B}{4} = \frac{A}{4} + \frac{B}{4} = \frac{-2}{3} + \frac{1}{9} = \frac{-6+1}{9} = \frac{-5}{9}.$$

Logo, a soma dá $-f(1) = 5/9$.

**Questão 4** (valor 2,5)

O método mestre não pode ser usado, pois as hipóteses não estão satisfeitas.

Fazendo a árvore de recursão, descobrimos que cada nível completo gasta $3n$ passos.

Além disso, a árvore é desbalanceada. O ramos mais curto tem comprimento $\log_{7/4} n$ e o mais longo, $\log_{7/3} n$.

Daí já podemos derivar um limite inferior. Até o nível mais curto temos uma árvore completa, logo o gasto até aí dá $3n \log_{7/4} n$, mostrando que a solução satisfaz $T(n) = \Omega(n \lg n)$.

Para o limite superior, vamos tentar algo da mesma ordem, ou seja, $cn \lg n$. Pelo método da substituição, supondo que vale para valores menores, vamos ver se vale para $n$:

$$\begin{aligned} T(n) &= T(3n/7) + T(4n/7) + 3n \\ &\leq (3cn/7)\lg(3n/7) + (4cn/7)\lg(4n/7) + 3n \\ &= (3cn/7)\lg 3 + (3cn/7)\lg n - (3cn/7)\lg 7 + \\ &\quad + (4cn/7)\lg 4 + (4cn/7)\lg n - (4cn/7)\lg 7 + 3n \\ &= (cn/7)(3\lg 3 + 4\lg 4) + cn\lg n - cn\lg 7 + 3n \\ &= cn\lg n + (cn/7)(3\lg 3 + 4\lg 4 - 7\lg 7) + 3n. \end{aligned}$$

Neste ponto observamos que $7\lg 7$ é um número grande, ficando entre 14 e 21, já que $\lg 7$ fica entre 2 e 3. Os números $3\lg 3$ e $4\lg 4$ são menores que ele, mesmo somados, pois $3\lg 3$ é menor que 6 e $4\lg 4$ é igual e 8.

Sendo assim, a constante $a = 3\lg 3 + 4\lg 4 - 7\lg 7$ entre parênteses é negativa, e tomando $c = -21a$ concluimos que $T(n) \leq cn\lg n$.

Para o caso base não podemos usar $n = 1$, mas há várias formas de resolver isto. Uma delas é pegar casos base com $n$ maior, por exemplo, $n = 2$, possivelmente tendo que aumentar $c$. Outra possibilidade é usar $cn\ln n + dn$ na prova. O termo $dn$ se comporta muito bem nesta recorrência e não vai mudar quase nada na prova, para qualquer $d$. Daí $d = 1$ serve para satisfazer o caso base, e assintoticamente continuamos a ter $T(n) = O(n\lg n)$.

Logo, a solução satisfaz $T(n) = \Theta(n\lg n)$.

# Fim das soluções