Aula Prática 2

ASA 2024/2025

Somatórios

- $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
- $\sum_{k=1}^{n} (a_k - a_{k-1}) = a_n - a_0$
- $\sum_{k=1}^{n} (a_k - a_{k+1}) = a_1 - a_{n+1}$
- $\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$, se $|x| < 1$
- $\sum_{k=1}^{n} \frac{1}{k} > \int_{1}^{n+1} \frac{1}{x} dx = log\,(n+1)$
- $\sum_{k=0}^{\infty} k\, x^k = \frac{1}{(1-x)^2}$, se $|x| < 1$
- $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$
- $\sum_{k=0}^{n} x^k = \frac{1-x^{n+1}}{1-x}$

Teorema Mestre

Sejam $a \geq 1, b > 1$ constantes e $f(n)$ uma função.
Para $T(n)$ definido por $T(n) = aT(n/b) + f(n)$:

- Caso 1: $T(n) = \Theta(n^{log_b a})$, se $f(n) = O(n^{log_b a - \epsilon})$ para $\epsilon > 0$
- Caso 2: $T(n) = \Theta(n^{log_b a} log\, n)$, se $f(n) = \Theta(n^{log_b a})$
- Caso 3: $T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{log_b a + \epsilon})$ para $\epsilon > 0$ e se $af(n/b) \leq cf(n)$ para $c < 1$ e n suficientemente grande

Teorema Mestre (simplificado)

Sejam $a \geq 1, b > 1, d \geq 0$ constantes,
seja $T(n)$ definido por $T(n) = aT(n/b) + O(n^d)$.

$$T(n) = \begin{cases} O(n^{log_b a}) & if\ d < log_b\, a \\ O(n^d log\, n) & if\ d = log_b\, a \\ O(n^d) & if\ d > log_b\, a \end{cases}$$

Q1 (T1 19/20): Considere a função recursiva:

```
int f(int n) {
  int i = 0, j = 0;

  for (i = 0; i < n; i++) { // Loop 1
    while (j - i < 2) {
      j++;
    }
  }

  if (n > 0)
    i = 2*f(n/2) + f(n/2) + f(n/2);

  while ( j > 0) { // Loop 2
    j = j / 2;
  }

  return j;
}
```

- Determine um upper bound medido em função do parâmetro n para o número de iterações dos loops **1** e **2** da função f.
- Determine o menor majorante assimptótico da função f em termos do número n utilizando os métodos que conhece.

Solução:

1. Consideramos os dois loops separadamente.
   - *Loop 1*: $O(n)$
   - *Loop 2*: $O(log\ n)$
2. Equação do tempo:
$$T(n) = 3.T(n/2) + O(n)$$
   Observando que $log_b\ a = log_2\ 3$ e que $n \in O(n^{log_2\ 3-\epsilon})$, aplicamos o Teorema Mestre, concluindo que: $T(n) \in O(n^{log_2\ 3})$.

Q2 (R1 19/20): Considere a função recursiva:

```
int f(int n) {
   int x = 0;

   for (int i = 0; i < n; i++) { // Loop 1
      for (int j=0; j < i; j++) { // Loop 2
         x++;
      }
   }

   if ((n > 0) && ((n%2) == 1)) {
      x = x + f(n - 1);
   }
   else if ((n > 0) && ((n%2) == 0)) {
      x = 2*f(n/2);
   }

   return x;
}
```

1. Determine um upper bound medido em função do parâmetro n para o número de iterações dos loops **1** e **2** por cada chamada à função f.
2. Determine o menor majorante assimptótico da função f em termos do número n utilizando os métodos que conhece.

Solução:

1. Upper bounds:
   - **Loop 1:** $\sum_{i=0}^{n-1} 1 = n \in O(n)$
   - **Loop 2:** $\sum_{i=0}^{n-1} \left(\sum_{j=1}^{i} 1\right) = \sum_{i=0}^{n-1} i = \sum_{i=1}^{n} i = \frac{n*(n+1)}{2} \in O(n^2)$
2. Analisamos separadamente os casos n é par e n é ímpar.
   - n é par: $T(n) = T(n/2) + O(n^2)$
   - n é ímpar: $T(n) = T(n-1) + O(n^2) = T((n-1)/2) + O(n^2) + O((n-1)^2) \leq T(n/2) + O(n^2)$

   Aplicando o Teorema Mestre ($a = 1$, $b = 2$, $d = 2$) concluímos que: $T(n) \in O(n^2)$.

Q3 (EN 23/24): Considere a função recursiva:

```
int f(int n) {
  int k = 0;

  if(n == 0)
    return 1;

  for (int i = n; i > 0; i/=2) {
    for (int j = 0; j < i; j++) { // Loop 1
      k = k + 1;
    }
    for (int l = 1; l <= n; l*=2) { // Loop 2
      k = k + 1;
    }
  }

  for (int i = 1; i < 3; i++) {
    k += 7*f(n/4) + f(n/4);
  }

  return k;
}
```

1. Determine um upper bound medido em função do parâmetro n para o número total de iterações dos loops **1** e **2** da função f. Deve justificar a resposta.
2. Determine o menor majorante assimptótico da função f em termos do número n utilizando os métodos que conhece.

Solução:

1. Analisamos cada loop separadamente:
   - *Loop 1:* Contamos o número de iterações do *Loop 1* por cada iteração do loop exterior que o contém. Para tal, precisamos de determinar o valor da variável i em função da iteração do loop exterior:
     - Iteração 0: $i = n = n/2^0$
     - Iteração 1: $i = n/2 = n/2^1$
     - Iteração 2: $i = n/4 = n/2^2$
     - ...
     - Iteração k: $i = n/2^k$

     O número de iterações do *Loop 1*, na iteração k do loop exterior, é dado pelo valor de $i = n/2^k$, e o número de iterações do loop exterior em função de n é dado pelo valor de $k = \log_2 n$. Assim, o número total de iterações do *Loop 1*:

$$\sum_{i=0}^{\log_2 n} \frac{n}{2^i} = n. \sum_{i=0}^{\log_2 n} (\frac{1}{2})^i = n\frac{1-(1/2)^{\log_2 n+1}}{1-1/2} = n\frac{1-(\frac{1}{2^{log_2 n}.2})}{1/2} = 2n.(1-\frac{1}{2n}) = 2n-1 \in \Theta(n)$$

- *Loop 2:* Contamos o número de iterações do *Loop 2* por cada iteração do loop exterior que o contém. Vemos que o número de iterações do loop exterior é dado por $k = \log_2 n$. Assim, o número de iterações do *Loop 2* é dado por:

$$\sum_{i=0}^{\log_2 n} \sum_{l=0}^{\log_2 n} 1 = \sum_{i=0}^{\log_2 n} \log_2 n = \log_2 n . \log_2 n = (\log_2 n)^2 \in \Theta((\log_2 n)^2)$$

2. Observamos que:

$$T(n) = 4.T(n/4) + \Theta(n)$$

Podemos aplicar o Teorema Mestre na forma simplificada notando que:
$a = 4$, $b = 4$, $d = 1$ e $log_b a = 1 == d = 1$.
Concluímos, pelo segundo caso do Teorema Mestre que $T(n) = O(n.log\, n)$.

Q4 (T1 20/21): Considere a função recursiva:

```
int f(int n) {
  int sum = 0;

  for (int j = n; j>0; j/=2) {
    for (int k=0; k<j; k+=1) { // Loop 1
      sum += 1;
    }
  }

  for (int i=1; i<n; i*=2) {
    for (int k=n; k>0; k/=2) { // Loop 2
      sum += 1;
    }
  }

  return sum+4*f(n/2);
}
```

1. Determine o menor majorante assimptótico medido em função do parâmetro n para o número total de iterações dos loops **1** e **2** por cada chamada à função f.

2. Determine o menor majorante assimptótico da função f, em função do parâmetro n, utilizando os métodos que conhece.

Solução:

1. Analisamos cada loop separadamente:

   - *Loop 1:* Contamos o número de iterações do *loop 1* por cada iteração do loop exterior que o contém. Para tal, precisamos de determinar o valor da variável j em função da iteração do loop exterior.
     - Iteração 0: $j = n = n/2^0$. *Iterações Loop 1:* $n/2^0$.
     - Iteração 1: $j = n/2 = n/2^1$. *Iterações Loop 1:* $n/2^1$.
     - Iteração 2: $j = n/4 = n/2^2$. *Iterações Loop 1:* $n/2^2$.

– ...
– Iteração k: $j = n/2^k$. *Iterações Loop 1:* $n/2^k$.

Dado que o loop exterior é executado $log\ n$ vezes, concluímos que o número total de iterações é do loop 1 é dado por:

$$\begin{aligned}\sum_{k=0}^{log\ n} \frac{n}{2^k} &= n.\sum_{k=0}^{log\ n} \frac{1}{2^k}\\ &= n.\frac{\frac{1}{2}^{(log\ n)+1}-1}{\frac{1}{2}-1}\\ &= 2n.\left(1-\frac{1}{2}^{(log\ n)+1}\right)\\ &= n.(2-\frac{1}{n}) = 2n-1 \in O(n)\end{aligned}$$

- *Loop 2:* O *loop 2* é executado $log\ n$ vezes por cada iteração do loop exterior que o contém. Por sua vez, o loop exterior é também executado $log\ n$ vezes. Pelo que concluímos que o *loop 2* é executado $O((log\ n)^2)$ vezes.

2. Observamos que:

$$T(n) = T(n/2) + O(n)$$

Podemos aplicar o Teorema Mestre na forma simplificada notando que $a = 1$, $b = 2$, $d = 1$ e $log_b\ a = 0$. Concluímos que $T(n) = O(n)$.

Q5 (R1 20/21): Considere a função recursiva:

```
int f(int n) {

  int i = 0, j=0, z=0;
  while (j + z < n) {  // Loop 1
    z += 1;
    j += i;
    i += 2;
  }

  int r = 0;
  if (n > 0) r = 3*f(n/2);

  j = 1; z = 0;
  while (j<n) {  // Loop 2
    j *= 2;
    z += 1;
  }

  return r+i+z;
}
```

1. Determine o menor majorante assimptótico medido em função do parâmetro n para o número total de iterações dos loops **1** e **2** por cada chamada à função f.

2. Determine o menor majorante assimptótico da função f, em função do parâmetro n, utilizando os métodos que conhece.

Solução:

1. Consideramos os dois loops separadamente.

   - *Loop 1*: Definimos $z(k)$, $i(k)$ e $j(k)$ como sendo os valores das variáveis z, i e j no final da k-ésima iteração do loop 1:
     - $z(k) = k$
     - $i(k) = 2 * k$
     - $j(k) = j(k-1) + i(k-1) = j(k-1) + 2 * (k-1)$
       $j(k) = 2 * ((k-1) + (k-2) + ... + 1) = 2 * \frac{k*(k-1)}{2} = k * (k-1)$

     Resolvemos a condição de paragem em função de k:

     $$j(k) + z(k) = n \Leftrightarrow k * (k-1) + k = n \Leftrightarrow k = \sqrt{n}$$

     Concluímos que o majorante assimptótico para o Loop 1 é $O(\sqrt{n})$.

   - *Loop 2*: Definimos $j(k)$ e $z(k)$ como sendo os valores das variáveis j e z no final da k-ésima iteração do loop 2:
     - $j(k) = 2^k$
     - $z(k) = k$

     Resolvemos a condição de paragem em função de k:

     $$j(k) = n \Leftrightarrow 2^k = n \Leftrightarrow k = log\ n$$

     Concluímos que o majorante assimptótico para o Loop 2 é $O(log\ n)$.

2. Observando que $log\ n \in O(\sqrt{n})$, obtemos a equação do tempo:

   $$T(n) = T(n/2) + O(\sqrt{n})$$

   Aplicando o Teorema Mestre, concluímos que $T(n) = O(\sqrt{n})$. Parâmetros: $a = 1$, $b = 2$ e $d = 1/2$. Notamos que $log_b\ a = 0 < 1/2$.

Q6 (EE1 20/21): Considere a função recursiva:

```
int f(int n) {
  int i = 0, j = n;

  if (n <= 1) return 1;

  while(j > 1) { // Loop 1
    i++;
    j = j / 2;
  }

  for (int k = 0; k < 8; k++)
    j += f(n/2);

  while (i > 0) { // Loop 2
    j = j + 2;
    i--;
  }
  return j;
}
```

1. Determine o menor majorante assimptótico medido em função do parâmetro n para o número total de iterações dos loops **1** e **2** por cada chamada à função f.
2. Determine o menor majorante assimptótico da função f, em função do parâmetro n, utilizando os métodos que conhece.

Solução:

1. Consideramos os dois loops separadamente.
   - *Loop 1*: $O(log\ n)$
   - *Loop 2*: $O(log\ n)$
2. Equação do tempo:
   $$T(n) = 8.T(n/2) + O(log\ n)$$
   Observando que $log_b\ a = log_2\ 8 = 3$ e que $log\ n \in O(n^{3-\epsilon})$, aplicamos o Teorema Mestre na forma geral, concluindo que: $T(n) \in O(n^3)$.