

Aula Prática 2

ASA 2023/2024

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 $|x| < 1$
- $\sum_{k=1}^n \frac{1}{k} > \int_1^{n+1} \frac{1}{x} dx = \log(n+1)$
- $\sum_{k=0}^{\infty} k x^k = \frac{x}{(1-x)^2}$, 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 $T(n) = aT(n/b) + O(n^d)$.

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

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 assintótico da função f em termos do número n utilizando os métodos que conhece.

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 assintótico da função f em termos do número n utilizando os métodos que conhece.

Q3 (EE 19/20): Considere a seguinte implementação naif de uma fila de prioridade mínima baseada em listas simplesmente ligadas. Uma fila de prioridade é guardada em memória como uma lista de nós, cada qual associado a uma prioridade `pri` e a um identificador `id`. A implementação é composta pelas funções:

- `Insert(Lst lst, Lst node)` que insere o nó `node` na lista `lst`;
- `Remove(Lst lst, int i)` que remove o nó com identificador `i` da lista `lst`;
- `ExtractQueue(Queue q)` que remove o nó com prioridade mínima da fila de prioridade `q`; e
- `DecreaseKey(Queue q, Lst node, int pri)` que diminui a prioridade do nó `node` para `pri` na fila de prioridade `q`.

```
typedef struct Node {
    int id;
    int pri;
    struct Node* next;
} *Lst;

typedef struct QueueNode {
    Lst hd;
} *Queue;

int ExtractQueue(Queue q) {
    if (q->hd == NULL) return -1;

    Lst hd = q->hd;
    q->hd = hd->next;
    return hd->id;
}

Lst Remove(Lst lst, int id) {
    if (lst == NULL) return lst;

    if (lst->id == id) return lst->next;

    Lst prev = lst;
    Lst cur = lst->next;
    while (cur != NULL) {
        if (cur->id == id) {
            prev->next = cur->next;
            break;
        }
        prev = cur;
        cur = cur->next;
    }
    return lst;
}

Lst Insert (Lst lst, Lst node) {
```

```

    if (lst == NULL) return node;
    if (node->pri <= lst->pri) {
        node->next = lst;
        return node;
    } else {
        Lst ret = Insert(lst->next, node);
        lst->next = ret;
        return lst;
    }
}

void DecreaseKey (Queue q, Lst node, int pri) {
    Lst lst = Remove(q->hd, node->id);
    node->pri = pri;
    lst = Insert(lst, node);
    q->hd = lst;
}

```

Determine o menor majorante assintótico para funções **Insert**, **Remove**, **ExtractQueue** e **DecreaseKey** em função do número de elementos, n , contidos na fila de prioridade ou lista que respectivamente recebem como argumento. Deve indicar para cada uma das funções a equação do tempo que expressa o número instruções executadas em função do tamanho do input (i.e. $T(n) = \dots$).

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 assintótico 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 assintótico da função f , em função do parâmetro n , utilizando os métodos que conhece.

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 assintótico 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 assintótico da função f , em função do parâmetro n , utilizando os métodos que conhece.

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) {  
        i++;  
        j = j / 2;  
    }  
  
    for (int k = 0; k < 8; k++)  
        j += f(n/2);  
  
    while (i > 0) {  
        j = j + 2;  
        i--;  
    }  
    return j;  
}
```

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