

## 1) Somatórios

### a) Progressão Aritmética (PA)

$$S_n = \sum_{0 \leq i \leq n} [a + b \cdot i] = \frac{(2a + bn)(n+1)}{2}$$

$$S_n = \sum_{0 \leq i \leq n} [0 + 1 \cdot i] = \frac{(2 \cdot 0 + 1 \cdot n) \cdot (n+1)}{2} = \frac{n \cdot (n+1)}{2}$$

### b) Propriedades

P1: Combinando Conjuntos

$$\sum_{i \in I} a_i + \sum_{i \in I'} a_i = \sum_{i \in I \cup I'} a_i + \sum_{i \in I \cap I'} a_i$$

P2: Base para a Pertubação

$$S_n + a_{n+1} = a_0 + \sum_{0 \leq i \leq n} a_{i+1}$$

### c) Progressão Geométrica → adquirida a partir da base para perturbação

$$S_n = \sum_{0 \leq i \leq n} a \cdot x^i = \frac{a - a \cdot x^{n+1}}{1 - x}, \text{ para } x \neq 1$$

### d) Prova por Indução

(Passo Base) → provar que a fórmula é verdadeira para o primeiro valor

(Indução) → provar que a fórmula é verdadeira para o primeiro valor

$$S_n = S_{n-1} + a_n$$

$S_{n-1}$  é a equação substituindo n por (n-1)  $S_n = \frac{2n^3 + 3n^2 + n}{6}$   
 $a_n$  = n-ésimo termo da sequência  $S_n = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$

## 2) Análise de Algoritmos

- 1)  $f(n) = O(f(n))$
- 2)  $c \times O(f(n)) = O(f(n))$
- 3)  $O(f(n)) + O(f(n)) = O(f(n))$
- 4)  $O(O(f(n))) = O(f(n))$
- 5)  $O(f(n)) + O(g(n)) = O(\text{máximo}(f(n), g(n)))$
- 6)  $O(f(n)) \times O(g(n)) = O(f(n) \times g(n))$
- 7)  $f(n) \times O(g(n)) = O(f(n) \times g(n))$

\*) As mesmas propriedades são aplicadas para  $\Omega$  e  $\Theta$

## 3) Estrutura de Dados Lineares

### a) Fila Flexível

```
void inserir(int x) {
    if (((ultimo + 1) % MAXTAM) == primeiro)
        exit(1);

    array[ultimo] = x;
    ultimo = (ultimo + 1) % MAXTAM;
}
```

```
int remover() {
    if (primeiro == ultimo)
        exit(1);

    int resp = array[primeiro];
    primeiro = (primeiro + 1) % MAXTAM;
    return resp;
}
```

```
void mostrar() {
    int i = primeiro;
    printf("T");
    while (i != ultimo) {
        printf(array[i] + " ");
        i = (i + 1) % MAXTAM;
    }
    printf("J");
}
```

```
int removerInicio() {
    if (n == 0)
        exit(1);

    int resp = array[0];
    n--;

    for (int i = 0; i < n; i++) {
        array[i] = array[i+1];
    }

    return resp;
}
```

```
int removerFim() {
    if (n == 0)
        exit(1);

    return array[--n];
}
```

```
int remover(int pos) {
    if (n == 0 || pos < 0 || pos >= n)
        exit(1);

    int resp = array[pos];
    n--;

    for (int i = pos; i < n; i++) {
        array[i] = array[i+1];
    }

    return resp;
}
```

```
void inserirInicio(int x) {
    if (n >= MAXTAM)
        exit(1);

    //levar elementos para o fim do array
    for (int i = n; i > 0; i--) {
        array[i] = array[i-1];
    }

    array[0] = x;
    n++;
}
```

```
void inserirFim(int x) {
    if (n >= MAXTAM)
        exit(1);

    array[n] = x;
    n++;
}
```

```
void inserir(int x, int pos) {
    if (n >= MAXTAM || pos < 0 || pos > n)
        exit(1);

    //levar elementos para o fim do array
    for (int i = n; i > pos; i--) {
        array[i] = array[i-1];
    }

    array[pos] = x;
    n++;
}
```