

ESTUDO SOMATÓRIOS

$\sum_{i=1}^{i \leq n}$ valor de início
limite superior
somando

$$\begin{array}{l} \text{lim. superior} \\ \sum_{\text{lim. inferior}}^{\text{somando}} \end{array} \quad \begin{array}{l} \text{os dois} \\ \text{limites} \end{array} \quad \sum_{0 \leq i \leq n-2}^{(n-i-1)}$$

$\sum_{i=1}^{i \leq n} a_i$ O a_i representa um termo
qualquer na posição i . O a
é a variável.

EXERCÍCIO

a) $\sum_{n=1}^5 n^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$

O somatório de uma constante é ela multiplicada
pelo limite superior - limite inferior + 1.
constante. ($\text{lim. superior} - \text{lim. inferior} + 1$)

b) $\sum_{1}^5 3i = 3 \cdot 1 + 3 \cdot 2 + 3 \cdot 3 + 3 \cdot 4 + 3 \cdot 5 = 3 \cdot \sum_{1}^5 i = 3 \cdot (1+2+3+4+5)$

c) $\sum_{1}^5 (3 - 2i) = (3 - 2 \cdot 1) + (3 - 2 \cdot 2) + (3 - 2 \cdot 3) + (3 - 2 \cdot 4) + (3 - 2 \cdot 5)$

Somatório de 1 até 5 de $(3 - 2i)$

d) $\sum_{1}^5 (ai + x) = (a \cdot 1 + x) + (a \cdot 2 + x) + (a \cdot 3 + x) + (a \cdot 4 + x) + (a \cdot 5 + x)$

então: $\sum_{1}^5 ai + \sum_{i=1}^{i=5} x = (a \cdot 1 + a \cdot 2 \dots) + (5x)$

então: $a \cdot \sum_{1}^5 i + 5x$

A expressão quando i
vale esse valor que está
embaixo
 $\rightarrow a_0, a_1 \text{ e } a_5$

e) $\sum_{0}^5 i \cdot (i-1) \cdot (5-i) \rightarrow$ Quando $i=0, i=1 \text{ e } i=5$ o somando será zero

$$2 \cdot (2-1) \cdot (5-2) + 3 \cdot (3-1) \cdot (5-3) + 4 \cdot (4-1) \cdot (5-4)$$

f) $\sum_{m=1}^5 (8j - 2m) = \sum_{m=1}^5 8j - \sum_{1}^5 2m = 5 \cdot 8j - 2 \cdot \sum_{1}^5 m = 40j - 30$

constante

$$8j \cdot \sum_{1}^5 1$$

MANIPULAÇÃO DE SOMATÓRIOS

As três regras básicas de transformação: distributividade, associatividade e comutatividade.

DISTRIBUTIVIDADE

$$\sum_i c \cdot a_i = c \cdot \sum_i a_i \quad c \cdot a_0 + c \cdot a_1 + c \cdot a_2 = c \cdot (a_0 + a_1 + a_2)$$

Permite mover constantes para dentro ou fora de um somatório.

Também se aplica à subtração:

$$\sum_i \frac{a_i}{c} = \frac{1}{c} \cdot \sum_i a_i$$

ASSOCIATIVIDADE

$$\sum_i (a_i + b_i) = \sum_i a_i + \sum_i b_i \quad \text{Permite quebrar um somatório em partes ou unir duas somatórias.}$$

$$(a_0 + b_0) + (a_1 + b_1) + (a_2 + b_2) = (a_0 + a_1 + a_2) + (b_0 + b_1 + b_2)$$

Também se aplica à subtração.

COMUTATIVIDADE

$$\sum_i a_i = \sum_{p(i)} a_{p(i)} \quad a_0 + a_1 + a_2 = a_2 + a_0 + a_1$$

Permite colocar os termos em qualquer ordem.

EXERCÍCIOS

1. Aplique a associatividade para unir os dois somatórios abaixo:

$$S_n = \sum_3^n a_i + \sum_1^n b_i$$

$$S_n = \left(\sum_3^n a_i + b_i \right) + b_1 + b_2 \quad \text{OU} \quad S_n = \left(\sum_1^n a_i + b_i \right) + a_1 + a_2$$

É como se fosse uma corrida e o b conseguiu a frente.

2. Usando a comutatividade, prove que os somatórios abaixo são iguais

$$\sum_{0 \leq i \leq 4} (3 + 2 \cdot i) = \sum_{0 \leq i \leq 4} (3 + 2 \cdot (4 - i)) = \sum_0^4 3 + \sum_0^4 2(4 - i)$$

Primeiro somatório: $(3 + 2 \cdot 0) + (3 + 2 \cdot 1) + (3 + 2 \cdot 2) + (3 + 2 \cdot 3) + (3 + 2 \cdot 4)$

No segundo: $15 + [2 \cdot (\underbrace{4 - 0})] + [2 \cdot (\underbrace{4 - 1})] + [2 \cdot (\underbrace{4 - 2})] + [2 \cdot (\underbrace{4 - 3})] + [2 \cdot (\underbrace{4 - 4})]$

Logo, por comutatividade, temos apenas a alteração da ordem dos elementos.

Nota: $(n - i)$ "vai mudar" um decremento no valor de i .

É como se fosse $\sum_0^n i$ ou $\sum_{0 \leq (n-i) \leq n}$ ou simplesmente $\sum_{0 \leq i \leq n}$

3. Aplique a regra de transformação para obter a fórmula fechada da soma S_n dos elementos de uma progressão aritmética.

$$\rightarrow S_n = \sum_{i=0}^n a + b \cdot i$$

RECORDANDO PROGRESSÃO ARITMÉTICA

- Uma PA é uma sequência cuja razão (diferença) entre dois termos consecutivos é constante. Por exemplo: $5, 7, 9, 11, 13 \dots$

$$\underbrace{7-5}_7 = a \quad \underbrace{13-11}_{13} = a$$

- Cada termo da PA será $a_i = a + b \cdot i$, onde a é o termo inicial, b é a razão e i é a ordem do termo.
- Na sequência acima, a e b serão 5 e 2, respectivamente. Logo, temos: $(5 + 2 \cdot 0), (5 + 2 \cdot 1), (5 + 2 \cdot 2), (5 + 2 \cdot 3) \dots$

Aplicando a comutatividade, podemos somar de maior para o menor, trocando i por $n - i$.

$$S_n = \sum_{0 \leq (n-i) \leq n} [a + b \cdot (n - i)] = \sum_{0 \leq i \leq n} [a + b \cdot (n - i)] = \sum_{i=0}^n [a + bn - bi]$$

Por conta dessa equivalência, podemos afirmar que: slide 30

$$2S_n = \sum_{0 \leq i \leq n} [a + b \cdot i] + \sum_{0 \leq i \leq n} [a + b \cdot n - b \cdot i]$$

Aplicando a comutatividade, podemos combinar os dois somatórios.

$$2S_n = \sum_{0 \leq i \leq n} [a + b.i + a + b.n - b.i] = \sum_{i=0}^n [2.a + b.n]$$

Usando a distributividade, temos:

$$2S_n = (2a + bn) \cdot \sum_{0 \leq i \leq n} 1 \quad \rightarrow \text{Como } [2a + bn] \text{ é uma constante nesse caso, visto que não depende de } i, \text{ ele pode "sair" do somatório.}$$

$n+1$

$$2S_n = (2a + bn) \cdot (n+1)$$

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

4. Sabendo a fórmula da soma de uma progressão aritmética qualquer, mostre a fórmula fechada para o somatório de Gauss.

$$\sum_{0 \leq i \leq n} i = 0 + 1 + 2 + 3 + 4 + \dots + n$$

nesses casos $a = 0$ (termo inicial) e $b = 1$ (diferença).

$$S_n = \frac{(2.a + b.n) \cdot (n+1)}{2} \quad S_n = \frac{(2.0 + 1.n)(n+1)}{2} = \frac{n(n+1)}{2}$$

5. Dada a fórmula fechada do somatório dos n primeiros números inteiros, mostre um algoritmo mais eficiente que o apresentado abaixo.

```
int somatorio (int n) {
    int soma = 0;
    for (int i = 0; i <= n; i++) {
        soma += i;
    }
    return soma;
}
```

Resposta:

```
int somatorio (int n) {
    return ((n * (n+1)) / 2);
}
```

6. O Algoritmo de Seleção realiza $\sum_{i=0}^{n-1} (n-i-1)$ comparações entre regiões. Mostre a fórmula fechada para esse somatório.

Aplicando a assosiatividade temos:

$$\sum_{i=0}^{n-1} (n-i-1) = \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i - \sum_{i=0}^{n-1} 1 \quad \{ 1.(n-1)$$

$n(n-1)$ $(n-1) \cdot (n-1+1)$

2

Quando temos uma constante no somatório, a fórmula é a seguinte:

$$\text{constante} \cdot (\limite{\text{superior}} + 1)$$

$$S = n(n-1) - \frac{(n-2)(n-1)}{2} - (n-1)$$

$$S = \frac{2n(n-1) - (n-2)(n-1) - 2(n-1)}{2}$$

$$S = \frac{2n^2 - 2n - [n^2 - 2n - n + 2] - 2n + 2}{2}$$

$$S = \frac{2n^2 - 2n - n^2 + 2n + n - 2 - 2n + 2}{2} \rightarrow S = \frac{n^2 - n}{2} \rightarrow \Theta(n^2)$$

7. Justifique ora a expressão

a) $\sum_{i=1}^n i = \sum_{i=0}^n i$ Verdadeira, pois $0 + 1 + 2 + \dots + n = 1 + 2 + \dots + n$. Somar um zero no primeiro termo das segundas não altera o resultado.

b) $\sum_{i=1}^n a_i \neq \sum_{i=0}^n a_i$ O dia somatório não é igual pois no primeiro caso o somatório apresenta o termo $n+1$ vez, enquanto no segundo a_i é somado $n+1$ vez.

Supondo que $n=3$

$a_1 + a_2 + a_3$ $a_0 + a_1 + a_2 + a_3$ Como não aderem o valor do termo, não podemos assumir que o termo inicial a_0 em $\sum_{0 \leq i \leq n} a_i$ será zero.

c) $\sum_{i=1}^n a_i = \sum_{i=0}^{n-1} a_{i+1}$ São iguais para a quantidade de termos somada em ambos é a mesma. No dia caso $a_1 + a_2 + a_3$ $a_0 + a_1 + a_2 + a_3$ a_i será somado de a_1 até a_n . Para quando $i=n-1$ No momento $n-1$, $i+1=n$!

PROPRIEDADE P1: COMBINANDO CONJUNTOS

Combina conjuntos de índices diferentes. No caso, se X e Y são dois conjuntos quaisquer de inteiros, então:

$$\sum_{i \in X} a_i + \sum_{j \in Y} a_j = \sum_{i \in X \cup Y} a_i + \sum_{i \in X \cap Y} a_i$$

A união garante todos os elementos e a interseção, os repetidos.

Exemplo: se $X = \{1, 2, 3\}$ e $Y = \{3, 5, 7\}$, então $A \cup B = \{1, 2, 3, 5, 7\}$ e $A \cap B = \{3\}$.

9. Sendo $1 \leq m \leq n$, aplique a propriedade P1 para unir as duas somatórias (quase disjuntas) abaixo.

$$\sum_{i=1}^m a_i + \sum_{i=m}^n a_i$$

$$a_1 + a_2 + \dots + a_m$$

$$a_m + a_{m+1} + a_{m+2} + \dots + a_n$$

$$\sum_{i=1}^n a_i + a_m$$

→ Tem que ser contados 2x
intervais: o que têm em comum
união

10. Sendo $1 \leq m \leq n$, aplique a propriedade P1 para unir as duas somatórias abaixo.

$$\sum_{i=1}^{m-3} a_i + \sum_{i=m}^n a_i$$

$$a_1 + a_2 + \dots + a_{m-3}$$

$$a_m + a_{m+1} + \dots + a_n$$

$$\sum_{i=1}^n a_i - a_{m-2} - a_{m-1}$$

PROPRIEDADE PA: BASE PARA A PERTURBAÇÃO

Dada uma soma genérica qualquer $S_n = \sum_{0 \leq i \leq n} a_i$, temos que:

$$S_{n+1} = a_0 + a_1 + a_2 + \dots + a_n + a_{n+1}$$

Poderia reescrever S_{n+1} de outras formas. A primeira é assim:

$$S_{n+1} = S_n + a_{n+1}$$

Vamos desenvolver a segunda:

$$S_{n+1} = \sum_{i=0}^{n+1} a_i \Rightarrow a_0 + \sum_{i=1}^{n+1} a_i \Rightarrow a_0 + \sum_{i=0}^n a_{i+1}$$

Quando chegarmos no termo n nesse caso, teremos a a_{n+1} , pois o i sempre cresce 1 unidade na frente da a

Portanto, na prática, para perturbar, resolvemos a igualdade abaixo:

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

Isso, frequentemente, resulta na equação fechada para S_n .

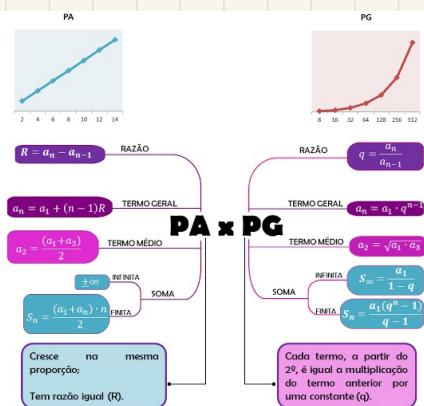
RECORDANDO PROGRESSÃO GEOMÉTRICA

Primeiro termo Posição do termo

$$a_n = ar^{n-1}$$

Termo geral Razão

Fórmulas	PA	PG
Termo Geral:	$a_n = a_1 + (n-1)r$	$a_n = a_1 \cdot q^{(n-1)}$
Soma dos termos:	$S_n = \frac{(a_1 + a_n) \cdot n}{2}$	$S_n = \frac{a_1(q^n - 1)}{q - 1}$ $S_\infty = \frac{a_1}{1-q}$ Condição: $-1 < q < 1$
Equivalência:	$a_n + a_m = a_p + a_t$ Onde: $n+m=p+t$	$a_n \cdot a_m = a_p \cdot a_t$ Onde: $n+m=p+t$



11. Aplique P2 para obter a fórmula fechada da soma S_n dos elementos de uma PG.

$$S_n = \sum_{i=0}^n a \cdot x^i \quad \text{Para... } a_i = a \cdot x^i$$

$$S_n + a_{n+1} = a \cdot x^0 + \sum_{i=0}^n a \cdot x^{i+1}$$

$$x^{i+1} = x^i \cdot x$$

$$\sum_{i=0}^n (a \cdot x^i \cdot x) = x \cdot \sum_{i=0}^n a \cdot x^i = x \cdot S_n$$

$$S_n + a \cdot x^{n+1} = a \cdot x^0 + x \cdot S_n \rightarrow S_n + a \cdot x^{n+1} = a + x \cdot S_n$$

$$S_n - x \cdot S_n = a - a \cdot x^{n+1}$$

$$S_n(1-x) = a - a \cdot x^{n+1} \rightarrow S_n = \frac{a - a \cdot x^{n+1}}{1-x} \text{ para } x \neq 1$$

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

Quando $x = 1$, temos:

$$\sum_{i=0}^n a \cdot 1^i = \sum_{i=0}^n a = (n+1) \cdot a$$

12. Encontre a fórmula fechada do somatório abaixo

$$S_n = \sum_{i=0}^n i \cdot 2^i$$

$$S_n + i_{n+1} = i_0 + \sum_{i=0}^n i_{i+1}$$

$$S_{n+1} = \sum_{i=0}^{n+1} i \cdot 2^i$$

$$S_n + (n+1)2^{n+1} = 0 \cdot 2^0 + \sum_{i=0}^n (i+1) \cdot 2^{i+1} \quad \checkmark$$

- Retirar o último elemento DE DENTRO DO SOMATÓRIO:

$$S_{n+1} = \sum_{i=0}^n i \cdot 2^i + (n+1)2^{n+1} = S_n + (n+1) \cdot 2^{n+1}$$

- Retirar o primeiro elemento:

$$S_{n+1} = \sum_{i=1}^{n+1} i \cdot 2^i + \underbrace{0 \cdot 2^0}_{0} = \sum_{i=1}^{n+1} i \cdot 2^i \rightarrow \sum_{i=0}^n (i+1) \cdot 2^{i+1}$$

nosso objetivo é sumir com \sum , substituindo - & por algo que já temos / conhecemos, e depois calcular S_n .

$$\sum_{i=0}^n (i+1) \cdot 2^{i+1} = \sum_{i=0}^n (i+1) \cdot 2^i \cdot 2 = 2 \cdot \sum_{i=0}^n (i+1) \cdot 2^i$$

$$2 \cdot \sum_0^n (i \cdot 2^i) + 2^i = 2 \cdot \left(\underbrace{\sum_0^n i \cdot 2^i}_{S_n} + \underbrace{\sum_0^n 2^i} \right)$$

$$= 2 \cdot \left(S_n + \left(\frac{1 - 1 \cdot 2^{n+1}}{1 - 2} \right) \right)$$

$$2 \cdot \left(S_n + \left(\frac{1 - 2^{n+1}}{-1} \right) \right) = S_n + (n+1) \cdot 2^{n+1}$$

Dividir por -1 é simplesmente inverter os sinais

$$2 \cdot (S_n - 1 + 2^{n+2}) = S_n + (n+1) \cdot 2^{n+2}$$

$$2S_n - 2 + 2^{n+2} = S_n + (n+1) \cdot 2^{n+2}$$

$$S_n - 2 + 2^{n+2} = (n+1) \cdot 2^{n+2}$$

$$S_n = (n+1) \cdot 2^{n+2} + 2 - 2^{n+2}$$

$$S_n = n \cdot 2^{n+2} + 2^{n+2} + 2 - 2 \cdot 2^{n+2}$$

$$S_n = 2^{n+2} (n+1 - 2) + 2$$

$$S_n = 2^{n+2} \cdot (n-1) + 2 \quad \text{EBAAA!}$$

PROVA POR INDUÇÃO

Se, em um parav de máquina (ou inspeção ou dedução), determina a resposta, basta prová-la por indução matemática.

1º parav (parav base): provar que a fórmula é verdadeira para o primeiro valor, substituindo n na equação pelo primeiro valor.

2º parav (indução propriamente dita): supõe-se que $n > 0$ e que a fórmula é válida quando substituímos n por $n-1$.

$$S_n = \underbrace{S_{n-1}}_{\downarrow} + A_n \quad \Rightarrow \text{n-ésimo termo da sequência}$$

É a equação substituindo n por $n-1$

1. Prove por indução que a fórmula abaixo para a soma dos quadrados perfeitos é verdadeira.

$$S_n = \sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6} \text{ para } n \geq 0$$

primeiros valer

1º passo $\rightarrow 0^2 = \frac{0(0+1)(2 \cdot 0+1)}{6} = 0$ verdadeiro!

2º passo $\rightarrow S_n = \frac{(n-1) \cdot [(n-1)+1] \cdot [2(n-1)+1]}{6} + n^2$

$$S_n = \frac{(n-1) \cdot (n) \cdot (2n-1)}{6} + n^2$$

$$6S_n = (n-1) \cdot n \cdot (2n-1) + 6n^2 \rightarrow (n^2-n)(2n-1) + 6n$$

$$6S_n = [2n^3 - n^2 - 2n^2 + n] + 6n^2 \rightarrow S_n = \frac{2n^3 + 3n^2 + n}{6} = \frac{n(n+1)(2n+1)}{6}$$

2. Encontre a fórmula fechada da somatória abaixo e, em seguida, prove-a usando indução matemática.

$$\sum_{i=0}^n (3+i)$$

ATENÇÃO! Nesse caso nem precisava usar perturbação!

$$= \sum_{i=0}^n 3 + \sum_{i=0}^n i \quad \text{Somatório de Gauss} \quad = 3(n+1) + \frac{n \cdot (n+1)}{2} = \frac{6(n+1) + n(n+1)}{2}$$

$$= \frac{6n+6+n^2+n}{2} = \frac{n^2+7n+6}{2} //$$

• 1º passo $\rightarrow 3+0 = \frac{0^2+7 \cdot 0+6}{2} \rightarrow 3 = \frac{6}{2} \checkmark$ verdade!

• 2º passo $\rightarrow S_n = S_{n-1} + a_n$

$$\underbrace{(n-1)^2 + 7(n-1) + 6}_{\substack{\text{produto} \\ \text{notável}}} + (3+n) = \frac{n^2 - 2n + 1 + 7n - 7 + 6 + 3 + n}{2} = S_n$$

$$\frac{n^2 + 5n + 6 + 2n}{2} = S_n \rightarrow 2S_n = n^2 + 7n + 6 \rightarrow S_n = \frac{n^2 + 7n + 6}{2} \checkmark$$

verdade!

3. Encontre a fórmula fechada da somatória abaixo e, em seguida, prove-a usando indução matemática.

$$\sum_{i=1}^n [(2i+1)^2 - (2i)^2]$$

$$\sum_{i=1}^n (2i+1)^2 - \sum_{i=1}^n (2i)^2 = S_n$$

produto notável

$$\sum_{i=1}^n (2i)^2 + 2 \cdot 2i \cdot 1 + 1^2 = \sum_{i=1}^n (2i)^2 + 2i + 1$$

$$S_n = \sum_{i=1}^n (2i)^2 + \sum_{i=1}^n 2i + \sum_{i=1}^n 1 - \sum_{i=1}^n (2i)^2$$

$$S_n = 4 \cdot \sum_{i=1}^n i + n$$

$1 \cdot (n-1+1) = n$

Somatório de Gauss

$$S_n = 4 \cdot \left[\frac{n \cdot (n+1)}{2} \right] + n \rightarrow S_n = 2 \cdot (n^2 + n) + n = 2n^2 + 3n,,$$

Precisa ter muita calma na hora de pensar na álgebra. A lógica da coisa eu já pequi, mas estou errando (CONTAS por falta de atenção!).

na realidade, nem precisava direcionar, era só ter reavaliação o P.N. dentro do somatório original.

1º passo: $[(2 \cdot 1 + 1)^2 - (2 \cdot 1)^2] = 5$
 $2 \cdot 1^2 + 3 \cdot 1 = 5,,$ verdadeiro!

2º passo: $S_n = S_{n-1} + a_n$

$$S_n = 2(n-1)^2 + 3(n-1) + (2n+1)^2 - (2n)^2$$

$$S_n = 2(n^2 - 2n + 1) + 3n - 3 + (2n)^2 + 4n + 1 - 4n^2$$

~~$$S_n = 2n^2 - 4n + 2 + 3n - 3 + 4n^2 + 4n + 1 - 4n^2$$~~

$$S_n = 4n^2 + 3n,,$$
 verdadeiro!

4. Encontre a fórmula fechada da somatória abaixo e, em seguida, prove-a usando indução matemática.

$$\sum_{i=1}^n [(5i+1)^2 - (5i-1)^2]$$

$$(25i^2 + 10i + 1) - (25i^2 - 10i + 1)$$

$$25i^2 + 10i + 1 - 25i^2 + 10i - 1$$

$$\sum_{i=1}^n 20i = 20 \cdot \sum_{i=1}^n i \rightarrow S_n = 20 \left[\frac{n(n+1)}{2} \right]$$

$$= 10(n^2 + n)$$

✓

$$\begin{aligned} \text{1º passo} &\rightarrow (5 \cdot 1 + 1)^2 - (5 \cdot 1 - 1)^2 = 36 - 16 = 20 \\ &\downarrow \\ &10(1^2 + 1) = 10 \cdot 2 = 20 \text{ verdadeiro!} \end{aligned}$$

$$2^{\circ} \text{ paravale} \rightarrow S_{n-1} + a_n = S_n$$

$$S_n = 10(n-1)^2 + 10(n-1) + (5n+1)^2 - (5n-1)^2$$

$$2.5n - 1 = +10n$$

$$5n = 10(n^2 - 2n + 1) + 10n - 10 + 25n^2 + 10n + 1 - 25n^2 + 10n - 1 \quad \text{verdadeiro!}$$

$$S_n = 10n^2 - 20n + 10 + 10n - 10 + 2S_{n-2} + 10n + 1 - 2S_{n-2} + 10n - 1 \rightarrow S_n = 10n^2 + 10n, //$$

5. Prove a fórmula apresentada anteriormente usando indução matemática.

$$\sum_{0 \leq i \leq n} i \cdot a^i = (n-1) \cdot a^{n+1} + a$$

$$\text{1º passo: } 0 \cdot \lambda^0 = 0$$

$$(0-1) \cdot \lambda^{0+1} + \lambda = -1 \cdot \lambda + \lambda = 0 \quad \text{OK}$$

2° paravas :

$$S_n = [(n-1)-1] \cdot 2^{(n-1)+1} + 2 + n \cdot 2^n$$

$$S_n = (n-2) \cdot 2^n + 2 + n \cdot 2^n$$

$$S_n = 2^n(n-2) + 2^n(n) + 2 \rightarrow S_n = 2^n(n-2+n) + 2 \rightarrow S_n = 2^n(2n-2) + 2$$

$$S_n = 2 \cdot 2^n (n-1) + 2 \rightarrow S_n = 2^{n+1} (n-1) + 2 \quad \underline{OK}$$

6. Aplique perturbación para encontrar la fórmula de somatorio abajo.

$$S_n = \sum_{i=0}^n i^2$$

$$\sum a_{j,i+1}$$

$$S_{n+1} = \sum_{i=0}^n i^2 + (n+1)^2$$

$\underbrace{\hspace{1cm}}_{S_n}$ $\underbrace{\hspace{1cm}}_{a_{n+1}}$

$$n+1 = \underbrace{0^2}_{\text{a sum}} + \sum_{0}^n (i+1)^2$$

$$5n + (n+1)^{\alpha} = \sum_{0}^n (\omega+1)^{\alpha}$$

$$\sum_{i=0}^n a_i x^i + a_0 \cdot \sum_{i=0}^n i x^i + \sum_{i=0}^n 1$$

$$S_n + (n+1)^2 = S_n + n(n+1) + (n+1)$$

I'm conclusive! Qa Sae amulam.

7. Se perturbarmos o somatório dos quadrados encontramos em certo ponto $\sum_{i=0}^n i$, então perturbando o somatório das unhas poderemos achar $\sum_{i=0}^n i^2$.

$$S_{\text{subo}, n} = \sum_{i=0}^n i^3$$

$$S_{\text{subo}, n+1} = \underbrace{\sum_{i=0}^n i^3}_{S_{\text{subo}}} + (n+1)^3 \quad S_{\text{subo}, n+1} = (0) + \sum_{i=0}^n (i+1)^3$$

$$(i+1)^3 = i^3 + 3i^2 \cdot 1 + 3i \cdot 1^2 + 1^3$$

$$\sum_0^n i^3 + 3i^2 + 3i + 1 = \underbrace{\sum_0^n i^3}_{S_{\text{subo}}} + 3 \cdot \underbrace{\sum_0^n i^2}_{\text{Squad.}} + 3 \cdot \underbrace{\sum_0^n i}_{\frac{3 \cdot n(n+1)}{2}} + \underbrace{\sum_0^n 1}_{n+1}$$

$$S_{\text{subo}} + (n+1)^3 = S_{\text{subo}} + 3 \cdot \text{Squad} + 3 \frac{n(n+1)}{2} + n+1$$

$$(n+1)^3 = 3 \cdot \text{Squad} + 3 \frac{n^2 + n}{2} + n + 1$$

$$(n+1)^3 = \frac{6 \cdot \text{Squad} + 3(n^2 + n) + 2n + 2}{2}$$

$$2(n+1)^3 = 6 \cdot \text{Squad} + 3(n^2 + n) + 2n + 2$$

$$-6 \cdot \text{Squad} = 3(n^2 + n) + 2n + 2 - 2(n+1)^3$$

$$-6 \cdot \text{Squad} = 3n^2 + 3n + 2n + 2 - 2(n^3 + 3n^2 + 3n + 1)$$

$$-6 \cdot \text{Squad} = \cancel{3n^2} + \cancel{3n} + \cancel{2n} + \cancel{2} - \cancel{2} \cdot (n^3 + 3n^2 + 3n + 1)$$

$$-6 \cdot \text{Squad} = -2n^3 - 3n^2 - n$$

$$\text{Squad} = \frac{2n^3 + 3n^2 + n}{6}$$

MAIS EXERCÍCIOS

1. Faça um método int somatorioPA (double a, double b, int n) que retorna o somatório das n primeiras termos de uma PA com termo inicial a e razão b.

```
int somatorioPA (double a, double b, int n){
```

```
    return (((a*a)+(b*n))*(n+1))/2;
```

```
}
```

BUSCA SEQUENCIAL

- Procura um elemento em um array que pode estar ordenado ou desordenado.
- Simplesmente varre o array total até "achar" o alvo.

```
public static int pesquisa_sequencial (int []arr, int target) {  
    for (int i = 0; i < arr.length(); i++) {  
        if (arr[i] == target) return i;  
    }  
}
```

Custo de complexidade

MELHOR CASO: o elemento buscado está na primeira posição do array. $\Omega(1)$.

PIOR CASO: o elemento alvo está na última posição do vetor ou não está no vetor. $O(n)$. Complexidade linear de tempo.

BUSCA BINÁRIA

- Algoritmo para encontrar um elemento em um array ordenado.
 - 1 - Encontra o elemento na posição do meio do vetor.
 - 2 - Compara esse elemento na posição central com o elemento alvo.
 - 3 - Se for MAIOR que o elemento alvo, passamos a analisar apenas as inícios do vetor até a posição do meio - 1.
 - 4 - Se for MENOR que o alvo, agora na próxima rodada moveremos o início do array para meio + 1.
 - 5 - Se o elemento encontrado na posição central for igual ao alvo, achamos o que queríamos e podemos retornar essa posição.

```
public static int binary_rec (int []arr, int target, int start, int end) {  
    if (start > end) return -1; // o elemento não existe no array. Evita OOB  
    int middle = (start+end)/2;  
    if (arr[middle] == target) return middle; // achamos!  
    if (arr[middle] > target) return binary_rec (arr, target, start, middle-1);  
    if (arr[middle] < target) return binary_rec (arr, target, middle+1, end);  
}
```

Custo de comparações

MELHOR CASO: o elemento procurado está exatamente no meio do vetor.
 $\Omega(1)$.

PIOR CASO: o elemento alvo está na primeira ou na última posição do vetor, ou não existe no arranjo analisado. $O(\log n)$

- É muito importante que as condições de saída valem escritas ANTES das chamadas recursivas na função, especialmente a condição que evita o erro de `out of bounds`.

```
public static int binary (int [] arr, int target) {
    int start = 0; int end = arr.length - 1;
    while (start <= end) {
        int mid = (start + end) / 2;
        if (arr [mid] == target) return mid;
        if (arr [mid] > target) end = mid - 1;
        if (arr [mid] < target) start = mid + 1;
    }
    return -1;
}
```



O logaritmo na base 2 de um número indica a quantidade de vezes que esse número deve ser dividido por 2 até chegar a 1.

ALGORITMO DE SELEÇÃO INSTÁVEL

- consiste em selecionar o menor elemento da porção não-ordenada do arranjo e trocá-lo de posição (`swap`) com o primeiro elemento dessa parte não-ordenada. NÃO varre "arredondando" os elementos a cada comparação, apenas fazemos o `swap` para resumir a porção dos menores elementos não-ordenados.

```
public static int[] selection (int [] arr) {
    for (int i = 0; i < arr.length - 1; i++) {
        int lowest = i; // O que importa é a POSIÇÃO. lembrando que a unidade que o menor elemento está na posição i (primeira posição da parte não-ordenada do vetor)
        for (int j = i + 1; j < arr.length; j++) {
            if (arr [j] < array [lowest]) lowest = j;
        }
        array = swap (array, i, lowest);
    }
    return array;
}
```

- Em poucas palavras: escolhe o menor elemento entre i e n e coloca na posição i. Repete para i+1 até n.

Custo de comparações

MELHOR CASO e PIOR CASO são iguais. Em ambos, temos $\Theta(n^2)$. Vamos desenvolver o somatório:

$$\sum_{i=0}^{n-a} n - i - 1$$

O i chega até a sair igual a $n-a$!

O for externo roda de $i=0$ até $n-1$. A cada vez que ele é realizado, o for interno roda $n-(i+1)$ vezes.

$$= \sum_{i=0}^{n-a} n - \sum_{i=0}^{n-a} i - \sum_{i=0}^{n-a} 1$$

\downarrow

$1.((n-a)-0+1)$
 $= 1.n = n-1$

$$S_n = (n^2 - n) - \left(\frac{n^2 - 3n + a}{a} \right) - (n-1)$$

$$S_n = \frac{a^2 n^2 - 2an - n^2 + 3n - a - 2n + a}{a}$$

Somatório de Gauss

$$(n-a).((n-a)+1) = \frac{(n-a)(n-1)}{2}$$

$$= \frac{n^2 - n - 2an + a}{2} = \frac{n^2 - 3n + a}{2}$$

$$C(n) = \frac{n^2 - n}{2}$$

$\Theta(n^2)$

$$n.((n-a)-0+1) = n.(n-1) = n^2 - n$$

Custo de movimentações

MELHOR CASO e PIOR CASO são iguais. Em ambos, temos $\Theta(n)$. Antes de desenvolver o somatório, vamos entender como as movimentações funcionam:

O primeiro ponto é que int lowest = i e lowest = j NÃO considera a movimentação, apenas a troca é simples. As movimentações devem envolver um elemento DENTRO do vetor (`arr[pos]`) e não apenas variação de inteiros usados para traçar as posições onde estamos analisando.

Isto é, assim, todas as movimentações do algoritmo de seleção ocorrem dentro do vetor `swap(array, i, lowest)`. Função essa que é chamada sempre imediatamente a cada vez que o for externo roda.

```
public static int[] swap(int[] array, int i, int pos) {
    int temp = array[i]
    array[i] = array[pos];
    array[pos] = temp;
    return array;
}
```

No caso, temos TRÊS movimentações

$$\sum_{i=0}^{n-2} 3 = 3 \cdot ((n-2) - 0 + 1) = 3(n-1) \quad M(n) = 3n - 3 \quad \Theta(n)$$

- Obs.: Um algoritmo é dito estável se depois da execução, os elementos com a mesma chave (igual) mantiverem a ordem relativa original entre as chaves repetidas. Isto é, a estabilidade permite ordenar os acrônimos com a chave para manter a sequência com a qual os elementos idênticos aparecem no vetor de entrada.

ALGORITMO DE INSERÇÃO ESTÁVEL

- Esse algoritmo constrói o vetor ordenado final ao selecionar um elemento por vez, compara-lo com seu anterior imediatamente sucessivamente (da direita para a esquerda) e ir "arrastando" os componentes do array para frente através de cópia até que a posição do elemento chave seja estabelecida.
- Encontramos a posição correta do elemento-chave quando chegamos ao fim do vetor ou quando ele é maior que o elemento na posição "de trás".

```
public static int[] insertion (int[] arr){
    int n = arr.length; int tmp = 0; int j = 0;

    for (int i = 1; i < n; i++) {
        tmp = arr[i] // Guardamos o elemento da vez numa
        j = i - 1;     variável temporária

        while (j >= 0 && arr[j] > tmp) { // As condições tem que estar
            arr[j + 1] = arr[j]           Nessa ordem para evitar que a com-
            j = j - 1;                   paração arr[-1] > tmp aconteça e dê
        }                                erro de out of bounds
        arr[j + 1] = tmp;               // Copionando os elementos para 1
    }                                posição a frente da atual.

    // Se j <= tmp, então a
    // posição correta do elemento arr[i]
    // é logo na frente de j.
}
```

- É muito eficiente para vetores quase ordenados.

Custo de comparações

MELHOR CASO: Quando o arranjo já está ordenado. Nesse cenário, temos $n-1$ comparações. Cada elemento i da rotina só é comparado 1 vez com o seu anterior para garantir que está no lugar. $\Omega(n)$.

PIOR CASO: Quando o vetor analisado está ordenado de trás para frente (ordem decrescente). Assim, cada elemento i precisa obrigatoriamente ser adicionado no final da parte já ordenada do vetor, sem possibilidade de sair do while antes.

$$\sum_{i=1}^{n-1} i = \frac{(n-1)((n-1)+1)}{2} \quad C(n) = \frac{n^2 - n}{2}$$

O for externo roda de $i=0$ até $i=n-1$. A cada iteração do loop interno (while) é feita uma comparação. $O(n^2)$.

Custo de movimentações

MELHOR CASO: Nunca entra no while (loop interno) pois o vetor já está ordenado. No loop externo (for) existem 2 movimentações por rodaada. Portanto, temos uma complexidade linear. $\Omega(n)$.

$$\sum_{i=1}^{n-1} 2 = 2 \cdot ((n-1)-1+1) = 2(n-1)$$

PIOR CASO: Quando o arranjo está na ordem inversa à desejada. Além das 2 movimentações a cada rodaada do loop externo, temos 1 movimentação a cada iteração do while, que roda $i-1$ vezes. $O(n^2)$.

$$\sum_{i=1}^{n-1} 2 + (i-1) = \underbrace{\sum_{j=1}^{n-1} 2}_{2(n-1)} + \underbrace{\sum_{j=1}^{n-1} i}_{\frac{n^2-n}{2}} - \underbrace{\sum_{i=1}^{n-1} 1}_{n-1} \quad M(n) = \frac{n^2 + n - 2}{2}$$

A menina que
não sabia
álgebra...

$$S_n = 2n - 2 + \left(\frac{n^2 - n}{2} \right) - n + 1 \rightarrow S_n = \frac{4n - 4 + n^2 - n - 2n - 2}{2}$$

- Uma outra maneira de calcular as movimentações é observar que o loop interno sempre realiza 1 movimentação A MENOS de que o número de comparações que o while faz de verificações. $M(n) = C(n) + 1$ no pior caso.
- O Algoritmo de Insersão é uma boa opção se desejarmos adicionar alguma itera em um array ordenado porque seu custo será linear.

BUBBLE SORT

ESTÁVEL

- O Bubble Sort começa no inicio do array e compara cada PAR de elementos adjacentes. Se eles estiverem na ordem errada (o primeiro é maior que o segundo), o algoritmo os troca de lugar. Apos cada passagem completa pelo array, o maior elemento "bolinha" até sua posição correta no final do vetor.
- Este processo é repetido para cada elemento do array, mas a cada passagem, a verificação é feita ate o elemento a menos no final, pois o maior elemento já está na posição correta.
- É estável (preserva a ordem original de registros iguais).
- É ineficiente para listas quase ordenadas devido à quantidade de comparações e trocas necessárias, mesmo quando o array já está quase ordenado.

```
public static int[] bubble (int []arr){  
    int n = arr.length;  
  
    for (int i=0; i<n-1; i++){  
  
        for (int j=0; j<n-i-1; j++){  
  
            if (arr [j] > arr [j+1]) swap (arr, j, j+1);  
        }  
    }  
}
```

Custo de comparações

MELHOR CASO e PIOR CASO serão iguais. Em ambos, temos $\Theta(n^2)$. Vamos desenvolver o somatório:

$$\sum_{i=0}^{n-2} n-i-2 = \underbrace{\sum_{i=0}^{n-2} n}_{n \cdot ((n-2)-0+1)} - \underbrace{\sum_{i=0}^{n-2} i}_{\frac{(n-2) \cdot (n-2+1)}{2}} - \sum_{i=0}^{n-2} 2 \left\{ 2 \cdot ((n-2)-0+1) \right\}$$

$$S_n = n^2 - n - \left(\frac{n^2 - 3n + 2}{2} \right) - (2n - 2) \rightarrow S_n = \frac{2n^2 - 2n - n^2 + 3n - 2 - 2n + 2}{2}$$

$$C(n) = \frac{n^2 - n}{2} \quad \Theta(n^2)$$

Caso de movimento retilíneo

MELHOR CASO: vetor já está ordenado. Nesse caso, nunca realizamos nenhuma movimentação.

Pior caso: vetor está em ordem decrescente. Desenvolvendo o somatório:

$$\sum_{i=0}^{n-2} 3 \cdot (n-i-2)$$

desenvolve-se: no final temos 3 movimentos.

$$= 3 \cdot \sum_{i=0}^{n-2} n - i - 2$$

} já resolvida.
} é o nº de componentes.

$$M(n) = \frac{3 \cdot (n^2 - n)}{2} \quad O(n^2)$$

ACABOU A TEORIA. BORA PRATICAR!

SLIDES AULÃO

- ① Usando a propriedade da soma, $S_n + a_{n+1} = a_0 + \sum_{0 \leq i \leq n} a_{i+1}$, encontre a fórmula fechada do somatório abaixo e, em seguida, prove por indução matemática. Nesta questão, você deve lembrar que $\sum_0^n a x^i = \frac{a - a x^{n+1}}{1 - x}$.

$$S_n = \sum_0^n (4^{\omega} \cdot 5^{\omega})$$

$$S_{n+1} = \underbrace{\sum_0^n (4^{\omega} \cdot 5^{\omega})}_{S_n} + \underbrace{(4^{n+1} \cdot 5^{(n+1)})}_{a_{n+1}}$$

$$S_n + (4^{n+1} \cdot 5^{(n+1)}) = 45n + 20 \left(\frac{1 - 4^{n+1}}{-3} \right)$$

$$S_{n+1} = \underbrace{(4^0 \cdot 5^0)}_{a_0} + \underbrace{\sum_0^n (4^{\omega+1} \cdot 5^{(\omega+1)})}_{\sum_0^n (4^{\omega+1} \cdot 5^{\omega+1})}$$

$$= \sum_0^n (4^{\omega+1} \cdot 5^{\omega+1})$$

$$= 4 \cdot \sum_0^n (4^{\omega} \cdot 5^{\omega} + 4^{\omega} \cdot 5)$$

$$= 4 \cdot (\sum_0^n 4^{\omega} \cdot 5^{\omega} + \sum_0^n 4^{\omega} \cdot 5)$$

- ② Encontre a fórmula fechada do somatório abaixo e, em seguida, prove-a usando indução matemática.

$$S_n = \sum_1^n [(4\omega+5)^{\omega} - (4\omega-5)^{\omega}] = 80 \cdot \sum_1^n \omega$$

$$(4\omega+5)^{\omega} = (4\omega)^{\omega} + 2 \cdot 4\omega \cdot 5 + 5^{\omega} = 16\omega^{\omega} + 40\omega + 25$$

$$(4\omega-5)^{\omega} = 16\omega^{\omega} - 40\omega + 25$$

$$16\omega^{\omega} + 40\omega + 25 - 16\omega^{\omega} + 40\omega - 25 = 80\omega$$

$$= 4 \cdot (S_n + 5 \cdot \sum_0^n 4^{\omega})$$

P.G. onde $a=1$ e $x=4$

$$= 4(S_n + 5 \cdot \frac{(1 - 4^{n+1})}{1 - 4})$$

$$= 4S_n + 20 \frac{(1 - 4^{n+1})}{-3}$$

$$S_n = 80 \cdot \left(\frac{n(n+1)}{2} \right) \rightarrow S_n = 40n(n+1) = 40n^2 + 40n //$$

Procedendo por indução:

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

$$1^{\text{º}} \text{ passo: } 40 \cdot (1^2) + 40 \cdot (1) = 40 + 40 = 80$$

2º passo:

$$S_n = 40(n-1)^2 + 40(n-1) + \underbrace{(4n+5)^2 - (4n-5)^2}_{80n}$$

$$S_n = 40(n^2 - 2n + 1) + 40n - 40 + 80n$$

$$S_n = 40n^2 - 80n + 40 + 40n - 40 + 80n$$

$$S_n = 40n^2 + 40n //$$

- ③ Encontre a fórmula fechada do somatório abaixo e, em seguida, prove-a usando indução matemática.

$$S_n = \sum_{i=1}^n [(2i+3)^2 - (2i-3)^2] = 24 \sum_{i=1}^n i$$

$$(2i+3)^2 = 4i^2 + 12i + 9 \quad 4i^2 + 12i + 9 - 4i^2 + 12i - 9 = 24i$$

$$(2i-3)^2 = 4i^2 - 12i + 9$$

$$S_n = 24 \cdot \frac{n(n+1)}{2} = 12n(n+1) = 12n^2 + 12n$$

Prova por indução:

$$1^{\text{º}} \text{ passo: } 12(1)^2 + 12(1) = 24 \text{ OK!}$$

$$2^{\text{º}} \text{ passo: } S_n = 12(n-1)^2 + 12(n-1) + 24n$$

$$S_n = 12(n^2 - 2n + 1) + 12n - 12 + 24n$$

$$S_n = 12n^2 - 24n + 12 + 12n - 12 + 24n = 12n^2 + 12n // \text{OK!}$$

- ④ Escreva, utilizando a notação \sum , o somatório de números de multiplicação em função de n . Depois use Pd e encontre a fórmula fechada, provando-a por indução matemática.

```
public void calcula (int n){
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            a *= 2;
            b *= 2;
            c *= 2;
        }
        b *= 3;
        c *= 3;
    }
}
```

$$\sum_{i=1}^n 2 + 3i = \sum_{i=1}^n 2 + 3 \cdot \sum_{i=1}^n i \quad \begin{matrix} \text{N} \\ \text{preciso de} \\ \text{perturbar para} \\ \text{achar a fórmula} \end{matrix}$$

$$S_n = 2(n-1+1) + 3 \cdot \frac{n \cdot (n+1)}{2}$$

$$S_n = 2n + \frac{3n^2 + 3n}{2}$$

$$S_n = \frac{4n + 3n^2 + 3n}{2} = \frac{3n^2 + 7n}{2} //$$

⑤ Apresente a função de complexidade para o número de multiplicações para o pior e o melhor caso usando a notação Θ . Descreva quando acontece cada um desses casos.

a)

```
int multiply (int a, int b) {
    if (a == 0 || b == 0) {
        return 0;
    } else if (a == 1) {
        return b;
    } else if (b == 1) {
        return a;
    } else if (a * b == 0) {
        return multiply (a/2, b);
    } else {
        return multiply (a/2, b) * 2 + b;
    }
}
```

Melhor caso : $\Omega(1)$ Onde quando $a \leq 1$ ou $b \leq 1$. Não foge nenhuma multiplicação.



Pior caso : Quando a é maior que 1 e ímpar
Complexidade $\Theta(\ln a)$. ($a \geq 3$)

$\sum_{i=0}^{\lfloor \frac{a}{2} \rfloor} \frac{a}{2}$ → A função que comece com $a = 1$, sempre saímos da função quando $a = 1$
~~X~~ não precisa de monitorar o somatório.

b)

```
static int power (int a, int b) {
    if (b < 0) return a;
    if (b == 0) return 1;
    int sum = a;
    for (int I = 0; I < b-1; I++) {
        sum *= a;
    }
    return sum;
}
```

Melhor caso : $\Theta(1)$. Realiza 0 multiplicações quando $b \leq 1$.



Pior caso : Quando $b > 1$. Realiza $b-1$ multiplicações. Complexidade linear $\Theta(b)$.

```
c) void printPairs (int [] array) {
    for (int i = 0; i < array.length; i++) {
        for (int j = 0; j < array.length; j++) {
            System.out.println (array [i] * array [j]);
        }
    }
}
```



Não tem melhor nem pior caso. Em todos os cenários a complexidade é $\Theta(n^2)$.

~~Se sobrar espaço achar o somatório de i^2~~

```
function selectionOPT {
| int left = 0; int right = arr.length;
| while (left < right) {
| | int lowest = left; int max = left;
| | for (int i = left; i < right; i++) {
| | | if (arr[i] < arr[lowest]) lowest = i;
| | | if (arr[i] > arr[max]) max = i;
| | }
| | swap(arr, lowest, left);
| | if (max == left) max = lowest;
| | swap(arr, max, right - 1);
| | left++; right--;
| }
| }
function bubbleOPT {
| int last_swap = arr.length - 1;
| for (int i = 0; i < arr.length - 1; i++) {
| | int new_last_swap = 0;
| | for (int j = 0; j < last_swap; j++) {
| | | if (arr[j] > arr[j + 1]) {
| | | | swap(arr, j, j + 1);
| | | | new_last_swap = j;
| | }
| | }
| | if (!new_last_swap) {
| | | i = arr.length;
| | } else {
| | | last_swap = new_last_swap;
| | }
| }
| }
```

$\text{const.} \cdot (\text{limSup} - \text{limInf} + 1)$

$$\text{distributividade } \sum_i c \cdot a_i = c \cdot \sum_i a_i$$

$$\text{associatividade } \sum_i (a_i + b_i) = \sum_i a_i + \sum_i b_i$$

comutatividade: permite botar os termos em qq ordem

$$\text{P.A.} \rightarrow \sum_{i=0}^n (a + b \cdot i) = (2a + bn) \cdot (n+1)$$

$$\text{GAUSS} \rightarrow \sum_{i=0}^n i = \frac{n \cdot (n+1)}{2}$$

$$\text{P.G.} \rightarrow \sum_{i=0}^n a \cdot x^i = \frac{a - a \cdot x^{n+1}}{1-x} \quad \text{p/ } x \neq 1$$

P1 Combinar conjuntos

$$\sum_{i \in X} a_i + \sum_{i \in Y} a_i = \sum_{i \in X \cup Y} a_i + \sum_{i \in X \cap Y} a_i$$

P2 Perturbação S_{n+1}

$$S_n + a_{n+1} = a_0 + \sum_{i=0}^n a_{i+1} \quad \frac{2n^3 + 3n^2 + n}{6}$$

INDUÇÃO

1 - Subst. n na eq. pelo 1º valor de i

2 - $S_n = \underbrace{S_{n-1}}_{\text{termo somado}} + a_n$ subst. i por n

minha eq. substituindo n por $n-1$

```
function selection{
    for(int i=0; i<n-1; i++){
        int lowest=i;
        for(int j=i+1; j<n; j++){
            if(arr[j] < arr[lowest]) lowest=j;
        }
        arr = swap(arr, i, lowest);
    }
    return arr;
}
```

```
function insertion{
    for(int i=1; i<n; i++){
        int tmp = arr[i];
        j = i-1;
        while(j >= 0 && arr[j] > tmp){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = tmp;
    }
}
```

```
function bubble{
    for(int i=0; i<n-1; i++){
        for(int j=0; j<n-i-1; j++){
            if(arr[j] > arr[j+1]) swap(arr, j, j+1);
        }
    }
}
```

busca sequencial (vetor \vec{n} ordenado)

MC $\rightarrow \Theta(1)$ PC $\rightarrow \Theta(n)$

busca binária (vetor ordenado)

MC $\rightarrow \Theta(1)$ PC $\rightarrow \Theta(\ln n)$ arr.length - 1

```
binary_rec(int[] arr, int target, int start, int end){
    if(start > end) return -1;
    int middle = (start + end) / 2;
    if(arr[middle] == target) return middle;
    if(arr[middle] > target) return binary_rec(arr, target, start, middle - 1);
    if(arr[middle] < target) return binary_rec(arr, target, middle + 1, end);
}
```

I Seleção

comparações

$$\text{MC e PC: } \Theta(n^2) \sum_{i=0}^{n-1} n-i-1 \quad C = \frac{n^2-n}{2}$$

E inserção

comparações

$$\text{MC: } \Theta(n) \text{ PC: } \Theta(n^2) \sum_{i=1}^{n-1} i \quad C \uparrow$$

E bolha

comparações

$$\text{MC e PC: } \Theta(n^2) \sum_{i=0}^{n-2} n-i-2 \quad C$$

Class Lista {

int []arr; int n;

Lista (int tam){

arr = new int [tam]; n = 0;

}

FILA: first in, first out

insere no final e remove

no inicio (ou o contrário)

arr = new int [tam]; n = 0;

PILHA: first in, last out

void inserirInício (int x){} insere e remove no

void inserirFim (int x){} •• final (topo).

void inserir (int x, int pos){}

int removerInício (){}•

int removerFim (){}•

int remover (int pos) {}

void mostrar (){}•

movimentações

$$\text{MC e PC: } \Theta(n) \sum_{i=0}^{n-2} 3 \quad M = 3n-3$$

$$\text{MC: } \Theta(n) \text{ PC: } \Theta(n^2) \sum_{i=1}^{n-1} 2+(i-1) \quad M = \frac{n^2+n-2}{2}$$

$$\text{MC: } \Theta(1) \text{ PC: } \Theta(n^2) \sum_{i=0}^{n-2} 3(n-i-2) \quad M = \frac{3(n^2-n)}{2}$$