**MARCOS ANTONIO LOMMEZ CANDIDO RIBEIRO**

• Quando existir um laço começando em 0 ate N teremos, N repetições, por outro lado quando começar em 'a' teremos n-a repetições
for (int i = 0; i<n; i++) = n
for (int i = a; i<n; i++) = n-a

• O laco do algoritmo de seleção não precisa ir até a posição n-1 (ultima), por isso seu limite superior será (n-2), enquanto o laco interno realiza i<n operações, o que pode ser expresso com a equação (n-i-1) onde (n-i) simula um decremento no valor de 'i'. Logo sua formula será:

$\sum_{i=0}^{n-2}(\mathrm{n}-\mathrm{i}-1)$

$\sum_{i=0}^{n-2}(\mathrm{n}) - \sum_{i=0}^{n-2}(i) - \sum_{i=0}^{n-2}(1)$ << associatividade

$n.(n-1) - \sum_{i=0}^{n-2}(i) - 1.(n-1)$ << substituição pela formula (n*n)

$n.(n-1) - \frac{(n-2)(n-1)}{2} - 1.(n-1)$ << substituição pelo somatório de Gauss $\frac{n(n-1)}{2}$

• Também podemos manipular somas através dos seus conjuntos

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

• E através da base para perturbação

$S_n + a_{n+1} = a_0 + \sum a_{i+1}$ onde $a_i = a.x^i$

• Exemplo aplicado a formula do somatório da PG

$S_n + a.x^{n+1} = a.x^0 + \sum a.x^{i+1}$ = $S_n + a.x^{n+1} = a.x^0 + \sum a.x^i.x$

$S_n + a.x^{n+1} = a.x^0 + x.\sum a.x^i$ = $S_n + a.x^{n+1} = a.x^0 + x.\ S_n$

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

$(1-x)S_n = a - a.x^{n+1}$ = $S_n = \frac{a - a.x^{n+1}}{(1-x)}$ (para i diferente de 1)

• Exemplo de Somatório encontrado através da aplicação da formula acima junto da base para perturbação:

$$\boxed{S_n = \sum_{0 \le i \le n} i.2^i = (n-1).2^{n+1} + 2}$$

• Outras formulas uteis:

$\sum_0^n i = \mathrm{n(n+1)}$

$\sum_0^n 1 = \mathrm{n+1}$

$\sum_0^n i^2 = \frac{n(n+1)(2n+1)}{6}$

$a_i = i^2$

• Uma formula de somatório pode ser provada correta utilizando a indução.
A mesma é dividida em duas partes, sendo

- Passo base: Substituir n pelo primeiro valor e provar seu resultado
- Indução propriamente dita: A formula devera ser valida ao substituir n por n-1

Exemplos:
Passo base

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

$$S_0 = \frac{0\,.\,(0+1)\,.\,(2.0+1)}{6} = 0$$

Indução propriamente dita:

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

$$S_{n-1} = \frac{(n-1)\,((n-1)+1)(2(n-1)+1)}{6} = \frac{(n-1)(n)(2n-1)}{6}$$

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

• Para calcular o numero de repetições de um laço do tipo logarítmico (n/=2) realizaremos $\lfloor lg(n) \rfloor + 1$, o +1 ocorre por causa que o logaritmo não conta o valor do laço onde n será igual a 1.

• A pesquisa sequencial possui custo Θ(n) e a pesquisa binaria possui custo Θ(lg(n))
O custo mínimo da ordenação é Θ(n.lg(n)), logo para um array não ordenado deveremos usar a pesquisa sequencial, a não ser sejam realizados n.$lg^2$ (n) pesquisas, a partir dai vale mais apena ordenar e pesquisar binariamente.

A definição das notações nasce do conceito de que:
Para uma função g(n) ser O(f(n)), ela devera possuir constantes ‘c’ e ‘m’ para que sua “reta” ultrapasse em determinado momento (m) a “reta” de (g).
Ao utilizar a notação estes valores ficarão implícitos, é necessário apenas que eles existam para que a premissa se torne verdadeira. Pelo mesmo lado o mesmo é necessário para Ω(f(n)), a diferença é que a “reta” não pode ultrapassar
E por fim para Θ(f(n)) estes valores deverão existir onde seja possível que a “reta” esteja entre O e Ω de forma justa.

• Exemplo de notação de complexidade

```
for (i = 0; i < n; i++) {
      for (j = 1; j <= n; j *= 2) {
            b--;
      }
}
```

função

f(n) = (lg(n) + 1) * n = n * lg(n) + n

complexidade

O(n *x* lg(n)), Ω(n *x* lg(n)) e Θ(n *x* lg(n))