DOM SEG TER QUA QUI SEX SÁB

AEDS II

23 09 21

MEGA INTENSIVÃO AEDS II

PARA A SEMANA QUE VEM

UNIDADE I:

1) NOÇÕES DE COMPLEXIDADE

CENÁRIOS POSSÍVEIS
- → MELHOR CASO: MENOR TEMPO DE EXECUÇÃO PARA TODAS AS ENTRADAS
- → PIOR CASO: MAIOR TEMPO DE EXECUÇÃO PARA TODAS AS ENTRADAS
- → CASO MÉDIO: MÉDIA DOS TEMPOS DE EXECUÇÃO PARA TODAS AS ENTRADAS

• CONTAGEM DE OPERAÇÕES COM CONDICIONAL: É O CUSTO DA CONDIÇÃO MAIS O DA LISTA VERDADEIRA OU FALSA

```
if (condição()){
|  listaV();
}else{
|  listaF();
}
```

⇒ • MELHOR CASO = condição() + min(listaV(), listaF())
⇒ • PIOR CASO = condição() + max(listaV(), listaF())

• CONTAGEM DE OPERAÇÕES COM REPETIÇÃO: É O CUSTO DA CONDIÇÃO MAIS O NÚMERO DE ITERAÇÕES VEZES A SOMA DOS CUSTOS DA CONDIÇÃO E DA REPETIÇÃO

```
while (condição()){
|  lista();
}
```

⇒ • CUSTO = condição() + n(lista() + condição())

• CONTAGEM DE OPERAÇÕES COM REPETIÇÃO (for): POSSUI 3 CASOS:

SE i=0

```
for (int i=0; i<n; i++){
|  lista();
}
```

CUSTO = n × lista()

SE i ≠ 0

```
for (int i=a; i<n; i++){
|  lista()
}
```

CUSTO = (n-a) × lista()

ANINHADA

```
for (int i=0; i<n; i++){
|  for (int j=0; j<a; j++){
|  |  lista()
|  }
}
```

CUSTO = n × a × lista()

2) NOTAÇÕES O, Ω E θ

- LIMITE SUPERIOR (O): A TAXA DE CRESCIMENTO DE $f(x)$ ASSINTOTICAMENTE MENOR OU IGUAL A DE $g(x)$

EXEMPLO ERRADO: $a = 2,4n \quad b = O(a) \longrightarrow$ ~~$b = 3n$~~ (NA VERDADE, $b \geq n \ln n$) !

- LIMITE INFERIOR (Ω): A TAXA DE CRESCIMENTO DE $f(x)$ É MAIOR OU IGUAL A DE $g(x)$

$a = 2,4n \quad b = \Omega(a) \longrightarrow b \leq n \downarrow$

- LIMITE ESTREITO (θ): A TAXA DE CRESCIMENTO DE $f(x)$ É IGUAL A DE $g(x)$

$a = 2,4n \quad b = \theta(a) \longrightarrow b = n -$

3) ALGORITMOS DE PESQUISA

- PESQUISA SEQUENCIAL: COMPARAÇÃO ENTRE ELEMENTOS DO ARRAY ATÉ SATISFAZER A CONDIÇÃO

```
for (int i=0; i<n; i++){
    if(array[i]==x){
        resp=true;
        i=n;
    }
}
```

COMPLEXIDADE:
MELHOR CASO: θ1
PIOR CASO: θn
CASO MÉDIO: θn

- PESQUISA BINÁRIA: PESQUISA A PARTIR DA METADE DO ARRAY. SE O ELEMENTO FOR MAIOR QUE A METADE, DESCARTE A OUTRA E VICE-VERSA

```
int dir=n-1, esq=0, meio;
while (esq<=dir) {
    meio=(esq+dir)/2;
    if(x==array[meio]){
        resp=true
        esq=n;
    }else if (x>array[meio]){
        esq=meio+1;
    }else{
        dir=meio-1;
    }
}
```

COMPLEXIDADE:
MELHOR CASO: θ1
PIOR CASO: θ(lg n)

DOM SEG TER QUA QUI SEX SÁB

4) ALGORITMO DE ORDENAÇÃO POR SELEÇÃO

AEDS 2 / FONTE / U04
IMPORTANTE!

→ A ORDENAÇÃO É DITA INTERNA QUANDO A LISTA DE ELEMENTOS CABE NA MEMÓRIA PRINCIPAL. CASO CONTRÁRIO, ELA É EXTERNA

• CHAVE DE PESQUISA: É O ATRIBUTO UTILIZADO PARA ORDENAR OS REGISTROS

• OPERAÇÕES FUNDAMENTAIS: COMPARAÇÃO E MOVIMENTAÇÃO ENTRE OS ELEMENTOS DO ARRAY

VÁRIOS ALGORITMOS DE ORDENAÇÃO INTERNO CHEGAM A ESSE LIMITE →

$\Theta(n * \log n)$

LIMITE INFERIOR EM TERMOS DO Nº DE COMPARAÇÕES

← LOGO, ESSA FÓRMULA TAMBÉM SE TRATA DA COMPLEXIDADE ÓTIMA PARA ORDENAÇÃO INTERNA EM NÚMERO DE COMPARAÇÕES DO PIOR E DO CASO MÉDIO

→ O ALGORITMO É DITO ESTÁVEL SE DEPOIS DA EXECUÇÃO, OS ELEMENTOS COM A MESMA CHAVE MANTIVEREM A ORDEM RELATIVA ENTRE AS CHAVES REPETIDAS

5 1 3 2 4 3 0

0 1 2 3 3 4 5

• FUNCIONAMENTO:

I) PROCURAR O MENOR ELEMENTO DO ARRAY

101 115 30 63 47 20

II) TROCAR DE POSIÇÃO COM O PRIMEIRO ELEMENTO

20 115 30 63 47 101

III) REPITA O PROCESSO A PARTIR DA PRÓXIMA POSIÇÃO

20 | 115 30 63 47 101 |

• ALGORITMO EM C LIKE:

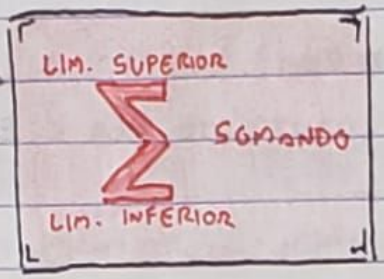
```
for (int i = 0; i < (n-1); i++){
    int menor = i;
    for (int j = (i+1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap (menor, i);
```

DOM SEG TER QUA QUI SEX SÁB
DOM LUN MAR MIÉ JUE VIE SÁB

UNIDADE II

1) SOMATÓRIO (Σ)

O CUSTO DE UM ALGORITMO É A SOMA DOS CUSTOS DAS SUAS OPERAÇÕES



$$\Rightarrow \sum_{i=0}^{n} = 0+1+2+3+\dots n$$

$$\sum_{i=1}^{i\leq n} a_i = \sum_{1}^{n} a_i = \sum_{1\leq i\leq n} a_i = \sum_{i=1}^{i\leq n} a_i$$

2) RELAÇÕES DE RECORRÊNCIA

TÉCNICA USADA PARA CALCULAR SOMAS. É DISCUTIDA EM TEORIA DOS GRAFOS E COMPUTABILIDADE

$$\begin{cases} fat(1) = 1 \\ fat(n) = n \cdot fat(n-1) \end{cases}$$

REPETE O PROCESSO ATÉ $fat(1)$ E DEPOIS VAI RESOLVENDO-OS.

3) SOMAS MÚLTIPLAS

OS TERMOS DO SOMATÓRIO PODEM SER ESPECIFICADO EM DOIS OU MAIS ÍNDICES

$$\sum_{1\leq i,j\leq 3} a_i b_j = \left(\sum_{i=1}^{i\leq 3} a_i\right)\left(\sum_{j=1}^{j\leq 3} b_j\right)$$

4) MANIPULAÇÃO DE SOMAS

- DISTRIBUTIVIDADE $\Rightarrow \sum_{i\in I} c \times a_i = \left(\sum_{i\in I} a_i\right) \times c$ C = CONSTANTE

$$\Rightarrow \sum_{i\in I} \frac{a_i}{c} = \frac{1}{c}\left(\sum_{i\in I} a_i\right)$$ C = CONSTANTE

- ASSOCIATIVIDADE ⇨ $\sum_{i \in I} (a_i + b_i) = \sum_{i \in I} a_i + \sum_{i \in I} b_i$

↳ $\sum_{i \in I} (a_i - b_i) = \sum_{i \in I} a_i - \sum_{i \in I} b_i$

- COMUTATIVIDADE ⇨ $\sum_{i \in I} a_i = \sum_{p(i) \in I} a_{p(i)}$

5) MÉTODOS GERAIS

- PROVA POR INDUÇÃO:

I) PROVE QUE A FÓRMULA É VERDADEIRA PARA O PRIMEIRO VALOR (SUBSTITUA n PELO PRIMEIRO VALOR)

II) SUPONDO QUE $n > 0$ E QUE A FÓRMULA É VOLIDA (I)

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

S_{n-1} = EQUAÇÃO FEITA EM (I)
a_n = nésimo TERMO DA SEQUÊNCIA

- PERTURBE A SOMA: APLICA AS REGRAS BÁSICAS DE TRANSFORMAÇÃO E OS PROPRIEDADES P1 E P2

$S_{CUBO_n} = \sum_{0 \leq i \leq n} i^3$ $\quad$ $S_{GAUSS} = \sum_{0 \leq i \leq n} i$

UNIDADE III

1) FUNÇÕES DE COMPLEXIDADE

- FUNÇÃO DE COMPLEXIDADE EM TEMPO: MEDE O TEMPO (NÚMERO DE EXECUÇÕES DA OPERAÇÃO RELEVANTE) DE EXECUÇÃO DO ALGORITMO PARA UM PROBLEMA DE TAMANHO h

- FUNÇÃO DE COMPLEXIDADE DE ESPAÇO: MEDE A QUANTIDADE DE MEMÓRIA NECESSÁRIA PARA EXECUTAR UM ALGORITMO DE TAMANHO n

• COMO CALCULAR A COMPLEXIDADE DE UM ALGORITMO?

I) **CONDICIONAL**: CUSTO DA CONDIÇÃO MAIS O DA LISTA V OU F

```
if(condição()){
    listaV();
}else{
    listaF();
}
```

CUSTO

• MELHOR CASO = condição + min (listaV(), listaF())

• PIOR CASO = condição + max (listaV(), listaF())

II) **REPETIÇÃO**: CUSTO DA CONDIÇÃO MAIS O NÚMERO DE INTERAÇÕES MULTIPLICADO PELO SOMA DOS CUSTOS DA CONDIÇÃO E DA LISTA A SER REPETIDA

```
while(condição()){
    lista();
}
```

CUSTO = condição + n (lista() + condição())

III) **OUTROS LAÇOS**: CONSIDERA O LIMITE SUPERIOR

IV) **MÉTODOS**: CONSIDERA O CUSTO DO MÉTODO

V) **RECURSIVIDADE**: UTILIZA EQUAÇÕES DE RECORRÊNCIA (TEORIA DOS GRAFOS E COMPUTABILIDADE)

• ALGORITMO ÓTIMO: É O ALGORITMO NO QUAL SEU CUSTO É IGUAL AO MENOR CUSTO POSSÍVEL

• CLASSE DE ALGORITMOS

• CONSTANTE = $O(1)$	• LINEAR-LOGARÍTMICO = $O(n \cdot \log n)$	• EXPONENCIAL = $O(2^n)$
• LOGARÍTMICO: $O(\log n)$	• QUADRÁTICO = $O(n^2)$	
• LINEAR = $O(n)$	• CÚBICO = $O(n^3)$	

DOM | SEG | TER | QUA | QUI | SEX | SÁB

- **ALGORITMO POLINOMIAL**: É CONSIDERADO QUANDO O ALGORITMO FOR IGUAL A $O(n^p)$ PARA ALGUM INTEIRO p. PROBLEMAS COM ALGORITMOS POLINOMIAIS SÃO TRATÁVEIS, ENQUANTO AQUELES SEM ESTE ALGORITMO É CONSIDERADO INTRATÁVEL

- **EXEMPLO**: FAÇA A CORRESPONDÊNCIA ENTRE CADA FUNÇÃO $f(n)$ COM SUA $g(n)$ EQUIVALENTE, EM TERMOS DO Θ. ESSA CORRESPONDÊNCIA ACONTECE QUANDO $f(n) = \Theta(g(n))$

f(n)	g(n)
$n + 30$	n^4
$n^2 + 2n - 10$	$3n - 1$
$n^3 * 3n$	$\lg(2n)$
$\lg(n)$	$n^2 + 3n$

UNIDADE IV

1) ALGORITMO DA BOLHA (BUBBLE SORT)

PC = PIOR CASO CM = CASO MÉDIO

COMPARAÇÕES = $\frac{n(n-1)}{2}$

MOVIMENTAÇÕES = $3.[(n(n-1))/2]$ PC

$3.((n(n-1))/4)$ CM

- **FUNCIONAMENTO**:

I) SELECIONE O ÚLTIMO ELEMENTO DO ARRAY

30 63 47 20

II) COMPARE O COM O ELEMENTO À SUA ESQUERDA

30 63 [47 20]

III) SE O ELEMENTO SELECIONADO FOR O MENOR, TROQUE-O DE LUGAR

30 63 20 47

IV) REPITA OS PASSOS II E III ATÉ NÃO POSSUIR VALOR MENOR À ESQUERDA

30 [63 20] 47

V) RETORNE AO PASSO I COM O PRÓXIMO ELEMENTO

20 30 63 47

- **ANÁLISE DO ALGORITMO**:

→ ELE É ESTÁVEL

→ REALIZA VÁRIAS COMPARAÇÕES REDUNDANTES

→ REALIZA UM NÚMERO QUADRÁTICO DE MOVIMENTAÇÕES

DOM DOM | SEG LUN | TER MAR | QUA MIÉ | QUI JUE | SEX VIE | SÁB SÁB

2) ALGORITMO DE INSERÇÃO (INSERTION SORT)

- FUNCIONAMENTO:

I) SELECIONE O PRIMEIRO ELEMENTO DO ARRAY

101 115 30 63

II) COMPARE COM O ELEMENTO A DIREITA

101 115 30 63

III) SE O NÚMERO SELECIONADO FOR MENOR QUE O DA DIREITA, VOLTE AO PASSO I COM O PRÓXIMO ELEMENTO

101 115 30 63

IV) SE O NÚMERO SELECIONADO FOR MAIOR QUE O DA DIREITA, COLOQUE O ELEMENTO MENOR EM UMA VARIÁVEL TEMPORÁRIA

101 115 30 63

TEMP = 30

COMPARE O ELEMENTO TEMP COM TODOS A SUA ESQUERDA. CASO SEJA MAIOR, DESLOQUE SUA POSIÇÃO NO ARRAY

101 → 115 → __ 63

30

V) REPITA O PROCESSO COM O PRÓXIMO ELEMENTO

30 101 115 63

- ANÁLISE DO ALGORITMO:
  - ELE É ESTÁVEL
  - É MAIS RECOMENDADO O SEU USO CASO O ARRAY ESTEJA "QUASE" ORDENADO
  - TAMBÉM RECOMENDADO CASO QUEIRA ADICIONAR NOVOS ITENS EM UM ARRAY ORDENADO

COMPARAÇÕES = $\Theta(n)$ MC, $\Theta(n^2)$ PC

MOVIMENTAÇÕES = $\Theta(n)$ MC, $\Theta(n^2)$ PC

MC = MELHOR CASO PC = PIOR CASO

DOM SEG TER QUA QUI SEX SÁB

RAZÃO DE EFICIÊNCIA NÃO CONHECIDA
COMPARAÇÕES: PROB 1 = $\Theta(n^{1,26})$ PROB 2 = $\Theta(n(\ln n)^2)$

3) SHELL SORT

• FUNCIONAMENTO:

I) POR SE TRATAR DE UMA SEQUÊNCIA h-ORDENADA, DIVIDA EM h OPERAÇÕES (DIMINUINDO O h APÓS CADA REPETIÇÃO). COMEÇANDO COM h=4

90 45 32 11 15 60 45 70

II) REALIZE O INSERTION SORT EM CADA GRUPO h

15 45 32 11 90 60 45 70

III) VOLTE AO PASSO I E REDUZA O h (EX.: h=2)

15 45 32 11 90 60 45 70

• ANÁLISE DO ALGORITMO:

→ É UMA VERSÃO MELHORADA DO INSERTION SORT
→ É O MAIS EFICIENTE DOS ALGORITMOS DE COMPLEXIDADE QUADRÁTICA (1,2,3)
→ ELE É NÃO-ESTÁVEL
→ BOM PARA ARQUIVOS DE TAMANHO MODERADO

4) QUICK SORT

COMPARAÇÕES: $n \cdot \lg(n) - n + 1$ MC, Θn^2 PC
MOVIMENTAÇÕES: 3 P/ RECURSIVA MC, $n/2$ PC

• FUNCIONAMENTO:

I) DIVIDA O ARRAY EM 2, COM O ELEMENTO CENTRAL (PIVÔ) PODENDO SER A MÉDIA, A MODA OU A MEDIANA ENTRE OS ARRAYS

90 45 32 11 27 75 60 70

II) SELECIONE OS ELEMENTOS MAIS À DIREITA E À ESQUERDA DO ARRAY

90 45 32 11 27 5 60 70

III) À ESQUERDA DO PIVÔ, ENCONTRE OS NÚMEROS MAIORES QUE ELE, À DIREITA DO PIVÔ, ENCONTRE OS NÚMEROS MENORES QUE ELE

90 45 32 11 2 75 60 70

IV) TROQUE DE LUGAR AS DUAS ARRAYS SELECIONADAS

2 45 32 11 90 75 60 70

V) VOLTE AO PASSO I, AGORA COM AS DUAS METADES DE CADA ARRAY

2 45 32 11 90 75 60 70

- ANÁLISE DO ALGORITMO:
  - → É EXTREMAMENTE EFICIENTE
  - → FAZ EM MÉDIA $\theta(n * \log(n))$ COMPARAÇÕES
  - → SUA IMPLEMENTAÇÃO É DELICADA E DIFÍCIL
  - → ELE É NÃO-ESTÁVEL

5) MERGESORT

COMPARAÇÕES: $\theta(n * \log(n))$
MOVIMENTAÇÕES: $\theta(n * \log(n))$

- FUNCIONAMENTO:

I) DIVIDA O ARRAY EM 2 ATÉ POSSUIR 2 COMPARAÇÕES

50 44 35 93 67 22 36 18

↓

50 44 35 93

↓

50 44

II) SE O PRIMEIRO VALOR FOR MAIOR QUE O SEGUNDO, TROQUE-OS DO LUGAR

44 50 35 93

III) VÁ RETORNANDO ÀS DIVISÕES ANTERIORES

35 44 50 93

44 50 35 93

i j

OBS = COLOQUE i E j PARA COMPARAR OS VALORES (if(i<j){...})

IV) CONTINUE RETORNANDO ATÉ O ARRAY ORIGINAL

18 22 35 36 44 50 67 93

- ANÁLISE DO ALGORITMO:
  - → POR CONTA DA RECURSIVIDADE, ELE DEMANDA MAIS MEMÓRIA
  - → ELE É ESTÁVEL
  - → INDEPENDENTE DOS CASOS, SEU NÚMERO DE COMPARAÇÕES E MOVIMENTAÇÕES SERÁ IGUAL

DOM SEG TER QUA QUI SEX SÁB

COMPARAÇÕES: $\Theta \log(n)$
MOVIMENTAÇÕES: $\Theta \log(n)$
COMPLEXIDADE: $\Theta(n \times \log(n))$

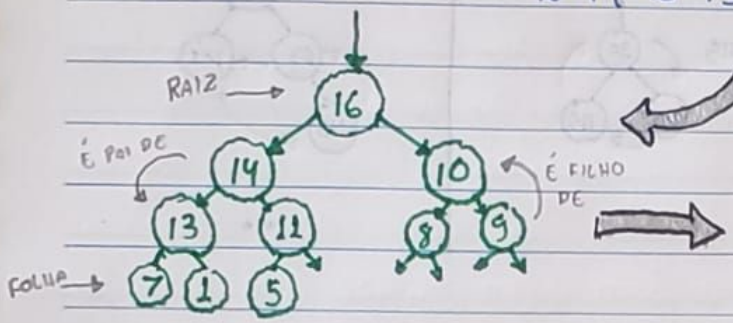
6) HEAPSORT

- FUNCIONAMENTO

→ **HEAP**: ALGORITMO DE SELEÇÃO QUE ENCONTRA O MAIOR ELEMENTO EM UMA LISTA, TROCA COM O ÚLTIMO E REPETE O PROCESSO

→ É NECESSÁRIO CONHECIMENTO SOBRE ÁRVORES BINÁRIAS

16 14 10 13 11 8 9 7 1 5



- array [1] É O RAIZ;
- array [i/2] É O PAI DE ARRAY [i] PARA i>1;
- SE array [2*i] E array [2*i+1] EXISTEM, ELES SÃO FILHOS DE array [i]
- SE i > (n/2), array [i] É UMA FOLHA

O HEAP SORT UTILIZA O HEAP INVERTIDO!

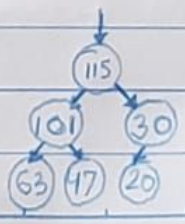
→ **PRINCÍPIO DE INSERÇÃO**: CRIE UMA NOVA FOLHA (CONTENDO O NOVO ELEMENTO NO ÚLTIMO NÍVEL DO HEAP. SE ESSE ESTIVER COMPLETO, RECOMECE UM NOVO NÍVEL. SE O NOVO ELEMENTO FOR MAIOR QUE O PAI, TROQUE-OS ATÉ QUE TODOS OS PAIS SEJAM MAIORES QUE SEUS FILHOS

→ **PRINCÍPIO DE REMOÇÃO**: ARMAZENE O ELEMENTO DA RAIZ EM UMA VARIÁVEL TEMPORÁRIA, SUBSTITUA O ELEMENTO RAIZ PELO DA ULTIMA FOLHA NO ÚLTIMO NÍVEL, REMOVA A ÚLTIMA FOLHA DO ÚLTIMO NÍVEL, TROQUE O ELEMENTO DA RAIZ COM O DE SEU MAIOR FILHO E REPITA ATÉ QUE TODOS OS PAIS SEJAM MAIORES QUE SEUS FILHOS

AGORA SIM PODEMOS PARTIR PARA O PASSO-A-PASSO:

I) CONSTRUA O HEAP INSERINDO SISTEMATICAMENTE CADA UM DOS ELEMENTOS DO ARRAY

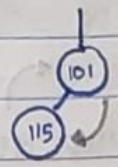
115 101 30 63 47 20



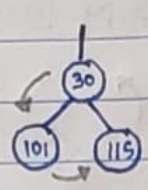
A ORGANIZAÇÃO DO ARRAY SE DÁ PELA ORDEM DE FUNCIONAMENTO DA ÁRVORE

II) REMOVA SISTEMATICAMENTE CADA ELEMENTO DO HEAP, RECONSTRUA-O E INSIRA O ELEMENTO REMOVIDO NA POSIÇÃO DO ARRAY IMEDIATAMENTE SEGUINTE AO TAMANHO CORRENTE DO HEAP

1) 101 < 115



2) 30 < 101 < 115



3) 30 < 63 < 101 < 115