Sistemas de Informação
6º Período
13/01/2021

IFMG - Ouro Branco
Diego Santos Seabra
0040251

Tarefa 2

Projeto e Análise de Algoritmos

Contents

1) Usando o pseudocódigo do algoritmo de INSERTION-SORT, ilustre as operações realizadas no arranjo A = [31, 41, 59, 26, 41, 58]

Solução:

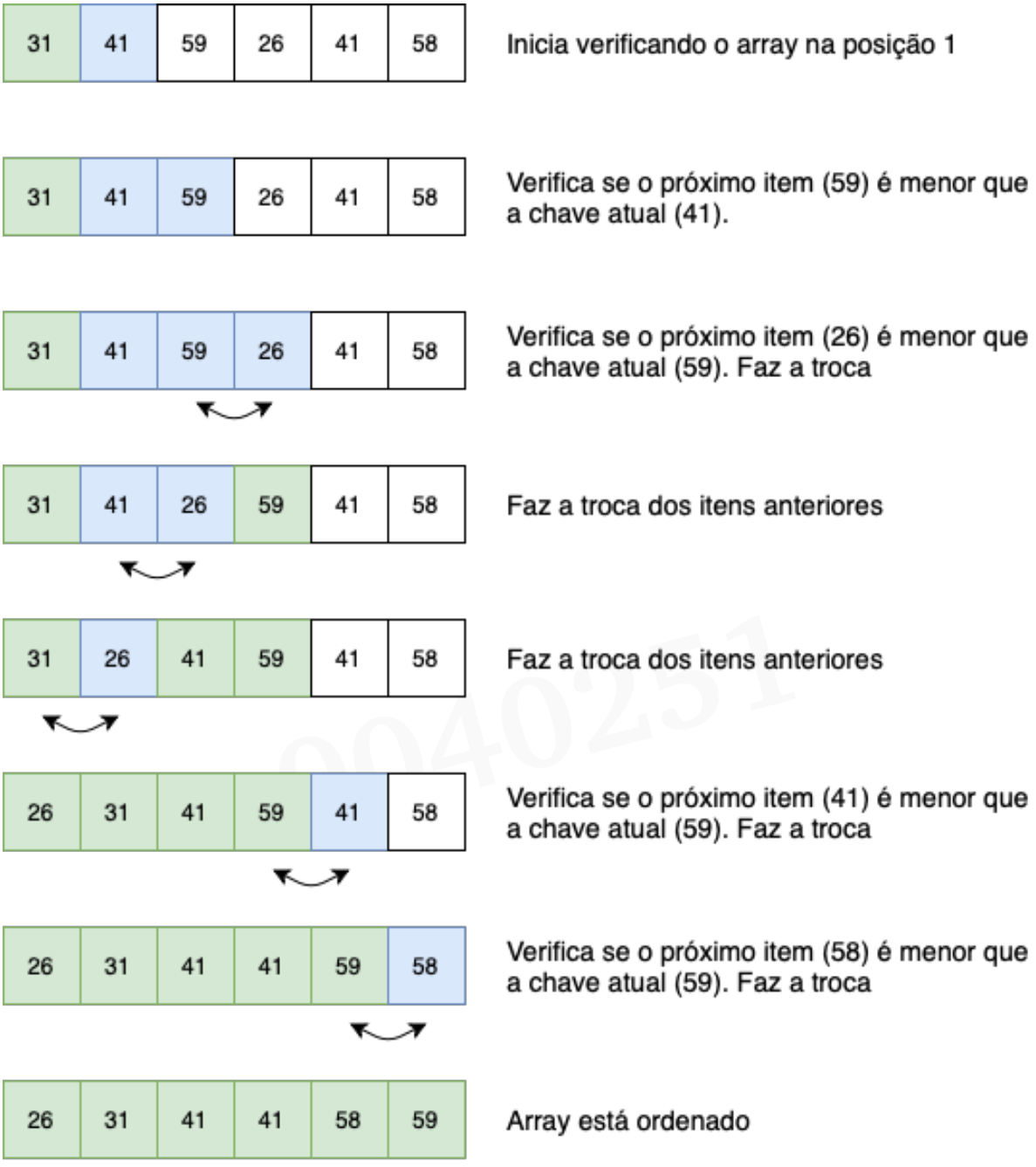


Fig. 1: Operações do Insertion Sort no Array $31, 41, 59, 26, 41, 58$

2) Reescreva o procedimento INSERTION-SORT para ordenar em ordem decrescente, em vez da ordem crescente.

Solução:

Pseudocódigo

```
1: ordena(A):
2: for b ← 2 to comprimento[A] do
3:     chave ← A[b]
4:     i ← b − 1
5:     while i > 0 e chave > A[i] do
6:         A[i + 1] ← A[i]
7:         i ← i − 1
8:     end while
9:     A[i + 1] ← chave
10: end for
```

Código Implementado

```c
//////////////////////////////////////
//                                  //
// Autor: Diego S. Seabra           //
// Matricula: 0040251               //
//                                  //
//////////////////////////////////////

#define n 6

#include <stdio.h>

// 31 41 59 26 41 58
void ordena(int arr[n])
{
    int chave, i;
    for (int b = 1; b < n; b++) // b = 2
    {
        chave = arr[b];
        i = b - 1;

        while (i ≥ 0 && chave > arr[i])  // i > 0
        {
            arr[i + 1] = arr[i];
            i = i - 1;
        }
        arr[i + 1] = chave;
    }
}

void imprimeArranjo(int arr[n])
{
    for (int i = 0; i < n; i++)
    {
        printf("%i ", arr[i]);
    }
    printf("\n");
}

int main()
{
    int arr[n] = {31, 41, 59, 26, 41, 58};
    printf("Inicial:\n");
    imprimeArranjo(arr);

    ordena(arr);

    printf("Ordenado:\n");
    imprimeArranjo(arr);
}
```

3) Considere o problema de pesquisa

a) Entrada: Uma sequência de n números A = <a1, a2, ..., an> e um valor v

b) Saída: Um índice i tal que v = A[i] ou o valor especial NIL, se não aparecer em A

c) Escreva um pseudocódigo para pesquisa linear, que faça a varredura da sequencia, procurando por v. Usando um loop invariante, prove que seu algoritmo é correto.

Solução:

Pseudocódigo

```
1: varre(A, v):
2: for i ← 1 to comprimento[A] do
3:     if A[i] == v then
4:         return i
5:     end if
6: end for
7: return NIL
```

Código Implementado

```c
//////////////////////////////////////
//                                  //
// Autor: Diego S. Seabra           //
// Matricula: 0040251               //
//                                  //
//////////////////////////////////////

#define n 6

#include <stdio.h>

int varre(int arr[n], int v)
{
    for (int i = 0; i < n; i++)
    {
        if(arr[i] == v){
            return i;
        }
    }
    return -1;
}

int main()
{
    int arr[n] = {31, 41, 59, 26, 44, 58};

    // Testes
    int res = varre(arr, 58);
    // int res = varre(arr, 99);
    // int res = varre(arr, 26);
    printf("%i\n", res);
}
```

Loop Invariante

Inicialização

Na etapa de inicialização do algoritmo, se o vetor estiver vazio, o loop invariante não será executado. Portanto o i se inicia a partir do 1.

Manutenção

Na etapa de manutenção do algoritmo, assumindo que o item não foi encontrado nas iterações anteriores, em cada iteração é checado o i-ésimo valor do arranjo com v, retornando i se $A[i] = v$; assim, devido ao algoritmo ter completado essa iteração, a linha 4 (return i) não foi executada, o que significa que a condição da linha 3 é falsa. Portanto seu inverso é verdadeiro e $A[i] \neq v$; a invariante ainda se mantém.

Finalização

Na etapa de finalização, há duas maneiras do loop ser finalizado:

1. A primeira é quando i é maior que o tamanho do arranjo, ou seja, $i = comprimento[A]+1$. Como demonstrado na parte de manutenção, o algoritmo executaria a linha depois do loop, retornando NIL, o que é o resultado correto (pois não foi encontrado um valor $A[i] == v$).
2. A segunda é a finalização na linha 4. Se isto ocorre, a condição da linha 3 era verdadeira ($A[i] == v$). Neste caso o algoritmo retorna i, que é o resultado correto (e esperado).

Como todos os casos de terminação (como demonstrado acima) são todos os casos possíveis, podemos afirmar que nosso algoritmo está correto.

4) Considere o problema de somar dois inteiros binários de n bits, armazenados em dois arranjos de n elementos A e B. A soma dos dois inteiros deve ser armazenada em forma binária em um arranjo de (n+1) elementos C. Escreva o pseudocódigo para somar os dois valores binários.

Solução:

Pseudocódigo

```
1: add(A,B):
2: carry, i ← 0
3: for i ← 1 to comprimento[A] do
4:     C[i] = (A[i] + B[i] + carry)%2
5:     carry = (A[i] + B[i] + carry)/2
6: end for
7: C[i] = carry
```

Código Implementado

```c
////////////////////////////////////////
//                                    //
// Autor: Diego S. Seabra             //
// Matricula: 0040251                 //
//                                    //
////////////////////////////////////////

#define n 3

#include <stdio.h>
#include <string.h>

void add(int A[n], int B[n], int C[n + 1])
{
    int carry, i = 0;
    for (i = 0; i < n; i++)
    {
        C[i] = (A[i] + B[i] + carry) % 2;  // resto
        carry = (A[i] + B[i] + carry) / 2; // quociente
    }
    C[i] = carry;
}

void imprime(int arr[n + 1])
{
    for (int i = 0; i < n + 1; i++)
    {
        printf("%i ", arr[i]);
    }
    printf("\n");
}

int main()
{
    int A[n] = {1, 1, 1};
    int B[n] = {1, 1, 1};
    int C[n + 1];

    // Inicia o array de resultado com 0's
    memset(C, 0, n + 1);

    add(A, B, C);
    imprime(C);
}
```

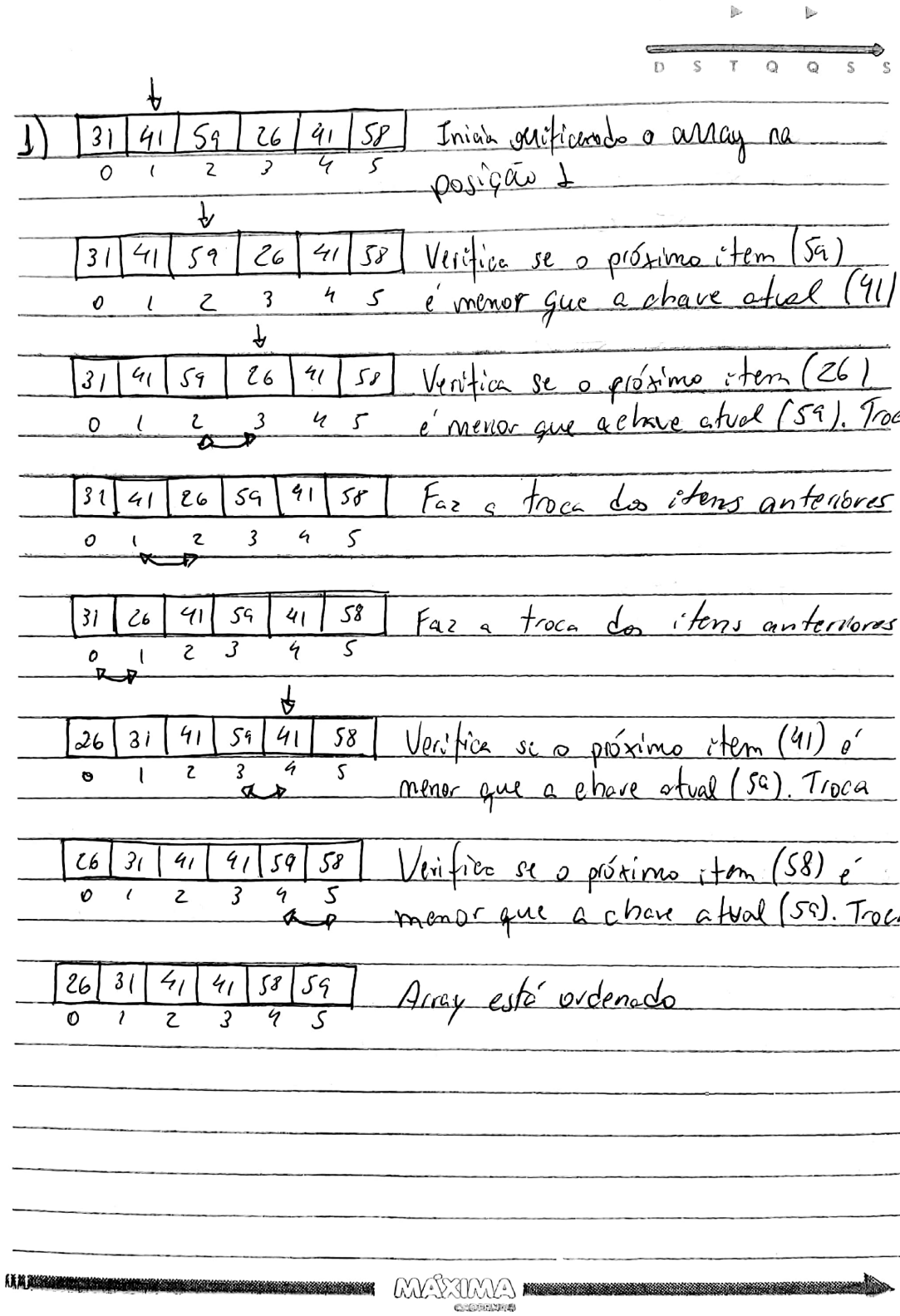
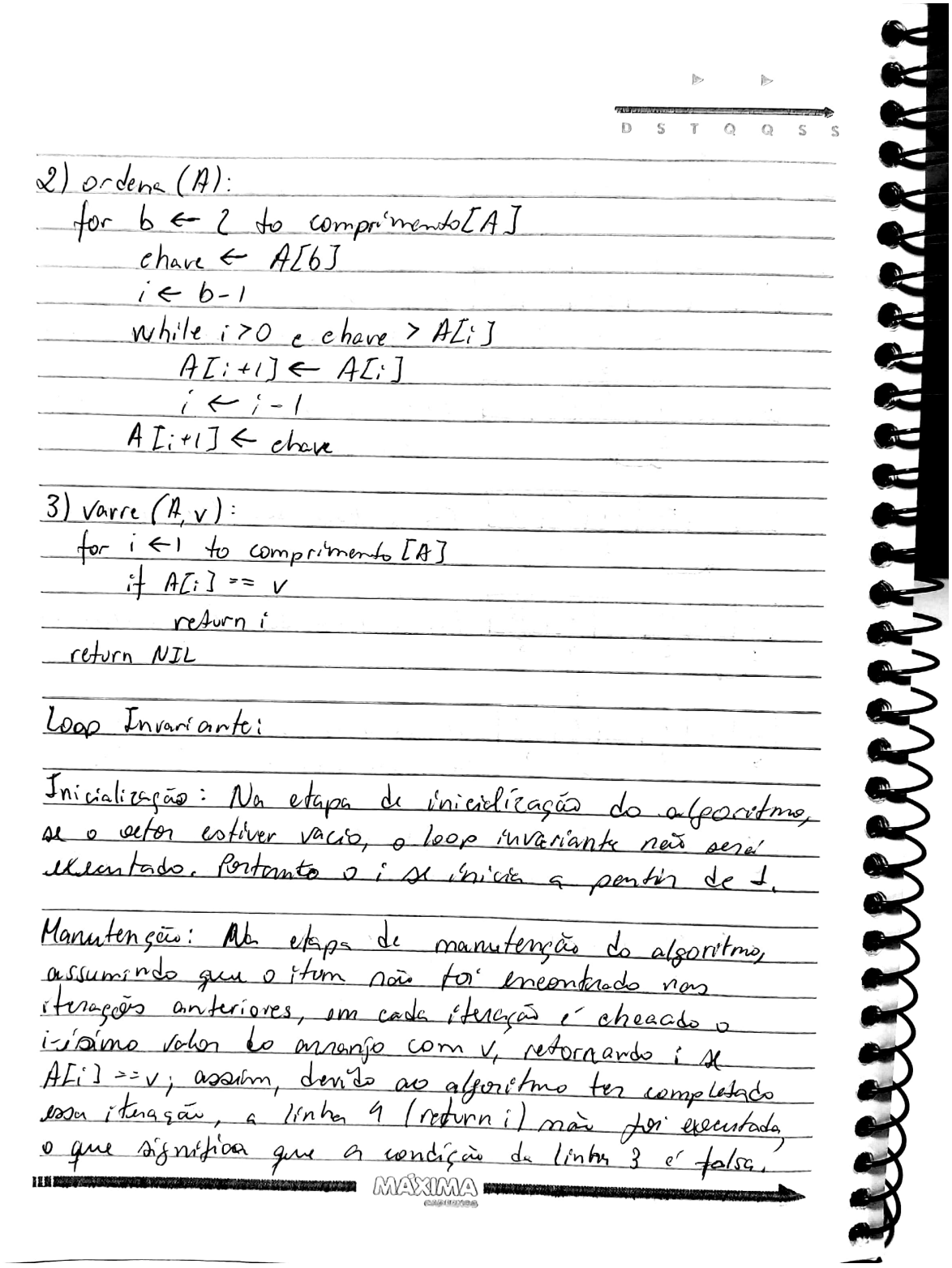


Fig. 2: Exercício 1



2) ordena(A):
for b ← 2 to comprimento[A]
chave ← A[b]
i ← b-1
while i>0 e chave > A[i]
A[i+1] ← A[i]
i ← i-1
A[i+1] ← chave

3) varre(A, v):
for i ← 1 to comprimento[A]
if A[i] == v
return i
return NIL

Loop Invariante:

Inicialização: Na etapa de inicialização do algoritmo, se o vetor estiver vazio, o loop invariante não será executado. Portanto o i se inicia a partir de 1.

Manutenção: Na etapa de manutenção do algoritmo, assumindo que o item não foi encontrado nas iterações anteriores, em cada iteração é checado o i-ésimo valor do arranjo com v, retornando i se A[i] == v; assim, dentro do algoritmo ter completado essa iteração, a linha 4 (return i) não foi executada, o que significa que a condição da linha 3 é falsa.

MÁXIMA

Fig. 3: Exercício 2 e 3

Portanto seu inverso é verdadeiro e A[i] ≠ v; a invariante se mantém.

Finalização: Na etapa de finalização, há duas maneiras do loop ser finalizado:

1. A primeira é quando i é maior que o tamanho do array, ou seja, i = comprimento[A] + 1. Como demonstrado na parte de manutenção, o algoritmo executará a linha depois do loop, retornando NIL, o que é o resultado correto (pois não foi encontrado um valor A[i] == v).

2. A segunda é a finalização na linha 4. Se isto ocorre, a condição da linha 3 era verdadeira (A[i] == v). Neste caso o algoritmo retorna i, que é o resultado correto (e esperado).

Como todos os casos de terminação (demonstrado acima) são todos os casos possíveis, podemos afirmar que nosso algoritmo está correto.

4) add(A, B):
carry, i ← 0
for i ← 1 to comprimento[A]
    C[i] = (A[i] + B[i] + carry) % 2
    carry = (A[i] + B[i] + carry) / 2
C[i] = carry

Fig. 4: Exercício 3 e 4

Diego Santos Seabra
0040251