Algoritmos e Estrutura de Dados II
(ESTCMP011)
2do PERIODO 2018

UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

Algoritmos de ordenação: outros.

Prof. Luis Cuevas Rodríguez, PhD
E-mail: lcuevasrodriguez@gmail.com / lrodriguez@uea.edu.br
Celular: 9298154648

EST
Escola Superior de Tecnologia da UEA

NUCOMP
NÚCLEO DE COMPUTAÇÃO

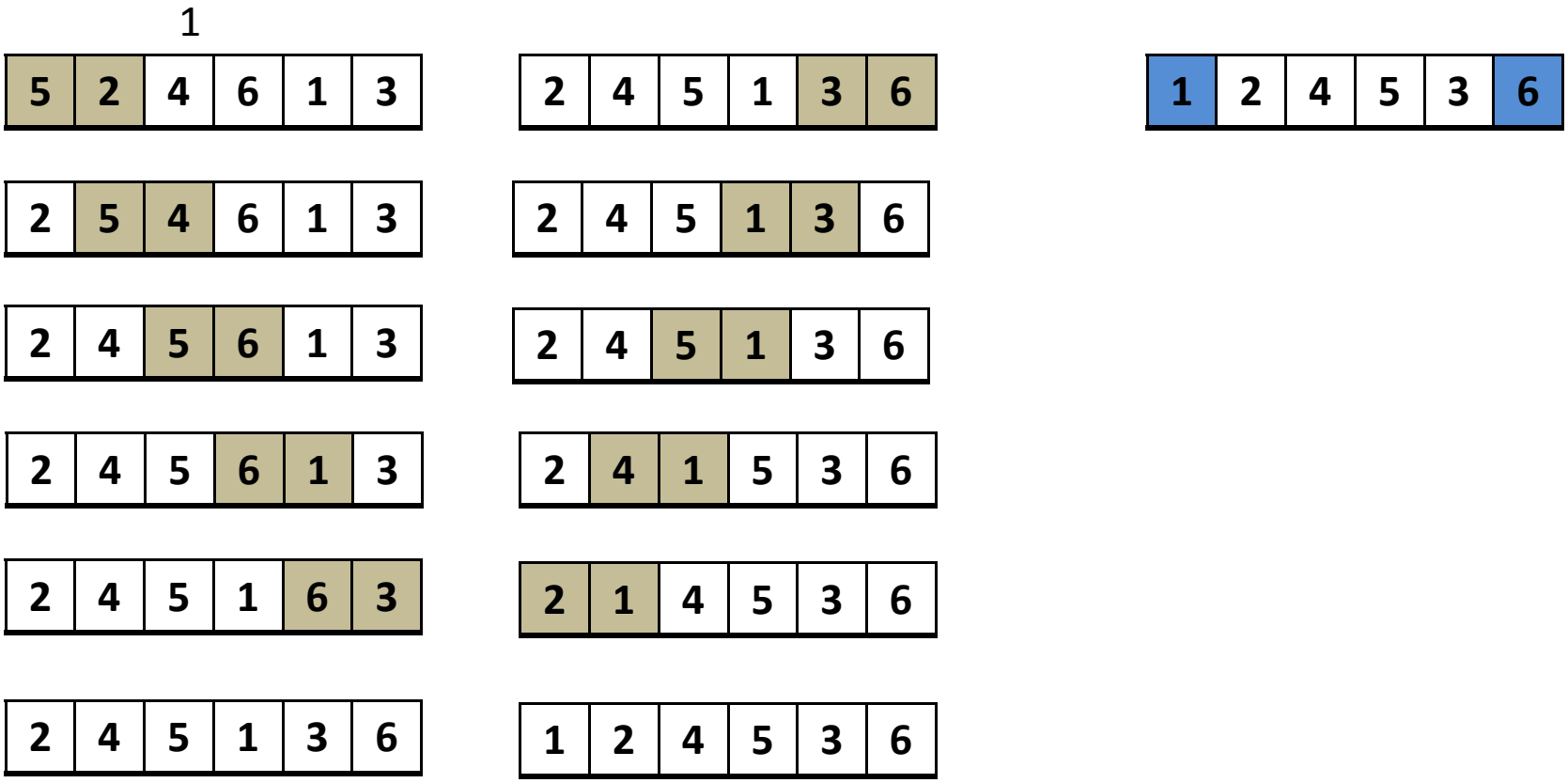
Conteúdo

- Especificado o algoritmo de ordenação por:
  - Cocktail sort
  - Counting sort (ordenação por contagem)
  - Bucket Sort (ordenação por baldes)
- Analisar seu tempo de execução.

Cocktail sort

- Shaker Sort, bubble sort bidirecional (que também pode se referir a uma variante do selection sort), ripple sort, shuttle sort ou happy hour sort.
- Variação do bubble sort.
  - Ordenar em ambas as direções em cada passagem através da lista.

Ordenação por flutuação (Cocktail sort)



Prof. Luis Cuevas Rodríguez, PhD

UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

Implementação

```
void cocktailsort(int a[], int n)
{
    bool troca = true;
    int temp;
    int inicio = 0;
    int fim = n - 1;

    while (troca) {
        troca = false;
        for (int i = inicio; i < fim; ++i) {
            if (a[i] > a[i + 1]) {
                temp=a[i];
                a[i]=a[i+1];
                a[i+1]=temp;
                troca = true;
            }
        }
        if (!troca)
            break;
        troca = false;
        fim--;
        for (int i = fim - 1; i >= inicio; --i) {
            if (a[i] > a[i + 1]) {
                temp=a[i];
                a[i]=a[i+1];
                a[i+1]=temp;
                troca = true;
            }
        }
        inicio++;
    }
```

Complexidade $\Theta(n^2)$

Exercícios

1. Ilustre a operação Cocktail sort sobre os vetores
   a) C=(27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4,8,9,0)
   b) A=(5, 3, 17, 10,84, 19,6,22,9)
   c) B=(5, 13, 2,25,7,17,20,8,4)

Counting sort

- Técnica baseada em chaves num intervalo específico.
- Contando o número de elmentos com valores-chave distintos. E usa uma aritmética para calcular a posição de cada elemento na seqüência de saída.
- Determinar, para cada entrada x, o número de elementos menor que x. Essa informação é usada para colocar o elemento x diretamente em sua posição no vetor de saída.

Counting sort

- Tem que ter definido um intervalo.
- Desvantagem :
  - Esta implementação tem a desvantagem de precisar de vetores auxiliares.
  - O Counting Sort ordena exclusivamente números inteiros pelo fato de seus valores servirem como índices no vetor de contagem.

Counting sort

```
//k é o maior valor do vetor A

//Criar vetor auxiliar com k+1 elementos e inicializar com zeros
for i ← 0 to k
    do C[i]←0

for j ← 1 to length[A]
    do C[A[j]] ← C[A[j]] + 1
//Agora C[i] contem o numero de elementos igual a i.

for i ← 1 to k
    do C[i] ← C[i] + C[i - 1]
//Agora C[i] contem o numero de elementos menor que ou igual a i.

for j ← length[A] downto 1
    do B[C[A[j]]] ← A[j]
        C[A[j]] ← C[A[j]] - 1

//Pseudocodigo do livro "Introduction to Algorithms"
//de Thomas H. Cormen...[et al.] - 2nd ed.
//The MIT Press (p. 168)
```

Counting sort

4	8	3	9	1	5	7

Intervalo: 10

Criar vetor auxiliar com k+1 elementos e inicializar com zeros

0	0	0	0	0	0	0	0	0	0	0

Contar os elementos

0	1	0	1	1	1	0	1	1	1	0

Contar numero de elementos menor que ou igual a i

0	1	1	2	3	4	4	5	6	7	7

Gerar saída

1	3	4	5	7	8	9

Prof. Luis Cuevas Rodríguez, PhD

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Counting sort

```
void countSort(int vetor[], int INTERV, int tamanho)
{
    char output[tamanho];
    int contar[INTERV + 1], i;
    for (i = 0; i <= INTERV; ++i)
        contar[i] = 0;
    for(i = 0; i < tamanho; i++)
        contar[vetor[i]]++;
    for (i = 1; i <= INTERV; ++i)
        contar[i] += contar[i-1];
    for (i = 0; i < tamanho; ++i)
    {
        output[contar[vetor[i]]-1] = vetor[i];
        contar[vetor[i]]--;
    }
    for (i = 0; vetor[i]; ++i)
        vetor[i] = output[i];
}
```

Complexidade $\Theta(n)$

Prof. Luis Cuevas Rodríguez, PhD

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

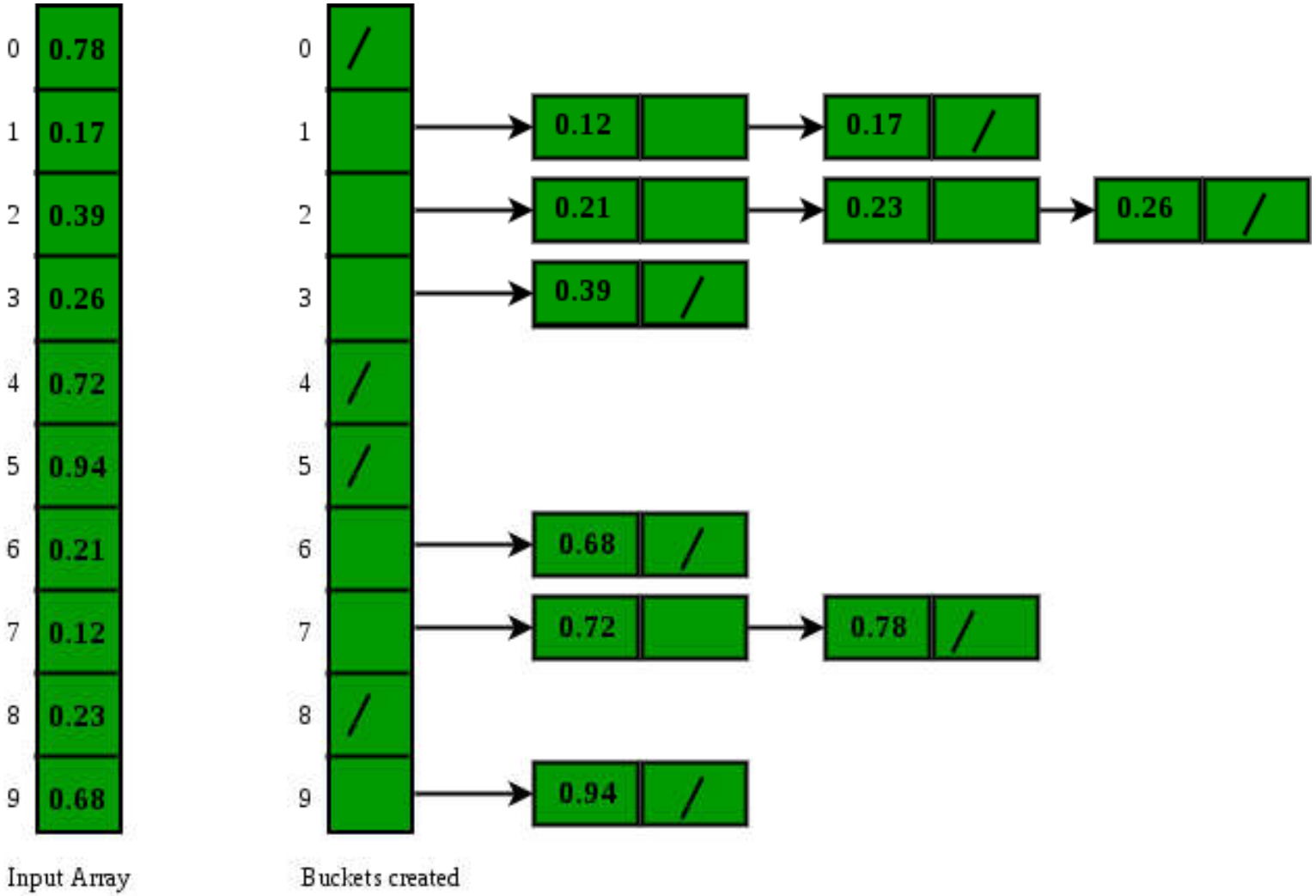
Exercícios

1. Ilustre a operação Counting sort sobre os vetores
   a) C=(27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4,8,9,0)
   b) A=(5, 3, 17, 10,84, 19,6,22,9)
   c) B=(5, 13, 2,25,7,17,20,8,4)

Bucket Sort

- Divide um vetor em um número finito de recipientes (baldes).
- Cada recipiente é então ordenado individualmente (seja usando um algoritmo de ordenação)
- Concatena os baldes que não estão vazios no vetor original.

Bucket Sort



UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Bucket Sort

- A classificação de bucket é principalmente útil quando:
  - as chaves são números de ponto flutuante
  - a entrada é distribuída uniformemente em um intervalo.