ESTRUTURA DE DADOS

AGENDA

- Visão geral
- Bubble sort
- Insertion sort
- Selection sort
- Merge sort
- Quick sort

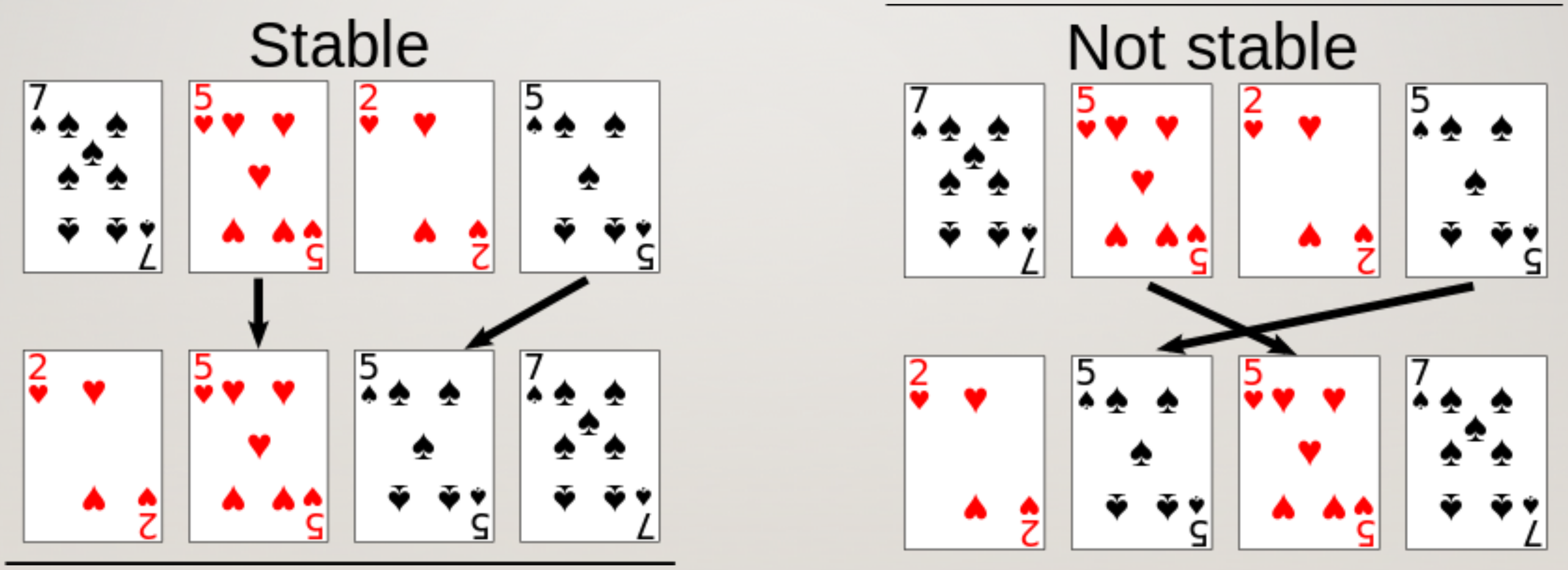
VISÃO GERAL

- Colocar um conjunto de dados em ordem
- Entrada: Conjunto de dados
- Saída: Dados em ordem
- Facilita a consulta dos dados

VISÃO GERAL

- Trabalha sobre as chaves de registros ou estruturas
- Chave de ordenação: qualquer chave com regra de ordenação bem definida

MÉTODOS ESTÁVEIS



IN PLACE

- “Rearranja os valores em um vetor A com no máximo um número constante desses valores sendo armazenados fora deste em um momento.”
- Usa uma quantidade constante de variáveis auxiliares para rearranjar os valores

BUBBLE SORT – T: $O(n^2)$ E: $O(n)$

```
BUBBLESORT(A)
for i = 1 to A.length − 1
    for j = A.length downto i + 1
        if A[j] < A[j − 1]
            exchange A[j] with A[j − 1]
```

INSERTION SORT - T: $O(n^2)$ E: $O(n)$

INSERTION-SORT(A)	*cost*	*times*
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
$i = j - 1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n-1$

SELECTION SORT - T: $O(n^2)$ E: $O(n)$

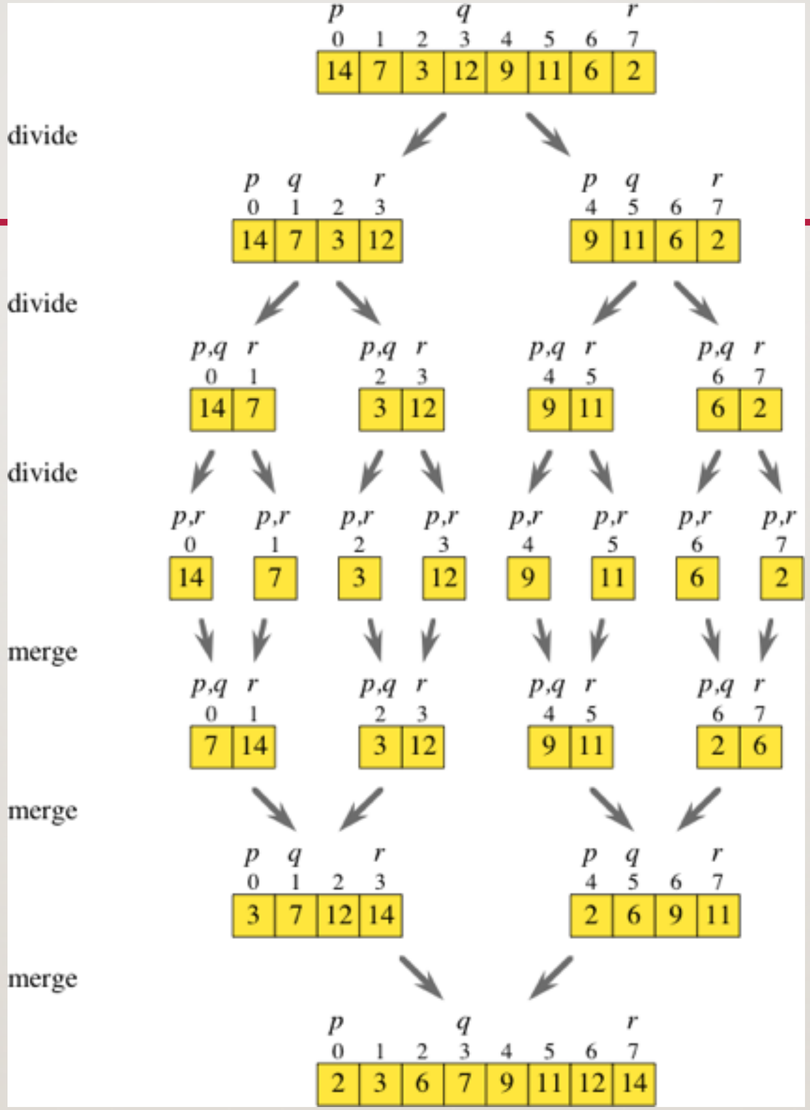
```
Algorithm SelectionSort
Inputs A: Array of Integers;
       N:Integer;
Variables i,j,min: Integer;

Begin
     for i:=0 to N-2 do
       min:=i;
       for j:=i+1 to N-1 do
         if (A[j]<A[min]) then min:=j fi
       od
       swap(A,i,min);
     od
End
```

MERGE SORT - T: $O(n \cdot logn)$ E: $O(n)$

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

```
MERGE(A, p, q, r)
n1 = q − p + 1
n2 = r − q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

MERGE SORT



https://pt.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/overview-of-merge-sort

QUICK SORT - T: $O(n^2)$ (MÉDIA $O(n \cdot logn)$) E: $O(n)$

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

QUICK SORT

- Laço de repetição pode ser implementado de forma eficiente na arquitetura
- Os dados podem ser preparados para evitar chegar a $O(n^2)$
- Tira vantagem da hierarquia de memória
- Pior caso: pivô sempre no extremo da lista
- Caso médio: pivô divide os dados em duas partes iguais

RESUMINDO

- http://bigocheatsheet.com/

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)