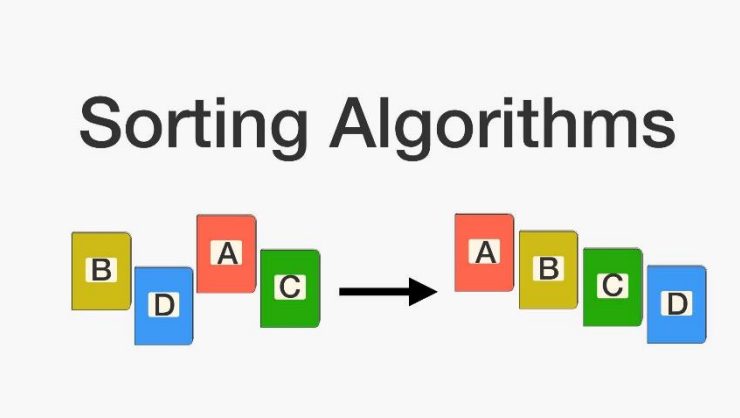
MAUÁ ESCOLA DE ENGENHARIA



Retirado de (https://embed-ssl.wistia.com/deliveries/70d6f4e10e2badb5ef394f00c17ad2bc1c14f6e7.jpg), em 18/08/2021

Quicksort
Busca Binária

Algoritmos de Ordenação

Algoritmos de Ordenação

- Existem diversos algoritmos de ordenação, onde cada um deles apresenta um conjunto de características específicas.
- A notação Big O é uma notação especial que diz o quão rápido um algoritmo pode ser executado.
- Cada algoritmo tem um tempo de execução diferente de acordo com a quantidade de dados que ele precisa ordenar.
- A notação Big O possibilita comparar o desempenho de algoritmos diferentes.
- Verificar: https://the-algorithms.com/

Algoritmos de Ordenação

Array Sorting Algorithms

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

Retirado de (https://camo.githubusercontent.com/903369da847e3a4b5078c8ce80bc9ce84801d2ace7346f4a0ab332be0781c653/687474703a2f2f626c6f672e62656e6f697476616c6c6f6e2e636f6d2f696d672f323031362d30332d31322d736f7274696e672d616c676f726974686d732d696e2d6a6176617363726970742f6269672d6f2e706e67), em 18/08/2021

Quicksort

Quick Sort

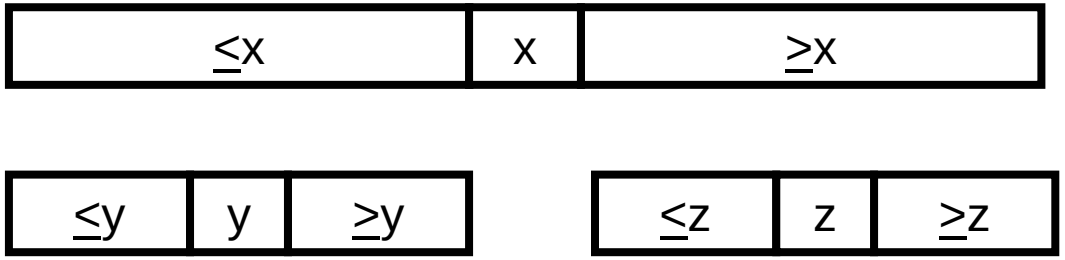
- Também conhecido como ordenação por partição;
- publicado por Hoare, em 1961;
- faz trocas entre elementos mais distantes entre si na lista;
- sequência reversa: apenas n/2 trocas;
- realiza trocas sobre um elemento (pivô) que particiona a lista de maneira conveniente.

Quick Sort

Escolher, de forma aleatória, um elemento **x** da sequência $v_0, v_1, \ldots, v_{n-1}$, de **n** elementos;

Percorrer sequência deixando os valores menores que **x** à sua esquerda e os maiores à sua direita;

Dessa maneira, teremos:



Repetir o processo de **partição** nas duas listas geradas, até que as novas listas possuam apenas um elemento;

A lista é dividida em listas menores e, depois, as listas menores vão aumentando, já ordenadas...

Quick Sort Recursivo

```
void quicksort(int *v, int l, int r) {
   int pivo, novoPivo;
   /* se a lista possui 2 ou mais elementos */
   if (l < r) {
      /* escolher o índice do meio como pivô */
      pivo = (l + r) / 2;
      novoPivo = particionar(v, l, r, pivo);
      /* ordenação recursiva dos subvetores */
      quicksort(v, l, novoPivo - 1);
      quicksort(v, novoPivo + 1, r);
   }
}
```

Quick Sort Recursivo

```
int particionar(int *v, int l, int r, int p) {
   int i, j;
   int valor_pivo = v[p];
   trocar(&v[p], &v[r]);
   j = l;
   for (i = l; i <= r - 1; i++) {
      if (v[i] <= valor_pivo) {
         trocar(&v[i], &v[j]);
         j++;
      }
   }
   trocar(&v[j], &v[r]);
   return j;
}
```

Busca Binária

O Problema da Busca

Dada uma sequência de elementos $<a_0, a_1, ..., a_{n-1}>$ e um elemento **x** fornecido, deseja-se localizar um elemento a_k tal que $x = a_k$.

Os elementos da sequencia $<a_0, a_1, ..., a_{n-1}>$ podem ser números, caracteres, cadeias de caracteres, registros na memória, registros de um arquivo, etc.

Busca Binária

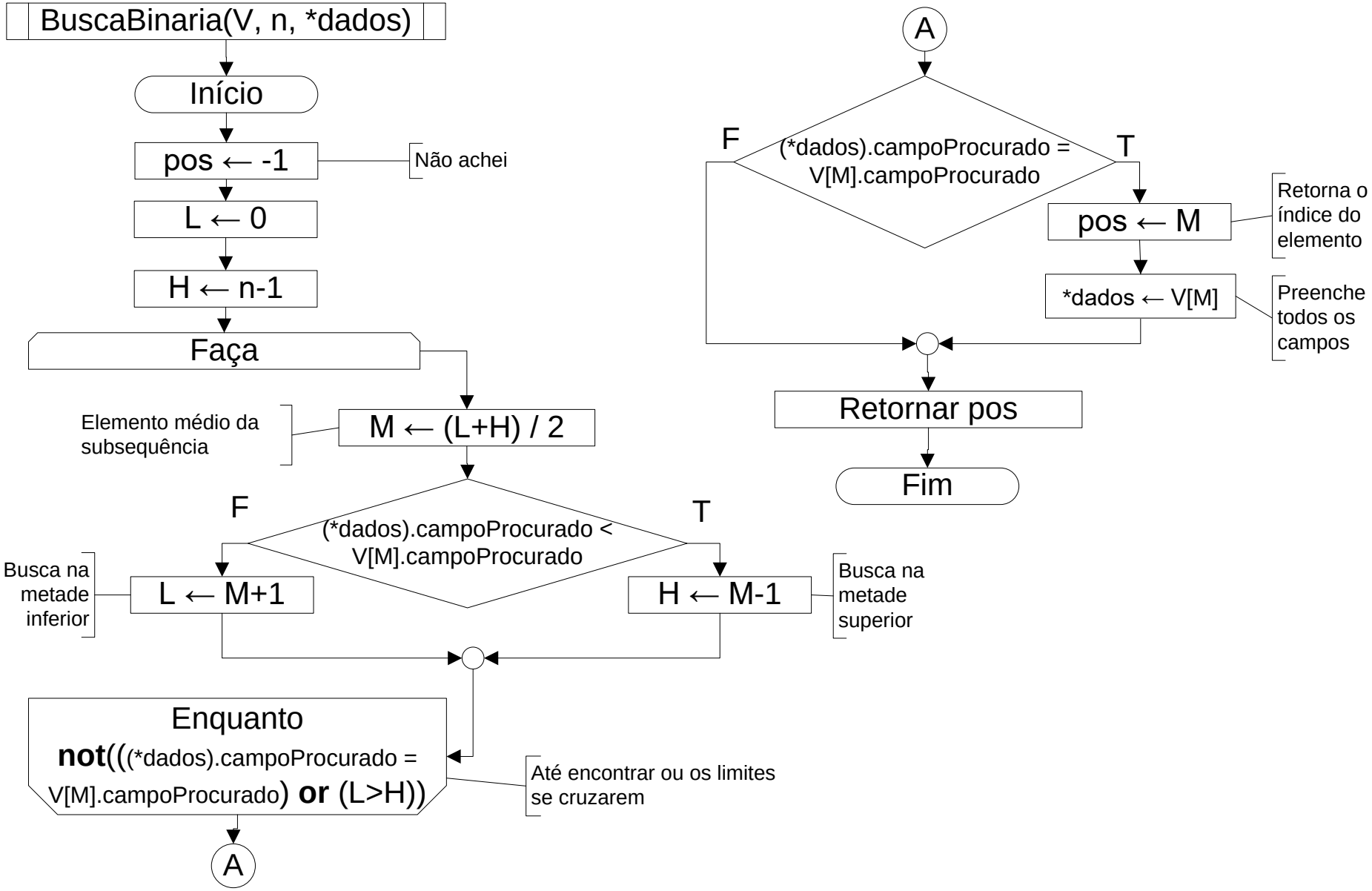
Baseia-se no princípio da bissecção.

A partir de uma sequência ORDENADA de elementos $<a_0, a_1, ..., a_{n-1}>$, verifica se o elemento procurado x é o valor a_k, sendo o elemento médio da sequência. Se $x = a_k$ encontramos o elemento.

Senão, se $x > a_k$, repete-se o passo anterior agora na subsequência $<a_{k+1}, a_{k+2}, ..., a_{n-1}>$, caso contrário na subsequência $<a_0, a_1, ..., a_{k-1}>$.

Repete-se este processo sucessivamente até que o elemento seja encontrado ou até que os extremos da subsequência se cruzem.

Busca Binária (iterativa)



Comparação de Strings

- Função para comparar strings:

```
int strcmp(char *, char *)
```

- Exemplo:
  - `strcmp("Goku", "Vegeta")` retorna <0
  - `strcmp("Gohan", "Gohan")` retorna 0
  - `strcmp("Vegeta", "Goku")` retorna >0