Algoritmos de Ordenação

Prof. Nelson Lima

Algoritmos de Ordenação

- São algoritmos que colocam os elementos de uma dada sequência em uma certa ordem (ascendente/descendente).
- As ordens mais usadas são a numérica e a lexicográfica (quando ordenamos palavras ou textos).

Algoritmos de Ordenação

- Os tipos de ordenação vão dos mais simples:
  - Bubble sort (Ordenação por trocas)
  - Selection sort (Ordenação por seleção)
  - Insertion sort (Ordenação por inserção)

  Aos mais sofisticados como:
  - Count sort
  - Quick sort
  - Merge sort
  - Heapsort ...
  - Shell sort ...
  - Radix sort ...
  - Bucket sort ...
  - Cocktail sort ...

  entre outros....

Bubble Sort

O algoritmo de ordenação Bubble Sort é um algoritmo simples, porém não muito eficiente em comparação com outros algoritmos de classificação. Ele funciona passando pela lista de itens várias vezes, comparando pares de itens adjacentes e os trocando se estiverem na ordem errada. O processo é repetido até que nenhuma troca seja necessária, o que significa que a lista está ordenada.

```
def bubble_sort(arr):
    n = len(arr)

    # Percorre todos os elementos do array
    for i in range(n):
        # Últimos i elementos já estão no lugar certo
        for j in range(0, n-i-1):
            # Troca se o elemento encontrado for maior que o próximo elemento
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

# Exemplo de uso
arr = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(arr)

print("Array ordenado:")
for i in range(len(arr)):
    print("%d" % arr[i], end=" ")
```

Selection Sort

Selection sort, *ou* ordenação por seleção, é um algoritmo de ordenação que procura passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o de segundo menor valor para a segunda posição, e assim é feito sucessivamente com os (n-1) elementos restantes, até os últimos dois elementos.

```
def selection_sort(arr):
    n = len(arr)

    # Percorre todos os elementos do array
    for i in range(n):
        # Encontra o índice do menor elemento restante
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j

        # Troca o menor elemento encontrado com o primeiro elemento não ordenado
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

# Exemplo de uso
arr = [64, 25, 12, 22, 11]
selection_sort(arr)

print("Array ordenado:")
for i in range(len(arr)):
    print("%d" % arr[i], end=" ")
```

Insertion Sort

Insertion sort, ou *ordenação por inserção*, é um algoritimo simples e eficiente quando aplicado a um pequeno número de elementos pouco desordenados.

Em termos gerais, ele percorre um vetor de elementos da esquerda para a direita e à medida que avança vai deixando os elementos mais à esquerda ordenados.

O algoritmo de inserção funciona da mesma maneira com que muitas pessoas ordenam cartas em um jogo de baralho como o pôquer.

```
def insertion_sort(arr):
    n = len(arr)

    # Percorre todos os elementos do array
    for i in range(1, n):
        key = arr[i]
        j = i - 1

        # Move os elementos de arr[0..i-1], que são maiores que key,
        # para uma posição à frente de sua posição atual
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

# Exemplo de uso
arr = [64, 25, 12, 22, 11]
insertion_sort(arr)

print("Array ordenado:")
for i in range(len(arr)):
    print("%d" % arr[i], end=" ")
```

Algoritmos de Ordenação