**INFORMATIKA**

**Algoritma Dasar dari sorting adalah proses mengurutkan elemen-elemen dalam suatu Kumpulan data sesuai dengan aturan tertentu. Beberapa algoritma dasar yang umum digunakan dalam sorting antara lain:**

1. **Bubble sort**

* ***Contoh kode:***

**def bubble\_sort(arr): n = len(arr)**

**for i in range(n): for j in range(0, n-i-1):**

**if arr[j] > arr[j+1]: arr[j], arr[j+1] = arr[j+1], arr[j]**

**# Contoh penggunaan:**

**arr = [64, 34, 25, 12, 22, 11, 90]bubble\_sort(arr)**

**print("Sorted array:", arr)**

1. **Insertion sort**

* ***Contoh kode:***

**def insertion\_sort(arr): for i in range(1, len(arr)):**

**key = arr[i] j = i - 1**

**while j >= 0 and key < arr[j]: arr[j + 1] = arr[j]**

**j -= 1 arr[j + 1] = key**

**# Contoh penggunaan:**

**arr = [64, 34, 25, 12, 22, 11, 90]insertion\_sort(arr)**

**print("Sorted array:", arr)**

1. **Selection sort**

* ***Contoh kode:***

**def selection\_sort(arr): n = len(arr)**

**for i in range(n): min\_index = i**

**for j in range(i+1, n): if arr[j] < arr[min\_index]:**

**min\_index = j arr[i], arr[min\_index] = arr[min\_index], arr[i]**

**# Contoh penggunaan:**

**arr = [64, 34, 25, 12, 22, 11, 90]selection\_sort(arr)**

**print("Sorted array:", arr)**

1. **Merge sort**

* ***Contoh kode:***

**def merge\_sort(arr): if len(arr) > 1:**

**mid = len(arr) // 2 L = arr[:mid]**

**R = arr[mid:]**

**merge\_sort(L) merge\_sort(R)**

**i = j = k = 0**

**while i < len(L) and j < len(R):**

**if L[i] < R[j]: arr[k] = L[i]**

**i += 1 else:**

**arr[k] = R[j] j += 1**

**k += 1**

**while i < len(L): arr[k] = L[i]**

**i += 1 k += 1**

**while j < len(R):**

**arr[k] = R[j] j += 1**

**k += 1**

**# Contoh penggunaan:arr = [64, 34, 25, 12, 22, 11, 90]**

**merge\_sort(arr)print("Sorted array:", arr)**

1. **Quick sort**

* ***Contoh kode:***

**def quick\_sort(arr): if len(arr) <= 1:**

**return arr else:**

**pivot = arr[0] less\_than\_pivot = [x for x in arr[1:] if x <= pivot]**

**greater\_than\_pivot = [x for x in arr[1:] if x > pivot] return quick\_sort(less\_than\_pivot) + [pivot] + quick\_sort(greater\_than\_pivot)**

**# Contoh penggunaan:**

**arr = [64, 34, 25, 12, 22, 11, 90]arr = quick\_sort(arr)**

**print("Sorted array:", arr)**

**Setiap algoritma memiliki kompleksitas waktu dan ruang yang berbeda, serta keunggulan dan kelemahan masing-masing tergantung pada jenis dan ukuran Kumpulan data yang diurutkan.**

**Berikut adalah Kekurangan sorting:**

1. **Waktu Eksekusi**
2. **Penggunaan memori**
3. **Stabilitas**
4. **Kasus terburuk**
5. **Ketergantungan terhadap jenis data**
6. **Kompabilitas dengan memori hierarkis**
7. **Kesulitan implementasi**

**Berikut adalah kelebihan sorting:**

1. **Memudahkan pencarian**
2. **Memfasilitasi analisis data**
3. **Membantu dalam visualisasi**
4. **Mengurangi kompleksitas algoritma lain**
5. **Memungkinkan proses paralel**