

# **LAPORAN PRAKTIKUM**

# **STRUKTUR DATA**

**MODUL KE-06**

**SORTING DALAM PYTHON**



**Disusun Oleh:**

**Nama** : Restu Wibisono

**NPM** : 2340506061

**Kelas** : 03 (Tiga)

**Program Studi S1 Teknologi Informasi**

**Fakultas Teknik, Universitas Tidar**

**Genap 2023/2024**

## **I.Tujuan Praktikum**

Adapun tujuan praktikum ini sebagai berikut :

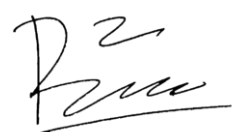
1. Mahasiswa mampu menerapkan algoritma Sorting pada bahasa pemrograman python2

## **II.Dasar Teori**

Mengurutkan data merupakan fondasi utama yang menjadi dasar bagi banyak algoritma lainnya. Hal ini terkait dengan beberapa konsep menarik yang akan Anda temui sepanjang perjalanan karier pemrograman Anda. Memahami bagaimana algoritma pengurutan bekerja di dalam Python adalah langkah awal yang penting untuk menerapkan algoritma dengan tepat dan efisien dalam menyelesaikan masalah dunia nyata. Pengurutan merupakan salah satu algoritma yang sering dipelajari dalam ilmu komputer. Terdapat banyak implementasi dan aplikasi berbeda dari pengurutan yang dapat meningkatkan efisiensi dan efektivitas kode Anda. Ada berbagai masalah yang dapat diselesaikan dengan pengurutan:

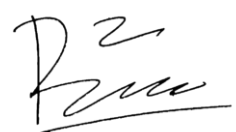
- Pencarian: Pencarian item dalam daftar menjadi lebih cepat jika daftar tersebut telah diurutkan.
- Seleksi: Memilih item dari daftar berdasarkan hubungannya dengan item lainnya menjadi lebih mudah dengan data yang diurutkan. Contohnya, mencari nilai k-th terbesar atau terkecil, atau mencari nilai median dari daftar, akan lebih mudah jika nilai nilai tersebut sudah dalam urutan menaik atau menurun.
- Penghapusan Duplikat: Menemukan nilai duplikat pada daftar menjadi lebih cepat ketika daftar sudah diurutkan.
- Analisis Distribusi: Menganalisis distribusi frekuensi item dalam suatu daftar akan lebih cepat jika daftar tersebut telah diurutkan. Contohnya, menemukan elemen yang paling sering atau paling jarang muncul menjadi lebih mudah dengan daftar yang telah diurutkan.

Tanda Tangan



Dalam bahasa Python, seperti banyak bahasa pemrograman tingkat tinggi lainnya, Anda dapat mengurutkan data secara langsung menggunakan fungsi `sorted()`.

Tanda Tangan

A stylized handwritten signature in black ink, consisting of a large capital 'P' followed by a series of loops and a long horizontal stroke at the bottom.

### III. Hasil dan Pembahasan

#### 1. Bubble Short

```
# Bubble Sort

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

    for i in range(n):
        # Buat tanda yang memungkinkan fungsi dihentikan lebih awal jika tidak ada
        # lagi yang perlu disortir
        already_sorted = True

        # Mulailah melihat setiap item dalam daftar satu per satu, bandingkan
        # dengan nilai yang berdekatan
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                # Jika item saat ini lebih besar dari item berikutnya, tukar
                # posisi mereka
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                # Karena Anda harus menukar dua elemen, setel tanda
                # 'already_sorted' ke 'False', agar algoritma tidak selesai sebelum
                # waktunya
                already_sorted = False

        # Jika tidak ada swap selama iterasi terakhir, array sudah diurutkan, dan
        # Anda dapat mengakhirkannya
        if already_sorted:
            break

    return arr

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

print("Array sebelum diurutkan: ", arr)
bubble_sort(arr)
print("Array setelah diurutkan: ", arr)
```

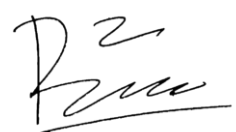
Python

Array sebelum diurutkan: [64, 34, 25, 12, 22, 11, 90]  
Array setelah diurutkan: [11, 12, 22, 25, 34, 64, 90]

(Gambar 3.1)

- Untuk mendklarasikan parameter 'bubble\_short' bisa menggunakan fungsi 'def bubble\_short', yaitu parameter array yang akan diurutkan.
- Menghitung panjang 'n = len(arr)' array yang diberikan dan akan menyimpan dalam variabel 'n'.
- Selanjutnya mulai iterasi 'for i in range(n)' untuk setiap elemen dalam array, 'i' akan berjalan mulai dari 0 hingga 'n-1'.
- Instalasi variabel 'already\_sorted' menjadi 'True', yang berfungsi untuk menandai apa array sudah di urutkan.

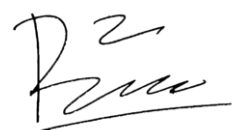
Tanda Tangan



- ‘for j in range(n – j – 1):’ adalah loop kekuda yang berfungsi untuk memeriksa setiap pasang elemen dalam array.
- Memeriksa apakah elemen saat ini lebih besar elemen dari berikutnya digunakan ‘if arr[j] > arr [j + 1].
- Jika ‘arr[j], arr[j + 1] = arr [j + 1], arr[j]’ ya, maka akan di lakukan pertukaran antar posisi antara elemen sekarang dan elemen selanjutnya.
- Saat pertukaran maka variabel ‘already\_sorted’ akan diatur menjadi ‘False’, yang berfungsi untuk menunjukkan array belum diurutkan semuanya.
- Setelah iterasi penuh atas array ‘if already\_sorted: break’ saat tidak ada pertukaran atau array sudah diurutkan, iterasi akan dihentikan dengan ‘break’.
- Untuk mengembalikan array yang telah diurutkan menggunakan ‘return arr’
- ‘arr = [64, 34, 25, 12, 22, 11, 90]’ Inisialisasi sebuah array yang akan diurutkan.
- Menggunakan fungsi ‘print("Array sebelum diurutkan: ", arr)’ mencetak array sebelum diurutkan.
- Memanggil fungsi ‘bubble\_sort’ untuk mengurutkan array.
- Terakhir menggunakan ‘print("Array setelah diurutkan: ", arr)’ untuk mencetak array setelah diurutkan.

## 2. Insertion Sort

Tanda Tangan



```

# Insertion Sort

def insertion_sort(arr):
    # Ulangi dari elemen kedua array hingga elemen terakhir
    for i in range(1, len(arr)):
        # Ini adalah elemen yang ingin kita posisikan pada tempat yang benar
        key_item = arr[i]

        # Inisialisasi variabel yang akan digunakan untuk menemukan posisi yang
        # benar dari elemen yang direfreshkan oleh 'key_item'
        j = i - 1

        # Jalankan daftar item (bagian kiri array) dan tentukan posisi yang benar
        # dari elemen yang direfreshkan oleh 'key_item' Lakukan ini hanya jika
        # 'key_item' lebih kecil dari nilai di dekatnya
        while j >= 0 and arr[j] > key_item:
            # Geser nilai satu posisi ke kiri dan ubah posisi j untuk menunjuk ke
            # elemen berikutnya (dari kanan ke kiri)
            arr[j + 1] = arr[j]
            j -= 1

        # Setelah selesai menggeser elemen, Anda dapat memposisikan 'key_item' di
        # lokasi yang benar
        arr[j + 1] = key_item

    return arr

# Contoh penggunaan
arr = [64, 34, 25, 12, 22, 11, 90]
print("Array sebelum diurutkan: ", arr)
insertion_sort(arr)
print("Array setelah diurutkan: ", arr)

```

✓ 0.0s Python

Array sebelum diurutkan: [64, 34, 25, 12, 22, 11, 90]  
 Array setelah diurutkan: [11, 12, 22, 25, 34, 64, 90]

(Gambar 3.2)

- Untuk mendklarasikan parameter 'insertion\_short' bisa menggunakan fungsi 'def insertion\_short(arr)', yaitu parameter array yang akan diurutkan.
- Memulai iterasi dari indeks kedua array hingga indeks terakhir menggunakan fungsi 'for i in range(1, len(arr)):', yang akan membandingkan setiap elemen dan memasukkan elemen ke dalam posisi benar.
- Selanjutnya mengambil elemen dengan 'key\_item = arr[i]' yang akan diposisikan di tempat dan menyimpan dalam variabel 'key\_item'.
- Inisialisasi variabel 'j' akan berfungsi untuk mencari posisi yang tepat untuk 'key\_item'.

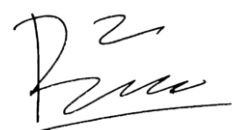
Tanda Tangan



- Loop while ‘while j >= 0 and arr[j] > key\_item:’ untuk mencari posisi yang tepat. Loop akan berjalan jika ‘j’ masih dalam rentang array dan selama elemen ‘j’ lebih besar dari ‘key\_item’.
- Menggeser elemen ke kanan dengan ‘arr[j + 1] = arr[j]’ jika elemen lebih besar dari ‘key\_item’.
- ‘j -= 1’ akan nilai ‘j’ untuk memeriksa elemen sebelumnya.
- Setelah menemukan posisi yang tepat dengan ‘arr[j + 1] = key\_item’, program akan memasukkan ‘key\_item’ ke posisi yang benar dalam array.
- ‘return arr’ berfungsi untuk array yang telah diurutkan.
- Inisialisasi sebuah array yang akan diurutkan ‘arr = [64, 34, 25, 12, 22, 11, 90]’.
- Menggunakan fungsi ‘print("Array sebelum diurutkan: ", arr)’  
Mencetak array sebelum diurutkan.
- Memanggil fungsi ‘insertion\_sort’ untuk mengurutkan array.
- Menggunakan fungsi ‘print("Array setelah diurutkan: ", arr)’  
Mencetak array setelah diurutkan.

### 3. Merge Sort

Tanda Tangan



```

# Merge Sort

def merge(left, right):
    # Jika array pertama kosong, maka tidak ada yang perlu digabungkan, dan Anda dapat mengembalikan
    # array kedua sebagai hasilnya
    if len(left) == 0:
        return right

    # Jika array kedua kosong, maka tidak ada yang perlu digabungkan, dan Anda dapat mengembalikan
    # array pertama sebagai hasilnya
    if len(right) == 0:
        return left

    result = []
    index_left = index_right = 0
    # Sekarang telusuri kedua array hingga semua elemen berhasil masuk ke dalam array yang dihasilkan
    while len(result) < len(left) + len(right):
        # Elemen perlu diurutkan untuk menambahkannya ke array yang dihasilkan, jadi Anda perlu
        # memutuskan apakah akan mendapatkan elemen berikutnya dari array pertama atau kedua
        if left[index_left] <= right[index_right]:
            result.append(left[index_left])
            index_left += 1
        else:
            result.append(right[index_right])
            index_right += 1
        # Jika Anda mencapai akhir dari salah satu array, maka Anda dapat menambahkan elemen yang
        # tersisa dari array lain ke hasil dan memutus perulangan
        if index_right == len(right):
            result.extend(left[index_left:])
            break
        if index_left == len(left):
            result.extend(right[index_right:])
            break
    return result

def merge_sort(array):
    # Jika array masukan berisi kurang dari dua elemen, kembalikan sebagai hasil fungsinya
    if len(array) < 2:
        return array
    midpoint = len(array) // 2
    # Urutkan array dengan membagi input secara rekursif menjadi dua bagian yang sama, mengurutkan
    # masing-masing bagian dan menggabungkannya menjadi hasil akhir
    return merge(
        merge_sort(array[:midpoint]),
        merge_sort(array[midpoint:]))

# Contoh penggunaan
arr = [64, 34, 25, 12, 22, 11, 90]
print("Array sebelum pengurutan:", arr)
arr = merge_sort(arr)
print("Array setelah pengurutan:", arr)

```

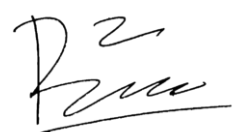
Python

Array sebelum pengurutan: [64, 34, 25, 12, 22, 11, 90]  
 Array setelah pengurutan: [11, 12, 22, 25, 34, 64, 90]

(Gambar 3.3)

- Deklarasi dari sebuah fungsi bernama ‘merge’ dengan dungsi ‘def merge(left, right):’ yang menerima dua parameter, yaitu dua array yang akan digabungkan dan diurutkan.
- ‘if len(left) == 0:’ dan ‘if len(right) == 0:’ akan mengecek jika salah satu array ada ingin di gabungkan kosong, fungsi akan langsung mengembalikan array lainnya, karena tidak perlu dilakukan penggabungan.
- Mendefinisikan array kosong dengan fungsi ‘result = []’ dan ‘index\_left = index\_right = 0’ dimana ‘result’ yang akan menyimpan

Tanda Tangan

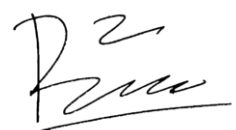




hasil dari penggabungan, serta dua variabel indeks 'index\_left' dan 'index\_right' yang berfungsi menandai posisi saat ini dalam masing-masing array.

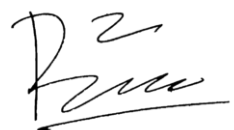
- 'while len(result) < len(left) + len(right):' Adalah loop while yang berfungsi menjalankan sampai semua elemen dari kedua array di gabungkan ke array hasil.
- Dalam loop while, program akan membandingkan elemen-elemen di 'left[index\_left]' dan 'right[index\_right]', lalu menambahkan elemen yang lebih kecil di dalam array 'result', serta menggeser indeks ke elemen berikutnya di array yang sesuai.
- Saat salah satu array telah mencapai akhirnya program akan menjalankan ('index\_right == len(right)' atau 'index\_left == len(left)'), dimana program akan menambahkan sisa dari elemen array lainnya ke 'result' lalu menghentikan loop.
- Mengembalikan array 'result' yang telah diurutkan.
- Deklarasi sebuah fungsi bernama 'merge\_sort' yang akan menerima satu parameter, yaitu array yang diurutkan menggunakan algoritma Merge Sort.
- Memeriksa jika panjang array kurang dari 2 dengan fungsi 'if len(array) < 2:', maka array tersebut sudah terurut (atau kosong).
- 'midpoint = len(array) // 2' Menghitung titik tengah array.
- Memanggil fungsi 'merge' untuk menggabungkan serta mengurutkan dari dua bagian dari array yang mana telah dibagi menjadi dua secara rekursif.
- Inisialisasi sebuah array yang akan diurutkan 'arr = [64, 34, 25, 12, 22, 11, 90]'.
- Menggunakan fungsi 'print("Array sebelum pengurutan:", arr)' Mencetak array sebelum diurutkan.
  - Memanggil fungsi 'merge\_sort' untuk mengurutkan array.

Tanda Tangan



- Menggunakan fungsi `'print("Array setelah pengurutan:", arr)'`  
Mencetak array setelah diurutkan.

Tanda Tangan

A handwritten signature in black ink, consisting of a stylized 'D' followed by a series of loops and a long horizontal stroke.

#### IV.Latihan

```
# Latihan Soal

def partition(array, low, high):
    pivot = array[high]
    i = low - 1
    for j in range(low, high):
        if array[j] <= pivot:
            i = i + 1
            array[i], array[j] = array[j], array[i]

    array[i + 1], array[high] = array[high], array[i + 1]
    return i + 1

def quick_sort(array, low, high):
    if low < high:
        pi = partition(array, low, high)
        quick_sort(array, low, pi - 1)
        quick_sort(array, pi + 1, high)

arr = [64, 34, 25, 12, 22, 11, 90]
n = len(arr)
print("Array sebelum pengurutan:", arr)
quick_sort(arr, 0, n - 1)
print("Array setelah pengurutan:", arr)
```

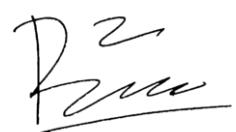
✓ 0.0s Python

Array sebelum pengurutan: [64, 34, 25, 12, 22, 11, 90]  
Array setelah pengurutan: [11, 12, 22, 25, 34, 64, 90]

(Gambar 4.1)

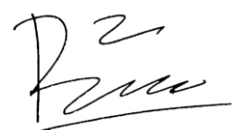
- ‘def partisi(arr, low, high):’ Mendefinisikan fungsi partisi untuk membagi array menjadi dua bagian (partisi) berdasarkan nilai pivot.
- Menginisialisasi ‘i = (low-1)’ dengan variabel i sebagai indeks untuk elemen terakhir dari partisi yang lebih kecil dari pivot.
- Memilih pivot dari array ‘pivot = arr[high]’, yang dalam kasus ini diambil sebagai elemen terakhir (arr[high]).
- Memulai iterasi dengan fungsi ‘for j in range(low, high):’ untuk setiap elemen dari low hingga high.
- Memeriksa ‘if arr[j] <= pivot:’ apakah nilai elemen saat ini kurang dari atau sama dengan pivot.
- Jika ‘i = i+1 dan arr[i], arr[j] = arr[j], arr[i]’ dalam kondisi terpenuhi, maka nilai i ditingkatkan dan dilakukan pertukaran nilai antara elemen ke-i dan elemen saat ini (arr[j]).
- Setelah selesai iterasi, dilakukan pertukaran pivot ke posisi yang benar dalam array.
- Mendefinisikan fungsi ‘def quickSort(arr, low, high)’ untuk melakukan sorting menggunakan algoritma Quick Sort.

Tanda Tangan



- Memeriksa jika panjang array 'if len(arr) == 1' arr hanya 1, maka array tersebut dianggap sudah terurut dan langsung dikembalikan.
- Memanggil fungsi partisi 'pi = partisi(arr, low, high)' untuk membagi array dalam dua bagian dan mendapatkan indeks pivot (pi).
- 'quickSort(arr, low, pi-1) dan quickSort(arr, pi+1, high)' Memanggil rekursif fungsi quickSort untuk melakukan sorting pada dua bagian array yang lebih kecil dari pivot dan dua bagian array yang lebih besar dari pivot.
- Mendefinisikan sebuah array dengan nama arr yang akan diurutkan menggunakan fungsi quicksort 'arr = [64, 34, 25, 12, 22, 11, 90]'.
- 'n = len(arr)' Menghitung panjang array arr.
- Memanggil fungsi 'quickSort(arr, 0, n-1)' untuk mengurutkan array arr mulai dari indeks 0 hingga indeks terakhir (n-1).
- Mencetak fungsi 'print("Array yang telah diurutkan adalah:", arr)' setelah proses pengurutan selesai.
- Outputnya adalah '[11, 12, 22, 25, 34, 64, 90]' Adalah array setelah di urutkan menggunakan algoritma Quick Sort.

Tanda Tangan

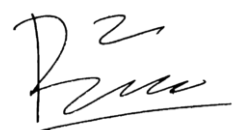


## V. Kesimpulan

Dalam praktik pemrograman di atas, kami menggunakan beberapa algoritma pengurutan yang umum digunakan, seperti Bubble Sort, Insertion Sort, Merge Sort, dan Quick Sort. Kami melakukan ini untuk mempelajari cara kerja masing-masing algoritma dan untuk membandingkan efisiensi dan kecepatan mereka dalam mengurutkan array.

Bubble Sort adalah algoritma yang sederhana tetapi memiliki kompleksitas waktu rata-rata  $O(n^2)$ , di mana  $n$  adalah jumlah elemen. Quick Sort memilih sebuah elemen sebagai pivot, membagi array menjadi dua bagian secara rekursif, mengurutkan masing-masing bagian, dan menggabungkannya kembali dengan kompleksitas waktu  $O(n \log n)$ . Sebaliknya, Merge Sort membagi array menjadi dua bagian, mengurutkan masing-masing bagian, dan menggabungkannya kembali dengan kompleksitas waktu  $O(n \log n)$ .

Tanda Tangan

A handwritten signature in black ink, consisting of a stylized 'P' followed by a series of loops and a horizontal stroke at the end.