

<p>Nama: Dimas Alip Priyono</p> <p>NIM: 064102400032</p>	 <p>Praktikum Algoritma & Pemrograman</p>	<h1>MODUL 10</h1> <p>Nama Dosen: Binti solihah, S.T, M.KOM</p> <p>Nama Asisten Labratorium:</p> <ol style="list-style-type: none">1. Yustianas Rombon - 0640023000152. Vira Aditya Kurniawan - 065002300012
<p>Hari/Tanggal: Jumat, 29 November 2024</p>		

Search, List & Sorting

1. Teori Singkat

Linear Search

Linear Search adalah sebuah algoritma pencarian, juga dikenal sebagai pencarian sekuensial, yang cocok untuk mencari sebuah nilai tertentu pada sebuah himpunan data. Algoritma ini beroperasi dengan memeriksa setiap elemen dari sebuah list sampai sebuah kecocokan ditemukan.

Binary Search

Binary Search atau sering disebut algoritma pencarian biner adalah sebuah teknik untuk menemukan nilai tertentu dalam sebuah larik linear, dengan menghilangkan setengah data pada setiap langkah, dipakai secara luas tetapi tidak secara ekslusif dalam ilmu komputer. Pada saat menggunakan binary search, data yang berada di dalam array harus diurutkan terlebih dahulu.

List



List adalah tipe data yang paling serbaguna yang tersedia dalam bahasa Python, yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya.

Sorting

Sorting merupakan suatu proses untuk menyusun kembali humpunan obyek menggunakan aturan tertentu. Sorting disebut juga sebagai suatu algoritma untuk meletakkan kumpulan elemen data kedalam urutan tertentu berdasarkan satu atau beberapa kunci dalam tiap-tiap elemen.

Metode-metode sorting meliputi:

1. Insertion Sort (Metode Penyisipan)
2. Selection Sort (Metode Seleksi)
3. Bubble sort(Metode Gelembung)
4. Shell Sort (Metode Shell)
5. Quick Sort (Metode Quick)
6. Merge Sort (Metode Penggabungan)

Contoh pembuatan list



```
● ● ●  
list1 = ['kimia', 'fisika', 1993, 2017]  
list2 = [1, 2, 3, 4, 5]  
list3 = ["a", "b", "c", "d"]
```

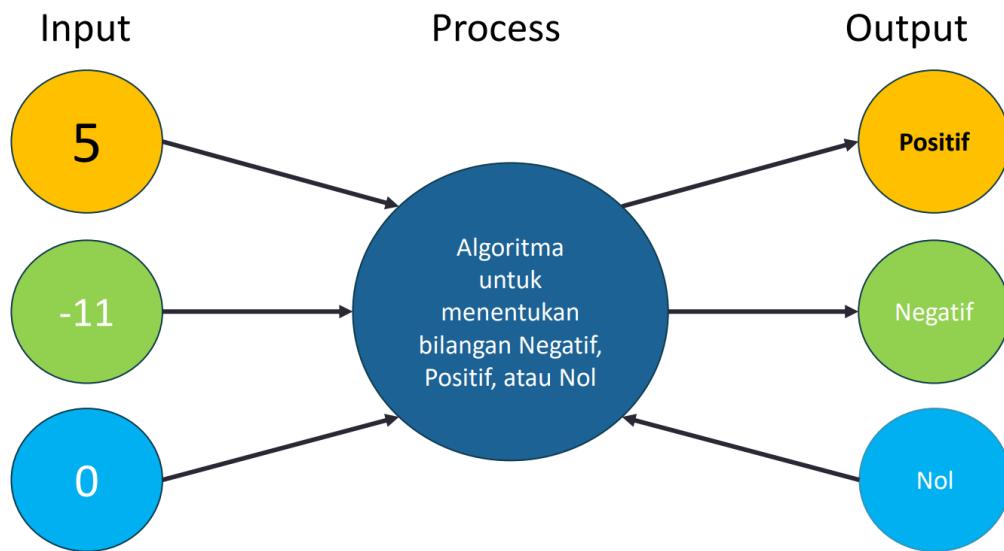
IPO (Input Process Output)

Konsep Dasar Input, Process, dan Output (IPO)

- Konsep input, process, dan output adalah prinsip dasar dalam pemrograman dan pengembangan algoritma.
- Setiap algoritma melibatkan tiga tahap utama: mengambil data masukan (input), melakukan operasi atau pengolahan data (process), dan menghasilkan hasil akhir (output).
- Konsep ini menggambarkan bagaimana algoritma beroperasi untuk memproses informasi.



Gambaran IPO (Menentukan Bilangan)



Notasi Algoritma Flowchart

1. Flowchart adalah representasi visual atau diagram alir yang digunakan untuk menggambarkan langkah-langkah dan urutan proses suatu algoritma atau program.
2. Flowchart menyajikan langkah-langkah dalam bentuk simbol-simbol grafis yang saling terhubung, membantu dalam memvisualisasikan bagaimana informasi mengalir dan bagaimana proses dilakukan.
3. Dalam kaitannya dengan notasi deskriptif, notasi algoritma yang menggunakan flowchart dapat lebih cepat dibaca dan dilihat alur dan hubungannya.

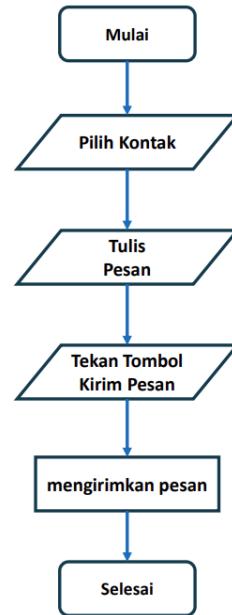
Simbol-simbol pada Flowchart

1. Setiap elemen flowchart dihubungkan oleh garis aliran bertanda panah
2. Garis aliran dimulai dari atas symbol dan keluar dari bagian bawah, kecuali symbol keputusan yang alirannya keluar dari bawah atau samping
3. Aliran bergerak dari atas ke bawah
4. Proses awal dan akhir menggunakan symbol terminal.



**Contoh sederhana
Penggunaan *flowchart*
untuk menunjukan algoritma**

**Kasus/Aliran:
Mengirim pesan WhatsApp**



2. Alat dan Bahan

Hardware : Laptop/PC

Software : Spyder (Anaconda Python)

3. Elemen Kompetensi

a. Latihan pertama

Buatlah sebuah fungsi binary search untuk mencari sebuah element didalam sebuah list tersebut yang dimana, jika list tersebut acak maka diurutkan terlebih dahulu dengan menggunakan fungsi sorting (implementasi bebas, boleh menggunakan bubblesort, dll) dan setelahnya baru dicari menggunakan fungsi binary search.

IPO (input process output)



Input

1. **Array awal:**
 - o arr - Berupa daftar angka untuk diurutkan. Contoh: [64, 34, 25, 12, 22, 11, 90].
2. **Angka yang dicari:**
 - o number_to_find - Angka dari input pengguna untuk dicari dalam array yang telah diurutkan.

Process

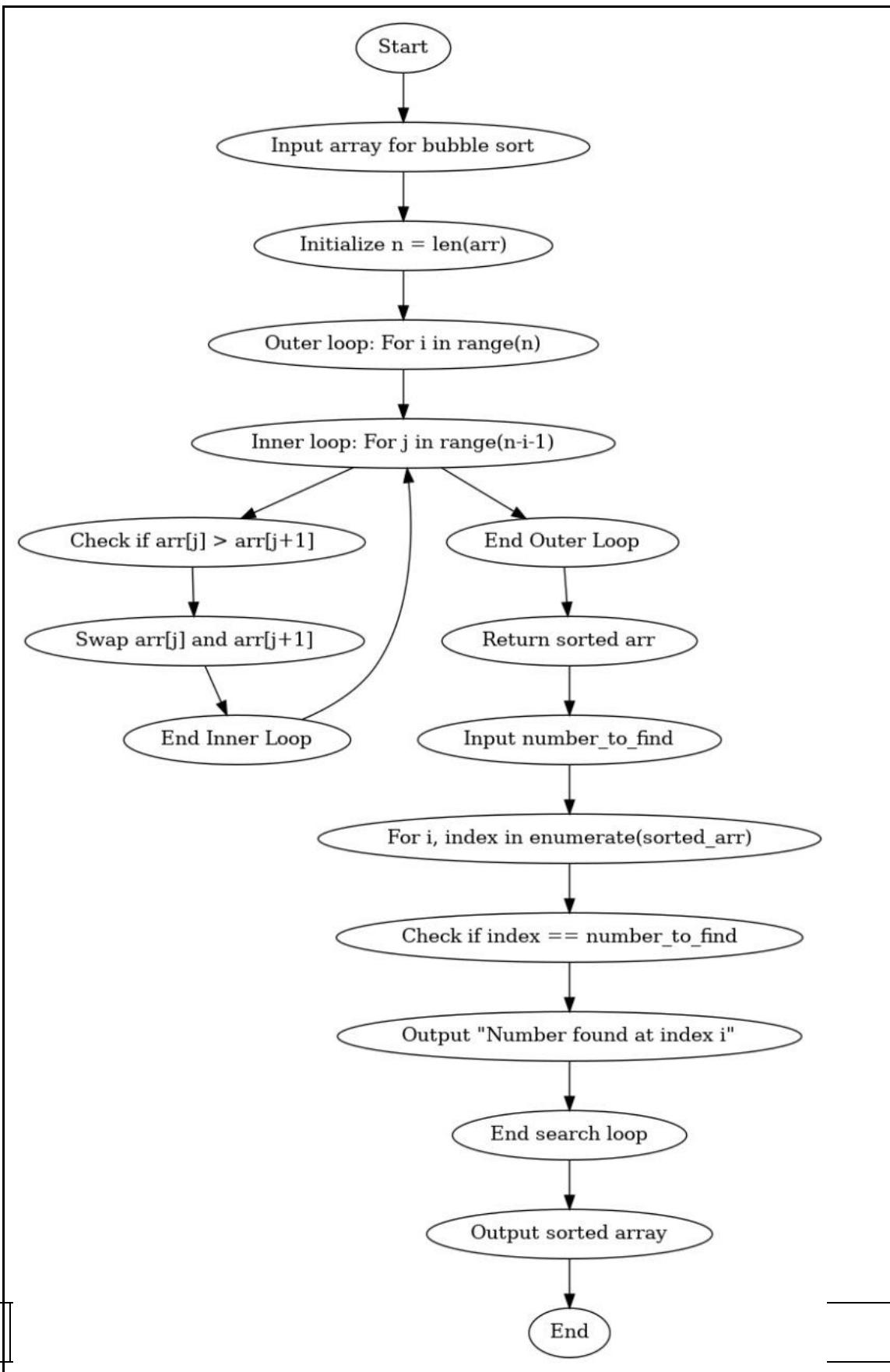
1. **Proses Pengurutan Bubble Sort:**
 - o Dilakukan iterasi ganda melalui array, dengan perbandingan antar elemen yang berdekatan.
 - o Jika elemen lebih besar dibandingkan elemen berikutnya, mereka ditukar hingga array tersortir dalam urutan menaik.
2. **Proses Pencarian Linear:**
 - o Melakukan pencarian angka dengan memeriksa setiap elemen dalam array yang telah diurutkan.
 - o Jika elemen yang cocok ditemukan, program mencatat indeks elemen tersebut.

Output

1. **Array yang telah diurutkan:**
 - o Hasil akhir dari algoritma Bubble Sort.
 - o Contoh: [11, 12, 22, 25, 34, 64, 90].
2. **Hasil pencarian angka:**
 - o Jika angka ditemukan, ditampilkan indeksnya.
 - o Contoh: Angka 22 ditemukan pada indeks ke-2.
 - o Jika tidak ditemukan, tidak ada output terkait pencarian (bisa ditambahkan pesan "angka tidak ditemukan").

Flowchart





Source Code

```
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]
    return arr

sorted_arr = bubble_sort([64, 34, 25, 12, 22, 11, 90])
print(sorted_arr)

number_to_find = int(input("Input angka yang dicari: "))

for i, index in enumerate(sorted_arr):
    if index == number_to_find:
        print(f"Angka {number_to_find} ditemukan pada indeks ke-{i}")
```

Output

```
→ [11, 12, 22, 25, 34, 64, 90]
Input angka yang dicari: 25
Angka 25 ditemukan pada indeks ke-3
```



b. Latihan Kedua

Buatlah sebuah fungsi sorting berdasarkan metode bubble sort menggunakan konsep rekursif dengan bahasa pemrograman Python.

IPO

IPO (Input-Process-Output) untuk Bubble Sort Rekursif

Input:

- Sebuah array tak terurut yang terdiri dari elemen-elemen integer.
- Contoh: [5, 2, 9, 1, 5, 6]

Process:

1. Pengecekan Panjang Array:

- Jika panjang array (n) kurang dari atau sama dengan 1, array dianggap sudah terurut dan langsung dikembalikan.

2. Proses Bubble:

- Iterasi pada elemen-elemen array untuk membandingkan pasangan elemen yang berdekatan:
- Jika elemen pertama lebih besar daripada elemen kedua, lakukan pertukaran (*swap*).

3. Rekursi:

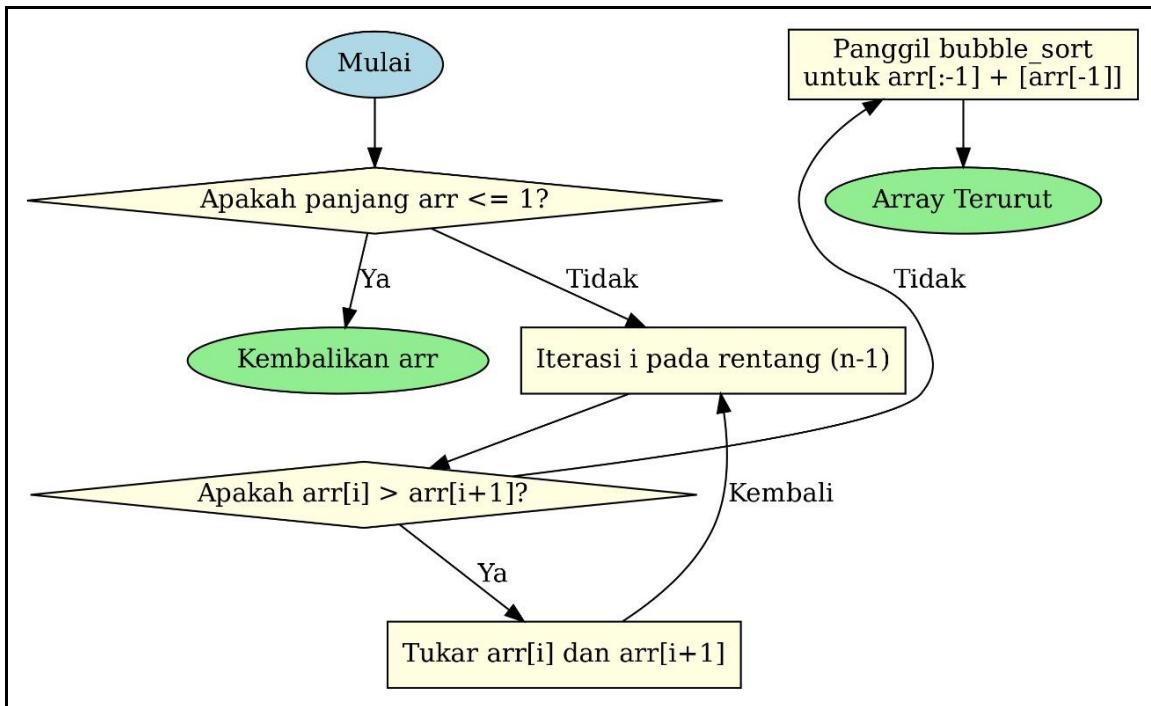
- Panggil kembali fungsi bubble_sort untuk sisa array (dari indeks pertama hingga indeks ke- $n-2$).
- Tambahkan elemen terbesar (yang sudah berada di posisi terakhir setelah satu iterasi) ke hasil rekursi.

Output:

- Array yang sudah terurut dalam urutan menaik.
- Contoh: [1, 2, 5, 5, 6, 9]



Flowchart



Source Code

```

def bubble_sort(arr):
    n = len(arr)
    if n <= 1:
        return arr
    for i in range(n - 1):
        if arr[i] > arr[i + 1]:
            arr[i], arr[i + 1] = arr[i + 1], arr[i]
    return bubble_sort(arr[:-1]) + [arr[-1]]

arr = [5, 2, 9, 1, 5, 6]
sorted_arr = bubble_sort(arr)
print("Array terurut:", sorted_arr)
    
```

Output



```
→ Array terurut: [1, 2, 5, 5, 6, 9]
```

4. File Praktikum

Github Repository:

```
https://github.com/Alip1023/prak10algo.git
```

5. Soal Latihan

Soal:

1. Mengapa dalam algoritma pencarian binary search himpunan datanya harus diurutkan terlebih dahulu? Jelaskan alasannya!
2. Deskripsikan serta narasikan jalannya alur source code program yang sebelumnya telah kalian buat pada Elemen Kompetensi Latihan Kedua!

Jawaban:

1. Data harus diurutkan dalam binary search karena algoritma ini bergantung pada relasi **lebih kecil** atau **lebih besar** untuk membagi himpunan secara logis. Tanpa urutan, algoritma tidak dapat bekerja dengan benar atau efisien.
2. Setiap fungsi dirancang dengan konsep dasar **rekursi**.
 - Proses berjalan melalui 3 tahap utama: **basis rekursi**, **pemanggilan fungsi rekursif**, dan **pengembalian hasil**.
 - Penanganan error diimplementasikan pada fungsi konversi untuk menangani input tidak valid.



6. Kesimpulan

- Dalam pengerjaan program dengan bahasa pemrograman Python, kita harus benar-benar teliti dalam menginputkan suatu fungsi untuk menampilkan suatu keluaran pada layar dengan sesuai.
- **Bubble Sort Rekursif** digunakan untuk mengurutkan data, dengan proses rekursif yang sederhana dan intuitif.
 - Setelah data diurutkan, **Binary Search** digunakan untuk mencari elemen dengan efisiensi tinggi (waktu pencarian logaritmik, $O(\log n)$).
 - Penggabungan kedua algoritma memungkinkan program menangani list acak secara otomatis sebelum pencarian dilakukan.

1. Cek List (✓)

No	Elemen Kompetensi	Penyelesaian	
		Selesai	Tidak Selesai
1.	Latihan Pertama	✓	
2.	Latihan Kedua	✓	

2. Formulir Umpam Balik

No	Elemen Kompetensi	Waktu Pengerjaan	Kriteria
1.	Latihan Pertama	60 Menit	Menarik
2.	Latihan Kedua	60 Menit	Menarik

Keterangan:

1. Menarik
2. Baik
3. Cukup
4. Kurang

