Nama: **Muhammad Ali Akbar Hidayatullah** **NIM:** **064002300034**		**MODUL 10** **Nama Dosen:** **Ratna Shofiati, S.Kom, M.Kom**
Hari/Tanggal: **Kamis 30 November 2023**	**Praktikum Algoritma & Pemrograman**	**Nama Asisten Labratorium:** 1. **Yuda Hadi Prasetyo - 065002100004** 2. **Muhammad Hasan Husein - 065002100009**

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.

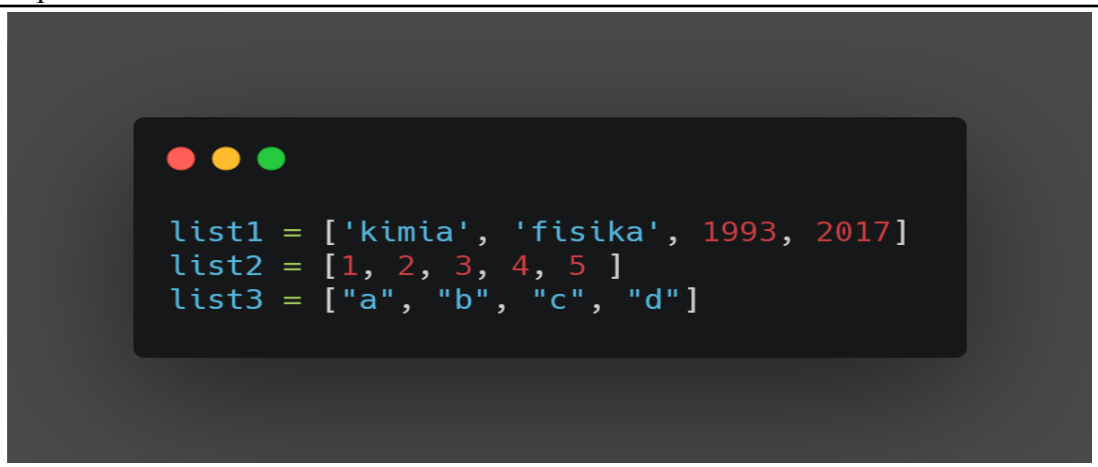
Sorting

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



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.

Source Code

```
def binarysearch(arr, start, end, x):
    if end >= start:
        mid = start + (end - start) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binarysearch(arr, start, mid - 1, x)
        else:
            return binarysearch(arr, mid + 1, end, x)
    else:
        return -1

def bubblesort(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

array = [87, 56, 34, 23, 89, 15, 2, 200, 28, 31]
sorted_array = bubblesort(array.copy())  # Sorting a copy of the original array

search_value = int(input("Masukkan angka yang ingin dicari: "))
result = binarysearch(sorted_array, 0, len(sorted_array) - 1, search_value)

if result != -1:
```

```
    print(f"Elemen {search_value} ditemukan pada baris {result}")
else:
    print(f"Elemen {search_value} tidak ditemukan.")
```

Output

```
Masukkan angka yang ingin dicari: 200
Elemen 200 ditemukan pada baris 9
```

```
Masukkan angka yang ingin dicari: 12
Elemen 12 tidak ditemukan.
```

b. Latihan Kedua

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

Source Code

```
def recursive_bubble_sort(arr, n):
    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]
    recursive_bubble_sort(arr, n - 1)
    return arr

def binary_search(arr, start, end, x):
    if end >= start:
        mid = start + (end - start) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binary_search(arr, start, mid - 1, x)
        else:
            return binary_search(arr, mid + 1, end, x)
    else:
        return -1

array = [70, 82, 27, 49, 31, 19, 4, 500, 25, 33]
sorted_array = recursive_bubble_sort(array.copy(), len(array))

search_value = int(input("Masukkan angka yang ingin dicari: "))
result = binary_search(sorted_array, 0, len(sorted_array) - 1, search_value)

if result != -1:
    print(f"Elemen {search_value} ditemukan pada indeks {result}")
else:
    print(f"Elemen {search_value} tidak ditemukan dalam list yang diurutkan.")
```

Output

```
Masukkan angka yang ingin dicari: 500
Elemen 500 ditemukan pada indeks 9
```

```
Masukkan angka yang ingin dicari: 70
Elemen 70 ditemukan pada indeks 7
```

```
Masukkan angka yang ingin dicari: 300
Elemen 300 tidak ditemukan dalam list yang diurutkan.
```

4. File Praktikum

Github Repository:

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. karena harus dari data yang terurut terlebih dahulu untuk efisiensi pencarian, dan apabila himpunan data tidak diurutkan, elemen yang dicari mungkin eror atau juga program akan bekerja namun tak maksimal itulah mengapa kita harus mengurutkan datanya terlebih dahulu
2. sebenernya masih sama dengan sebelumnya namun di latihan kedua ini menggunakan fungsi rekursif dan fungsi ini mengimplementasikan algoritmaa bubble sort secara rekursif. fungsi binary_search dipanggil untuk mencari nilai dalam array yang diurutkan lalu hasil pencarian diambil dan diperika

6. Kesimpulan

a. Dalam pengerjaan program dengan bahasa pemrograman Python, kita harus benar-benar teliti dalam menginputkan suatu fungsi untuk menampilkan suatu keluaran pada layar dengan sesuai.
b. Kita dapat mengetahui untuk membuat algoritma pencarian binary search dapat gunakan perintah dari def bubble sort atau def recursif bubble sort untuk merunning programnya ini berguna untuk mengurut komponen angkanya dan mencari angka dari urutan data angka yang ingin dicari.

7. Cek List (✔)

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

8. Formulir Umpan Balik

No	Elemen Kompetensi	Waktu Pengerjaan	Kriteria
1.	Latihan Pertama	20 Menit	Menarik
2.	Latihan Kedua	30 Menit	Menarik

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