`LAPORAN PRAKTIKUM

MODUL VIII
ALGORITMA SEARCHING

Disusun oleh:
Brian Farrel Evandhika
NIM: 2311102037

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023

A. TUJUANPRAKTIKUM

a. Menunjukkan beberapa algoritma dalam Pencarian.

b. Menunjukkan bahwa pencarian merupakan suatu persoalan yang bisa diselesaikan dengan beberapa algoritma yang berbeda.

c. Dapat memilih algoritma yang paling sesuai untuk menyelesaikan suatu permasalahan pemrograman.

B. Dasar Teori

Dasar Teori Algoritma searching adalah dasar dari banyak operasi komputasi yang bertujuan menemukan data tertentu dalam sebuah struktur data. Struktur data ini bisa berupa array, linked list, tree, graph, atau bentuk lain yang menyimpan informasi. Proses pencarian melibatkan penelusuran elemen-elemen dalam struktur tersebut untuk menemukan lokasi atau keberadaan elemen yang dicari. Efektivitas algoritma searching diukur berdasarkan waktu yang dibutuhkan untuk menemukan elemen tersebut dan jumlah perbandingan yang dilakukan selama pencarian.

Algoritma searching dapat diklasifikasikan menjadi dua kategori utama: algoritma searching linear dan non-linear. Algoritma linear search, seperti namanya, melakukan pencarian dengan memeriksa setiap elemen dalam urutan berurutan dari awal hingga akhir. Salah satu contohnya adalah linear search atau sequential search, yang sangat sederhana namun kurang efisien untuk jumlah data yang besar karena memiliki kompleksitas waktu O(n). Di sisi lain, algoritma non-linear seperti binary search bekerja dengan cara yang lebih efisien namun memerlukan data yang terurut.

Binary search adalah salah satu algoritma searching non-linear yang sangat efisien, terutama untuk data yang terurut. Algoritma ini bekerja dengan membagi struktur data menjadi dua bagian pada setiap langkah dan memeriksa elemen tengah. Jika elemen tengah bukan yang dicari, pencarian dilanjutkan pada salah satu separuh bagian data yang masih relevan. Dengan kompleksitas waktu O(log n), binary search jauh lebih cepat dibandingkan linear search untuk data yang besar. Namun, prasyarat utama penggunaannya adalah data harus dalam keadaan terurut.

Selain binary search, ada beberapa algoritma searching lain yang menawarkan efisiensi berbeda-beda berdasarkan karakteristik data yang dihadapi. Jump search, misalnya, menggabungkan kelebihan dari linear dan binary search dengan melakukan lompatan beberapa elemen pada satu waktu dan kemudian melakukan pencarian linear dalam interval kecil tersebut. Interpolation search memanfaatkan distribusi data untuk memprediksi lokasi elemen yang dicari, sangat efektif pada data yang terdistribusi secara merata dengan kompleksitas waktu terbaik O(log log n). Exponential search digunakan untuk data yang sangat besar dengan memulai pencarian secara eksponensial sebelum menggunakan binary search pada interval yang ditemukan.

Kompleksitas waktu dari algoritma searching sangat bervariasi tergantung pada metode yang digunakan. Linear search memiliki kompleksitas O(n) yang tidak efisien untuk data besar, sementara binary search menawarkan kompleksitas O(log n). Jump search dan interpolation search memiliki kompleksitas yang dapat lebih baik dalam kondisi tertentu, dengan interpolation search mencapai O(log log n) pada distribusi data yang ideal. Ruang tambahan yang dibutuhkan untuk kebanyakan algoritma ini biasanya O(1), membuatnya hemat dalam penggunaan memori.

GUIDED

Guided 1

```cpp
#include <iostream>
using namespace std;
int main()
{
    int n = 10;
    int data[n] = {9, 4, 1, 7, 5, 12, 4, 13, 4, 10};
    int cari = 10;
    bool ketemu = false;
    int i;
    // algoritma Sequential Search
    for (i = 0; i < n; i++)
    {
        if (data[i] == cari)
        {
            ketemu = true;
            break;
        }
    }
    cout << "\t Program Sequential Search Sederhana\n " << endl;
    cout << "data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}" << endl;
    if (ketemu)
    {
        cout << "\n angka " << cari << " ditemukan pada indeks ke - " << i <<
endl;
    }
    else
    {
        cout << cari << " tidak dapat ditemukan pada data." << endl;
    }
    return 0;
}
```

Screenshot Program

```
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITME\Praktikum Modul 3\2311102037_Brian-Farre
l-Evandhika_IF-11-A_Praktikum-Struktur-Data>  & 'c:\Users\MSI GAMING\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debug
Adapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-glfwlogg.xdu' '--stdout=Microsoft-MIEngine-Out-ot3hzt2b.ze2
' '--stderr=Microsoft-MIEngine-Error-03nhjtwn.okr' '--pid=Microsoft-MIEngine-Pid-i02neozs.ony' '--dbgExe=C:\msys64\ucrt64\bin\gdb
.exe' '--interpreter=mi'
        Program Sequential Search Sederhana

data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}

 angka 10 ditemukan pada indeks ke - 9
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITME\Praktikum Modul 3\2311102037_Brian-Farre
l-Evandhika_IF-11-A_Praktikum-Struktur-Data>
```

Deskripsi Program

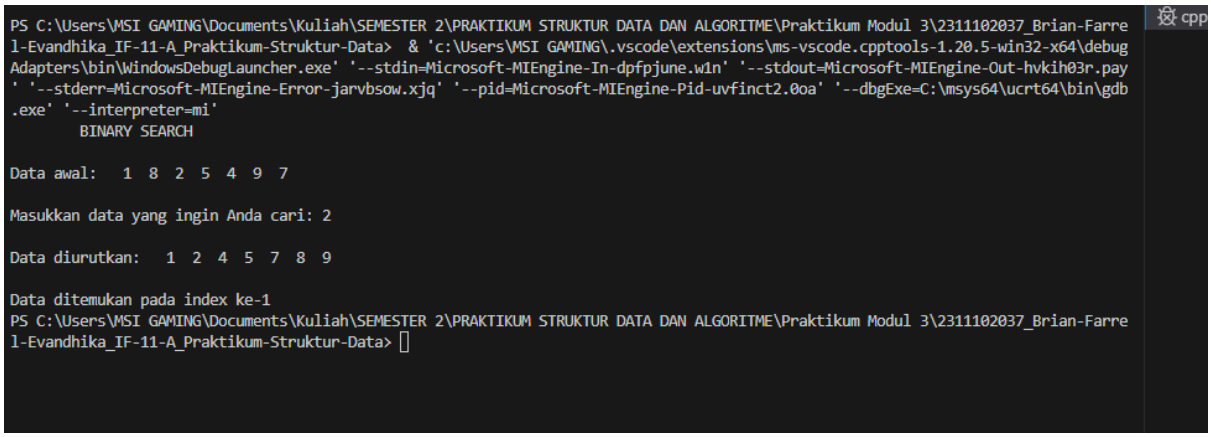
Program di atas adalah implementasi sederhana dari algoritma Sequential Search (pencarian berurutan) dalam bahasa C++. Program ini mencari elemen tertentu dalam sebuah array. Array `data` berisi 10 elemen yang terdiri dari angka-angka {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}. Program mencari angka yang disimpan dalam variabel `cari`, yang dalam hal ini adalah 10. Algoritma Sequential Search bekerja dengan memeriksa setiap elemen array mulai dari indeks 0 hingga 9. Jika elemen yang dicari ditemukan, program menandai bahwa elemen tersebut ditemukan (`ketemu = true`) dan mencetak pesan yang menunjukkan indeks di mana elemen tersebut ditemukan. Jika elemen tidak ditemukan setelah memeriksa seluruh array, program mencetak pesan bahwa elemen tersebut tidak ada dalam data. Program ini menggunakan loop for untuk iterasi melalui elemen-elemen array dan kondisi if untuk memeriksa kesamaan elemen dengan elemen yang dicari.

Guided 2

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
// Deklarasi array dan variabel untuk pencarian
int arrayData[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;
void selection_sort(int arr[], int n)
{
    int temp, min;
    for (int i = 0; i < n - 1; i++)
    {
        min = i;
        for (int j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[min])
            {
                min = j;
            }
        }
        // Tukar elemen
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}
void binary_search(int arr[], int n, int target)
{
    int awal = 0, akhir = n - 1, tengah, b_flag = 0;
    while (b_flag == 0 && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (arr[tengah] == target)
        {
            b_flag = 1;
            break;
        }
        else if (arr[tengah] < target)
```

```
        {
            awal = tengah + 1;
        }
        else
        {
            akhir = tengah - 1;
        }
    }
    if (b_flag == 1)
        cout << "\nData ditemukan pada index ke-" << tengah << endl;
    else
        cout << "\nData tidak ditemukan\n";
}
int main()
{
    cout << "\tBINARY SEARCH" << endl;
    cout << "\nData awal: ";
    // Tampilkan data awal
    for (int x = 0; x < 7; x++)
    {
        cout << setw(3) << arrayData[x];
    }
    cout << endl;
    cout << "\nMasukkan data yang ingin Anda cari: ";
    cin >> cari;
    // Urutkan data dengan selection sort
    selection_sort(arrayData, 7);
    cout << "\nData diurutkan: ";
    // Tampilkan data setelah diurutkan
    for (int x = 0; x < 7; x++)
    {
        cout << setw(3) << arrayData[x];
    }
    cout << endl;
    // Lakukan binary search
    binary_search(arrayData, 7, cari);
    return 0;
}
```

Screenshot Program



Deskripsi Program

Program di atas mengimplementasikan algoritma Selection Sort untuk mengurutkan array dan Binary Search untuk mencari elemen dalam array yang telah diurutkan. Program dimulai dengan menampilkan elemen-elemen awal dari array `data` yang berisi {1, 8, 2, 5, 4, 9, 7}. Pengguna kemudian diminta memasukkan elemen yang ingin dicari. Array `data` diurutkan menggunakan fungsi `selection_sort`, dan array yang telah diurutkan ditampilkan. Setelah itu, program melakukan pencarian elemen yang dimasukkan pengguna menggunakan fungsi `binary_search`. Jika elemen ditemukan, program mencetak pesan yang menunjukkan indeks di mana elemen tersebut berada; jika tidak, program mencetak pesan bahwa elemen tidak ditemukan.

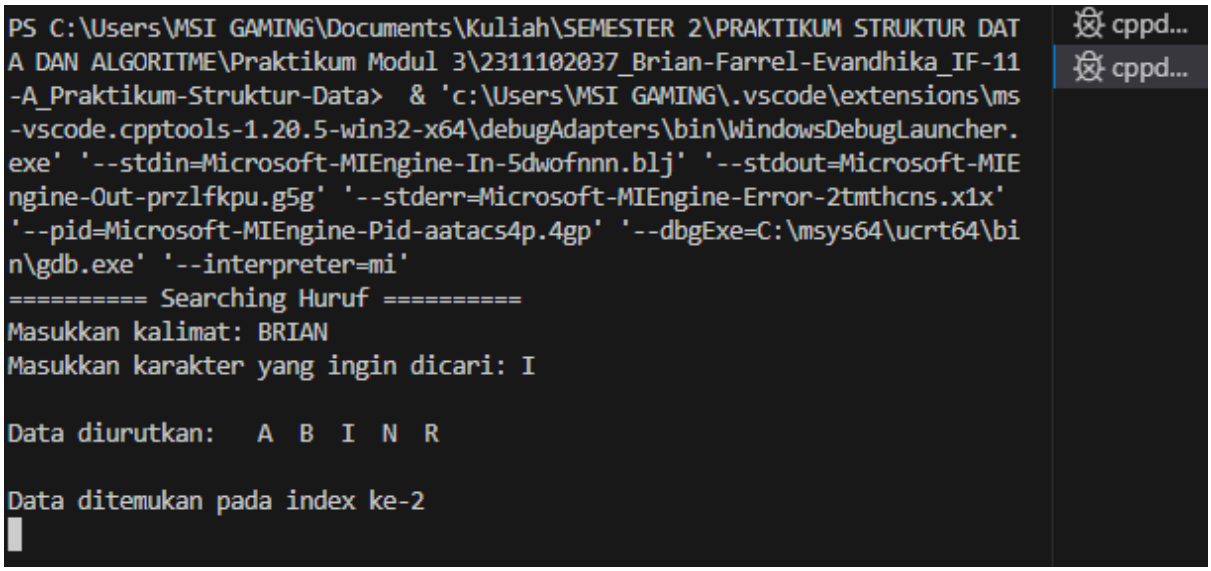
UNGUIDED

Unguided 1

```
#include <iostream> // Library standar yang digunakan untuk input dan output
#include <conio.h> // Library standar yang digunakan untuk _getche()
#include <cstring> // Library standar yang digunakan untuk strlen()
#include <iomanip> // Library standar yang digunakan untuk setw()

using namespace std; // Untuk mempersingkat penulisan kode program

char dataArray[100]; // Data yang akan diurutkan
char cari; // Data yang dicari

void Selection_Sort(char arr[], int n) // Fungsi untuk mengurutkan data
menggunakan Selection Sort
{
    int temp, min, i, j;
    for (i = 0; i < n - 1; i++)
    {
        min = i;
        for (j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[min])
            {
                min = j;
            }
        }
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}
void BinarySearch(char arr[], int n) // Fungsi untuk mencari data menggunakan
Binary Search
{
    int awal, akhir, tengah;
    bool b_flag = false;
    awal = 0;
    akhir = n - 1;
    while (!b_flag && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (arr[tengah] == cari)
        {
            b_flag = true;
        }
        else if (arr[tengah] < cari)
        {
```

```
            awal = tengah + 1;
        }
        else
        {
            akhir = tengah - 1;
        }
    }
    if (b_flag)
    {
        cout << "\nData ditemukan pada index ke-" << tengah << endl;
    }
    else
    {
        cout << "\nData tidak ditemukan" << endl;
    }
}

int main() // Fungsi utama
{
    cout << "========== Searching Huruf ==========" << endl;

    cout << "Masukkan kalimat: ";
    cin.getline(dataArray, 100); // Memasukkan data yang akan diurutkan,
getline() digunakan untuk membaca inputan yang mengandung spasi

    int n = strlen(dataArray);

    cout << "Masukkan karakter yang ingin dicari: ";
    cin >> cari;

    cout << "\nData diurutkan: ";
    Selection_Sort(dataArray, n); // Memanggil fungsi Selection_Sort

    for (int x = 0; x < n; x++)
    {
        cout << setw(3) << dataArray[x];
    }
    cout << endl;

    BinarySearch(dataArray, n);  // Memanggil fungsi BinarySearch

    _getche(); // Menunggu inputan dari user
    return 0;
}
```

Screenshot Program



Deskripsi Program

Kodingan ini adalah program C++ yang mengimplementasikan algoritma Selection Sort untuk mengurutkan data berupa kalimat yang diinput oleh user, dan kemudian menggunakan algoritma Binary Search untuk mencari karakter tertentu dalam kalimat yang telah diurutkan. Program ini meminta user untuk memasukkan kalimat dan karakter yang ingin dicari, kemudian mengurutkan kalimat menggunakan Selection Sort dan mencari karakter tersebut menggunakan Binary Search. Hasilnya, program akan menampilkan kalimat yang telah diurutkan dan informasi apakah karakter yang dicari ditemukan atau tidak, serta pada index ke-berapa karakter tersebut ditemukan.

Unguided 2

```cpp
#include <iostream> // Library standar yang digunakan untuk input dan output
#include <string>   // Library standar yang digunakan untuk string

using namespace std; // Untuk mempersingkat penulisan kode program

int carivokal(string kata, char huruf) // fungsi untuk mencari jumlah huruf vokal (sequential search)
{
    int jumlah = 0;
    for (int i = 0; i < kata.length(); i++) // Sequential search dilakukan dengan iterasi melalui setiap karakter dalam string
    {
        if (kata[i] == huruf) // Memeriksa apakah karakter pada indeks saat ini sama dengan huruf yang dicari
        {
            jumlah++; // Jika ditemukan huruf yang sesuai, jumlahnya ditambah
        }
    }
    return jumlah; // Mengembalikan jumlah huruf yang ditemukan
}
```

```cpp
void hitungvokal(string kata) // fungsi untuk menampilkan jumlah huruf vokal
{
    int jumlah;
    char vokal[10] = {'a', 'i', 'u', 'e', 'o', 'A', 'I', 'U', 'E', 'O'};
    for (int i = 0; i < 10; i++)
    {
        jumlah = carivokal(kata, vokal[i]);
        cout << "Jumlah huruf " << vokal[i] << " : " << jumlah << endl;
    }
}

int main() // fungsi utama
{
    cout << "========== Menghitung Jumlah Huruf Vokal ==========" << endl;
    string kata;
    cout << "Masukkan kata : ";
    cin >> kata;
    hitungvokal(kata);
    return 0;
}
```

Screenshot Program

```
========== Menghitung Jumlah Huruf Vokal ==========
Masukkan kata : BRIAN
Jumlah huruf a : 0
Jumlah huruf i : 0
Jumlah huruf u : 0
Jumlah huruf e : 0
Jumlah huruf o : 0
Jumlah huruf A : 1
Jumlah huruf I : 1
Jumlah huruf U : 0
Jumlah huruf E : 0
Jumlah huruf O : 0
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR DAT
A DAN ALGORITME\Praktikum Modul 3\2311102037_Brian-Farrel-Evandhika_IF-11
-A_Praktikum-Struktur-Data>
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR DAT
A DAN ALGORITME\Praktikum Modul 3\2311102037_Brian-Farrel-Evandhika_IF-11
-A_Praktikum-Struktur-Data>
```

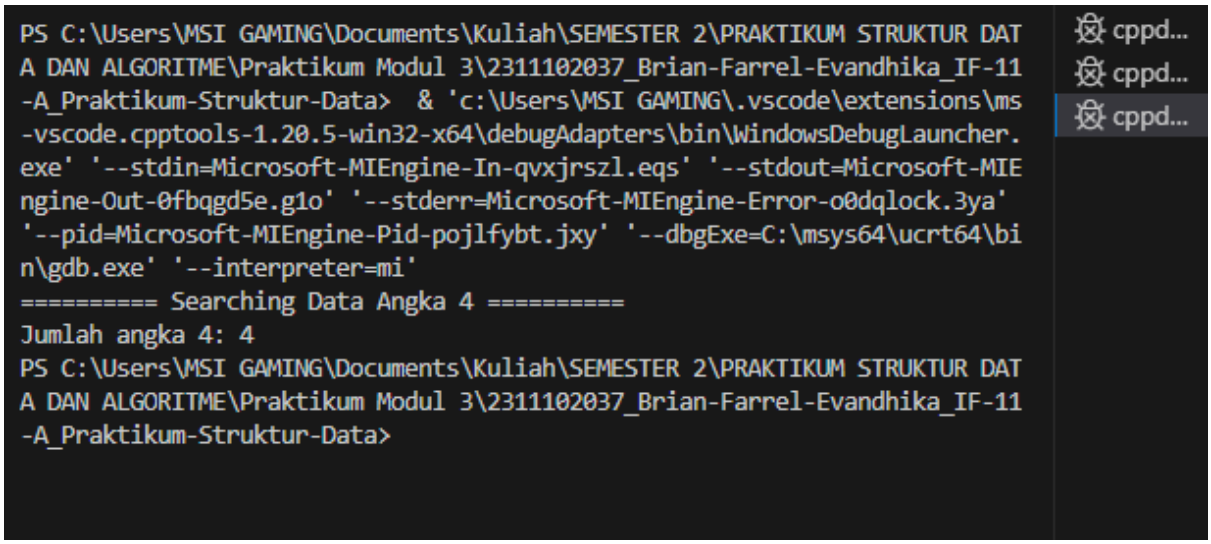
Deskripsi Program

Kodingan ini adalah program C++ yang mengimplementasikan algoritma Selection Sort untuk mengurutkan data berupa kalimat yang diinput oleh user, dan kemudian menggunakan algoritma Binary Search untuk mencari karakter tertentu dalam kalimat yang telah diurutkan. Program ini meminta user untuk memasukkan kalimat dan karakter yang ingin dicari, kemudian mengurutkan kalimat menggunakan Selection Sort dan mencari karakter tersebut menggunakan Binary Search. Hasilnya, program akan menampilkan kalimat yang telah diurutkan dan informasi apakah karakter yang dicari ditemukan atau tidak, serta pada index ke-berapa karakter tersebut ditemukan.

Unguided 3

```cpp
#include <iostream> // Library standar yang digunakan untuk input dan output

using namespace std; // Untuk mempersingkat penulisan kode program

int sequentialSearch(int arr[], int n, int key) // Fungsi untuk mencari jumlah kemunculan suatu angka dalam array
{
    int count = 0;
    for (int i = 0; i < n; ++i)
    {
        if (arr[i] == key)
        {
            count++;
        }
    }
    return count;
}

int main() // fungsi utama
{
    cout << "========== Searching Data Angka 4 ==========" << endl;
    int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}; // Data yang akan dicari
    int size = sizeof(data) / sizeof(data[0]); // Menghitung jumlah elemen dalam array
    int key = 4; // Angka yang dicari
    int result = sequentialSearch(data, size, key); // Memanggil fungsi sequentialSearch
    cout << "Jumlah angka 4: " << result << endl;
    return 0;
}
```

Screenshot Program



Deskripsi Program

Kodingan ini adalah program C++ yang mengimplementasikan algoritma Sequential Search untuk mencari jumlah kemunculan suatu angka dalam array. Program ini meminta user untuk memasukkan array dan angka yang ingin dicari, kemudian menggunakan algoritma Sequential Search untuk mencari jumlah kemunculan angka tersebut dalam array. Hasilnya, program akan menampilkan jumlah kemunculan angka yang dicari dalam array

Kesimpulan

Kodingan pertama adalah program C++ yang mengimplementasikan algoritma Selection Sort untuk mengurutkan data berupa kalimat yang diinput oleh user, dan kemudian menggunakan algoritma Binary Search untuk mencari karakter tertentu dalam kalimat yang telah diurutkan. Kodingan kedua adalah program C++ yang mengimplementasikan algoritma Sequential Search untuk mencari jumlah kemunculan suatu angka dalam array. Kodingan tersebut menggunakan algoritma pencarian yang berbeda, yaitu Selection Sort dan Binary Search untuk kodingan pertama, dan Sequential Search untuk kodingan kedua. Kodingan tersebut juga menggunakan array untuk menyimpan data yang akan dicari, dan menggunakan fungsi untuk mengimplementasikan algoritma pencarian. Kodingan tersebut memiliki tujuan yang berbeda, yaitu untuk mencari karakter tertentu dalam kalimat yang telah diurutkan untuk kodingan pertama, dan untuk mencari jumlah kemunculan suatu angka dalam array untuk kodingan kedua. Kodingan tersebut dapat digunakan untuk mengurutkan data dan mencari data tertentu dalam data yang telah diurutkan, sehingga dapat membantu dalam pengolahan data.

Daftar Pustaka

[1] https://www.geeksforgeeks.org/searching-algorithms/

[2] https://www.trivusi.web.id/2022/11/pengertian-algoritma-pencarian.html