

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN
PEMROGRAGAMAN**

**MODUL VIII
ALGORITMA SEARCHING**



Disusun oleh :

ZEFANYA.B.T.TARIGAN

2311102028

IF-11-A

Dosen Pengampu :

Wahyu Andi Saputra, S. Pd., M. Eng

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

2024

BAB I

A. TUJUAN PRAKTIKUM

- 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.

BAB II

B. DASAR TEORI

Pencarian (Searching) yaitu proses menemukan suatu nilai tertentu pada kumpulan data. Hasil pencarian adalah salah satu dari tiga keadaan ini: data ditemukan, data ditemukan lebih dari satu, atau data tidak ditemukan. Searching juga dapat dianggap sebagai proses pencarian suatu data di dalam sebuah array dengan cara mengecek satu persatu pada setiap index baris atau setiap index kolomnya dengan menggunakan teknik perulangan untuk melakukan pencarian data. Terdapat 2 metode pada algoritma Searching, yaitu:

a. Sequential Search

Sequential Search merupakan salah satu algoritma pencarian data yang biasa digunakan untuk data yang berpola acak atau belum terurut. Sequential search juga merupakan teknik pencarian data dari array yang paling mudah, dimana data dalam array dibaca satu demi satu dan diurutkan dari index terkecil ke index terbesar, maupun sebaliknya. Konsep Sequential Search yaitu:

- Membandingkan setiap elemen pada array satu per satu secara berurut.
- Proses pencarian dimulai dari indeks pertama hingga indeks terakhir.
- Proses pencarian akan berhenti apabila data ditemukan. Jika hingga akhir array data masih juga tidak ditemukan, maka proses pencarian tetap akan dihentikan.

- Proses perulangan pada pencarian akan terjadi sebanyak jumlah N elemen pada array.

Algoritma pencarian berurutan dapat dituliskan sebagai berikut :

- 1) $i \leftarrow 0$
- 2) $\text{ketemu} \leftarrow \text{false}$
- 3) Selama (tidak ketemu) dan ($i \leq N$) kerjakan baris 4
- 4) Jika ($\text{Data}[i] = x$) maka $\text{ketemu} \leftarrow \text{true}$, jika tidak $i \leftarrow i + 1$
- 5) Jika (ketemu) maka i adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

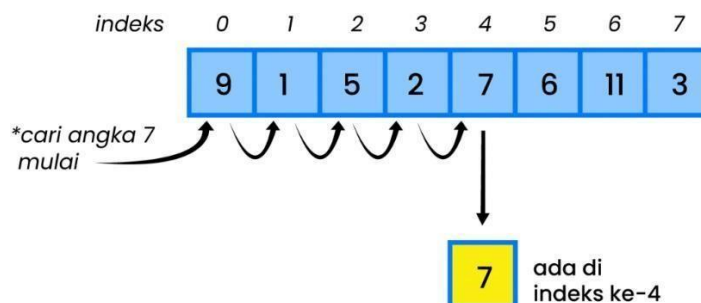
Di bawah ini merupakan fungsi untuk mencari data menggunakan pencarian sekuensial.

```
int SequentialSearch (int x)
{
    int i = 0;
    bool ketemu = false;
    while ((!ketemu) && (i < Max)){
        if (Data[i] == x)
            ketemu = true;
        else
            i++;
    }
    if (ketemu)
        return i;
    else
        return -1;
}
```

Fungsi diatas akan mengembalikan indeks dari data yang dicari. Apabila data tidak ditemukan maka fungsi diatas akan mengembalikan nilai -1.

Contoh dari Sequential Search, yaitu:

Int A[8] = {9,1,5,2,7,6,11,3}



Gambar 1. Ilustrasi Sequential Search

Misalkan, dari data di atas angka yang akan dicari adalah angka 7 dalam array A, maka proses yang akan terjadi yaitu:

- Pencarian dimulai pada index ke-0 yaitu angka 9, kemudian dicocokkan dengan angka yang akan dicari, jika tidak sama maka pencarian akan dilanjutkan ke index selanjutnya.
- Pada index ke-1, yaitu angka 1, juga bukan angka yang dicari, maka pencarian akan dilanjutkan pada index selanjutnya.
- Pada index ke-2 dan index ke-3 yaitu angka 5 dan 2, juga bukan angka yang dicari, sehingga pencarian dilanjutkan pada index selanjutnya.
- Pada index ke-4 yaitu angka 7 dan ternyata angka 7 merupakan angka yang dicari, sehingga pencarian akan dihentikan dan proses selesai.

b. Binary Search

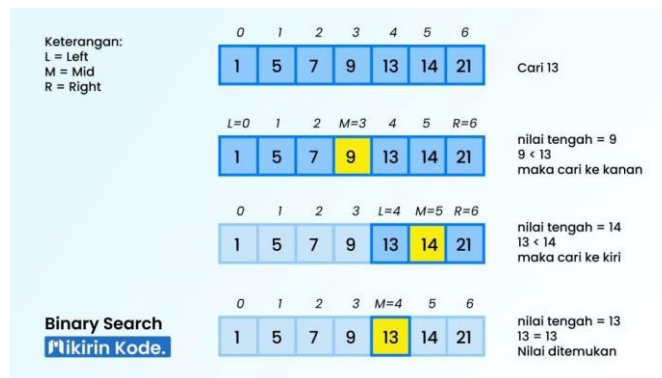
Binary Search termasuk ke dalam interval search, dimana algoritma ini merupakan algoritma pencarian pada array/list dengan elemen terurut. Pada metode ini, data harus diurutkan terlebih dahulu dengan cara data dibagi menjadi dua bagian (secara logika), untuk setiap tahap pencarian. Dalam penerapannya algoritma ini sering digabungkan dengan algoritma sorting karena data yang akan digunakan harus sudah terurut terlebih dahulu. Konsep Binary Search:

- Data diambil dari posisi 1 sampai posisi akhir N.
- Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
- Selanjutnya data yang dicari akan dibandingkan dengan data yang berada di posisi tengah, apakah lebih besar atau lebih kecil.
- Apabila data yang dicari lebih besar dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kanan dari data tengah. Proses pencarian selanjutnya akan dilakukan pembagian data menjadi dua bagian pada bagian kanan dengan acuan posisi data tengah akan menjadi posisi awal untuk pembagian tersebut.
- Apabila data yang dicari lebih kecil dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kiri dari data tengah. Proses pencarian selanjutnya akan dilakukan pembagian data menjadi dua bagian pada bagian kiri. Dengan acuan posisi data tengah akan menjadi posisi akhir untuk pembagian selanjutnya.
- Apabila data belum ditemukan, maka pencarian akan dilanjutkan dengan kembali membagi data menjadi dua.
- Namun apabila data bernilai sama, maka data yang dicari langsung ditemukan dan pencarian dihentikan.

Algoritma pencarian biner dapat dituliskan sebagai berikut :

- 1) $L = 0$
- 2) $R = N - 1$
- 3) `ketemu = false`
- 4) Selama $(L \leq R)$ dan (tidak ketemu) kerjakan baris 5 sampai dengan 8
- 5) $m = (L + R) / 2$
- 6) Jika $(Data[m] = x)$ maka `ketemu = true`
- 7) Jika $(x < Data[m])$ maka $R = m - 1$
- 8) Jika $(x > Data[m])$ maka $L = m + 1$
- 9) Jika (`ketemu`) maka m adalah indeks dari data yang dicari, jika tidak data tidak ditemukan

Contoh dari Binary Search, yaitu:



Gambar 2. Ilustrasi Binary Search

- Terdapat sebuah array yang menampung 7 elemen seperti ilustrasi di atas. Nilai yang akan dicari pada array tersebut adalah 13.
- Jadi karena konsep dari binary search ini adalah membagi array menjadi dua bagian, maka pertama tama kita cari nilai tengahnya dulu, total elemen dibagi 2 yaitu $7/2 = 4.5$ dan kita bulatkan jadi 4.
- Maka elemen ke empat pada array adalah nilai tengahnya, yaitu angka 9 pada indeks ke 3.
- Kemudian kita cek apakah $13 > 9$ atau $13 < 9$?
- 13 lebih besar dari 9, maka kemungkinan besar angka 13 berada setelah 9 atau di sebelah kanan. Selanjutnya kita cari ke kanan dan kita dapat mengabaikan elemen yang ada di kiri.

- Setelah itu kita cari lagi nilai tengahnya, didapatlah angka 14 sebagai nilai tengah. Lalu, kita bandingkan apakah $13 > 14$ atau $13 < 14$?
- Ternyata 13 lebih kecil dari 14, maka selanjutnya kita cari ke kiri.
- Karna tersisa 1 elemen saja, maka elemen tersebut adalah nilai tengahnya. Setelah dicek ternyata elemen pada indeks ke-4 adalah elemen yang dicari, maka telah selesai proses pencariannya.

BAB III

GUIDED

```
#include <iostream>
using namespace std;
int main()
{
    int n = 10;
    int data[] = {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 << "\tProgram Sequential Search Sederhana\n" << endl;
    cout << "Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}" << endl;

    if (ketemu)
```

```

        {
            cout << "\nAngka " << cari << " ditemukan pada indeks ke-" <<
i << endl;
        }
        else
        {
            cout << cari << " tidak dapat ditemukan pada data." <<
endl;
        }

        return 0;
    }
}

```

OUTPUT

```

gine-In-szh2y3a.3o3' '--stdout=Microsoft-MIEngine-Out-bat5
gine-Error-pfn5pjwp.vhh' '--pid=Microsoft-MIEngine-Pid-dxke
db.exe' '--interpreter=mi'
PS D:\KULIAH\SEMESTER2\struktur data praktikum> ./G1m8
    Program Sequential Search Sederhana

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

Angka 10 ditemukan pada indeks ke-9
PS D:\KULIAH\SEMESTER2\struktur data praktikum>

```

DESKRIPSI

Program ini adalah contoh sederhana dari implementasi algoritma Sequential Search dalam C++. Algoritma ini dapat bekerja dengan cara memeriksa setiap elemen dalam array secara berurutan sehingga akan menemukan nilai yang dicari atau hingga seluruh array telah diperiksa. Meskipun sederhana dan mudah diimplementasikan, algoritma ini mungkin tidak terlalu efisien untuk array yang sangat besar karena memerlukan waktu pencarian yang berbanding lurus dengan ukuran array.

Program ini berjudul "Program Sequential Search Sederhana", program yang mengindikasikan bahwa program ini merupakan contoh dasar dari algoritma pencarian berurutan. Tujuan dari program ini adalah untuk mencari sebuah nilai tertentu dalam sebuah array menggunakan metode Sequential Search. Dalam hal ini, program mencari nilai 10 dalam array yang sudah ditentukan.

Guided 2

Source Code

```
#include <iostream>
#include <iomanip>
#include <conio.h>
using namespace std;
int data[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;

void selection_sort()
{
    int temp, min, i, j;
    for (i = 0; i < 6; i++)
    {
        min = i;
        for (j = i + 1; j < 7; j++)
        {
            if (data[j] < data[min])
            {
                min = j;
            }
        }
        temp = data[i];
        data[i] = data[min];
        data[min] = temp;
    }
}

void binarysearch()
{
    int awal, akhir, tengah, b_flag = 0;
```



```

awal = 0;
akhir = 6;
while (b_flag == 0 && awal <= akhir)
{
    tengah = (awal + akhir) / 2;
    if (data[tengah] == cari)
    {
        b_flag = 1;
        break;
    }
    else if (data[tengah] < cari)
    {
        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:";
    for (int x = 0; x < 7; x++)
    {
        cout << setw(3) << data[x];
    }
}

```

```

cout << endl;

    cout << "\nMasukkan data yang ingin Anda cari: ";
    cin >> cari;

    cout << "\nData diurutkan:";
    selection_sort();
    for (int x = 0; x < 7; x++)
    {
        cout << setw(3) << data[x];
    }
    cout << endl;
    binarysearch();
    return 0;
}

```

OUTPUT

```

PS D:\KULIAH\SEMESTER2\struktur data praktikum> ./G2m8
    BINARY SEARCH

Data:  1  8  2  5  4  9  7

Masukkan data yang ingin Anda cari: 8

Data diurutkan:  1  2  4  5  7  8  9

Data ditemukan pada index ke 5

```

DESKRIPSI

Program ini mendemonstrasikan tentang penggunaan algoritma Binary Search yang efisien untuk mencari nilai dalam sebuah array. Namun, karena Binary Search mensyaratkan data harus terurut, program akan menggunakan algoritma Selection Sort untuk mengurutkan data terlebih dahulu. Meskipun Selection Sort tidak seefisien algoritma pengurutan lain untuk data besar, ia termasuk cukup sederhana untuk diimplementasikan. Kombinasi dari kedua algoritma ini dapat menunjukkan bahwa dalam beberapa kasus, preprocessing data (seperti pengurutan) dapat membantu mengoptimalkan operasi selanjutnya (seperti pencarian).

BAB IV

UNGUIDED

```
#include <iostream>
#include <algorithm>
#include <string>
using namespace std;

// Fungsi binary search untuk mencari huruf dalam string
int binarySearch(const string &str, char target)
{
    int left = 0;
    int right = str.length() - 1;

    while (left <= right)
    {
        int mid = left + (right - left) / 2;

        if (str[mid] == target)
        {
            return mid; // huruf ditemukan
        }
        else if (str[mid] < target)
        {
            left = mid + 1;
        }
        else
        {
            right = mid - 1;
        }
    }
}
```

```
return -1; // huruf tidak ditemukan
}

int main()
{
    string kalimat;
    char huruf;

    // Input kalimat
    cout << "Masukkan kalimat: ";
    getline(cin, kalimat);

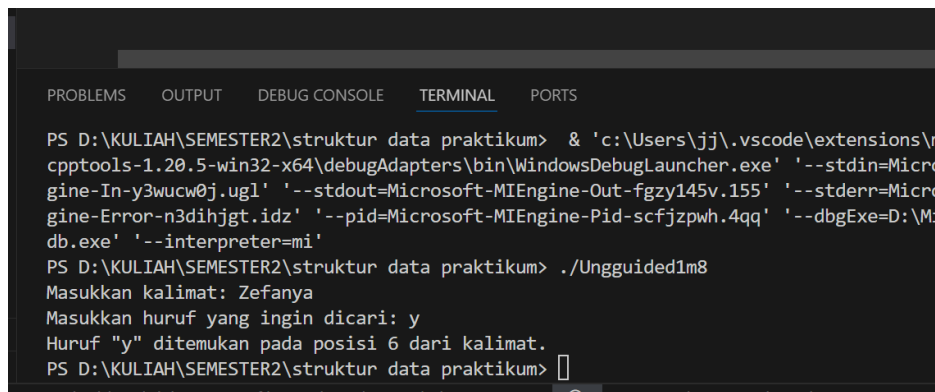
    // Input huruf yang dicari
    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> huruf;

    // Melakukan binary search pada kalimat
    int posisi = binarySearch(kalimat, huruf);

    // Mengecek apakah huruf ditemukan atau tidak
    if (posisi != -1)
    {
        cout << "Huruf \"" << huruf << "\" ditemukan pada posisi " <<
posisi + 1 << " dari kalimat." << endl;
    }
    else
    {
        cout << "Huruf \"" << huruf << "\" tidak ditemukan dalam
kalimat." << endl;
    }

    return 0;
}
```

OUTPUT



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\KULIAH\SEMESTER2\struktur data praktikum> & 'c:\Users\jj\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-y3wucw0j.ugl' '--stdout=Microsoft-MIEngine-Out-fgzy145v.155' '--stderr=Microsoft-MIEngine-Error-n3dihjgt.idz' '--pid=Microsoft-MIEngine-Pid-scfjzpw.4qq' '--dbgExe=D:\Microsoft\VisualStudio\15\bin\amd64\miengine.exe' '--interpreter=mi'
PS D:\KULIAH\SEMESTER2\struktur data praktikum> ./Ungguided1m8
Masukkan kalimat: Zefanya
Masukkan huruf yang ingin dicari: y
Huruf "y" ditemukan pada posisi 6 dari kalimat.
PS D:\KULIAH\SEMESTER2\struktur data praktikum>
```

DESKRIPSI

Program C++ tersebut adalah implementasi dari algoritma Binary Search yang dimodifikasi untuk mencari sebuah huruf spesifik dalam sebuah string (kalimat). Program ini bertujuan untuk mencari keberadaan dan posisi sebuah huruf tertentu dalam sebuah kalimat yang diinputkan oleh pengguna. Pencarian pada program ini dilakukan menggunakan algoritma Binary Search, yang biasanya digunakan pada array atau list yang terurut.

Program ini mencoba menerapkan algoritma Binary Search untuk mencari huruf dalam string, yang merupakan pendekatan secara tidak konvensional. Meskipun kode fungsi `binarySearch` sudah benar, asumsi bahwa string input sudah terurut membuat program ini tidak praktis untuk pencarian huruf dalam kalimat umum. Program ini bisa menjadi titik awal yang baik untuk diskusi tentang pemilihan algoritma yang tepat berdasarkan sifat-sifat data dan operasi yang dilakukan.

UNGUIDED 2

Source code

```
#include <iostream>
#include <string>
#include <cctype> // Untuk fungsi isalpha dan tolower
using namespace std;

// Fungsi untuk menghitung jumlah huruf vokal dalam sebuah kalimat
int hitungVokal(const string &kalimat)
{
    int jumlahVokal = 0;

    for (char huruf : kalimat)
    {
        // Mengonversi huruf menjadi huruf kecil
        char lowerHuruf = tolower(huruf);

        // Memeriksa apakah huruf merupakan huruf vokal
        if (lowerHuruf == 'a' || lowerHuruf == 'e' || lowerHuruf == 'i'
            || lowerHuruf == 'o' || lowerHuruf == 'u')
        {
            jumlahVokal++;
        }
    }

    return jumlahVokal;
}

int main()
{
    string kalimat;

    // Input kalimat dari pengguna
    cout << "Masukkan sebuah kalimat: ";
    getline(cin, kalimat);
```

```
// Menghitung jumlah huruf vokal dalam kalimat

int jumlahVokal = hitungVokal(kalimat);

// Menampilkan hasil

cout << "Jumlah huruf vokal dalam kalimat adalah: " << jumlahVokal
<< endl;

return 0;

}
```

OUTPUT

```
PS D:\KULIAH\SEMESTER2\struktur data praktikum> & 'c:\Users\jj\.vscode\extensions\ms-vs-
cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft
gine-In-midkpd1p.x43' '--stdout=Microsoft-MIEngine-Out-rv1pdhfx.jpg' '--stderr=Microsoft
gine-Error-24ddtui3.cjq' '--pid=Microsoft-MIEngine-Pid-ihceviod.slh' '--dbgExe=D:\MinGW\
db.exe' '--interpreter=mi'
PS D:\KULIAH\SEMESTER2\struktur data praktikum> ./Ungguided2m8
Masukkan sebuah kalimat: gorengan
Jumlah huruf vokal dalam kalimat adalah: 3
PS D:\KULIAH\SEMESTER2\struktur data praktikum> |
```

DESKRIPSI PROGRAM

Program C++ tersebut dirancang untuk menghitung jumlah huruf vokal yang terdapat dalam sebuah kalimat yang diinputkan oleh pengguna. Tujuan utama program ini adalah untuk menganalisis sebuah kalimat dan menentukan berapa banyak huruf vokal (a, e, i, o, u) yang muncul di dalamnya, tanpa memandang apakah huruf tersebut dalam bentuk huruf besar atau huruf kecil.

Program ini menyajikan solusi yang bersih dan efektif untuk menghitung huruf vokal dalam sebuah kalimat. Dengan menggunakan fungsi-fungsi C++ modern dan library standar, program ini mendemonstrasikan pemrograman yang baik: modular, efisien, dan mudah dipahami. Meskipun sederhana, program ini bisa menjadi dasar untuk tugas-tugas analisis teks yang lebih kompleks.

UNGUIDED 3

Source code

```
#include <iostream>

using namespace std;

// Fungsi untuk mencari jumlah kemunculan suatu angka dalam array dengan
Sequential Search

int hitungAngka(const int data[], int ukuran, int angka)
{
    int jumlah = 0;
    for (int i = 0; i < ukuran; ++i)
    {
        if (data[i] == angka)
        {
            jumlah++;
        }
    }
    return jumlah;
}

int main()
{
    const int ukuran = 10;
    int data[ukuran] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int angkaYangDicari = 4;

    // Menghitung jumlah kemunculan angka 4 dalam data menggunakan
    Sequential Search

    int jumlahAngka4 = hitungAngka(data, ukuran, angkaYangDicari);

    // Menampilkan hasil
    cout << "Jumlah angka 4 dalam data adalah: " << jumlahAngka4 << endl;

    return 0;
}
```


OUTPUT

```
cpptools-1.20.5-win32-x64\debugAdapters\bin\windowsDebugLauncher.exe --std
gine-In-cbbo1qej.j5i' '--stdout=Microsoft-MIEngine-Out-buqeb2o4.o13' '--std
gine-Error-fbbxvsmw.aey' '--pid=Microsoft-MIEngine-Pid-bmqj5ur2.bue' '--dbgE
db.exe' '--interpreter=mi'
PS D:\KULIAH\SEMESTER2\struktur data praktikum> ./Ungguided3m8
Jumlah angka 4 dalam data adalah: 4
PS D:\KULIAH\SEMESTER2\struktur data praktikum> █
```

DESKRIPSI

Program C++ tersebut dirancang untuk menghitung berapa kali sebuah angka tertentu muncul dalam sebuah array. Program ini menggunakan teknik Sequential Search untuk melakukan pencarian dan penghitungan. Tujuan utama program ini adalah untuk menganalisis sebuah array yang berisi angka-angka integer dan menghitung berapa kali suatu angka spesifik muncul dalam array tersebut. Dalam kasus ini, program khusus menghitung berapa kali angka 4 muncul dalam array yang sudah ditentukan.

Program ini menyajikan solusi yang efektif dan mudah dipahami untuk menghitung kemunculan suatu angka dalam array menggunakan Sequential Search. Meskipun sederhana, program ini mengilustrasikan konsep-konsep penting dalam pemrograman C++ seperti arrays, fungsi, dan looping. Cocok untuk pembelajaran dasar algoritma pencarian dan manipulasi array.

BAB V

KESIMPULAN

Algoritma searching adalah komponen fundamental dalam ilmu komputer, menawarkan beragam teknik untuk menemukan informasi dalam kumpulan data. Setiap algoritma memiliki karakteristik unik yang membuatnya sesuai untuk skenario tertentu. Pemilihan algoritma yang tepat melibatkan pemahaman mendalam tentang struktur data, karakteristik data, dan kebutuhan spesifik aplikasi. Studi tentang algoritma pencarian tidak hanya meningkatkan efisiensi program tetapi juga memperdalam pemahaman tentang desain algoritma, analisis kompleksitas, dan trade-off dalam pemrograman. Dalam era big data dan AI, keahlian dalam memilih dan mengoptimalkan algoritma pencarian menjadi semakin penting, menjadikan ini area yang dinamis dan terus berkembang dalam ilmu komputer. Mempelajari bidang ini tidak hanya meningkatkan keterampilan teknis tetapi juga mengasah pola pikir analitis yang penting dalam ilmu komputer dan pengembangan perangkat lunak.

E. DAFTAR PUSTAKA

Karumanchi, N.2016. *Data structures and algorithms made easy: Concepts, problems, interview Questions*. CareerMonk Publications.

Anita Sindar, RMS.2019. *Struktur Data dan Algoritma dengan c++*. Diakses dari https://books.google.co.id/books?hl=id&lr=&id=GP_ADwAAQBAJ&oi=fnd&pg=PR7&dq=algoritma+searching+c%2B%2B&ots=86f6Pu-MgO&sig=eBjm5jn0JtzqvH1lClzkj8W1d2g&redir_esc=y#v=onepage&q=algoritma%20searching%20c%2B%2B&f=false