

LAPORAN PRAKTIKUM

MODUL VIII ALGORITMA SEARCHING



Disusun oleh:
Muhammad Rifki Fadhilah
NIM: 2311102032

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

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2024**

BAB I

TUJUAN PRAKTIKUM

1. Menunjukkan beberapa algoritma dalam Pencarian.
2. Menunjukkan bahwa pencarian merupakan suatu persoalan yang bisa diselesaikan dengan beberapa algoritma yang berbeda.
3. Dapat memilih algoritma yang paling sesuai untuk menyelesaikan suatu permasalahan pemrograman.

BAB II

DASAR TEORI

1. Sequential Search(Pencarian Sekuensial)

Pencarian sekuenisal adalah metode pencarian sederhana yang digunakan untuk mencari elemen tertentu dalam sebuah daftar atau array. Algoritma ini bekerja dengan cara membandingkan setiap elemen dalam daftar dengan elemen yang dicari, satu per satu, mulai dari awal hingga akhir.

Proses pencarian dimulai dengan membandingkan elemen pertama dalam daftar dengan elemen yang dicari. Jika elemen tersebut sama dengan elemen yang dicari, pencarian selesai dan algoritma mengembalikan indeks elemen tersebut. Jika tidak, algoritma akan melanjutkan pencarian dengan membandingkan elemen kedua dalam daftar, dan begitu seterusnya, hingga seluruh elemen dalam daftar telah dibandingkan.

Jika elemen yang dicari tidak ditemukan dalam daftar, algoritma akan mengembalikan nilai yang menunjukkan bahwa elemen tersebut tidak ada dalam daftar.

Kelebihan dari pencarian sekuenisal adalah sederhana dan mudah dipahami. Namun, kelemahannya adalah waktu eksekusi yang relatif lambat, terutama pada daftar yang besar, karena algoritma ini harus memeriksa setiap elemen secara berurutan.

2. Binary Search(Pencarian Biner)

Pencarian biner adalah metode pencarian yang efisien digunakan untuk mencari elemen dalam sebuah daftar yang telah diurutkan. Algoritma ini bekerja dengan membandingkan elemen yang dicari dengan elemen tengah dari daftar. Jika elemen yang dicari lebih kecil dari elemen tengah, pencarian dilanjutkan hanya pada setengah daftar

bagian kiri dari elemen tengah. Sebaliknya, jika elemen yang dicari lebih besar dari elemen tengah, pencarian dilanjutkan hanya pada setengah daftar bagian kanan dari elemen tengah. Proses ini diulang secara rekursif pada setengah daftar yang dipilih, hingga elemen yang dicari ditemukan atau daftar hanya memiliki satu elemen yang tidak sesuai.

Kelebihan dari pencarian biner adalah waktu eksekusi yang cepat karena setiap langkah membagi jumlah elemen yang perlu diperiksa menjadi setengahnya. Namun, kelemahannya adalah daftar harus diurutkan terlebih dahulu sebelum pencarian dilakukan, dan algoritma ini tidak efektif pada daftar yang tidak diurutkan.

BAB III

GUIDED

1. Guided 1

Source code

```
#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;
}

```

Screenshoot program

```
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8> cd 'd:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output'
```

- PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output> & .\guided1.exe'
Program Sequential Search Sederhana

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

angka 10 ditemukan pada indeks ke - 9

```
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output> 
```

Deskripsi program

Program ini adalah implementasi pencarian sekuensial (sequential search) untuk mencari nilai tertentu dalam larik data. Pencarian sekuensial dilakukan dengan cara mengiterasi melalui setiap elemen larik secara berurutan untuk mencari nilai yang sesuai.

Dalam program ini, sebuah larik `data` dengan ukuran 10 diinisialisasi dengan nilai tertentu. Kemudian, sebuah nilai `cari` (dalam kasus ini, 10) ditentukan untuk dicari dalam larik. Sebuah variabel boolean `ketemu` digunakan untuk menunjukkan apakah nilai tersebut ditemukan atau tidak. Variabel `i` digunakan sebagai indeks iterasi dalam pencarian.

Pencarian sekuensial dilakukan dengan iterasi melalui setiap elemen larik menggunakan loop ``for``. Jika nilai yang dicari ditemukan, variabel ``ketemu`` diubah menjadi ``true`` dan iterasi dihentikan dengan menggunakan ``break``.

Setelah pencarian selesai, hasil pencarian dicetak ke layar dengan pesan yang sesuai, baik nilai tersebut ditemukan pada indeks ke berapa atau tidak ditemukan sama sekali.

2. Guided 2

Source code

```
#include <iostream>
using namespace std;
#include <conio.h>
#include <iomanip>

int data[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;

void selection_sort()
{
    int temp, min, i, j;
    for (i = 0; i < 7; 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 = 7;
```

```

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 << "\n Data ditemukan pada index ke- " << tengah <<
endl;
else
    cout << "\n Data tidak ditemukan\n";
}

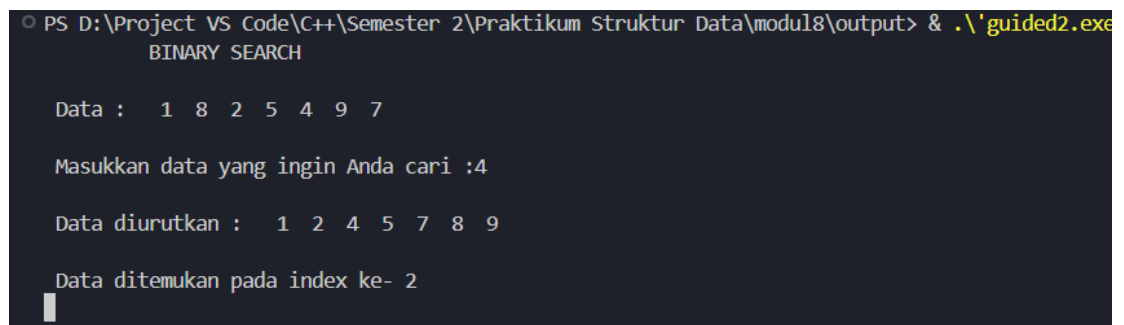
int main()
{
    cout << "\t BINARY SEARCH " << endl;
    cout << "\n Data : ";
    // tampilkan data awal
    for (int x = 0; x < 7; x++)
        cout << setw(3) << data[x];
    cout << endl;
    cout << "\n Masukkan data yang ingin Anda cari :";
    cin >> cari;
    cout << "\n Data diurutkan : ";
    // urutkan data dengan selection sort
    selection_sort();
    // tampilkan data setelah diurutkan

```



```
for (int x = 0; x < 7; x++)  
    cout << setw(3) << data[x];  
  
cout << endl;  
binarysearch();  
_getche();  
return EXIT_SUCCESS;  
}
```

Screenshoot program



```
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output> & .\'guided2.exe  
BINARY SEARCH  
  
Data : 1 8 2 5 4 9 7  
  
Masukkan data yang ingin Anda cari :4  
  
Data diurutkan : 1 2 4 5 7 8 9  
  
Data ditemukan pada index ke- 2
```

Deskripsi program

Program ini adalah implementasi pencarian biner (binary search) pada larik data yang telah diurutkan menggunakan algoritma selection sort. Pencarian biner bekerja dengan membagi-bagi interval pencarian menjadi dua bagian dan memeriksa apakah nilai yang dicari berada di setengah interval yang sesuai. Jika iya, pencarian dilakukan pada setengah tersebut; jika tidak, pencarian dilanjutkan pada setengah lainnya.

Dalam program ini, terdapat fungsi `selection_sort` untuk mengurutkan larik data menggunakan algoritma selection sort. Setelah diurutkan, fungsi `binarysearch` digunakan untuk melakukan pencarian biner terhadap data yang telah diurutkan.

Pertama-tama, program akan menampilkan data awal yang belum diurutkan, kemudian meminta pengguna memasukkan nilai yang ingin dicari (`cari`). Setelah itu, data diurutkan menggunakan selection sort, dan hasil pengurutan ditampilkan.

Selanjutnya, fungsi `binarysearch` akan mencari nilai yang dimasukkan pengguna (`cari`) dalam larik data yang telah diurutkan. Jika nilai ditemukan, program akan menampilkan indeks di mana nilai tersebut ditemukan; jika tidak ditemukan, program akan memberikan pesan bahwa data tidak ditemukan.

Program kemudian menunggu input dari pengguna sebelum berakhir.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

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

string kalimat;
char cari;

void binarysearch()
{
    int awal, akhir, tengah, b_flag = 0;
    awal = 0;
    akhir = kalimat.length() - 1;
    while (b_flag == 0 && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (kalimat[tengah] == cari)
        {
            b_flag = 1;
            break;
        }
        else if (kalimat[tengah] < cari)
            awal = tengah + 1;
        else
            akhir = tengah - 1;
    }
    if (b_flag == 1)
        cout << "\n Huruf ditemukan pada posisi ke-" << tengah <<
endl;
    else
```

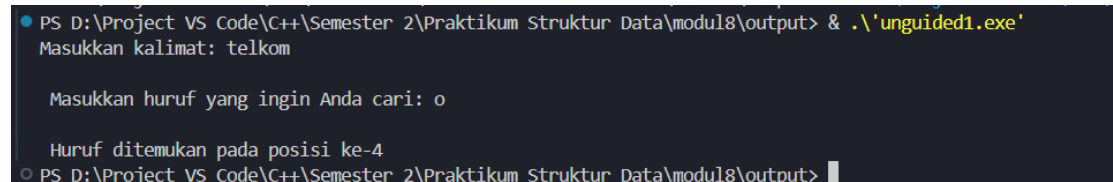
```

        cout << "\n Huruf tidak ditemukan\n";
    }

int main()
{
    cout << "Masukkan kalimat: ";
    getline(cin, kalimat);
    cout << "\n Masukkan huruf yang ingin Anda cari: ";
    cin >> cari;
    binarysearch();
    return 0;
}

```

Screenshoot program



```

PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output> & .\'unguided1.exe'
Masukkan kalimat: telkom

Masukkan huruf yang ingin Anda cari: o

Huruf ditemukan pada posisi ke-4
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output>

```

Deskripsi program

Program ini adalah implementasi pencarian biner (binary search) untuk mencari keberadaan sebuah huruf dalam sebuah kalimat yang dimasukkan oleh pengguna. Pencarian biner dilakukan dengan membagi-bagi interval pencarian (dalam hal ini, indeks kalimat) menjadi dua bagian dan memeriksa apakah huruf yang dicari berada di setengah interval yang sesuai. Jika ya, pencarian dilakukan pada setengah interval tersebut; jika tidak, pencarian dilanjutkan pada setengah lainnya.

Pada awal program, pengguna diminta memasukkan sebuah kalimat. Kalimat ini disimpan dalam variabel `kalimat`. Selanjutnya, pengguna diminta memasukkan huruf yang ingin dicari dalam kalimat tersebut, dan huruf tersebut disimpan dalam variabel `cari`.

Fungsi ``binarysearch`` kemudian digunakan untuk melakukan pencarian biner terhadap kalimat yang telah dimasukkan. Fungsi ini akan mencari keberadaan huruf yang dimasukkan dalam kalimat. Jika huruf ditemukan, program akan menampilkan posisi huruf tersebut dalam kalimat (indeks); jika tidak ditemukan, program akan memberikan pesan bahwa huruf tidak ditemukan.

Program kemudian berakhir setelah pencarian selesai.

2. Unguided 2

Source code

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

int hitungVokal(string kalimat) {
    int jumlah = 0;
    for (char huruf : kalimat) {
        // Konversi huruf ke huruf kecil untuk pengecekan yang
        // lebih mudah
        char huruf_kecil = tolower(huruf);
        // Cek apakah huruf adalah huruf vokal
        if (huruf_kecil == 'a' || huruf_kecil == 'e' ||
            huruf_kecil == 'i' ||
            huruf_kecil == 'o' || huruf_kecil == 'u') {
            jumlah++;
        }
        // Cek huruf vokal kapital
        else if (huruf == 'A' || huruf == 'E' || huruf == 'I' ||
            huruf == 'O' || huruf == 'U') {
            jumlah++;
        }
    }
    return jumlah;
}

int main() {
    string kalimat;
    cout << "Masukkan kalimat: ";
    getline(cin, kalimat);

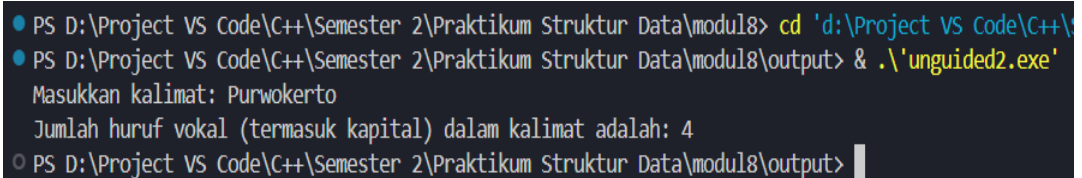
    int jumlah_vokal = hitungVokal(kalimat);

    cout << "Jumlah huruf vokal (termasuk kapital) dalam kalimat
```

```
adalah: " << jumlah_vokal << endl;

    return 0;
}
```

Screenshoot program



```
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8> cd 'd:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output> & .\'unguided2.exe\'
Masukkan kalimat: Purwokerto
Jumlah huruf vokal (termasuk kapital) dalam kalimat adalah: 4
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output>
```

Deskripsi program

Program ini adalah implementasi sederhana untuk menghitung jumlah huruf vokal (termasuk huruf vokal kapital) dalam sebuah kalimat yang dimasukkan oleh pengguna. Program menggunakan fungsi `hitungVokal` untuk melakukan perhitungan tersebut.

Fungsi `hitungVokal` menerima sebuah parameter berupa string `kalimat` dan mengembalikan jumlah huruf vokal dalam kalimat tersebut. Fungsi ini menggunakan loop `for` yang mengiterasi setiap karakter dalam kalimat. Setiap karakter diubah menjadi huruf kecil menggunakan fungsi `tolower` untuk memudahkan pengecekan. Jika karakter adalah huruf vokal (baik huruf kecil maupun huruf kapital), jumlah vokal akan ditambahkan.

Dalam `main` function, pengguna diminta memasukkan sebuah kalimat. Kalimat tersebut disimpan dalam variabel `kalimat`. Selanjutnya, fungsi `hitungVokal` dipanggil dengan parameter `kalimat` dan hasilnya disimpan dalam variabel `jumlah_vokal`. Akhirnya, program akan mencetak jumlah huruf vokal (termasuk huruf vokal kapital) dalam kalimat tersebut.

3. Unguided 3

Source code

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

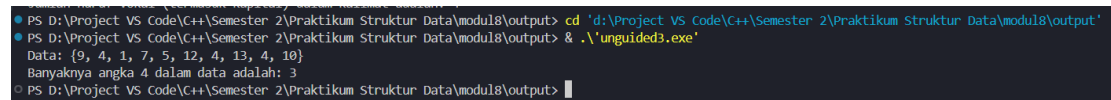
int main() {
    int n = 10;
    int data[n] = {9, 4, 1, 7, 5, 12, 4, 13, 4, 10};
    int cari = 4;
    int jumlah = 0;

    // Algoritma Sequential Search untuk menghitung jumlah angka
    4
    for (int i = 0; i < n; i++) {
        if (data[i] == cari) {
            jumlah++;
        }
    }

    cout << "Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}" << endl;
    cout << "Banyaknya angka " << cari << " dalam data adalah: "
    << jumlah << endl;

    return 0;
}
```

Screenshoot program



```
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output> cd 'd:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output'
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output> & .\unguided3.exe
Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}
Banyaknya angka 4 dalam data adalah: 3
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul8\output> |
```

Deskripsi program

Program ini menggunakan algoritma pencarian sekuensial (sequential search) untuk menghitung jumlah kemunculan angka 4 dalam sebuah larik bilangan bulat.

Pertama, program mendefinisikan sebuah larik ``data`` dengan 10 elemen dan menginisialisasinya dengan beberapa nilai. Kemudian, program mendefinisikan variabel ``cari`` yang menyimpan nilai yang ingin dicari, dalam hal ini adalah angka 4. Selanjutnya, program menggunakan sebuah loop ``for`` untuk mengiterasi melalui setiap elemen dalam larik ``data``. Setiap kali nilai elemen sama dengan nilai yang dicari (``cari``), variabel ``jumlah`` akan ditambah satu.

Setelah selesai mengiterasi semua elemen, program akan mencetak larik ``data`` dan jumlah kemunculan angka yang dicari (``cari``) dalam larik tersebut.

BAB IV

KESIMPULAN

Pencarian data dalam algoritma dapat dilakukan dengan dua metode utama, yaitu sequential search (pencarian sekuensial) dan binary search (pencarian biner). Sequential search bekerja dengan membandingkan setiap elemen dalam daftar secara berurutan dengan elemen yang dicari, sementara binary search membagi daftar menjadi dua bagian dan hanya memeriksa satu bagian yang mungkin mengandung elemen yang dicari. Meskipun sequential search sederhana, namun memiliki kompleksitas waktu yang tinggi terutama pada daftar yang besar. Di sisi lain, binary search lebih efisien karena membutuhkan lebih sedikit langkah, terutama pada daftar yang besar dan sudah diurutkan sebelumnya.