

LAPORAN PRAKTIKUM

MODUL 8 SEARCHING



**Disusun oleh:
Rakha Yudhistira
NIM: 2311102010**

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

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.

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.

BAB III

GUIDED

1. Guided 1

Source code

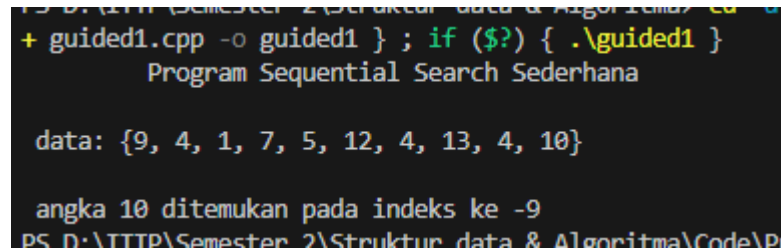
```
#include <iostream>
using namespace std;
int main()
{
    int n = 10;
    int data[n] = {9, 4, 1, 7, 5, 12, 4, 13, 4, 10};
    int cari = 10;
    bool ketemu = false;
    int i;
    // algoritma Sequential Search
    for (i = 0; i < n; i++)
    {
        if (data[i] == cari)
        {
            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:\TTP\Semester 2\Struktur data & Algoritma> g++ + guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
Program Sequential Search Sederhana

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

angka 10 ditemukan pada indeks ke -9
PS D:\TTP\Semester 2\Struktur data & Algoritma> Code\Pr

```

Deskripsi program

Program ini bertujuan untuk mencari nilai tertentu dalam sebuah array data menggunakan algoritma sequential search. Program ini secara efektif menerapkan algoritma sequential search untuk mencari nilai tertentu dalam array data. Program ini mudah dipahami dan diimplementasikan, dan dapat dimodifikasi untuk digunakan dengan array data dan nilai pencarian yang berbeda.

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()
{
    // searching
    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

```
odul 8\" ; if ($?) { g++ guided2.cpp -o gui
    BINARY SEARCH

Data : 1 8 2 5 4 9 7

Masukkan data yang ingin Anda cari :5

Data diurutkan : 1 2 4 5 7 8 9

Data ditemukan pada index ke-3
```

Deskripsi program

Program ini menggabungkan dua algoritma, selection sort dan binary search, untuk mencapai efisiensi dalam pengurutan dan pencarian data. Selection sort digunakan untuk mengurutkan data sebelum pencarian, dan binary search digunakan untuk mencari data yang diurutkan dengan cepat.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

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

void selection_sort(char arr[], int n)
{
    int min;
    char temp;
    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;
            }
        }
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}

void binary_search(char arr[], int n, char target)
{
    int awal = 0, akhir = n - 1, tengah;
    bool ditemukn = false;
    while (!ditemukn && awal <= akhir)
```

```

    {
        tengah = (awal + akhir) / 2;
        if (arr[tengah] == target)
        {
            ditemukn = true;
            break;
        }
        else if (arr[tengah] < target)
        {
            awal = tengah + 1;
        }
        else
        {
            akhir = tengah - 1;
        }
    }
    if (ditemukn)
        cout << "\nHuruf ditemukan pada urutan ke-" << (tengah+1)
<< endl;
    else
        cout << "\nHuruf tidak ditemukan\n";
}

int main()
{
    char kalimat[100], huruf;
    cout << "\nMasukkan sebuah kalimat    : ";
    cin.getline(kalimat, 100);
    int panjang = 0;
    while (kalimat[panjang] != 0 )
    {
        panjang++;
    }
    selection_sort(kalimat, panjang);

```

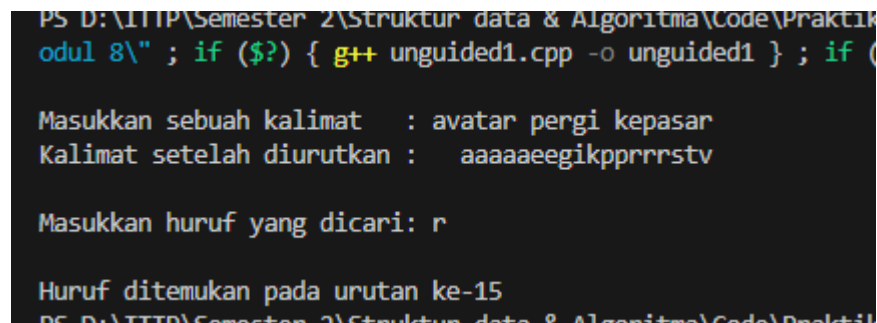
```

    cout << "Kalimat setelah diurutkan : " << kalimat << endl;
    cout << "\nMasukkan huruf yang dicari: ";
    cin >> huruf;
    binary_search(kalimat, panjang, huruf);

    return 0;
}

```

Screenshoot program



The screenshot shows a terminal window with the following output:

```

PS D:\ITIP\Semester 2\Struktur data & Algoritma\Code\Praktik
odul 8\" ; if ($?) { g++ unguided1.cpp -o unguided1 } ; if (

Masukkan sebuah kalimat : avatar pergi kepasar
Kalimat setelah diurutkan : aaaaaeegikpprrrstv

Masukkan huruf yang dicari: r

Huruf ditemukan pada urutan ke-15
PS D:\ITIP\Semester 2\Struktur data & Algoritma\Code\Praktik

```

Deskripsi program

Program ini mengurutkan kalimat yang dimasukkan pengguna dari A ke Z menggunakan algoritma selection sort, dan kemudian mencari huruf tertentu dalam kalimat yang sudah diurutkan tersebut menggunakan algoritma binary search. Program ini menampilkan hasil pencariannya, apakah huruf yang dicari ditemukan atau tidak, dan pada urutan keberapa jika ditemukan.

2. Unguided 2

Source code

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

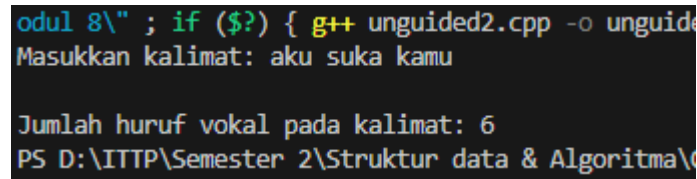
bool is_vokal(char c) {
    char vokal[] = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O',
'U'};
    for (int i = 0; i < 10; i++) {
        if (c == vokal[i]) {
            return true;
        }
    }
    return false;
}

int main() {
    char kalimat[100];
    cout << "Masukkan kalimat: ";
    cin.getline(kalimat, 100);

    int jumlah_vokal = 0;
    for (int i = 0; kalimat[i] != 0; i++) {
        if (is_vokal(kalimat[i])) {
            jumlah_vokal++;
        }
    }

    cout << "\nJumlah huruf vokal pada kalimat: " << jumlah_vokal
<< endl;
    return 0;
}
```

Screenshoot program



```
odul 8\" ; if ($?) { g++ unguided2.cpp -o unguide
Masukkan kalimat: aku suka kamu

Jumlah huruf vokal pada kalimat: 6
PS D:\ITTP\Semester 2\Struktur data & Algoritma\
```

Deskripsi program

Program ini berfungsi untuk menghitung jumlah huruf vokal dalam sebuah kalimat dengan cara mendefinisikan fungsi `is_vokal` untuk memeriksa apakah suatu karakter adalah vokal dan kemudian menghitung jumlah karakter vokal dalam kalimat yang dimasukkan pengguna.

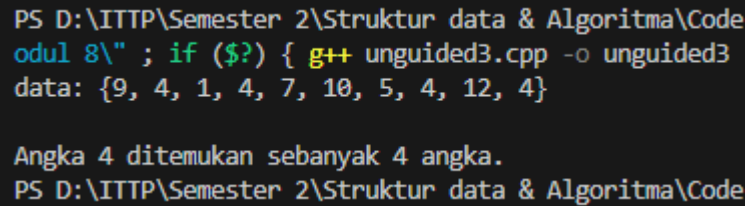
3. Unguided 3

Source code

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

int main()
{
    int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int n = 10;
    int cari = 4;
    int count = 0;
    for (int i = 0; i < n; i++)
    {
        if (data[i] == cari)
        {
            count++;
        }
    }
    cout << "data: {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}" << endl;
    cout << "\nAngka " << cari << " ditemukan sebanyak " << count
    << " angka." << endl;
    return 0;
}
```

Screenshoot program

A screenshot of a terminal window with a dark background. The text is displayed in a monospaced font with syntax highlighting. The first line shows the command to compile a C++ file. The second line shows the output of the program, which is a message indicating that the number 4 was found 4 times in the array. The third line shows the prompt for the next command.

```
PS D:\ITTP\Semester 2\Struktur data & Algoritma\Code
odul 8\" ; if ($?) { g++ unguided3.cpp -o unguided3
data: {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}

Angka 4 ditemukan sebanyak 4 angka.
PS D:\ITTP\Semester 2\Struktur data & Algoritma\Code
```

Deskripsi program

Program C++ ini secara efisien menghitung kemunculan angka tertentu dalam array. Ini dimulai dengan mendeklarasikan data array yang berisi sepuluh nilai integer. Variabel *n* menyimpan ukuran array (10 dalam kasus ini). Ia kemudian mendefinisikan *cari* untuk menampung nomor yang frekuensinya perlu ditentukan (setel ke 4 di sini). Dengan menggunakan perulangan *for*, program melakukan iterasi melalui setiap elemen dalam array. Di dalam loop, ia memeriksa apakah elemen saat ini (*data[i]*) cocok dengan nomor target (*cari*). Jika ada kecocokan, jumlah variabel penghitung bertambah. Setelah melakukan iterasi pada semua elemen, program akan mencetak array asli diikuti dengan pesan yang menunjukkan berapa kali nomor target (*cari*) ditemukan dalam array.

BAB IV

KESIMPULAN

Struktur data searching menyediakan berbagai metode untuk menemukan elemen dalam kumpulan data dengan efisiensi dan kompleksitas yang berbeda-beda. Memahami metode dan karakteristiknya membantu memilih metode yang tepat untuk aplikasi yang sesuai. **Pencarian (Searching)** adalah proses menemukan elemen tertentu dalam kumpulan data. Dalam struktur data, terdapat berbagai metode pencarian yang dapat digunakan, dengan efisiensi dan kompleksitas yang berbeda-beda.

DAFTAR PUSTAKA

Asisten Praktikum. (2024). MODUL 8 SEARCHING. Learning Managament System

Karumanchi, N. (2016). Data Structures and algorithms made easy: Concepts, problems, Interview Questions. CareerMonk Publications.