LAPORAN PRAKTIKUM

MODUL VIII
ALGORITMA SEARCHING

Disusun oleh:
Nadhif Atha Zaki
NIM: 2311102007

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

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

DASAR TEORI

Pencarian (Searching) adalah proses menemukan nilai tertentu dalam sebuah kumpulan data. Hasil pencarian dapat berupa data yang ditemukan, lebih dari satu data yang ditemukan, atau ketiadaan data. Proses pencarian juga dapat dianggap sebagai pencarian data dalam sebuah array dengan memeriksa setiap index secara berurutan menggunakan teknik perulangan. Ada dua metode utama dalam algoritma Searching, yaitu Sequential Search.

A. Sequential Search

Sequential Search adalah algoritma pencarian yang sering digunakan untuk data yang tidak terurut atau berpola acak. Algoritma ini mencari data dalam array dengan membaca setiap elemen secara berurutan, mulai dari index pertama hingga terakhir. Proses pencarian akan berhenti ketika data ditemukan. Jika tidak ditemukan, pencarian akan berakhir setelah melewati seluruh array. Proses perulangan pada pencarian Sequential Search akan dilakukan sebanyak jumlah elemen dalam array.

Algoritma pencarian berurutan dapat dijelaskan sebagai berikut: Pertama, indeks i diatur ke 0 dan variabel ketemu diatur ke false. Selama ketemu masih false dan i kurang dari atau sama dengan jumlah elemen dalam array, program akan memeriksa setiap elemen. Jika data ditemukan pada index i, variabel ketemu akan diubah menjadi true. Jika tidak, indeks i akan ditingkatkan. Jika data ditemukan (ketemu true), indeks i akan menunjukkan posisi data tersebut. Jika tidak, artinya data tidak ditemukan dalam array.

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.

B. Binary Search

Binary Search merupakan algoritma pencarian yang termasuk dalam kategori interval search, yang digunakan pada array atau list yang memiliki elemen terurut. Dalam metode ini, langkah pertama adalah mengurutkan data, kemudian data dibagi secara logis menjadi dua bagian pada setiap tahap pencarian. Algoritma Binary Search sering dikombinasikan dengan algoritma sorting karena data harus diurutkan terlebih dahulu sebelum pencarian dilakukan. Konsep dasar dari Binary Search adalah sebagai berikut:

Data yang akan dicari diambil dari posisi 1 hingga posisi akhir N dalam array atau list yang sudah terurut.

Setiap tahap pencarian, data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.

Data yang dicari akan dibandingkan dengan data yang berada di posisi tengah untuk menentukan apakah lebih besar atau lebih kecil.

Jika data yang dicari lebih besar dari data tengah, maka pencarian dilanjutkan pada bagian kanan dari data tengah dengan pembagian data menjadi dua bagian.

Jika data yang dicari lebih kecil dari data tengah, pencarian dilanjutkan pada bagian kiri dengan pembagian data menjadi dua bagian.

Proses pencarian terus dilanjutkan dengan membagi data menjadi dua sampai data yang dicari ditemukan atau seluruh bagian telah diperiksa.

Jika data yang dicari ditemukan, pencarian dihentikan. Jika data belum ditemukan, pencarian akan terus dilanjutkan dengan membagi data menjadi dua.

Binary Search merupakan algoritma yang efisien untuk pencarian dalam data yang sudah terurut, karena pada setiap iterasi jumlah data yang perlu diperiksa berkurang menjadi setengah dari iterasi sebelumnya, sehingga kompleksitas waktu yang dihasilkan adalah O(log n).

BAB III

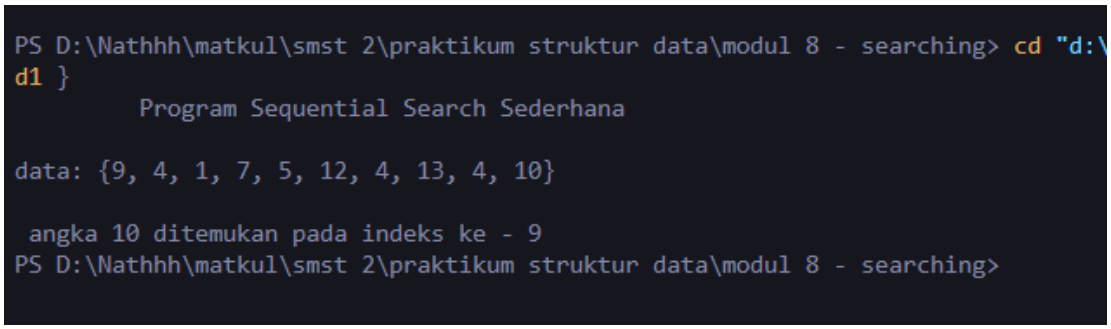
GUIDED

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:



Deskripsi program:

Program di atas merupakan implementasi sederhana dari algoritma Sequential Search dalam bahasa C++. Program ini digunakan untuk mencari angka tertentu dalam sebuah array. Pada awalnya, program mendeklarasikan sebuah array data dengan 10 elemen yang berisi angka-angka: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}. Variabel cari diinisialisasi dengan nilai 10, yang merupakan angka yang akan dicari dalam array tersebut. Selain itu, terdapat variabel ketemu yang merupakan variabel boolean untuk menandai apakah angka yang dicari ditemukan atau tidak, dengan nilai awal false. Variabel i digunakan sebagai indeks untuk iterasi melalui elemen-elemen array.

2. Guided 2

Source code

```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;
}
```

Screenshoot program:

```
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 8 - searching> cd "
d2 }
        BINARY SEARCH

Data awal:   1  8  2  5  4  9  7

Masukkan data yang ingin Anda cari: 9

Data diurutkan:   1  2  4  5  7  8  9

Data ditemukan pada index ke-6
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 8 - searching>
```

Deskripsi program:

Program di atas adalah implementasi gabungan dari algoritma Selection Sort dan Binary Search dalam bahasa C++. Pertama-tama, program mendeklarasikan sebuah array arrayData yang berisi 7 elemen: {1, 8, 2, 5, 4, 9, 7} serta sebuah variabel cari untuk menyimpan nilai yang akan dicari. Fungsi selection_sort diimplementasikan untuk mengurutkan elemen-elemen dalam array. Dalam fungsi ini, elemen-elemen array diurutkan dengan menemukan elemen terkecil dan menukarnya dengan elemen di posisi awal.

LATIHAN KELAS - UNGUIDED

1.

Buatlah sebuah program untuk mencari sebuah huruf pada sebuah kalimat yang sudah di input dengan menggunakan Binary Search!

Source code

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

// Fungsi untuk melakukan Binary Search
bool binarySearch(const string &sortedStr, char target)
{
    int left = 0;
    int right = sortedStr.length() - 1;

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

        if (sortedStr[middle] == target)
        {
            return true; // Huruf ditemukan
        }
        else if (sortedStr[middle] < target)
        {
            left = middle + 1;
        }
        else
        {
            right = middle - 1;
        }
    }
    return false; // Huruf tidak ditemukan
}

int main()
```

```
{
    string inputStr;
    char targetChar;

    // Mengambil input dari pengguna
    cout << "Masukkan sebuah kalimat: ";
    getline(cin, inputStr);

    // Mengurutkan kalimat
    sort(inputStr.begin(), inputStr.end());

    // Mengambil input huruf yang ingin dicari
    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> targetChar;

    // Mencari huruf menggunakan Binary Search
    if (binarySearch(inputStr, targetChar))
    {
        cout << "Huruf '" << targetChar << "' ditemukan dalam
kalimat." << endl;
    }
    else
    {
        cout << "Huruf '" << targetChar << "' tidak ditemukan
dalam kalimat." << endl;
    }

    return 0;
}
```

Screenshoot program

```
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 8 - searching> cd "d:\Nat
ided1 }
Masukkan sebuah kalimat: nadhif
Masukkan huruf yang ingin dicari: i
Huruf 'i' ditemukan dalam kalimat.
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 8 - searching>
```

Deskripsi dan Penjelasan Program:

Program di atas merupakan implementasi dari algoritma Binary Search dalam bahasa C++ untuk mencari sebuah huruf dalam sebuah kalimat yang sudah diurutkan. Program ini dimulai dengan mendeklarasikan fungsi binarySearch yang menerima string terurut dan karakter target sebagai parameter. Fungsi ini menggunakan metode Binary Search untuk menentukan apakah karakter target ada dalam string tersebut, dengan memeriksa elemen tengah dari bagian yang relevan dari string dalam loop.

Pada fungsi main, program meminta pengguna untuk memasukkan sebuah kalimat menggunakan getline(cin, inputStr), sehingga seluruh kalimat termasuk spasi dapat diambil sebagai input. Selanjutnya, kalimat tersebut diurutkan menggunakan fungsi sort dari library <algorithm>, sehingga string menjadi terurut berdasarkan urutan karakter ASCII.

Setelah kalimat diurutkan, program meminta pengguna untuk memasukkan huruf yang ingin dicari dalam kalimat tersebut. Dengan menggunakan fungsi binarySearch, program memeriksa apakah huruf tersebut ada dalam string yang sudah diurutkan.

Jika huruf ditemukan, program mencetak pesan bahwa huruf tersebut ditemukan dalam kalimat. Sebaliknya, jika huruf tidak ditemukan, program mencetak pesan bahwa huruf tersebut tidak ditemukan. Program ini menunjukkan bagaimana Binary Search dapat digunakan pada string yang sudah diurutkan untuk mencari karakter tertentu dengan efisien.

2. .

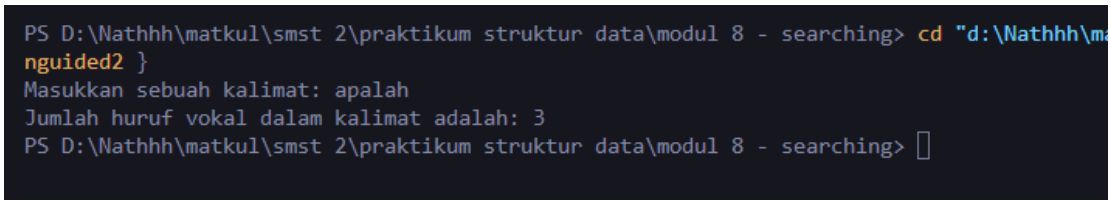
Buatlah sebuah program yang dapat menghitung banyaknya huruf vocal dalam sebuah kalimat!

Souce Code:

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

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

    for (char c : kalimat)
    {
        if (vokal.find(c) != string::npos)
        {
            jumlahVokal++;
        }
    }

    return jumlahVokal;
}

int main()
{
    string kalimatInput;

    // Mengambil input dari pengguna
    cout << "Masukkan sebuah kalimat: ";
    getline(cin, kalimatInput);

    // Menghitung jumlah huruf vokal
    int jumlahVokal = hitungVokal(kalimatInput);

    // Menampilkan hasil
```

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

    return 0;
}
```

Screenshot Program:



Deskripsi dan Penjelasan Program:

Program di atas adalah sebuah implementasi dalam bahasa C++ yang digunakan untuk menghitung jumlah huruf vokal dalam sebuah kalimat. Fungsi hitungVokal didefinisikan untuk menerima sebuah string sebagai parameter dan mengembalikan jumlah huruf vokal di dalamnya. String vokal berisi semua huruf vokal, baik huruf kecil maupun huruf besar.

Dalam fungsi hitungVokal, sebuah loop for digunakan untuk memeriksa setiap karakter dalam string kalimat. Jika karakter tersebut ditemukan dalam string vokal, maka penghitung vokal (jumlahVokal) akan ditambah satu. Fungsi ini kemudian mengembalikan nilai dari jumlahVokal setelah loop selesai.

Pada fungsi main, program meminta pengguna untuk memasukkan sebuah kalimat menggunakan getline(cin, kalimatInput) untuk

memastikan seluruh kalimat, termasuk spasi, ditangkap. Setelah kalimat diinput oleh pengguna, program memanggil fungsi hitungVokal untuk menghitung jumlah huruf vokal dalam kalimat tersebut.

Hasil perhitungan jumlah huruf vokal kemudian ditampilkan kepada pengguna dengan menggunakan cout. Program ini merupakan contoh sederhana bagaimana string dapat diproses untuk menghitung frekuensi karakter tertentu, dalam hal ini huruf vokal, yang sangat berguna dalam berbagai aplikasi pemrosesan teks.

3. .

Diketahui data = 9, 4, 1, 4, 7, 10, 5, 4, 12, 4. Hitunglah berapa banyak angka 4 dengan menggunakan algoritma Sequential Search!

Souce Code:

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

// Fungsi untuk menghitung berapa banyak angka 4 menggunakan
Sequential Search
int hitungAngka4(const int data[], int ukuran)
{
    int jumlah4 = 0;

    for (int i = 0; i < ukuran; ++i)
    {
        if (data[i] == 4)
        {
            jumlah4++;
        }
    }

    return jumlah4;
}

int main()
```

```
{
    // Data yang diberikan
    int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int ukuran = sizeof(data) / sizeof(data[0]);

    // Menghitung jumlah angka 4
    int jumlah4 = hitungAngka4(data, ukuran);

    // Menampilkan hasil
    cout << "Jumlah angka 4 dalam data adalah: " << jumlah4 <<
endl;

    return 0;
}
```

Screenshot Program:

```
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 8 - searching> cd
nguided3 }
Jumlah angka 4 dalam data adalah: 4
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 8 - searching>
```

Deskripsi dan Penjelasan Program:

Program di atas merupakan sebuah implementasi dalam bahasa C++ yang digunakan untuk menghitung berapa banyak angka 4 yang terdapat dalam sebuah array menggunakan algoritma Sequential Search. Fungsi hitungAngka4 didefinisikan untuk menerima array dan ukuran array sebagai parameter dan mengembalikan jumlah kemunculan angka 4 dalam array tersebut.

Dalam fungsi hitungAngka4, dilakukan iterasi melalui setiap elemen dalam array menggunakan loop for. Setiap elemen dicek apakah sama dengan angka 4. Jika iya, variabel jumlah4 yang bertindak sebagai penghitung akan bertambah satu. Setelah selesai melakukan iterasi, fungsi ini akan mengembalikan jumlah kemunculan angka 4 dalam array tersebut.

Pada fungsi main, array data yang berisi serangkaian angka disediakan secara langsung. Selanjutnya, ukuran array dihitung menggunakan sizeof(data) / sizeof(data[0]). Hal ini dilakukan untuk memastikan program dapat bekerja dengan array berukuran apapun.

Setelah itu, program memanggil fungsi hitungAngka4 dengan menyediakan array dan ukuran array sebagai argumen, dan menyimpan hasilnya dalam variabel jumlah4. Hasil perhitungan jumlah angka 4 kemudian ditampilkan kepada pengguna menggunakan cout. Program ini memberikan contoh bagaimana algoritma Sequential Search dapat diterapkan untuk mencari jumlah kemunculan suatu nilai tertentu dalam sebuah array dengan sederhana dan efisien.

BAB IV

KESIMPULAN

Sequential Search:

Sequential Search adalah algoritma pencarian sederhana yang efektif untuk mencari elemen dalam sebuah daftar atau array.

Algoritma ini bekerja dengan memeriksa setiap elemen satu per satu dari awal hingga akhir, dan pencarian berhenti saat elemen yang dicari ditemukan atau semua elemen telah diperiksa.

Keuntungan dari Sequential Search adalah kemudahan implementasi dan aplikasinya pada data yang tidak terurut.

Binary Search:

Binary Search adalah algoritma pencarian yang efisien untuk mencari elemen dalam sebuah array yang sudah diurutkan.

Algoritma ini bekerja dengan membagi array menjadi dua bagian dan mencari nilai target dengan membandingkannya dengan elemen tengah.

Binary Search memiliki kompleksitas waktu O(log n), yang membuatnya sangat cepat untuk mencari dalam data yang besar.

Namun, persyaratan utamanya adalah data harus sudah diurutkan terlebih dahulu sebelum pencarian dapat dilakukan.

Implementasi Praktis:

Implementasi dari algoritma pencarian dapat disesuaikan dengan kebutuhan aplikasi dan jenis data yang dihadapi.

Penggunaan algoritma pencarian yang tepat sangat tergantung pada sifat data, seperti apakah data sudah terurut atau tidak, serta kompleksitas waktu yang diinginkan.

Kompleksitas Waktu:

Kompleksitas waktu merupakan faktor penting dalam memilih algoritma pencarian yang tepat.

Algoritma dengan kompleksitas waktu yang lebih rendah akan lebih efisien dalam menangani data yang besar.

Dengan memahami berbagai jenis algoritma pencarian dan karakteristiknya, kita dapat memilih algoritma yang paling sesuai dengan kebutuhan spesifik dalam menyelesaikan masalah pencarian dalam pemrograman.