LAPORAN PRAKTIKUM
MODUL VII
ALGORITMA SEARCHING

Disusun oleh:

Raihan Ramadhan

NIM: 2311102040

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

Praktikum ini bertujuan untuk memastikan bahwa mahasiswa memiliki pemahaman yang baik tentang apa itu algoritma search, bagaimana cara kerjanya, dan mengapa penting dalam pemrograman. Mahasiswa diharapkan dapat menjelaskan dengan jelas apa itu algoritma search dan bagaimana ia bekerja sebagai bagian dari struktur data. Praktikum ini memberikan kesempatan kepada mahasiswa untuk mengimplementasikan algoritma search dalam bahasa pemrograman c++. Melalui latihan konkret, mahasiswa dapat belajar bagaimana mendefinisikan dan menggunakan algoritma search dalam konteks nyata. Mahasiswa mungkin akan diberikan situasi atau masalah yang memerlukan penggunaan algoritma search, dan mereka akan belajar bagaimana cara menguji keefektifan solusi mereka menggunakan algoritma search.

BAB II
DASAR TEORI

Pencarian (Searching) adalah proses menemukan nilai tertentu dalam sekumpulan data. Hasil dari pencarian dapat berupa salah satu dari tiga kemungkinan: data ditemukan, data ditemukan lebih dari sekali, atau data tidak ditemukan. Pencarian juga dapat diartikan sebagai proses mencari data dalam sebuah array dengan memeriksa satu per satu setiap indeks baris atau kolom menggunakan teknik perulangan untuk menemukan data tersebut.

a. Sequential Search adalah salah satu algoritma pencarian yang sering digunakan untuk data yang acak atau belum terurut. Algoritma ini juga merupakan teknik pencarian data dalam array yang paling sederhana, di mana data dalam array dibaca satu per satu, baik dari indeks terkecil ke 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 ← 0

    - ketemu ← false
    - 3) Selama (tidak ketemu) dan (i <= N) kerjakan baris 4
    - 4) Jika (Data[i] = x) maka ketemu ← true, jika tidak i ← i + 1
    - 5) Jika (ketemu) maka i adalah indeks dari data yang dicari, jika tidak data tidak ditemukan

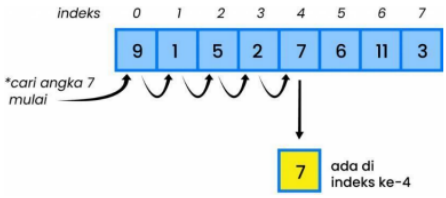
Di bawah ini merupakan fungsi untuk mencari data menggunakan pencarian sekuensial.

```
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}

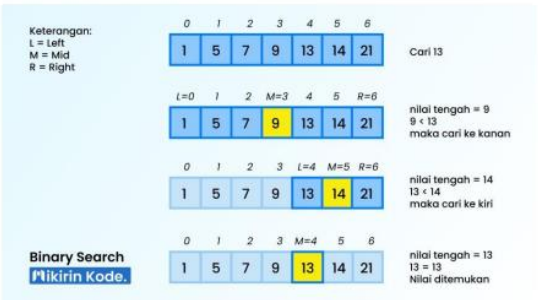


b. Binary Search

Binary Search adalah bagian dari interval search, yaitu algoritma pencarian yang diterapkan pada array atau daftar dengan elemen yang sudah terurut. Metode ini memerlukan data yang sudah diurutkan, di mana data dibagi menjadi dua bagian (secara logis) pada setiap tahap pencarian. Dalam praktiknya, algoritma ini sering dikombinasikan dengan algoritma sorting karena data yang akan dicari harus sudah terurut sebelumnya. Konsep Binary Search:

- Data diambil dari posisi 1 sampai posisi akhir N.
- Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
- Apabila data yang dicari lebih besar dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kanan dari data tengah.
- Apabila data yang dicari lebih kecil dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kiri dari data tengah.
- 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.

Contoh dari Binary Search, yaitu:



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;
}
```

Screenshot Output

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 8> cd
($?) { .\Guided_1 }
        Program Sequential Search Sederhana

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

 angka 10 ditemukan pada indeks ke -9
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 8>
```

Deskripsi Program

Kode program tersebut adalah implementasi dari algoritma Sequential Search untuk mencari elemen dalam array. Pertama, sebuah array data dengan 10 elemen didefinisikan, dan variabel cari diatur untuk mencari nilai 10. Algoritma Sequential Search dilakukan dengan perulangan yang memeriksa setiap elemen dalam array dari indeks 0 hingga indeks 9. Jika elemen yang dicari ditemukan, variabel ketemu diatur menjadi true dan perulangan dihentikan dengan break. Hasil pencarian kemudian ditampilkan: jika elemen ditemukan, program akan mencetak indeks di mana elemen tersebut ditemukan; jika tidak, program akan mencetak bahwa elemen tidak ditemukan.

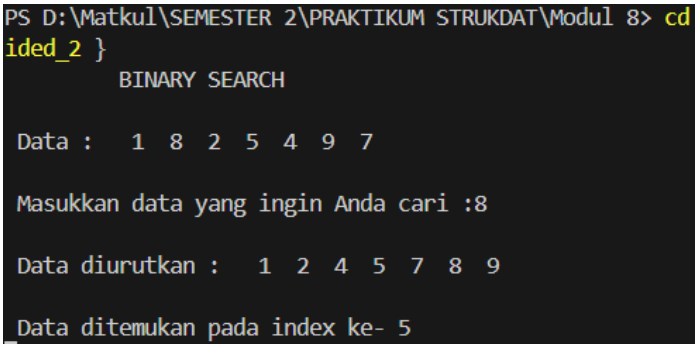
2. Guided 2

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;
}
```

Screenshot Output



Deskripsi Program

Program tersebut mengimplementasikan algoritma Selection Sort dan Binary Search untuk mencari elemen dalam array. Pertama, array data berisi tujuh elemen didefinisikan dan pengguna diminta memasukkan nilai yang ingin dicari. Algoritma Selection Sort digunakan untuk mengurutkan array data dari nilai terkecil ke terbesar. Setelah data diurutkan, algoritma Binary Search digunakan untuk mencari elemen yang dimasukkan pengguna dalam array yang sudah terurut. Hasil pencarian ditampilkan, menunjukkan apakah elemen ditemukan beserta indeksnya atau tidak ditemukan.

BAB IV

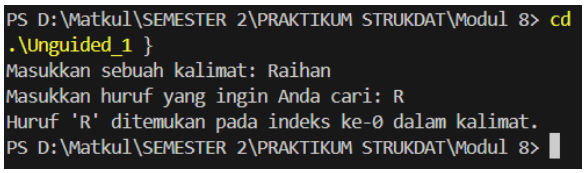
UNGUIDED

1. Unguided 1

Source Code

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

int binary_search(const string &str, char target) {
    int left = 0, right = str.length() - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (str[mid] == target) {
            return mid;
        } else if (str[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}
int main() {
    string kalimat;
    char cari;

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

    cout << "Masukkan huruf yang ingin Anda cari: ";
    cin >> cari;
    sort(kalimat.begin(), kalimat.end());

    int index = binary_search(kalimat, cari);

    if (index != -1) {
        cout << "Huruf '" << cari << "' ditemukan pada indeks ke-" <<
index << " dalam kalimat." << endl;
    } else {
        cout << "Huruf '" << cari << "' tidak ditemukan dalam kalimat."
<< endl;
    }

    return 0;
}
```

Screenshot Output



Deskripsi Program

Program tersebut merupakan yang bertujuan untuk mencari sebuah huruf dalam sebuah kalimat yang diinputkan oleh pengguna. Pertama-tama, pengguna diminta untuk memasukkan sebuah kalimat, kemudian diminta untuk menginputkan huruf yang ingin dicari. Selanjutnya, kalimat tersebut diurutkan menggunakan fungsi sort dari STL (Standard Template Library). Setelah proses pengurutan selesai, program akan melakukan pencarian menggunakan algoritma Binary Search untuk menemukan huruf yang dimasukkan pengguna dalam kalimat yang sudah diurutkan. Hasil pencarian akan ditampilkan, menunjukkan apakah huruf tersebut ditemukan beserta indeksnya atau tidak ditemukan.

2. **Unguided 2**
   SourceCode

```cpp
#include <iostream>
#include <string>

using namespace std;

int hitungVokal(const string &kalimat)
{
    string vokal = "aiueoAIUEO";
    int jumlah_vokal = 0;

    for (char karakter : kalimat)
    {
        for (char v : vokal)
        {
            if (karakter == v)
            {
                jumlah_vokal++;
                break;
            }
        }
    }

    return jumlah_vokal;
}
```

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

    int jumlah = hitungVokal(kalimat);
    cout << "Jumlah huruf vokal dalam kalimat adalah: " << jumlah << endl;

    return 0;
}
```

Screenshot Output

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 8> cd
.\Unguided_2 }
Masukkan sebuah kalimat: Raihan
Jumlah huruf vokal dalam kalimat adalah: 3
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 8>
```

Deskripsi Program

Program ini menghitung jumlah huruf vocal menggunakan sequential search dalam sebuah kalimat yang dimasukkan oleh pengguna. Fungsi hitungVokal menerima sebuah string dan menghitung jumlah vokal dengan memeriksa setiap karakter dalam kalimat terhadap string vokal yang berisi huruf vokal. Pengguna diminta memasukkan kalimat melalui`getline(cin, kalimat). Fungsi hitungVokal kemudian dipanggil untuk menghitung jumlah vokal dalam kalimat tersebut. Hasilnya ditampilkan dengan cout, menunjukkan jumlah huruf vokal dalam kalimat yang dimasukkan.

3. Unguided 3

Source code

```cpp
#include <iostream>

int sequentialSearch(int arr[], int n, int key)
{
    int count = 0;
    for (int i = 0; i < n; ++i)
    {
        if (arr[i] == key)
        {
            count++;
        }
    }
    return count;
}
```

```cpp
int main()
{
    int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int key = 4;
    int n = sizeof(data) / sizeof(data[0]);

    int count = sequentialSearch(data, n, key);
    std::cout << "Jumlah angka " << key << " dalam array adalah: " << count
<< std::endl;

    return 0;
}
```

Screenshot Output

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 8> cd
.\Unguided_3 }
Jumlah angka 4 dalam array adalah: 4
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 8>
```

Deskripsi Program

Program tersebut merupakan implementasi algoritma Sequential Search. Algoritma tersebut digunakan untuk mencari jumlah kemunculan suatu angka dalam array. Pertama, terdapat fungsi sequentialSearch yang melakukan iterasi pada array untuk menghitung jumlah kemunculan angka yang dicari. Selanjutnya, dalam fungsi main, array data disimpan dalam variabel data, dan angka yang dicari ditetapkan sebagai key. Program kemudian memanggil fungsi pencarian dan menampilkan jumlah kemunculan angka tersebut.

BAB V

KESIMPULAN

Setelah menganalisis dan mengimplementasikan algoritma sequential search dan binary search berdasarkan parameter waktu pencarian, dapat disimpulkan bahwa binary search menawarkan performa pencarian yang lebih cepat daripada sequential search. Binary search cenderung stabil dan memerlukan waktu yang singkat untuk menemukan data. Di sisi lain, sequential search membutuhkan waktu lebih lama dan kecepatannya bervariasi tergantung pada lokasi data dalam array. Meskipun sequential search bisa cepat jika data ditemukan di awal, pencarian akan menjadi lebih lambat jika data berada di akhir array. Oleh karena itu, binary search lebih sesuai untuk aplikasi seperti pencarian data dalam aplikasi dengan jumlah data yang besar.

DAFTAR PUSTAKA

Asisten Praktikum “MODUL 8 ALGOTIMA SEARCHING”. Learning Management System,31 Mei 2024.

Markuci Dian, Prianto Cahyo,” ANALISIS PERBANDINGAN PENGGUNAAN ALGORITMA SEQUENTIAL SEARCH DAN BINARY SEARCH PADA APLIKASI SURAT PERJALANAN DINAS”.Chrome,2 Juni 2024. https://ejournal.itn.ac.id/index.php/jati/article/download/4569/3037 .