LAPORAN PRAKTIKUM

MODUL VIII
ALGORITMA SEARCHING

Disusun oleh:
Maulana Ghani Rolanda
NIM: 2311102012

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023

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

Linear search adalah algoritma pencarian yang paling sederhana di mana kita memeriksa setiap elemen dalam daftar satu per satu hingga elemen yang dicari ditemukan atau seluruh daftar telah diperiksa. Algoritma ini memiliki kompleksitas waktu O(n), di mana n adalah jumlah elemen dalam daftar.

Binary search adalah algoritma pencarian yang lebih efisien daripada linear search. Algoritma ini hanya dapat digunakan pada daftar yang sudah diurutkan. Binary search bekerja dengan membagi daftar menjadi dua bagian dan membandingkan elemen tengah dengan elemen yang dicari. Jika elemen tengah bukan elemen yang dicari, pencarian dilanjutkan pada separuh daftar yang relevan. Algoritma ini memiliki kompleksitas waktu O(log n).

Depth-First Search (DFS) dan Breadth-First Search (BFS) adalah algoritma pencarian yang digunakan dalam struktur data graf dan pohon. DFS mengeksplorasi sepanjang satu cabang sebelum beralih ke cabang lain, menggunakan tumpukan atau rekursi. BFS mengeksplorasi semua tetangga dari suatu simpul sebelum beralih ke simpul berikutnya, menggunakan antrian. Algoritma ini sering digunakan dalam pencarian jalur dan masalah jaringan.

Jump search adalah algoritma pencarian yang lebih efisien daripada linear search pada daftar yang diurutkan. Algoritma ini bekerja dengan melompat sejumlah langkah tetap (sqrt(n)) dan kemudian melakukan pencarian linear dalam blok yang ditemukan. Kompleksitas waktu adalah O(sqrt(n)).

Exponential search digunakan untuk mencari dalam daftar yang diurutkan, terutama ketika ukuran daftar tidak diketahui. Algoritma ini mencari elemen dengan meningkatkan ukuran blok pencarian secara eksponensial dan kemudian melakukan binary search pada blok yang relevan. Kompleksitas waktu adalah O(log n).

Interpolation search adalah algoritma yang digunakan untuk mencari elemen dalam daftar yang diurutkan, terutama ketika elemen-elemen dalam daftar didistribusikan secara merata. Algoritma ini memperkirakan posisi elemen yang dicari berdasarkan nilai elemen-elemen dalam daftar. Kompleksitas waktu dalam kasus terbaik adalah O(log log n), tetapi dalam kasus terburuk bisa mencapai O(n).

Algoritma pencarian ini merupakan bagian penting dalam ilmu komputer dan digunakan dalam berbagai aplikasi seperti basis data, mesin pencari, dan pemrosesan

graf. Mereka memiliki keunggulan dan keterbatasan masing-masing, tergantung pada struktur data dan kondisi pencarian.

BAB III

GUIDED

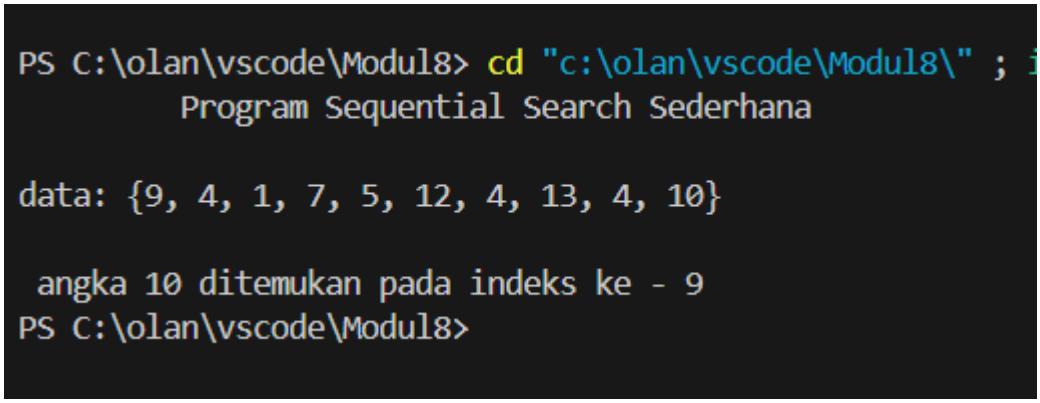
1. Guided 1

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 tersebut adalah implementasi sederhana dari algoritma pencarian sekuensial (Sequential Search). Pertama-tama, program mendeklarasikan variabel yang diperlukan, termasuk jumlah elemen dalam larik data, larik data itu sendiri, nilai yang akan dicari, dan variabel penanda apakah nilai yang dicari telah ditemukan. Kemudian, program menggunakan loop for untuk mencari nilai yang dicari dalam larik data. Setiap elemen dalam larik data diperiksa secara berurutan. Jika nilai yang dicari ditemukan pada suatu indeks dalam larik, variabel penanda diatur

menjadi true, dan loop berhenti. Setelah pencarian selesai, program mencetak hasil pencarian dengan menampilkan indeks di mana nilai yang dicari ditemukan, atau pesan bahwa nilai tidak ditemukan dalam larik. Ini adalah implementasi dasar dari algoritma pencarian sekuensial yang sederhana, di mana pencarian dilakukan dengan menguji setiap elemen satu per satu secara berurutan dalam larik.

2. Guided 2

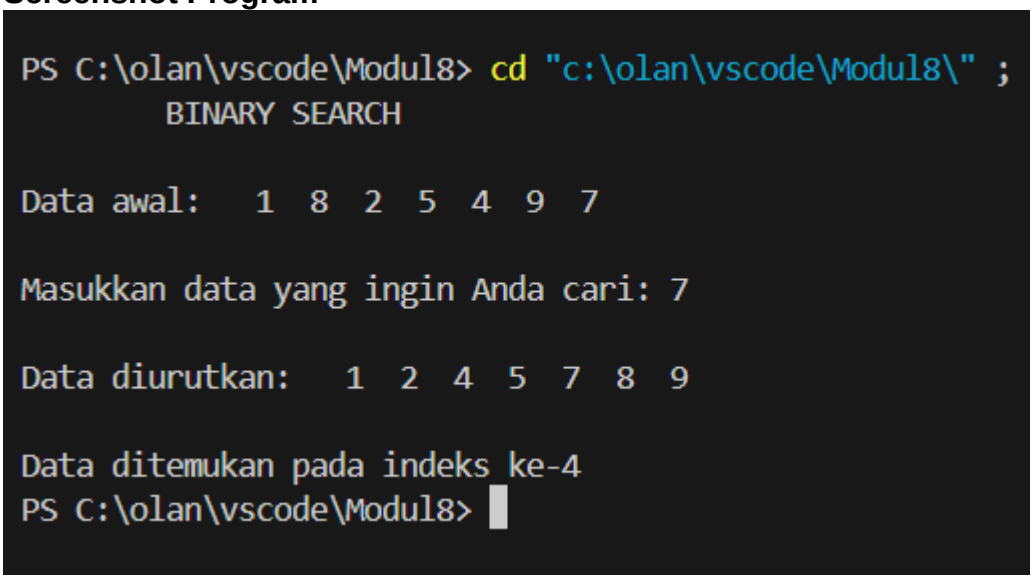
Source code

```
#include <iostream>
#include <iomanip>

using namespace std;

// Deklarasi array dan variabel untuk pencarian
int arr[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;

// Fungsi selection sort untuk mengurutkan arr
void selectionSort(int arr[], int n) {
    int temp, minIndex;

    for (int i = 0; i < n - 1; i++) {
        minIndex = i;

        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }

        // Tukar elemen
        temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
```

```
}

// Fungsi binary search untuk mencari data dalam array yang telah
diurutkan
void binarySearch(int arr[], int n, int target) {
    int start = 0, end = n - 1, middle, found = 0;

    while (start <= end && found == 0) {
        middle = (start + end) / 2;

        if (arr[middle] == target) {
            found = 1;
        } else if (arr[middle] < target) {
            start = middle + 1;
        } else {
            end = middle - 1;
        }
    }

    if (found == 1) {
        cout << "\nData ditemukan pada indeks ke-" << middle << endl;
    } else {
        cout << "\nData tidak ditemukan\n";
    }
}

int main() {
    cout << "\tBINARY SEARCH" << endl;

    cout << "\nData awal: ";
    // Menampilkan data awal
    for (int i = 0; i < 7; i++) {
        cout << setw(3) << arr[i];
    }
    cout << endl;

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

    // Mengurutkan data dengan selection sort
    selectionSort(arr, 7);
```

```
    cout << "\nData diurutkan: ";
    // Menampilkan data setelah diurutkan
    for (int i = 0; i < 7; i++) {
        cout << setw(3) << arr
[i];
    }
    cout << endl;

    // Melakukan binary search
    binarySearch(arr, 7, cari);

    return 0;
}
```

Screenshot Program



Deskripsi Program

Program tersebut mengimplementasikan algoritma Binary Search untuk mencari nilai tertentu dalam sebuah array yang telah diurutkan sebelumnya menggunakan algoritma Selection Sort. Pertama, program mendeklarasikan array `arr` yang berisi data yang akan dicari, serta variabel `cari` untuk menyimpan nilai yang akan dicari dalam array. Selanjutnya, terdapat fungsi `selectionSort()` yang digunakan untuk mengurutkan array menggunakan algoritma Selection Sort. Algoritma ini secara berulang mencari elemen terkecil dan menukar posisinya dengan elemen pertama, kemudian mencari elemen terkecil kedua dan menukar posisinya dengan elemen kedua, dan seterusnya. Setelah pengurutan selesai, program mencetak array yang telah

diurutkan. Selanjutnya, program meminta pengguna untuk memasukkan nilai yang ingin dicari dalam array. Kemudian, program menggunakan fungsi `binarySearch()` untuk mencari nilai yang dimasukkan pengguna dalam array yang telah diurutkan. Algoritma Binary Search membagi array menjadi dua bagian, memeriksa nilai tengahnya, dan memutuskan untuk melanjutkan pencarian di setengah bagian mana dari array berdasarkan perbandingan antara nilai tengah dengan nilai yang dicari. Hasil pencarian, berupa indeks di mana nilai ditemukan, atau pesan bahwa nilai tidak ditemukan, kemudian dicetak ke layar. Dengan memanfaatkan algoritma pengurutan Selection Sort dan algoritma pencarian Binary Search, program ini dapat mencari nilai secara efisien dalam array yang telah diurutkan.

LATIHAN KELAS - UNGUIDED

1. **Unguided 1**

   Source code

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

using namespace std;

// Fungsi untuk melakukan binary search
int binarySearch(const string& sortedStr, char target) {
    int left = 0;
    int right = sortedStr.size() - 1;

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

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

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

int main() {
    string kalimat;
    char huruf;

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

    // Menyimpan indeks asli dari setiap karakter sebelum
diurutkan
    vector<pair<char, int>> indexedKalimat;
    for (int i = 0; i < kalimat.size(); ++i) {
        indexedKalimat.push_back({kalimat[i], i});
    }

    // Mengurutkan kalimat berdasarkan karakter
    sort(indexedKalimat.begin(), indexedKalimat.end());

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

    // Membuat string terurut untuk binary search
    string sortedKalimat;
    for (const auto& p : indexedKalimat) {
        sortedKalimat.push_back(p.first);
    }

    // Menggunakan binary search untuk mencari huruf
    int index = binarySearch(sortedKalimat, huruf);

    // Menampilkan hasil
    if (index != -1) {
        cout << "Huruf " << huruf << " ditemukan pada indeks " <<
indexedKalimat[index].second << endl;
    } else {
```

```
        cout << "Huruf " << huruf << " tidak ditemukan dalam
kalimat." << endl;
    }

    return 0;
}
```

Screenshoot program

```
PS C:\olan\vscode\Modul8> cd "c:\olan\vscode\Modul8\" ;
Masukkan sebuah kalimat: tempe goreng
Masukkan huruf yang ingin dicari: g
Huruf g ditemukan pada indeks 11
PS C:\olan\vscode\Modul8>
```

Deskripsi Program

Program tersebut merupakan implementasi algoritma Binary Search untuk mencari keberadaan sebuah huruf dalam sebuah kalimat yang dimasukkan oleh pengguna. Pertama, pengguna diminta untuk memasukkan sebuah kalimat, yang disimpan dalam variabel string `kalimat`. Setiap karakter dalam kalimat awal disimpan bersama dengan indeks aslinya sebelum diurutkan, menggunakan vektor `indexedKalimat`. Selanjutnya, vektor `indexedKalimat` diurutkan berdasarkan karakternya. Pengguna juga diminta untuk memasukkan huruf yang ingin dicari dalam kalimat. String terurut `sortedKalimat` dibuat dengan mengambil karakter dari vektor `indexedKalimat` setelah diurutkan, yang akan digunakan untuk pencarian menggunakan algoritma Binary Search. Fungsi `binarySearch()` kemudian dipanggil untuk mencari keberadaan huruf dalam kalimat yang sudah diurutkan. Hasilnya adalah indeks dalam vektor `indexedKalimat`. Terakhir, program mencetak hasil pencarian, yaitu indeks di mana huruf ditemukan dalam kalimat awal. Jika huruf tidak ditemukan, program mencetak pesan yang

sesuai. Dengan memanfaatkan algoritma Binary Search, program ini dapat mencari keberadaan sebuah huruf dalam kalimat secara efisien, terutama pada kalimat yang panjang.

2. **Unguided 2**

Source code

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

using namespace std;

int main() {
    string kalimat;
    int count = 0;

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

    for (char c : kalimat) {
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ||
            c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U') {
            count++;
        }
    }

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

    return 0;
}
```

Screenshoot program

```
PS C:\olan\vscode\Modul8> cd "c:\olan\vscode\Modul8\" ; 
Masukkan sebuah kalimat: tempe goreng
Jumlah huruf vokal dalam kalimat adalah: 4
PS C:\olan\vscode\Modul8>
```

Deskripsi program

Program tersebut merupakan program sederhana yang bertujuan untuk menghitung jumlah huruf vokal dalam sebuah kalimat yang dimasukkan oleh pengguna. Pertama, program meminta pengguna untuk memasukkan sebuah kalimat menggunakan fungsi `getline(cin, kalimat)`, yang memungkinkan pengguna untuk memasukkan kalimat yang mengandung spasi. Selanjutnya, program menggunakan loop `for` untuk mengiterasi setiap karakter dalam kalimat. Setiap karakter diperiksa untuk menentukan apakah itu merupakan huruf vokal (baik huruf vokal kecil maupun huruf vokal besar). Jika sebuah karakter adalah huruf vokal, variabel `count` yang bertindak sebagai penghitung jumlah huruf vokal, akan ditambah satu. Setelah proses pencarian selesai, program mencetak jumlah huruf vokal yang telah dihitung ke layar. Program ini memberikan hasil yang sederhana dan langsung, yaitu jumlah huruf vokal dalam kalimat yang dimasukkan oleh pengguna.

3. Unguided 3

Source Code

```cpp
#include <iostream>
#include <vector>

using namespace std;

// Fungsi untuk menghitung jumlah angka 4 menggunakan Sequential Search
int hitungAngkaEmpat(const vector<int>& data) {
    int count = 0;
    for (int num : data) {
        if (num == 4) {
            count++;
        }
    }
    return count;
```

```
}

int main() {
    vector<int> data = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};

    // Menghitung jumlah angka 4
    int jumlahEmpat = hitungAngkaEmpat(data);

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

    return 0;
}
```

Screenshot Program

```
PS C:\olan\vscode\Modul8> cd "c:\olan\vscode\Modul8\" ;
Jumlah angka 4 dalam data: 4
PS C:\olan\vscode\Modul8>
```

Deskripsi Program

Program tersebut merupakan implementasi sederhana untuk menghitung jumlah kemunculan angka 4 dalam sebuah vektor yang telah diberikan. Pertama, program mendefinisikan sebuah fungsi `hitungAngkaEmpat()` yang menerima sebuah vektor integer sebagai argumen. Fungsi ini menggunakan algoritma Sequential Search untuk mengiterasi setiap elemen vektor dan menghitung jumlah kemunculan angka 4. Dalam setiap iterasi, jika elemen vektor adalah angka 4, variabel penghitung akan bertambah satu. Kemudian, di dalam fungsi `main()`, sebuah vektor integer diinisialisasi dengan sejumlah angka. Fungsi `hitungAngkaEmpat()` dipanggil dengan vektor tersebut sebagai argumen, dan hasilnya disimpan dalam variabel. Akhirnya, program mencetak jumlah angka 4 yang telah dihitung ke layar. Implementasi menggunakan algoritma Sequential Search memungkinkan program untuk menghitung jumlah tersebut dengan cara yang sederhana dan langsung.

BAB IV

KESIMPULAN

Kesimpulan dari Laporan Praktikum kali ini sebagai berikut:

1. **Pencarian dalam Data**: Program-program tersebut mengilustrasikan berbagai algoritma pencarian yang digunakan dalam komputasi, termasuk Sequential Search (pencarian sekuensial) dan Binary Search (pencarian biner). Selain itu, terdapat juga contoh penggunaan algoritma pencarian di dalam sebuah kalimat dan dalam vektor.

2. **Implementasi Algoritma**: Setiap program memiliki tujuan yang jelas dalam mengimplementasikan algoritma pencarian tertentu. Misalnya, penggunaan Binary Search pada array yang telah diurutkan untuk meningkatkan efisiensi pencarian, atau penggunaan Sequential Search untuk menghitung jumlah kemunculan angka tertentu dalam vektor.

3. **Efisiensi dan Kekurangan**: Program-program tersebut juga memberikan gambaran tentang efisiensi dan kekurangan dari masing-masing algoritma. Contohnya, Binary Search memiliki kompleksitas waktu yang lebih baik daripada Sequential Search, namun memerlukan array yang sudah diurutkan. Sementara itu, Sequential Search sederhana namun mungkin kurang efisien untuk data yang besar.

4. **Penggunaan Algoritma**: Program-program ini juga menunjukkan bagaimana pemilihan algoritma pencarian yang tepat dapat mempengaruhi kinerja aplikasi. Misalnya, untuk data yang tidak diurutkan, Sequential Search bisa menjadi pilihan yang masuk akal, sementara untuk data yang sudah diurutkan, Binary Search akan lebih efisien.

Dengan demikian, melalui program-program ini, pengguna dapat memahami konsep dasar dari berbagai algoritma pencarian dan memahami kapan dan bagaimana cara menggunakan algoritma yang sesuai untuk menyelesaikan masalah pencarian tertentu.

DAFTAR PUSTAKA

Geeks For Geeks 1 April 2024. Searching Algorithms, Diakses pada 4 Juni 2024 dari https://www.geeksforgeeks.org/searching-algorithms/