LAPORAN PRAKTIKUM

MODUL VIII
ALGORITMA SEARCHING

Disusun oleh:
Anita Nurazizah Agussalim
NIM: 2311102017

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 mmenyelesaikan suatu masalah 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 pengulangan pada pencarian akan terjadi sebanyak jumlah N elemen pada array.

   Algoritma pencarian berurutan dapat dituliskan sebagai berikut:

   1) i ← 0
   2) 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.

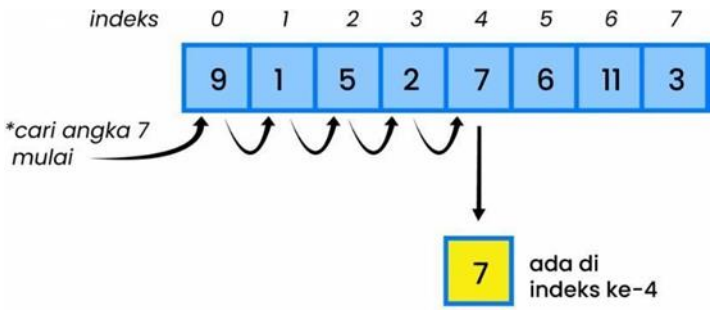
   Di bawah ini merupakan fungsi untuk mencari data menggunakan pencarian sekuensial.

```
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}



Gambar 1. Ilustrasi Sequential Search

Misalkan, dari data di atas angka yang akan dicari adalah angka 7 dalam array A, maka proses yang akan terjadi yaitu:

- Pencarian dimulai pada index ke-0 yaitu angka 9, kemudian dicocokkan dengan angka yang akan dicari, jika tidak sama maka pencarian akan dilanjutkan ke index selanjutnya.
- Pada index ke-1, yaitu angka 1, juga bukan angka yang dicari, maka pencarian akan dilanjutkan pada index selanjutnya.

- Pada index ke-2 dan index ke-3 yaitu angka 5 dan 2, juga bukan angka yang dicari, sehingga pencarian dilanjutkan pada index selanjutnya.
- Pada index ke-4 yaitu angka 7 dan ternyata angka 7 merupakan angka yang dicari, sehingga pencarian akan dihentikan dan proses selesai.

b. Binary Search

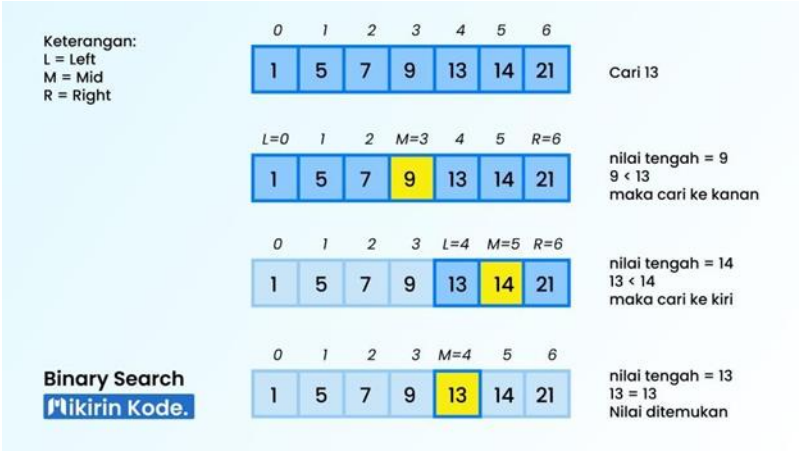
Binary Search termasuk ke dalam interval search, dimana algoritma ini merupakan algoritma pencarian pada array/list dengan elemen terurut. Pada metode ini, data harus diurutkan terlebih dahulu dengan cara data dibagi menjadi dua bagian (secara logika), untuk setiap tahap pencarian. Dalam penerapannya 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.

Algoritma pencarian biner dapat dituliskan sebagai berikut :

1) L ☐ 0
2) R ☐ N – 1
3) ketemu ☐ false
4) Selama (L <= R) dan (tidak ketemu) kerjakan baris 5 sampai dengan 8
5) m ☐ (L + R) / 2
6) Jika (Data[m] = x) maka ketemu ☐ true
7) Jika (x < Data[m]) maka R ☐ m – 1
8) Jika (x > Data[m]) maka L ☐ m + 1
9) Jika (ketemu) maka m adalah indeks dari data yang dicari, jika tidak data tidak ditemukan

Contoh dari Binary Search, yaitu:



Gambar 2. Ilustrasi Binary Search

- Terdapat sebuah array yang menampung 7 elemen seperti ilustrasi di atas. Nilai yang akan dicari pada array tersebut adalah 13.
- Jadi karena konsep dari binary search ini adalah membagi array menjadi dua bagian, maka pertama tama kita cari nilai tengahnya dulu, total elemen dibagi 2 yaitu 7/2 = 4.5 dan kita bulatkan jadi 4.
- Maka elemen ke empat pada array adalah nilai tengahnya, yaitu angka 9 pada indeks ke 3.

- Kemudian kita cek apakah 13 > 9 atau 13 < 9?
- 13 lebih besar dari 9, maka kemungkinan besar angka 13 berada setelah 9 atau di sebelah kanan. Selanjutnya kita cari ke kanan dan kita dapat mengabaikan elemen yang ada di kiri.
- Setelah itu kita cari lagi nilai tengahnya, didapatlah angka 14 sebagai nilai tengah. Lalu, kita bandingkan apakah 13 > 14 atau 13 < 14?
- Ternyata 13 lebih kecil dari 14, maka selanjutnya kita cari ke kiri.
- Karna tersisa 1 elemen saja, maka elemen tersebut adalah nilai tengahnya. Setelah dicek ternyata elemen pada indeks ke-4 adalah elemen yang dicari, maka telah selesai proses pencariannya.

BAB III

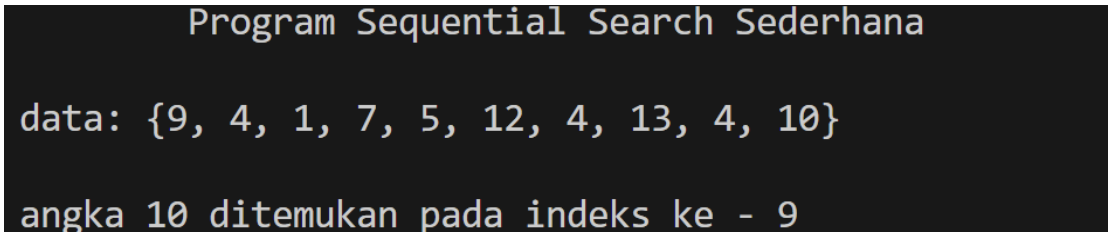
GUIDED

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



Deskripsi program

Program di atas adalah contoh implementasi algoritma Sequential Search untuk mencari nilai tertentu dalam array. Program ini terlebih dahulu mendefinisikan array data berisi 10 elemen dan nilai yang ingin dicari. Kemudian, program melakukan perulangan untuk membandingkan setiap elemen array dengan nilai yang dicari. Jika ditemukan, program akan menandai variabel ketemu sebagai true dan menyimpan indeks elemen tersebut. Setelah perulangan selesai, program akan menampilkan pesan yang menunjukkan apakah nilai yang dicari ditemukan atau tidak, beserta indeksnya jika ditemukan.

2. Guided 2

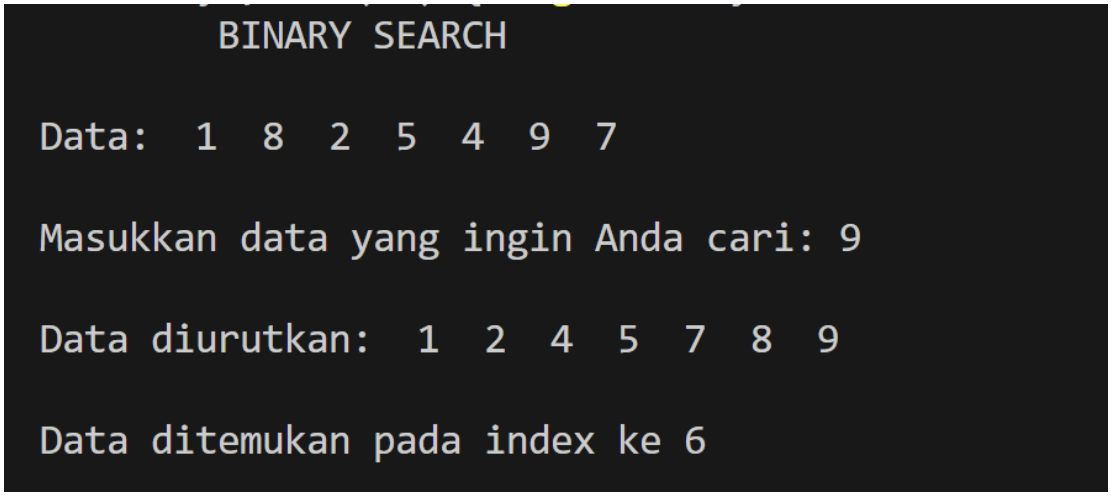
SOURCE CODE

```
#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 ()
{

    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 << "\nData ditemukan pada index ke "<< tengah << endl;
        else
        cout << "\nData tidak ditemukan\n";
}
int main()
{
    cout << "\tBINARY SEARCH" << endl;
```

```
    cout << "\nData:";

    for (int x = 0; x < 7; x++)
        cout << setw(3) << data[x];
    cout << endl;
    cout << "\nMasukkan data yang ingin Anda cari: ";
    cin >> cari;
    cout << "\nData diurutkan:";

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

SCREENSHOT PROGRAM



DESKRIPSI PROGRAM

Program ini mengimplementasikan dua algoritma pencarian: Selection Sort dan Binary Search. Selection Sort digunakan untuk mengurutkan data terlebih dahulu sebelum melakukan pencarian dengan Binary Search. Binary Search bekerja dengan cara membagi data menjadi dua bagian secara berulang,

berdasarkan nilai tengah, hingga nilai yang dicari ditemukan atau dipastikan tidak ada dalam data. Program ini menampilkan data awal, meminta nilai yang ingin dicari, mengurutkan data, melakukan pencarian dengan Binary Search, dan menampilkan hasil pencarian.

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>
#include <string>
#include <cstring>

using namespace std;

void binary_search(const char arr[], int size, char cari) {
    int awal = 0;
    int akhir = size - 1;
    bool found = false;

    while (awal <= akhir) {
        int tengah = (awal + akhir) / 2;
        if (arr[tengah] == cari) {
            found = true;
            cout << "\nHuruf '" << cari << "' ditemukan pada indeks ke-
" << tengah << endl;
            break;
        } else if (arr[tengah] < cari) {
            awal = tengah + 1;
        } else {
            akhir = tengah - 1;
        }
    }

    if (!found) {
        cout << "\nHuruf '" << cari << "' tidak ditemukan dalam
kalimat." << endl;
    }
}

int main() {
```

```
    string kalimat = "anitA";
    char cari;

    cout << "Kata: anitA" << endl;
    kalimat.erase(remove(kalimat.begin(), kalimat.end(), ' '),
kalimat.end());

    int size = kalimat.length();
    char* array = new char[size];
    strncpy(array, kalimat.c_str(), size);

    cout << "Masukkan huruf yang ingin Anda cari: ";
    cin >> cari;

    binary_search(array, size, cari);

    delete[] array;

    return 0;
}
```

OUTPUT PROGRAM

```
Kata: anitA
Masukkan huruf yang ingin Anda cari: t

Huruf 't' ditemukan pada indeks ke-3
```

DESKRIPSI PROGRAM

Program ini mengimplementasikan algoritma binary search untuk mencari huruf tertentu dalam sebuah kalimat.Pertama, program mengubah kalimat yang dimasukkan (misalnya "anitA") menjadi array karakter dan menghapus spasi. Kemudian, program melakukan binary search pada array karakter untuk menemukan huruf yang dicari. Jika huruf ditemukan, program akan menampilkan pesan bahwa huruf tersebut ditemukan pada indeks tertentu dalam array. Jika huruf tidak ditemukan, program akan

menampilkan pesan bahwa huruf tersebut tidak ditemukan dalam kalimat. Setelah selesai, program menghapus array karakter yang dialokasikan secara dinamis.

2. Buatlah sebuah program yang dapat menghitung banyaknya huruf vocal dalam sebuah kalimat

SOURCE CODE

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

// Untuk fungsi isalpha dan tolower
using namespace std;
// Fungsi untuk menghitung jumlah huruf vokal dalam sebuah kalimat
int hitungVokal(const string & kalimat) {
int jumlahVokal = 0;

for (char huruf : kalimat) {
// Mengonversi huruf menjadi huruf kecil
 char lowerHuruf = tolower(huruf);

// Memeriksa apakah huruf merupakan huruf vokal
if (lowerHuruf == 'a' || lowerHuruf == 'e' || lowerHuruf == 'i' ||
lowerHuruf == 'o' || lowerHuruf == 'u') {
jumlahVokal++;
}
}

return jumlahVokal;
}

int main() {
string kalimat;

// Input kalimat dari pengguna
cout << "Masukkan sebuah kalimat: "; getline(cin, kalimat);
```

```
// Menghitung jumlah huruf vokal dalam kalimat
int jumlahVokal = hitungVokal(kalimat);

// Menampilkan hasil
cout << "Jumlah huruf vokal dalam kalimat adalah: " << jumlahVokal <<
endl;

return 0;
}
```

OUTPUT PROGRAM

```
Masukkan sebuah kalimat: anitalucu
Jumlah huruf vokal dalam kalimat adalah: 5
```

DESKRIPSI PROGRAM

Program di atas adalah sebuah program C++ yang menghitung jumlah huruf vokal (a, e, i, o, u) dalam sebuah kalimat yang dimasukkan oleh pengguna. Program menggunakan fungsi `hitungVokal` yang menerima sebuah string sebagai argumen dan menghitung jumlah huruf vokal dengan mengonversi setiap karakter dalam string menjadi huruf kecil menggunakan fungsi `tolower`. Jika karakter tersebut adalah huruf vokal, maka variabel penghitung `jumlahVokal` akan ditambah satu. Hasil perhitungan kemudian ditampilkan di layar.

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

SOURCE CODE

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

```cpp
// Fungsi untuk mencari jumlah kemunculan suatu angka dalam array
dengan Sequential Search
int hitungAngka(const int data[], int ukuran, int angka) { int jumlah =
0;
for (int i = 0; i < ukuran; ++i) { if (data[i] == angka) {
jumlah++;
}
}
return jumlah;
}

int main() {
const int ukuran = 10;
int data[ukuran] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}; int
angkaYangDicari = 4;

// Menghitung jumlah kemunculan angka 4 dalam data menggunakan
Sequential Search
int jumlahAngka4 = hitungAngka(data, ukuran, angkaYangDicari);

// Menampilkan hasil
cout << "Jumlah angka 4 dalam data adalah: " << jumlahAngka4 << endl;

return 0;
}
```

SCREENSHOT PROGRAM

```
Jumlah angka 4 dalam data adalah: 4
```

DESKRIPSI PROGRAM

Program di atas adalah sebuah program C++ yang mencari dan menghitung jumlah kemunculan suatu angka dalam sebuah array menggunakan metode Sequential Search. Program mendefinisikan sebuah fungsi `hitungAngka` yang menerima sebuah array `data`, ukuran array `ukuran`, dan angka yang dicari `angka`, kemudian menghitung berapa kali angka tersebut muncul dalam array.

Dalam fungsi `main`, sebuah array `data` berukuran 10 didefinisikan, dan angka yang dicari adalah 4. Fungsi `hitungAngka` digunakan untuk menghitung jumlah kemunculan angka 4 dalam array, dan hasilnya ditampilkan ke layar.

BAB IV

KESIMPULAN

Pencarian (searching) adalah proses menemukan nilai tertentu dalam kumpulan data, dengan hasil pencarian bisa berupa data ditemukan, data ditemukan lebih dari satu, atau data tidak ditemukan. Dua metode utama dalam algoritma pencarian adalah Sequential Search dan Binary Search. Sequential Search adalah metode sederhana yang cocok untuk data acak atau belum terurut, dimana setiap elemen array diperiksa satu per satu dari awal hingga akhir. Binary Search, di sisi lain, lebih efisien tetapi memerlukan data yang sudah terurut, dengan prinsip membagi data menjadi dua bagian dan membandingkan nilai tengah hingga data ditemukan atau seluruh elemen habis diperiksa. Sequential Search berguna untuk data yang tidak terurut, sedangkan Binary Search optimal untuk data yang terurut.

DAFTAR PUSTAKA

Asisten Praktikum. (2024). *MODUL 8: ALGORITMA SEARCHING.*

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