LAPORAN PRAKTIKUM

MODUL 8
SEARCHING

Disusun oleh:
Denny Budiansyach
NIM: 2311102022

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.

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.

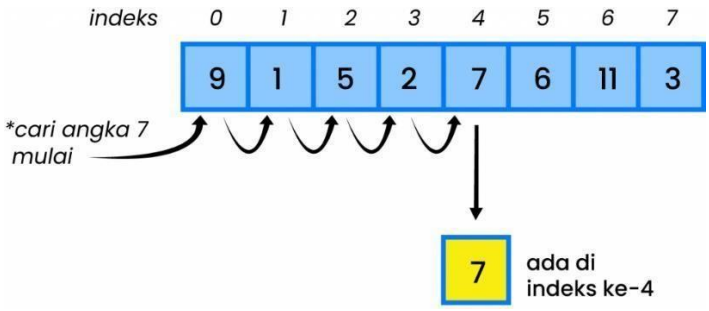
Di bawah ini merupakan fungsi untuk mencari data menggunakan pencarian sekuensial.

```
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

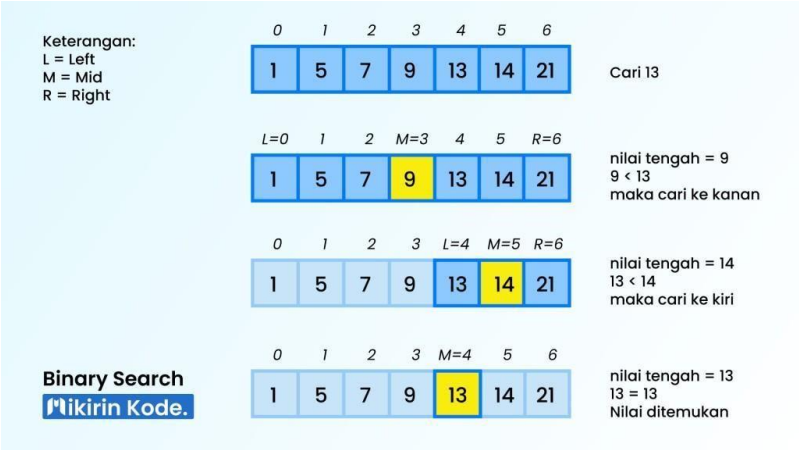
6) Jika (Data[m] = x) maka ketemu □ true

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.

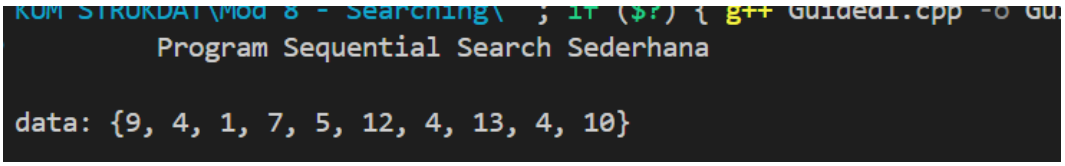
BAB III

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;
}
```

Screenshoot program



Deskripsi program

Program diatas adalah bentuk implementasi sequential search yang digunakan untuk menemukan suatu nilai dalam sebuah array. Pertama tama, sebuah array yang berisi sepuluh elemen dideklarasikan bersama dengan nilai yang akan dicari (10). Program melakukan iterasi melalui setiap elemen dalam array dan memeriksa apakah elemen tersebut sama dengan nilai yang dicari. Jika nilai yang dicari berhasil ditemukan, variabel `ketemu` diatur menjadi `true` dan loop berhenti. Setelah pencarian selesai, program menampilkan hasil pencarian. Jika nilai ditemukan, program mencetak indeks di mana nilai tersebut berada; jika tidak, program mencetak bahwa nilai tersebut tidak ditemukan dalam array. Program juga menampilkan pesan pembuka yang menyatakan bahwa ini adalah program pencarian berurutan sederhana.

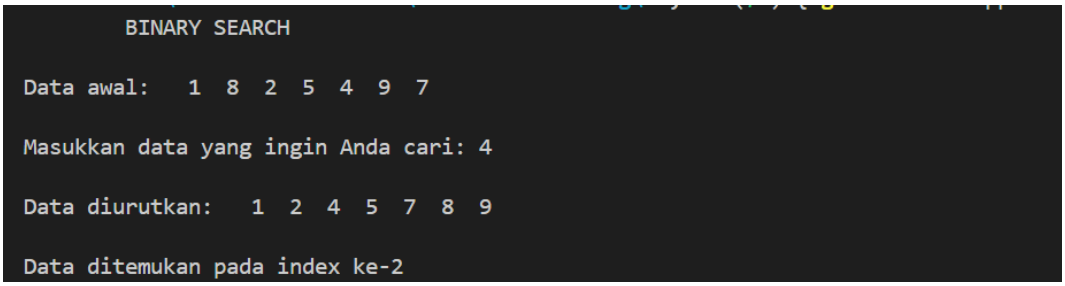
GUIDED 2

Guided 2 Source code

```
#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



Deskripsi program

Program di atas adalah bentuk implementasi dari binary search pada sebuah array yang telah diurutkan menggunakan metode selection sort. Pertama tama, array yang berisi tujuh elemen dideklarasikan dan ditampilkan. Kemudian, program meminta pengguna untuk memasukkan nilai yang ingin dicari. Array tersebut diurutkan dengan selection sort, di mana elemen terkecil ditemukan dan ditukar dengan elemen pertama, lalu proses ini diulangi untuk elemen-elemen berikutnya. Setelah array terurut, program akan menjalankan binary search dengan membagi array menjadi dua dan membandingkan elemen tengah dengan nilai yang dicari. Jika nilai yang dicari lebih kecil dari elemen tengah, pencarian dilanjutkan di bagian kiri array, dan sebaliknya jika nilainya lebih besar. Proses ini diulangi hingga nilai ditemukan atau seluruh array telah diperiksa. Hasil pencarian kemudian ditampilkan, menunjukkan apakah nilai ditemukan dan berada di indeks berapa, atau menyatakan bahwa nilai tidak ditemukan.

LATIHAN KELAS - UNGUIDED

1. **Unguided 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

```
Masukkan sebuah kalimat   : qwertyuiopasdfghjklzxcvbnm
Kalimat setelah diurutkan : abcdefghijklmnopqrstuvwxyz

Masukkan huruf yang dicari: q

Huruf ditemukan pada urutan ke-17
```

Deskripsi program

Program di atas adalah bentuk implementasi dari binary search. Program akan melakukan pencarian binary pada karakter-karakter dalam sebuah kalimat yang telah diurutkan menggunakan metode selection sort. Pertama tama program akan meminta pengguna untuk memasukkan sebuah kalimat. Panjang kalimat dihitung dengan iterasi hingga mencapai karakter null (0). Kemudian, kalimat tersebut diurutkan menggunakan selection sort, di mana setiap karakter dibandingkan dengan karakter lainnya dan diurutkan dalam urutan menaik. Setelah kalimat diurutkan, program meminta pengguna untuk memasukkan huruf yang ingin dicari. Pencarian binary dilakukan pada array karakter yang telah diurutkan. Proses binary search membagi array menjadi dua bgian, kemduian memeriksa karakter tengah, dan menentukan apakah pencarian harus dilanjutkan di bagian kiri atau kanan array, tergantung pada perbandingan dengan huruf yang dicari. Jika huruf ditemukan, program menampilkan indeks di mana huruf tersebut berada; jika tidak, program menyatakan bahwa huruf tidak ditemukan.

2. **Unguided Source code**

```cpp
#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 sebuah 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 dalam kalimat: " << jumlah_vokal
<< endl;
    return 0;
}
```

Screenshoot program

```
TIKUM STRUKDAT\Mod 8 - Searching\" ; if ($?) { g++ UnGuided
Masukkan sebuah kalimat: aku suka makan pedas

Jumlah huruf vokal dalam kalimat: 8
```

Deskripsi program

Program di atas adlah program yang akan menghitung jumlah huruf vokal dalam sebuah kalimat. Pertama, program mendefinisikan fungsi is_vokal yang memeriksa apakah suatu karakter termasuk dalam huruf vokal (a, e, i, o, u, dan versi kapitalnya). Fungsi ini menggunakan array yang berisi huruf vokal dan melakukan iterasi untuk membandingkan karakter input dengan elemen-elemen dalam array. Jika karakter ditemukan dalam array, fungsi is_vokal akan mengembalikan TRUE, menandakan bahwa karakter tersebut adalah huruf vokal. Di dalam fungsi main, program meminta pengguna untuk memasukkan sebuah kalimat. Kemudian, program menginisialisasi variabel jumlah_vokal untuk menghitung jumlah huruf vokal dalam kalimat. Program melakukan iterasi melalui setiap karakter dalam kalimat, menggunakan fungsi is_vokal untuk memeriksa apakah karakter tersebut adalah huruf vokal. Jika iya, nilai dari variabel jumlah_vokal akan ditambahkan satu. Setelah iterasi selesai, program menampilkan jumlah huruf vokal yang ditemukan dalam kalimat.

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

    return 0;
}
```

Screenshoot program

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

Angka 4 ditemukan sebanyak 4 angka.
```

Deskripsi program

Program di atas menghitung berapa kali suatu angka muncul dalam sebuah array. Pertama tama program akan mendefinisikan sebuah array “data” yang berisi sepuluh elemen int. Kemudian, program mendefinisikan variabel `n` yang menyimpan panjang array “data”. Kemudian program melakukan iterasi melalui setiap elemen array dimana pada setiap iterasi, program memeriksa apakah elemen saat ini sama dengan nilai pada variabel `target`. Jika iya, variabel `count` yang menyimpan jumlah kemunculan `target` ditambah satu. Setelah iterasi selesai, program mencetak array `data` dan menampilkan berapa kali angka `target` muncul dalam array tersebut.

BAB IV
KESIMPULAN

Searching merupakan suatu proses untuk menemukan suatu nilai tertentu dalam kumpulan data, dengan hasil berupa ditemukannnya data, ditemukan lebih dari satu, atau tidak ditemukan. Terdapat dua metode dalam algoritma pencarian, yaitu Sequential Search, yang digunakan untuk data tidak terurut, dengan konsep membandingkan setiap elemen satu per satu, dan Binary Search, yang digunakan untuk data terurut, dengan konsep membagi data menjadi dua bagian dan memeriksa bagian mana yang berpotensi mengandung nilai yang dicari. Perbedaan metode tersebut dapat dilihat dalam contoh penggunaannya, di mana Sequential Search cocok untuk data tak terurut, dan Binary Search cocok untuk data yang telah diurutkan.

DAFTAR PUSTAKA

GeeksforGeeks. (2024, April 1). Diambil kembali dari geeksforgeeks.com: https://www.geeksforgeeks.org/searching-algorithms/

Asisten Praktikum. (2024). *MODUL 8 – ALGORITMA SEARCHING*, Learning Managament System.