

**LAPORAN PRAKTIKUM**

**MODUL VIII**

**ALGORITMA SEARCHING**



**Disusun oleh:**  
**Imelda Fajar Awalina Crisyanti**  
**NIM: 2311102004**

**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**

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:

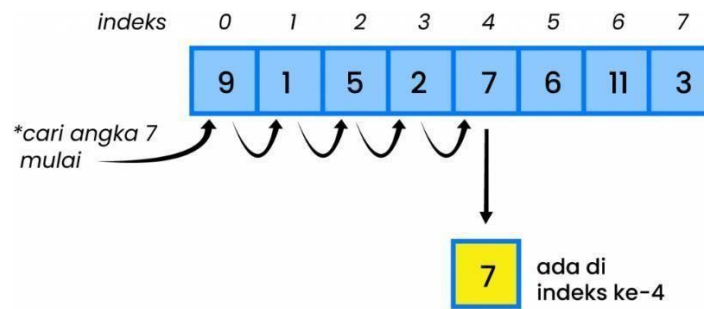
- Membandingkan setiap elemen pada array satu per satu secara berurut.
- Proses pencarian dimulai dari indeks pertama hingga indeks terakhir.
- Proses pencarian akan berhenti apabila data ditemukan. Jika hingga akhir array data masih juga tidak ditemukan, maka proses pencarian tetap akan dihentikan.
- Proses perulangan pada pencarian akan terjadi sebanyak jumlah N elemen pada array.

Algoritma pencarian berurutan dapat dituliskan sebagai berikut :

- 1)  $i \leftarrow 0$
- 2)  $ketemu \leftarrow false$
- 3) Selama (tidak  $ketemu$ ) dan  $(i \leq N)$  kerjakan baris 4
- 4) Jika  $(Data[i] = x)$  maka  $ketemu \leftarrow true$ , jika tidak  $i \leftarrow i + 1$
- 5) Jika  $(ketemu)$  maka  $i$  adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

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 :

$L \leftarrow 0$

2)  $R \leftarrow N - 1$

ketemu  $\leftarrow$  false

Selama ( $L \leq R$ ) dan (tidak ketemu) kerjakan baris 5 sampai dengan 8 5)  $m \leftarrow (L + R) / 2$

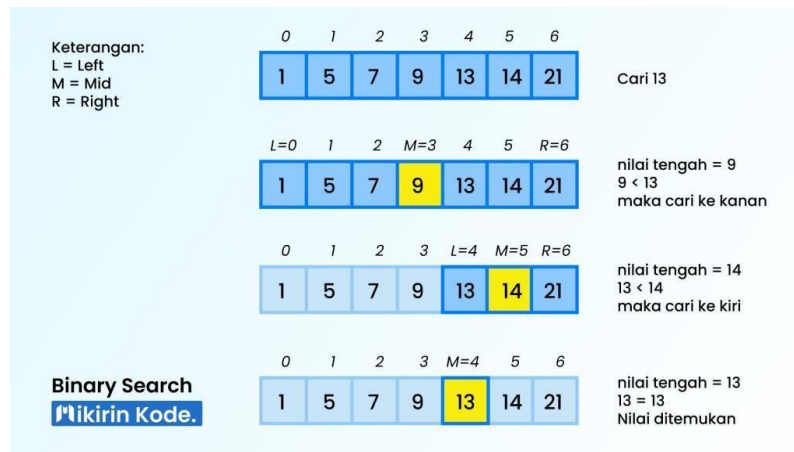
Jika ( $Data[m] = x$ ) maka ketemu  $\leftarrow$  true

Jika ( $x < Data[m]$ ) maka  $R \leftarrow m - 1$

Jika ( $x > Data[m]$ ) maka  $L \leftarrow m + 1$

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

### Program Sequential Search Sederhana

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

angka 10 ditemukan pada indeks ke -9

#### Deskripsi program

Program di atas merupakan implementasi sederhana dari algoritma pencarian sekuensial (Sequential Search) dalam bahasa C++. Berikut adalah deskripsi singkat mengenai program tersebut:

n: Menyimpan ukuran array, yaitu 10. Data: Array yang berisi 10 elemen dengan nilai-nilai yang telah ditentukan. cari: Angka yang ingin dicari dalam array, yaitu 10. ketemu: Variabel boolean untuk menandakan apakah angka yang dicari ditemukan atau tidak, awalnya diinisialisasi dengan nilai false. i: Variabel untuk iterasi.

Program melakukan iterasi dari indeks 0 hingga indeks 9 (seluruh elemen dalam array data). Dalam setiap iterasi, program mengecek apakah elemen array pada indeks ke-i sama dengan angka yang dicari (cari). Jika ditemukan kesamaan, variabel ketemu diatur menjadi true dan iterasi dihentikan (break).

Program menampilkan pesan pengantar "Program Sequential Search Sederhana". Menampilkan data array yang digunakan dalam pencarian. Jika angka yang dicari ditemukan dalam array, program akan menampilkan indeks di mana angka tersebut ditemukan. Jika angka yang dicari tidak ditemukan, program akan memberikan pesan bahwa angka tersebut tidak ditemukan dalam data.

## 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++)

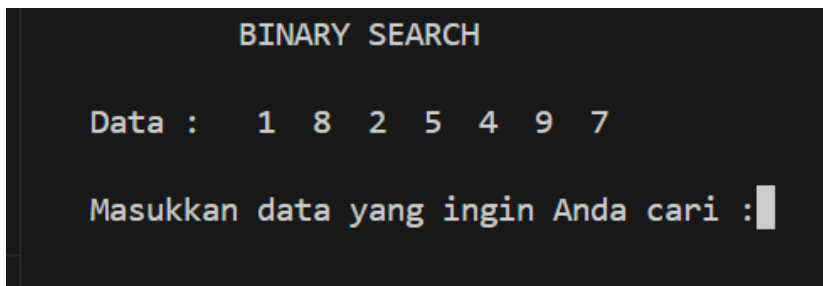
```

```

        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;
}

```

### Screenshoot program



```

BINARY SEARCH

Data :  1  8  2  5  4  9  7

Masukkan data yang ingin Anda cari : 

```

### Deskripsi program

Program di atas adalah implementasi dari algoritma pengurutan dan pencarian menggunakan selection sort dan binary search dalam bahasa C++. Berikut adalah deskripsi singkat dari program tersebut:

Fungsi ini mengurutkan array data menggunakan algoritma selection sort. Dalam setiap iterasi, elemen terkecil dari bagian yang belum diurutkan ditemukan dan ditukar dengan elemen pertama dari bagian tersebut.

Fungsi ini melakukan pencarian biner pada array data yang sudah diurutkan. awal, akhir, dan tengah digunakan untuk menentukan batas pencarian. Jika elemen di tengah array adalah elemen yang dicari, pencarian selesai. Jika elemen di tengah lebih kecil dari elemen yang dicari, pencarian dilanjutkan pada bagian kanan. Jika elemen di tengah lebih besar, pencarian dilanjutkan pada bagian kiri. Hasil pencarian ditampilkan apakah data ditemukan beserta indeksnya atau data tidak ditemukan.

Menampilkan judul program "BINARY SEARCH". Menampilkan data awal yang belum diurutkan. Meminta pengguna memasukkan angka yang ingin dicari (cari). Mengurutkan array data menggunakan selection\_sort. Menampilkan data setelah diurutkan. Melakukan pencarian data menggunakan binarysearch. Menunggu pengguna untuk menekan sembarang tombol sebelum program selesai dengan \_getche().

## LATIHAN KELAS - UNGUIDED

### 1. Unguided 1

#### Source code

```
#include <iostream>
#include <string>
#include <algorithm> // Untuk sort
using namespace std;

void binary_search(const string &sorted_str, char cari)
{
    int awal = 0;
    int akhir = sorted_str.length() - 1;
    int tengah;
    bool b_flag = false;

    while (awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (sorted_str[tengah] == cari)
        {
            b_flag = true;
            break;
        }
        else if (sorted_str[tengah] < cari)
        {
            awal = tengah + 1;
        }
        else
        {
            akhir = tengah - 1;
        }
    }

    if (b_flag)
    {
        cout << "\n Huruf '" << cari << "' ditemukan pada index ke-"
        << tengah << " setelah pengurutan.\n";
    }
    else
```

```

    {
        cout << "\n Huruf '" << cari << "' tidak ditemukan.\n";
    }
}

int main()
{
    string kalimat;
    char cari;

    cout << "\t BINARY SEARCH PADA KALIMAT" << endl;
    cout << "\n Masukkan kalimat: ";
    getline(cin, kalimat);

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

    // Menghapus spasi dari kalimat untuk tujuan pencarian
    kalimat.erase(remove(kalimat.begin(), kalimat.end(), ' '),
kalimat.end());

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

    cout << "\n Kalimat setelah diurutkan: " << kalimat << endl;

    binary_search(kalimat, cari);

    return 0;
}

```

## Screenshoot program

```

BINARY SEARCH PADA KALIMAT

Masukkan kalimat: i m e l d a f a j a r

Masukkan huruf yang ingin Anda cari: f

Kalimat setelah diurutkan: aaadefijlmr

```

## Deskripsi program

Input Kalimat: Program meminta pengguna untuk memasukkan sebuah kalimat. Input Huruf: Program meminta pengguna untuk memasukkan huruf yang ingin dicari dalam kalimat. Menghapus Spasi: Program menghapus semua spasi dalam kalimat untuk memudahkan pencarian. Mengurutkan Kalimat: Program mengurutkan karakter-karakter dalam kalimat menggunakan fungsi sort dari library <algorithm>. Binary Search: Program melakukan pencarian huruf menggunakan Binary Search pada kalimat yang sudah diurutkan. Menampilkan Hasil: Program menampilkan hasil pencarian, apakah huruf ditemukan atau tidak dan di indeks ke berapa jika ditemukan.

Dengan program ini, Anda dapat mencari sebuah huruf dalam kalimat yang telah diinputkan dengan efisien menggunakan Binary Search.

## 2. Unguided 2

### Source code

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

// Fungsi untuk mengecek apakah sebuah karakter adalah huruf vokal
bool isVokal(char ch) {
    ch = tolower(ch); // Mengubah huruf menjadi huruf kecil untuk
    memudahkan pengecekan
    return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch ==
    'u');
}

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

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

    // Iterasi melalui setiap karakter dalam kalimat
    for (char ch : kalimat) {
        if (isVokal(ch)) {
            jumlahVokal++;
        }
    }
}
```

```

    }

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

    return 0;
}

```

### Screenshoot program

```

Masukkan sebuah kalimat: i m e l d a f a j a r
Jumlah huruf vokal dalam kalimat: 5

```

### Deskripsi program

**Input Kalimat:** Program meminta pengguna untuk memasukkan sebuah kalimat menggunakan `getline(cin, kalimat)` untuk membaca seluruh baris termasuk spasi. **Fungsi isVokal:** Fungsi ini mengecek apakah karakter yang diberikan adalah huruf vokal ('a', 'e', 'i', 'o', 'u'). Fungsi ini mengubah karakter menjadi huruf kecil terlebih dahulu untuk memudahkan pengecekan. **Menghitung Huruf Vokal:** Program iterasi melalui setiap karakter dalam kalimat menggunakan loop `for`. Jika karakter adalah vokal (dieksekusi oleh fungsi `isVokal`), program meningkatkan penghitung `jumlahVokal`. **Menampilkan Hasil:** Program menampilkan jumlah huruf vokal yang ditemukan dalam kalimat.

Dengan program ini, Anda dapat menghitung jumlah huruf vokal dalam kalimat yang dimasukkan oleh pengguna.

## 3. Unguided 3

### Source code

```

#include <iostream>
using namespace std;

int main() {
    int n = 10;
    int data[n] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int cari = 4;
    int jumlah = 0;
}

```

```

// Algoritma Sequential Search untuk menghitung jumlah angka 4
for (int i = 0; i < n; i++) {
    if (data[i] == cari) {
        jumlah++;
    }
}

cout << "\tProgram Menghitung Jumlah Angka 4 dengan Sequential
Search\n" << endl;
cout << "Data: {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}" << endl;
cout << "Angka " << cari << " muncul sebanyak " << jumlah << "
kali." << endl;

return 0;
}

```

### Screenshoot program

```

Program Menghitung Jumlah Angka 4 dengan Sequential Search

Data: {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}
Angka 4 muncul sebanyak 4 kali.

```

### Deskripsi program

Deklarasi dan Inisialisasi Array: Program mendeklarasikan array data dengan 10 elemen dan mengisinya dengan nilai yang telah ditentukan. Variabel cari dan jumlah: Variabel cari menyimpan angka yang ingin dicari (dalam hal ini angka 4), dan variabel jumlah digunakan untuk menghitung jumlah kemunculan angka 4. Algoritma Sequential Search: Program melakukan iterasi melalui setiap elemen dalam array menggunakan loop for. Jika elemen saat ini sama dengan nilai cari, maka jumlah ditingkatkan. Menampilkan Hasil: Setelah loop selesai, program menampilkan jumlah kemunculan angka 4 dalam array.

Dengan menggunakan program ini, Anda dapat mengetahui berapa kali angka 4 muncul dalam array.



## **BAB IV**

### **KESIMPULAN**

Searching merupakan suatu proses untuk menemukan suatu nilai tertentu dalam kumpulan data, dengan hasil berupa ditemukannya data, ditemukan lebih dari satu, atau tidak ditemukan. Ada dua macam metode dalam algoritma pencarian yang kita pelajari antara lain:

#### **1. Pencarian Sequential (Pencarian Urutan)**

Pada metode ini, elemen-elemen dalam daftar diperiksa satu per satu secara berurutan hingga elemen yang dicari ditemukan atau sampai akhir daftar tercapai. Metode ini sederhana dan mudah diimplementasikan, namun tidak efisien untuk daftar yang besar karena membutuhkan waktu yang lama untuk memeriksa semua elemen.

#### **2. Pencarian Biner (Binary Search)**

Metode ini hanya dapat digunakan pada daftar yang terurut. Algoritma pencarian biner membagi daftar menjadi dua bagian secara berulang, dan kemudian mencari elemen yang dicari di bagian yang kemungkinan besar berisi elemen tersebut. Metode ini jauh lebih efisien daripada pencarian sequential untuk daftar yang besar.

Jadi kesimpulannya Gunakan pencarian sequential untuk data kecil atau data yang tidak terurut. Gunakan pencarian binary untuk data besar yang sudah terurut.