

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL VIII  
ALGORITMA SEARCHING**



**Disusun Oleh :**

NAMA : Pandia Arya Brata

NIM : 2311102076

**Dosen**

Wahyu Andi Saputra, S.Pd.,M.Eng.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA INSTITUT TEKNOLOGI TELKOM  
PURWOKERTO 2024**

## **MODUL 8**

### **ALGORITMA SEARCHING**

#### **A. 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.

## B. 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.

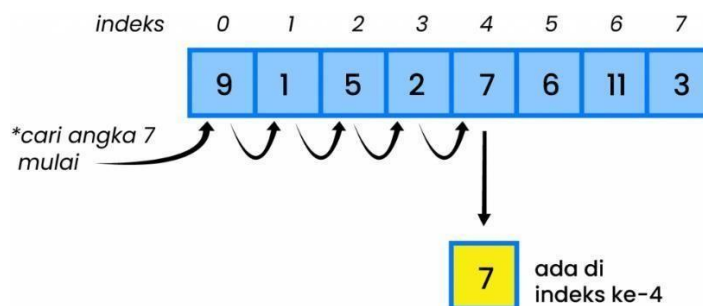
Di bawah ini merupakan fungsi untuk mencari data menggunakan pencarian sekuensial.

```
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:

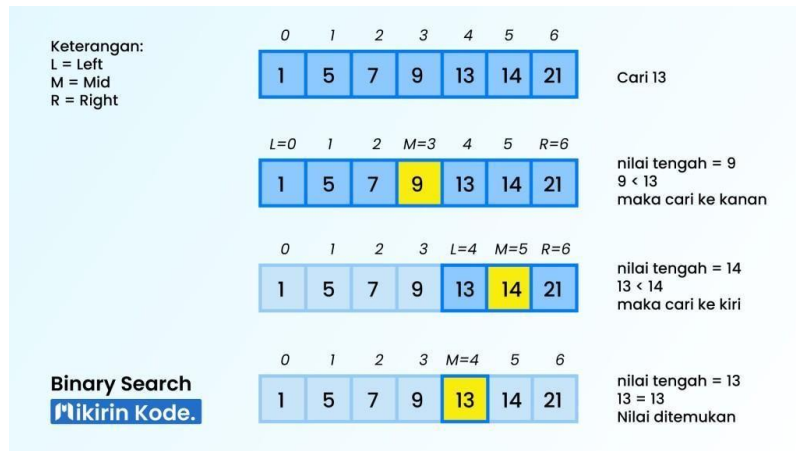
- Data diambil dari posisi 1 sampai posisi akhir N.
- Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
- Selanjutnya data yang dicari akan dibandingkan dengan data yang berada di posisi tengah, apakah lebih besar atau lebih kecil.
- Apabila data yang dicari lebih besar dari 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)  $L \leftarrow 0$
- 2)  $R \leftarrow N - 1$
- 3)  $\text{ketemu} \leftarrow \text{false}$
- 4) Selama  $(L \leq R)$  dan (tidak ketemu) kerjakan baris 5 sampai dengan 8
- 5)  $m \leftarrow (L + R) / 2$
- 6) Jika  $(\text{Data}[m] = x)$  maka  $\text{ketemu} \leftarrow \text{true}$
- 7) Jika  $(x < \text{Data}[m])$  maka  $R \leftarrow m - 1$
- 8) Jika  $(x > \text{Data}[m])$  maka  $L \leftarrow m + 1$
- 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.

### C. GUIDED

1. Buatlah sebuah project dengan menggunakan sequential search sederhana untuk melakukan pencarian data.

**Source code:**

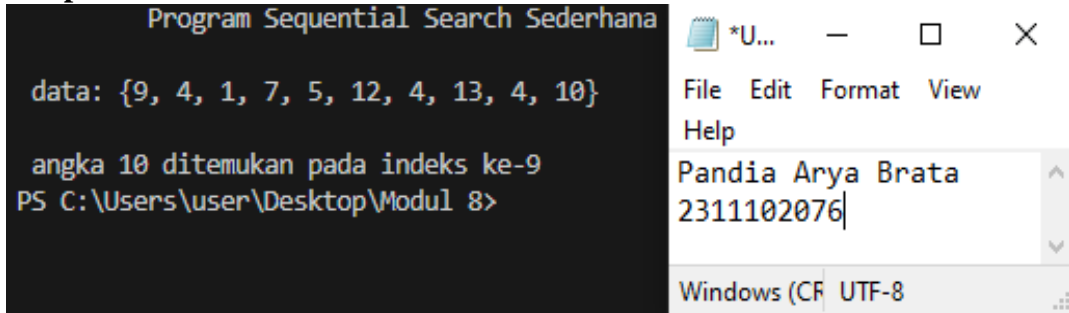
```
#include <iostream>
using namespace std;

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;
};
```

### Output:



### Deskripsi program :

Program ini menunjukkan cara kerja algoritma pencarian sekuensial untuk mencari nilai tertentu dalam array data. Meskipun algoritma ini sederhana dan mudah dipahami, namun kinerjanya tidak seefisien algoritma pencarian lainnya seperti pencarian biner, terutama untuk field data yang besar.

2. Buatlah sebuah project untuk melakukan pencarian data dengan menggunakan Binary Search.

### Source code:

```
#include <iostream>
#include <conio.h>
#include <iomanip>
using namespace std;
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;
```



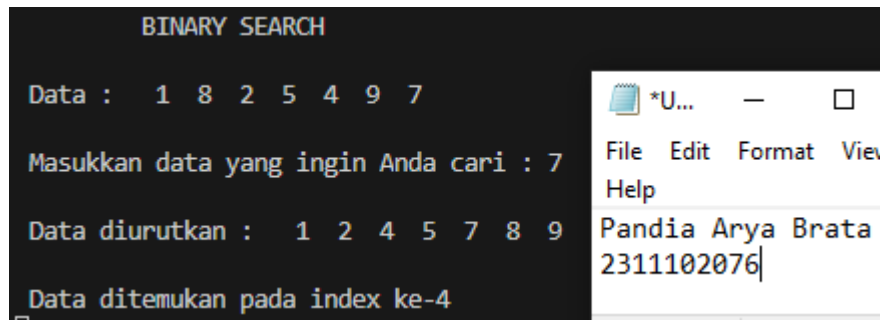
```

    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;
}

```

### Output:



```
BINARY SEARCH
Data : 1 8 2 5 4 9 7
Masukkan data yang ingin Anda cari : 7
Data diurutkan : 1 2 4 5 7 8 9
Data ditemukan pada index ke-4
```

### Deskripsi program :

Program ini menggabungkan dua algoritma yang berguna: Selection Sort untuk mengurutkan data, dan pencarian biner untuk menemukan nilai dalam data yang diurutkan. Program ini menunjukkan kepada Anda bagaimana kedua algoritma bekerja dengan cara yang jelas dan mudah dipahami.

#### D. 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 <string>
#include <algorithm>
using namespace std;

bool binarySearch(string sentence, char target)
{
    int left = 0;
    int right = sentence.length() - 1;
    while (left <= right)
    {
        int mid = left + (right - left) / 2;
        if (sentence[mid] == target)
        {
            return true;
        }
        if (sentence[mid] < target)
        {
            left = mid + 1;
        }
        else
        {
            right = mid - 1;
        }
    }
    return false;
}

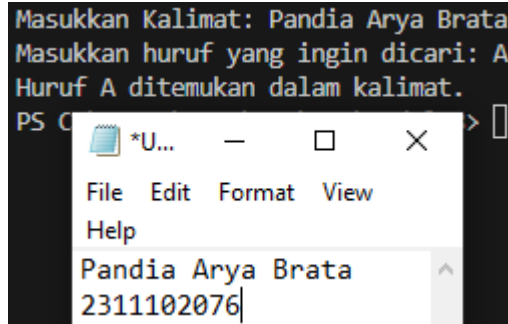
int main()
{
    string sentence;
    char target;
    cout << "Masukkan Kalimat: ";
    getline(cin, sentence);
    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> target;
    // Mengubah kalimat menjadi lowercase agar pencarian case insensitive
    transform(sentence.begin(), sentence.end(), sentence.begin(), ::tolower);
    /// Mencari huruf dalam kalimat menggunakan Binary
    bool found = binarySearch(sentence, tolower(target));
    if (found) {
        cout << "Huruf " << target << " ditemukan dalam kalimat." << endl;
    } else {
        cout << "Huruf " << target << " tidak ditemukan dalam kalimat." << endl;
    }
}
```

```
<<endl;
}
return 0;
}
```

#### Screenshot

#### output

:



#### Deskripsi program

:

1. Program menggunakan header untuk input-output, untuk memanipulasi string, dan untuk menggunakan fungsi transform.
2. Fungsi `binarySearch` adalah implementasi algoritma Binary Search untuk mencari huruf dalam string `sentence`. Fungsi ini menerima dua parameter, yaitu kalimat `sentence` dan huruf target. Fungsi ini mengembalikan `true` jika huruf ditemukan dalam kalimat, dan `false` jika tidak ditemukan.
3. Di dalam fungsi `main`, variabel `sentence` digunakan untuk menyimpan kalimat yang diinputkan oleh pengguna, dan variabel `target` digunakan untuk menyimpan huruf yang ingin dicari.
4. Pengguna diminta untuk memasukkan kalimat menggunakan `getline(cin, sentence)`, dan huruf yang ingin dicari menggunakan `cin >> target`.
5. Menggunakan fungsi `transform`, kalimat diubah menjadi lowercase agar pencarian bersifat case-insensitive. Ini dilakukan agar huruf besar dan huruf kecil dianggap sama dalam pencarian.
6. Pemanggilan fungsi `binarySearch` dilakukan untuk mencari huruf target dalam kalimat `sentence`. Hasil pencarian disimpan dalam variabel `found`.
7. Hasil pencarian dicetak di layar. Jika huruf ditemukan, program mencetak pesan bahwa huruf tersebut ditemukan dalam kalimat. Jika huruf tidak ditemukan, program mencetak pesan bahwa huruf tidak ditemukan dalam kalimat.

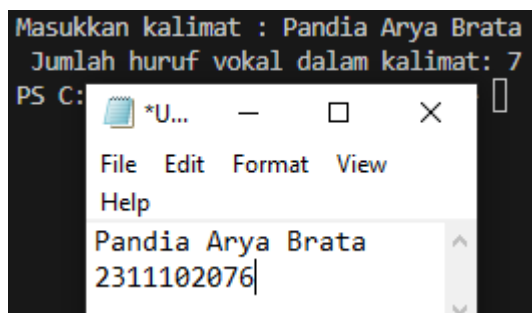
2. Buatlah sebuah program yang dapat menghitung banyaknya huruf vocal dalam sebuah kalimat!

**Source code :**

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

using namespace std;
int countVowels(string sentence)
{
    int count = 0;
    string vowels = "aeiou";
    // Mengubah kalimat menjadi lowercase agar pencarian case
    insensitive
    transform(sentence.begin(), sentence.end(),
    sentence.begin(), ::tolower);
    for (char c : sentence)
    {
        if (vowels.find(c) != string::npos)
        {
            count++;
        }
    }
    return count;
}
int main()
{
    string sentence;
    cout << "Masukkan kalimat : ";
    getline(cin, sentence);
    int vowelCount = countVowels(sentence);
    cout << " Jumlah huruf vokal dalam kalimat: " << vowelCount
    << endl;
    return 0;
}
```

**Screenshot output :**



**Deskripsi program :**

1. Program menggunakan header untuk input-output, untuk memanipulasi string, dan untuk menggunakan fungsi transform.
  2. Fungsi countVowels digunakan untuk menghitung jumlah huruf vokal dalam string sentence. Fungsi ini menerima satu parameter, yaitu kalimat sentence. Fungsi ini mengembalikan jumlah huruf vokal dalam kalimat.
  3. Di dalam fungsi main, variabel sentence digunakan untuk menyimpan kalimat yang diinputkan oleh pengguna.
  4. Pengguna diminta untuk memasukkan kalimat menggunakan getline(cin, sentence).
  5. Menggunakan fungsi transform, kalimat diubah menjadi lowercase agar pencarian bersifat case-insensitive.
  6. Fungsi countVowels dipanggil untuk menghitung jumlah huruf vokal dalam kalimat. Fungsi ini menggunakan loop for untuk memeriksa setiap karakter dalam kalimat. Jika karakter termasuk dalam string vowels (yaitu huruf vokal), maka count akan bertambah.
  7. Hasil perhitungan jumlah huruf vokal dicetak 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!

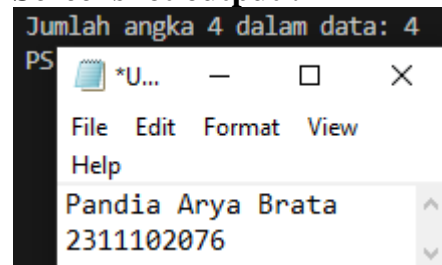
**Source code :**

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

int main()
{
    int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int size = sizeof(data) / sizeof(data[0]); //Mendapatkan
    ukuran array
    int target = 4;
    int count = 0; // Variabel untuk menghitung jumlah kemunculan
    angka 4
    for (int i = 0; i < size; i++)
    {
        if (data[i] == target)
        {
            count++;
        }
    }
    std::cout << "Jumlah angka 4 dalam data: " << count
    << std::endl;
    return 0;
}
```

```
}
```

### Screenshot output :



### Deskripsi program :

1. Program menggunakan header untuk input-output.
2. Fungsi countOccurrences digunakan untuk menghitung berapa kali angka target muncul dalam array data. Fungsi ini menerima tiga parameter, yaitu array data, ukuran array size, dan angka target. Fungsi ini mengembalikan jumlah kemunculan angka target dalam array.
3. Di dalam fungsi main, array data yang diberikan adalah {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}. Variabel size dihitung dengan membagi ukuran total array data dengan ukuran satu elemen array.
4. Variabel target diinisialisasi dengan nilai 4, yang merupakan angka yang ingin dihitung kemunculannya dalam array.
5. Fungsi countOccurrences dipanggil untuk menghitung berapa kali angka 4 muncul dalam array data. Fungsi ini menggunakan loop for untuk memeriksa setiap elemen array. Jika elemen array sama dengan angka target, maka count akan bertambah.
6. Hasil perhitungan jumlah kemunculan angka 4 dicetak di layar.

## KESIMPULAN

- Algoritma searching digunakan untuk mencari elemen tertentu dalam kumpulan data.
- Tujuannya adalah untuk menentukan apakah elemen tersebut ada dalam kumpulan data dan di mana letaknya jika ada.
- Algoritma Sequential Search, juga dikenal sebagai Linear Search, adalah metode sederhana untuk mencari elemen secara berurutan satu per satu dalam kumpulan data. Ini dapat digunakan untuk mencari elemen dalam array atau list.
- Algoritma Binary Search digunakan untuk mencari elemen dalam kumpulan data yang terurut secara terus-menerus, seperti array yang telah diurutkan. Ini bekerja dengan membandingkan elemen yang dicari dengan elemen tengah kumpulan data, dan secara rekursif mengurangi rentang pencarian hingga elemen ditemukan atau rentang pencarian habis.
- Binary Search memiliki keunggulan dalam efisiensi waktu dibandingkan dengan Sequential Search. Pencarian dalam Binary Search memiliki kompleksitas waktu logaritmik  $O(\log n)$ , di mana  $n$  adalah jumlah elemen dalam kumpulan data yang terurut. Sementara itu, kompleksitas waktu Sequential Search adalah linier  $O(n)$ , di mana  $n$  adalah jumlah elemen dalam kumpulan data.
- Pemilihan algoritma searching yang tepat tergantung pada sifat data yang akan dicari. Jika data sudah terurut, Binary Search akan lebih efisien. Namun, jika data tidak terurut atau hanya memiliki jumlah elemen yang sedikit, Sequential Search dapat menjadi pilihan yang lebih sederhana.
- Penting untuk memahami cara kerja dan kompleksitas waktu dari setiap algoritma searching agar dapat memilih yang tepat untuk kasus yang spesifik.
- Faktor seperti ukuran data, kebutuhan waktu eksekusi, dan sifat data harus dipertimbangkan.
- Implementasi algoritma searching dapat dilakukan dalam berbagai bahasa pemrograman, termasuk C++. Pemahaman tentang sintaksis dan struktur dasar bahasa pemrograman sangat membantu dalam mengimplementasikan algoritma searching secara efektif.



## **E. DAFTAR PUSTAKA**

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