

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL VIII  
ALGORITMA SEARCHING**



**Disusun Oleh:**

Arvan Murbiyanto

2311102074

**Dosen**

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

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

## Modul VIII

### ALGORITMA SEARCHING

#### A. Tujuan

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.

#### 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.

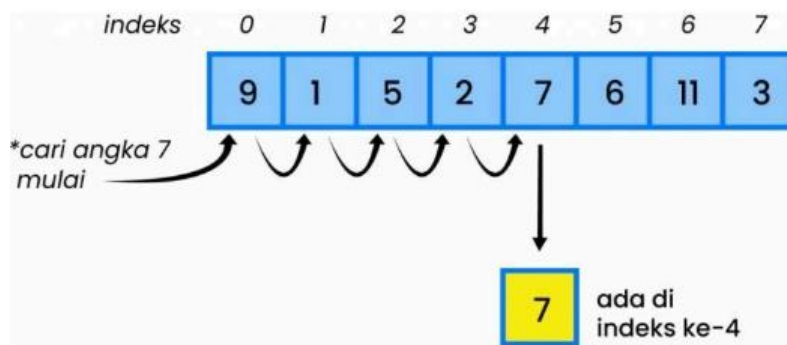
```

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 adalah sebuah algoritma pencarian dengan cara membagi data menjadi dua bagian setiap kali terjadi proses pencarian untuk menemukan nilai tertentu dalam sebuah larik (array) linear. Sebuah pencarian biner mencari nilai tengah (median), melakukan sebuah perbandingan untuk menentukan apakah nilai yang dicari ada sebelum atau sesudahnya, kemudian mencari setengah sisanya dengan cara yang sama. Pencarian Biner (Binary Search) dilakukan untuk :

- Memperkecil jumlah operasi perbandingan yang harus dilakukan antara data yang dicari dengan data yang ada di dalam tabel, khususnya untuk jumlah data yang sangat besar ukurannya.
- Beban komputasi juga lebih kecil karena pencarian dilakukan dari depan, belakang, dan tengah.
- Prinsip dasarnya adalah melakukan proses pembagian ruang pencarian secara berulang-ulang sampai data ditemukan atau sampai ruang pencarian tidak dapat dibagi lagi (berarti ada kemungkinan data tidak ditemukan).
- Syarat utama untuk pencarian biner adalah data di dalam tabel harus sudah terurut.

Algoritma pencarian biner dapat dituliskan sebagai berikut :

- 1)  $L \leftarrow 0$ .
- 2)  $R \leftarrow N - 1$
- 3) 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 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.

## GUIDED 1

```
#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:

```
pp -o guided1per0 } ; 17 (#?) { 17guided1per0 }
    Program Sequential Search Sederhana

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

angka 10 ditemukan pada indeks ke-9
PS C:\Users\ASUS\OneDrive\Desktop\praktikum struktur
```

Deskripsi:

Pada program di atas membuat struktur data antrian menggunakan linked list, Setiap elemen dalam antrian direpresentasikan oleh Node yang memiliki dua atribut : nama dan nim. Setiap Node juga memiliki pointer next yang menunjuk ke elemen berikutnya dalam antrian. Implementasi fungsi antrian meliputi enqueue (menambah antrian), dequeue (menghapus antrian), lihat antrian, jumlah antrian, dan hapus semua antrian. Program utama menampilkan menu pilihan untuk pengguna, termasuk opsi untuk menambah, menghapus, dan melihat antrian.

## GUIDED 2

```
#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;
}

```

Output:

```

BINARY SEARCH

Data awal:  1  8  2  5  4  9  7

Masukkan data yang ingin Anda cari: 8

Data diurutkan:  1  2  4  5  7  8  9

Data ditemukan pada index ke-5
PS C:\Users\ASUS\OneDrive\Desktop\praktiku

```

Deskripsi:

Pada kode di atas, Anda mendeklarasikan sebuah array bernama arrayData dengan 7 elemen bertipe int. Selanjutnya, terdapat dua fungsi utama: selection\_sort dan binary\_search. Fungsi selection\_sort mengurutkan elemen-elemen dalam arrayData menggunakan algoritma selection sort, sementara fungsi binary\_search melakukan pencarian data dalam arrayData menggunakan algoritma binary search.

### C. UNGUIDED

1. Buatlah sebuah program untuk mencari sebuah huruf pada sebuah kalimat yang sudah di input dengan menggunakan Binary Search!

SOURCE CODE : UNGUIDED 1

```
#include <iostream>
#include <conio.h>
#include <iomanip>

using namespace std;

string sentence;
char c;

void toLower() {
    string temp;
    for (int i = 0; i < sentence.length(); i++) {
        temp += tolower(sentence[i]);
    }
    sentence = temp;
}

void selection_sort()
{
    int min, i, j;
    char temp;
    toLower();
    for (i = 0; i < sentence.length(); i++)
    {
        min = i;
        for (j = i + 1; j < sentence.length(); j++)
        {
            if (sentence[j] < sentence[min])
            {
                min = j;
            }
        }
    }
}
```



```

        temp = sentence[i];
        sentence[i] = sentence[min];
        sentence[min] = temp;
    }
}

void binarysearch()
{
    int awal, akhir, tengah, b_flag = 0;
    awal = 0;
    akhir = sentence.length();
    while (b_flag == 0 && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (sentence[tengah] == c)
        {
            b_flag = 1;
            break;
        }
        else if (sentence[tengah] < c)
            awal = tengah + 1;
        else
            akhir = tengah - 1;
    }
    if (b_flag == 1)
        cout << "\n Karakter '" << c << "' ditemukan pada index ke -"
        << tengah << endl;
    else
        cout << "\nData tidak ditemukan\n";
}

int main()
{
    cout << "\t -----BINARY SEARCH----- " << endl;
    cout << "Masukkan kalimat : ";
    getline(cin, sentence);
    cout << "\nMasukkan karakter yang ingin Anda cari : ";
    cin >> c;
    c = tolower(c);
    cout << "\nKalimat yang diurutkan berdasarkan karakter : ";
    selection_sort();
    for (int x = 0; x < sentence.length(); x++)
        cout << sentence[x];
    cout << endl;
    binarysearch();
    return EXIT_SUCCESS;
}

```

## SCREENSHOT OUTPUT

```
Masukkan kalimat : cristianorojalinemarbogor
Masukkan karakter yang ingin Anda cari : j
Kalimat yang diurutkan berdasarkan karakter : aaabcegiijlmnnooooorrrsty
Karakter 'j' ditemukan pada index ke - 10
PS C:\Users\ASUS\OneDrive\Desktop\praktikum struktur data\modul8.cpp> █
```

## DESKRIPSI:

Program di atas mengimplementasikan algoritma Binary Search untuk mencari karakter tertentu dalam sebuah kalimat. Pengguna diminta memasukkan kalimat (sebuah string) dan karakter yang ingin dicari. Kalimat tersebut kemudian diurutkan berdasarkan karakternya, dan pencarian karakter dilakukan dengan menggunakan algoritma binary search. Hasil dari pencarian (apakah karakter ditemukan atau tidak) ditampilkan sebagai output. Algoritma binary search efisien karena membagi interval pencarian menjadi dua bagian secara berulang hingga menemukan karakter yang dicari atau menentukan bahwa karakter tersebut tidak ada dalam kalimat.

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

## SOURCE CODE : UNGUIDED 2

```
#include <iostream>
using namespace std;
int main()
{
    string s;
    int count = 0;

    cout << "\t -----Jumlah huruf vokal-----" << endl;
    cout << "\nMasukkan kalimat : ";
    getline(cin, s);

    for (int i = 0; i < s.length(); i++) {
        char c = tolower(s[i]);
        if (c == 'a' || c == 'i' || c == 'u' || c == 'e' || c == 'o')
            count++;
    }

    cout << "\nKalimat yang dimasukkan memiliki " << count << " huruf vokal.";
    return 0;
}
```

## SRENSHOOT OUTPUT:

```
-----Jumlah huruf vokal-----  
  
Masukkan kalimat : cristiano rojali pernah ke indonesia  
  
Kalimat yang dimasukkan memiliki 15 huruf vokal.  
PS C:\Users\ASUS\OneDrive\Desktop\praktikum struktur data\
```

## DESKRIPSI

Program di atas menghitung jumlah huruf vocal pada sebuah kalimat

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 : UNGUIDED 3

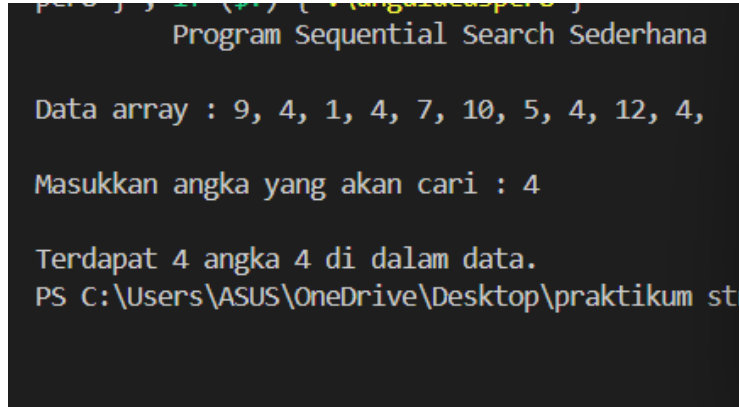
```
#include <iostream>  
using namespace std;  
  
int arr[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};  
int arrLength = sizeof(arr)/sizeof(arr[0]);  
int cari;  
  
int seqSearch() {  
    int count = 0;  
    for (int i = 0; i < arrLength; i++)  
    {  
        if (arr[i] == cari)  
        {  
            count++;  
        }  
    }  
    return count;  
}  
  
int main()  
{  
    // algoritma Sequential Search  
  
    cout << "\t Program Sequential Search Sederhana\n" << endl;  
    cout << "Data array : ";  
    for (int x : arr) {  
        cout << x << ", ";  
    }  
    cout << "\n\nMasukkan angka yang akan cari : ";  
    cin >> cari;
```

```

        cout << "\nTerdapat " << seqSearch() << " angka " << cari << "
di dalam data.";
        return 0;
    }

```

#### SCREENSHOOT OUTPUT



```

Program Sequential Search Sederhana

Data array : 9, 4, 1, 4, 7, 10, 5, 4, 12, 4,

Masukkan angka yang akan cari : 4

Terdapat 4 angka 4 di dalam data.
PS C:\Users\ASUS\OneDrive\Desktop\praktikum st

```

#### DESKRIPSI:

Program di atas menghitung berapa banyak angka 4 dengan menggunakan algoritma Sequential Search!

#### D. KESIMPULAN

Sequential Search adalah algoritma pencarian yang memeriksa setiap elemen pada array secara berurutan. Proses pencarian dimulai dari indeks pertama hingga akhir. Algoritma ini cocok untuk data yang belum terurut. Jika data ditemukan, proses pencarian berhenti. Jika tidak, pencarian tetap dihentikan setelah memeriksa seluruh elemen pada array.

Binary Search, di sisi lain, bekerja dengan cara yang berbeda. Algoritma ini cocok untuk data yang sudah terurut. Binary search memeriksa elemen di tengah-tengah koleksi. Jika elemen yang dicari lebih kecil atau lebih besar dari elemen yang ditemukan, maka sub-array didefinisikan dan pencarian dilanjutkan. Binary search lebih efisien untuk data yang terurut.

#### E. REFERENSI

[1] Asprak “Modul 8 ALGORITMA SEARCHING” Learning Meaning system

[2] blogspot

<http://blog-sharings.blogspot.com/2012/07/konsep-binary-search.html>

[3] Wordpress

<https://tirago4.wordpress.com/2016/10/11/pencarian-searching-didalam-algoritma/>