

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN ALGORITMA**

**MODUL 8  
ALGORITMA SEARCHING**



**Dosen : Wahyu Andi Saputra, S.Pd., M.Eng.**

**Disusun oleh:**

**MUHAMMAD AULIA MUZZAKI NUGRAHA**

**(2311102051)**

**IF-11-B**

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

**2024**

# BAB I

## DASAR TEORI

### 1. Konsep Dasar Searching

- Searching adalah proses mendapatkan (retrieve) informasi berdasarkan kunci tertentu dari sejumlah informasi yang lebih disimpan.

### 2. Sequential Search

Konsep dasar Algoritma Sequential Search:

- Terdapat larik yang berisi n sejumlah data (  $L[0], L[1], \dots, L[n-1]$  )
- i adalah bilangan indeks terkecil, yang memenuhi kondisi  $0 < k < n - 1$
- k adalah data yang dicari  $\Rightarrow L[i]=k$

Algoritma Sequential Search:

- Berikut ini merupakan implementasi algoritma pencarian secara sekuensial.
- Subrutin akan menghasilkan nilai balik berupa:

a. -1 jika data yang dicari tidak ditemukan dan

b. Bilangan antara 0 samapai dengan n-1 (dengan n adalah jumlah elemen larik) jika data yang dicari ditemukan

Pseudocode Sequential Search:

```
SUBROUTIN cari (L, n, k)
  JIKA n ≤ 0 MAKA
    posisi ← - 1
  SEBALIKNYA
    ketemu ← SALAH
    i ← 0
    ULANG SELAMA (i < n-1) DAN (TIDAK ketemu)
      JIKA K = L[i] MAKA
        posisi ← i
        ketemu ← BENAR
      SEBALIKNYA
        i ← i + 1
    AKHIR-JIKA
  AKHIR ULANG
  JIKA TIDAK ketemu MAKA
    posisi ← -1
  AKHIR-JIKA
  AKHIR-JIKA
  NILAI-BALIK posisi
AKHIR-SUBROUTIN
```

Dengan **L = larik**; **n = jml elemen larik**; dan **k = data yang dicari**

Fungsi Sequential Search dalam bahasa C++:

```
int sequential_search(int data [], int n, int k) {
    int posisi, i, ketemu;

    if (n <= 0)
        posisi = -1;
    else {
        ketemu = 0;
        i = 1;
        while ((i < n - 1) && ! ketemu) {
            if (data [i] == k) {
                posisi = i;
                ketemu = 1;
            } else {
                i++;
            }

            if (!ketemu) {
                posisi = -1;
            }
        }
    }
    return posisi;
}
```

Dari fungsi tersebut dapat diketahui jika program menemukan kunci yang dicari maka fungsi akan mengembalikan posisi indeks dari kunci tersebut pada array. Sedangkan jika program tidak menemukan kunci maka program akan mengemballlikan nilai -1.

### Contoh

Diketahui array suatu integer terdiri dari 7 elemen sebagai berikut : {1, 0, 3, 2, 5, 7, 9}. Contoh Program untuk mencari angka/kunci 3 pada array tersebut adalah sebagai berikut :

```

#include <iostream>

using namespace std;

int sequential_search(int data [], int n, int k) {
    int posisi, i, ketemu;

    if (n <= 0)
        posisi = -1;
    else {
        ketemu = 0;
        i = 1;
        while ((i < n -1) && ! ketemu) {
            if (data [i] == k) {
                posisi = i;
                ketemu = 1;
            } else {
                i++;
            }
        }

        if (!ketemu) {
            posisi = -1;

```

```

        }
    }
    return posisi;
}

int main() {

    int n      = 7;
    int data[] = {1, 0, 3, 2, 5, 7, 9};
    int k      = 4;

    int posisi = sequential_search(data, n, k);

    if(posisi != -1) {
        cout << "kunci " << k << " ditemukan pada posisi indeks ke-"
<< posisi << endl;
    } else {
        cout << "kunci " << k << " tidak ditemukan" << endl;
    }

    return 0;
}

```

### 3. Binary Search

#### Konsep dasar Algoritma Bubble Sort

- Apabila kumpulan data sudah dalam keadaan **terurut**, pencarian data dengan menggunakan pencarian sekuensial akan memakan waktu yang lama jika jumlah data dalam kumpulan data tersebut sangat banyak.
- Pencarian biner dilakukan dengan membagi larik menjadi dua bagian dengan jumlah yang sama atau berbeda 1 jika jumlah data semula ganjil
- Data yang dicari kemudian dibandingkan dengan data tengah pada bagian pertama
- Dalam hal ini akan terjadi 3 kemungkinan yang terjadi :
  - Data yang dicari sama dengan elemen tengah pada bagian pertama dalam larik. Jika kondisi ini terpenuhi, data yang dicari berarti ditemukan.
  - Data yang dicari bernilai kurang dari nilai elemen tengah pada bagian pertama dalam larik. Pada keadaan ini, pencarian diteruskan pada bagian pertama.
  - Data yang dicari bernilai lebih dari nilai elemen tengah pada bagian pertama dalam larik. Pada keadaan ini, pencarian diteruskan pada bagian kedua.

### Pseudocode Binary Search

```

SUBROUTIN cari_biner (L, n, k)
  ada ← SALAH
  bawah ← 0
  atas ← n - 1
  ULANG SELAMA atas ≥ bawah
    tengah ← (atas + bawah) / 2 // Pembagi Bulat
    JIKA k > L[tengah] MAKA // Data dicari dikanan
      bawah ← tengah + 1
    SEBALIKNYA
      JIKA k < L[tengah] MAKA // Data dicari dikiri
        atas = tengah - 1

```

```

      SEBALIKNYA
        ada ← BENAR
        posisi ← tengah
        bawah ← atas + 1 // Supaya perulangan berakhir
      AKHIR-JIKA
    AKHIR-JIKA
  AKHIR-ULANG
  JIKA TIDAK ada MAKA
    posisi ← -1
  AKHIR-JIKA
  NILAI-BALIK posisi
AKHIR-SUBROUTIN

```

### Fungsi Binary Search dalam Bahasa C++:

```
int function_binary_search (int data [], int n, int k) {
    int atas, bawah, tengah, posisi;
    bool ada;

    ada      = false;
    bawah    = 0;
    atas     = n - 1;
    posisi   = -1;

    while (bawah <= atas) {
        tengah = (atas + bawah)/2;
        if(k == data[tengah]) {
            posisi = tengah;
            break;
        }
        else if(k < data[tengah]) atas = tengah - 1;
        else if(k > data[tengah]) bawah = tengah + 1;
    }

    return posisi;
}
```

## BAB II

### GUIDED

#### LATIHAN – GUIDED

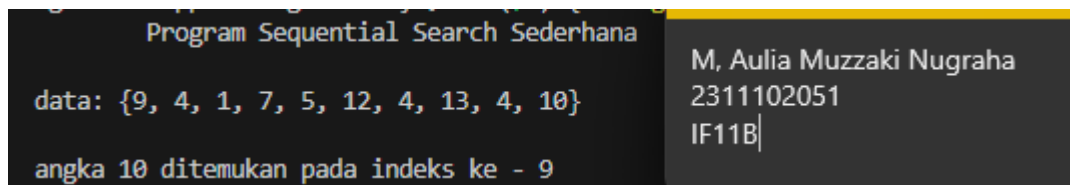
##### 1. Guided 1

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;
}
```

##### Screenshoot program



### Deskripsi program

Program menggunakan perulangan for untuk memeriksa setiap elemen dalam array data secara berurutan. Pada setiap iterasi, program membandingkan nilai elemen saat ini dengan nilai cari. Jika nilai tersebut sama dengan cari, variabel ketemu diatur menjadi true, dan perulangan dihentikan dengan menggunakan perintah break. Jika perulangan selesai dan nilai cari tidak ditemukan, variabel ketemu tetap bernilai false.

### 2. Guided 2

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

### 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;
```



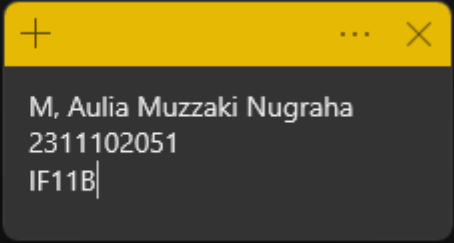
```

    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 :4
Data diurutkan : 1 2 4 5 7 8 9
Data ditemukan pada index ke- 2
```



### Deskripsi program

Algoritma utama yang digunakan dalam program ini adalah Selection Sort untuk mengurutkan array, dan Binary Search untuk melakukan pencarian biner pada array yang telah terurut. Algoritma Binary Search bekerja dengan cara membagi array menjadi dua bagian pada setiap iterasi, dan membuang setengah bagian yang tidak mungkin mengandung elemen yang dicari. Proses ini diulangi sampai elemen ditemukan atau seluruh array telah diperiksa.

## BAB III

### UNGUIDED

#### TUGAS – UNGUIDED

##### 1. 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 kiri = 0;
    int kanan = sentence.length() - 1;
    while (kiri <= kanan)
    {
        int mid = kiri + (kanan - kiri) / 2;
        if (sentence[mid] == target)
        {
            return true;
        }
        if (sentence[mid] < target)
        {
            kiri = mid + 1;
        }
        else
        {
            kanan = mid - 1;
        }
    }
    return false;
}

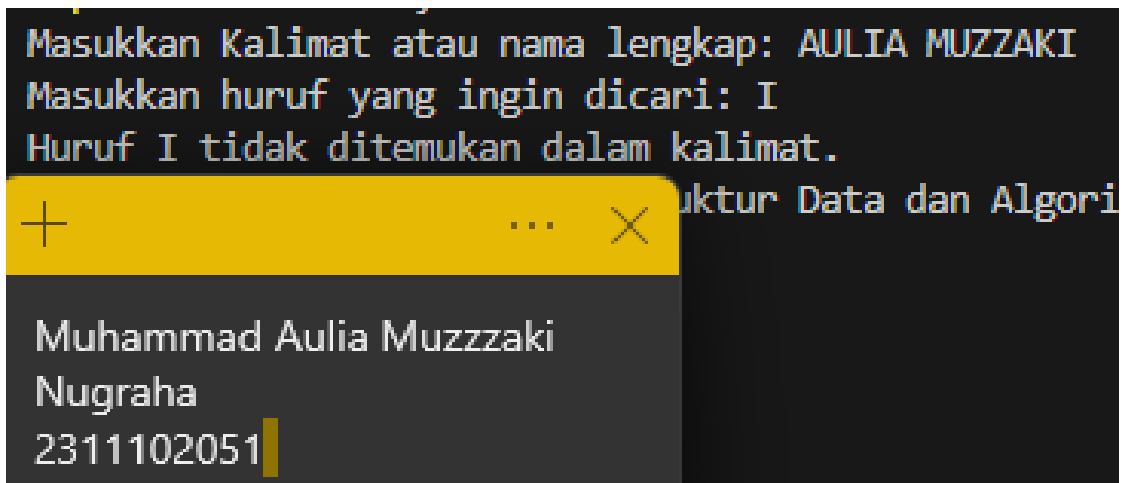
int main()
{
    string sentence;
    char tuju;
    cout << "Masukkan Kalimat atau nama lengkap: ";
    getline(cin, sentence);
    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> tuju;
    transform(sentence.begin(), sentence.end(), sentence.begin(),
::tolower); // Mengubah kalimat menjadi lowercase agar pencariancase
intensitive
```

```

    bool found = binarySearch(sentence, tolower(tuju)); // Mencari
huruf dalam kalimat menggunakan Binary
    if (found)
    {
        cout << "Huruf " << tuju << " ditemukan dalam kalimat." <<
endl;
    }
    else
    {
        cout << "Huruf " << tuju << " tidak ditemukan dalam kalimat."
<< endl;
    }
    return 0;
}

```

### Screenshot Program



### Deskripsi program

Program akan memberikan output kepada pengguna apakah karakter target ditemukan atau tidak dalam kalimat yang dimasukkan sebelumnya. Jika ditemukan, program akan menampilkan pesan bahwa karakter tersebut ditemukan di dalam kalimat. Jika tidak ditemukan, program akan menampilkan pesan bahwa karakter tersebut tidak ada di dalam kalimat yang dimasukkan.

### 2. Guided 2

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

### Source code

```

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

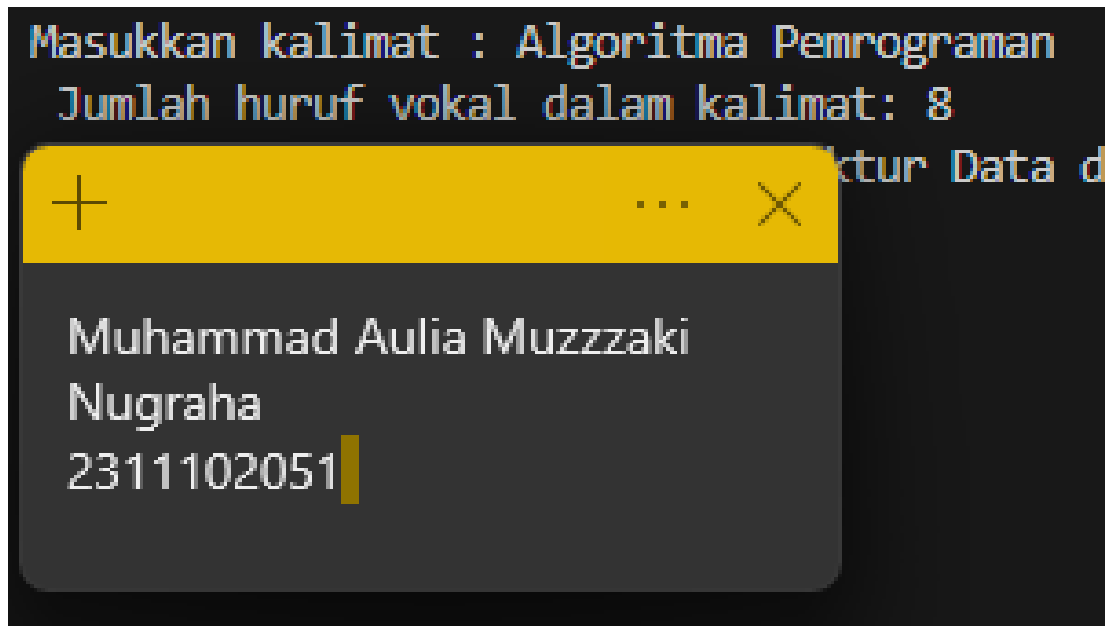
```

```
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;
}
```

**Screenshoot program**



### Deskripsi program

Fungsi melakukan iterasi melalui setiap karakter dalam kalimat menggunakan perulangan for. Pada setiap iterasi, fungsi memeriksa apakah karakter tersebut merupakan huruf vokal atau bukan dengan menggunakan fungsi find dari kelas string. Jika karakter tersebut merupakan huruf vokal, maka variabel count akan diinkrementasi.

### 3. Guided 3

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

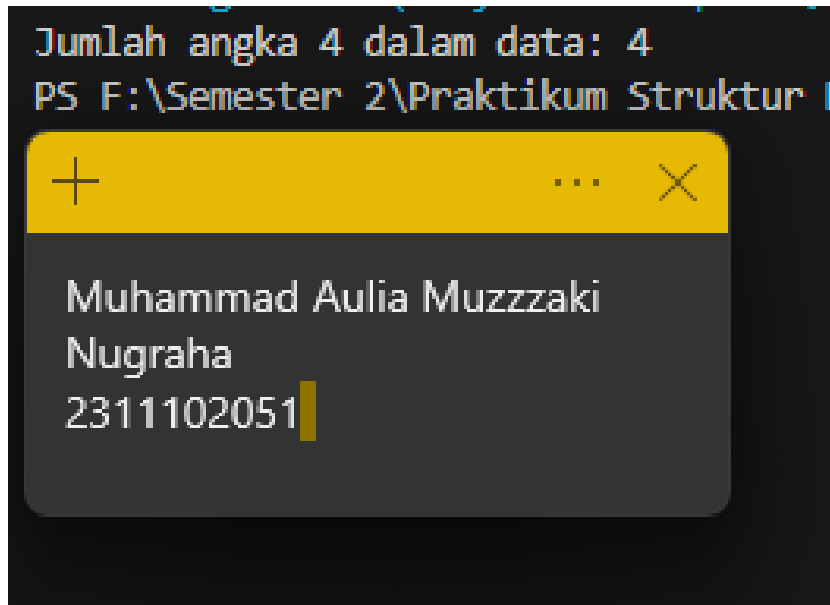
### Source code

```
#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++;
        }
    }
}
```

```
cout << "Jumlah angka 4 dalam data: " << count << endl;  
return 0;  
}
```

### Screenshoot program



### Deskripsi program

program mendefinisikan sebuah array bernama data yang berisi sepuluh elemen berupa angka-angka bulat. Kemudian, program menghitung ukuran array tersebut dengan menggunakan ekspresi `sizeof(data) / sizeof(data[0])`. Hasil perhitungan ini disimpan dalam variabel `size`. Selanjutnya, program mendefinisikan sebuah variabel target yang berisi angka 4, serta sebuah variabel `count` yang diinisialisasi dengan nilai 0. Variabel `count` ini akan digunakan untuk menghitung jumlah kemunculan angka 4 dalam array.

## **BAB IV**

### **KESIMPULAN**

Dalam mempelajari praktikum Algoritma Searching, dua algoritma yang telah dipelajari adalah Sequential Search (Pencarian Linear) dan Binary Search (Pencarian Biner). Sequential Search merupakan algoritma pencarian paling sederhana yang mencari data secara berurutan dari awal hingga akhir dalam sebuah kumpulan data. Algoritma ini cocok digunakan untuk data yang tidak terurut dan memiliki kompleksitas waktu terburuk  $O(n)$ , di mana  $n$  adalah jumlah elemen dalam kumpulan data.

Di sisi lain, Binary Search merupakan algoritma pencarian yang efisien untuk data yang telah terurut. Prinsip kerjanya adalah membagi data menjadi dua bagian dan mencari di bagian yang sesuai dengan nilai yang dicari. Algoritma ini memiliki kompleksitas waktu  $O(\log n)$ , yang berarti lebih cepat daripada Sequential Search untuk data besar yang terurut. Dalam praktikum, implementasi kedua algoritma ini dalam bahasa pemrograman seperti Python, Java, atau C++ memberikan pengalaman praktis dalam memahami kinerja dan karakteristik masing-masing algoritma melalui berbagai kasus uji dengan data yang berbeda-beda.



## DAFTAR PUSTAKA

- [1] Asisten Praktikum, "*Modul 8 Searching*", Learning Management System, 2023.
- [2] Asisten Praktikum, "*Modul 8 Algoritma Searching*", 2024.