

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN  
PEMROGRAMAN  
MODUL VIII  
ALGORITMA SEARCHING**



**Disusun oleh :**

**Ahmadan Syaridin**

**2311102038**

**IF-11-A**

**Dosen Pengampu :**

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

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

**2024**

# **BAB I**

## **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.

## BAB II

### 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.

#### 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:

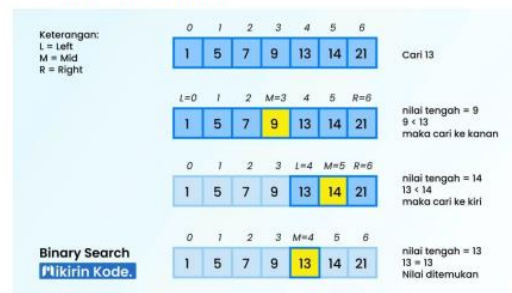
- Data diambil dari posisi 1 sampai posisi akhir N.
- Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
- Selanjutnya data yang dicari akan dibandingkan dengan data yang berada di posisi tengah, apakah lebih besar atau lebih kecil.
- Apabila data yang dicari lebih besar dari 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  $(L \leq R)$  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

### C. GUIDED

#### 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;

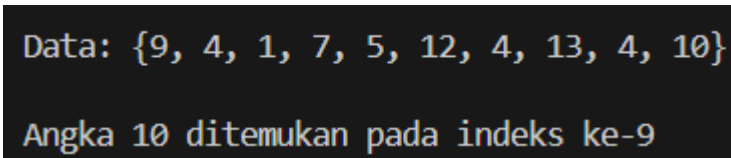
    // Algoritma Sequential Search
    for (i = 0; i < n; i++) {
        if (data[i] == cari) {
            ketemu = true;
            break;
        }
    }

    cout << "\tProgram Sequential Search Sederhana\n" << endl;
    cout << "Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}" << endl;

    if (ketemu) {
        cout << "\nAngka " << cari << " ditemukan pada indeks ke-" << i << endl;
    } else {
        cout << cari << " tidak dapat ditemukan pada data." << endl;
    }

    return 0;
}
```

#### OUTPUT



```
Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}
Angka 10 ditemukan pada indeks ke-9
```

#### DESKRIPSI PROGRAM

Program ini menggunakan algoritma pencarian berurutan untuk mencari elemen tertentu (angka 10) dalam array berukuran 10. Program memeriksa setiap elemen dari awal hingga akhir array, menandai variabel ketemu sebagai benar jika elemen ditemukan, dan mencetak indeks elemen tersebut. Jika elemen tidak ditemukan, program mencetak pesan bahwa elemen tersebut tidak ada. Program juga menampilkan data array sebagai referensi.

## GUIDED 2

### SOURCE CODE

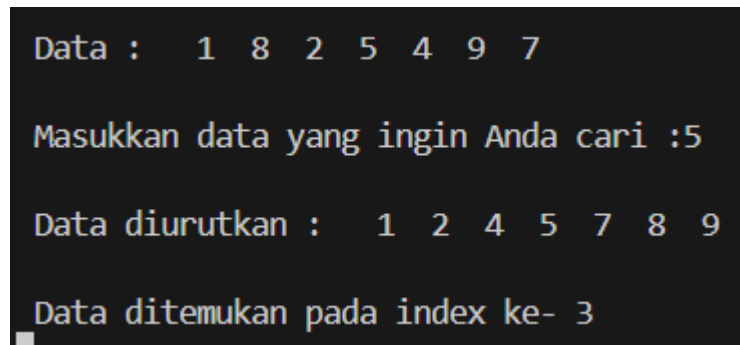
```
#include <iostream>
using namespace std;
#include <conio.h>
#include <iomanip>
int data1[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;
void selection_sort()
{
    int temp, min, i, j;
    for (i = 0; i < 7; i++)
    {
        min = i;
        for (j = i + 1; j < 7; j++)
        {
            if (data1[j] < data1[min])
            {
                min = j;
            }
        }
        temp = data1[i];
        data1[i] = data1[min];
        data1[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 (data1[tengah] == cari)
        {
            b_flag = 1;
            break;
        }
        else if (data1[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 data1 awal
```

```

for (int x = 0; x < 7; x++)
    cout << setw(3) << data1[x];
cout << endl;
cout << "\n Masukkan data yang ingin Anda cari :";
cin >> cari;
cout << "\n Data diurutkan : ";
// urutkan data1 dengan selection sort
selection_sort();
// tampilkan data1 setelah diurutkan
for (int x = 0; x < 7; x++)
    cout << setw(3) << data1[x];
cout << endl;
binarysearch();
_getche();
return EXIT_SUCCESS;
}

```

## OUTPUT



```

Data :   1  8  2  5  4  9  7

Masukkan data yang ingin Anda cari :5

Data diurutkan :   1  2  4  5  7  8  9

Data ditemukan pada index ke- 3

```

## DESKRIPSI PROGRAM

Program ini mengurutkan array data1 berukuran 7 menggunakan selection sort, lalu melakukan pencarian biner untuk mencari angka yang dimasukkan oleh pengguna. Jika angka ditemukan, program mencetak indeksinya; jika tidak, program mencetak pesan bahwa angka tersebut tidak ditemukan. Output program menampilkan array sebelum dan setelah diurutkan.

## BAB IV

### D. UNGUIDED

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

#### SOURCE CODE

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

using namespace std;

// Fungsi binary search untuk mencari huruf dalam string
int binarySearch(const string &str, char target) {
    int left = 0;
    int right = str.length() - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (str[mid] == target) {
            return mid; // huruf ditemukan
        } else if (str[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1; // huruf tidak ditemukan
}

int main() {
    string kalimat;
    char huruf;

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

    // Mengurutkan kalimat terlebih dahulu agar binary search
    dapat bekerja
    sort(kalimat.begin(), kalimat.end());

    // Input huruf yang dicari
    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> huruf;

    // Melakukan binary search pada kalimat
    int posisi = binarySearch(kalimat, huruf);

    // Mengecek apakah huruf ditemukan atau tidak
    if (posisi != -1) {
        cout << "Huruf \"" << huruf << "\" ditemukan pada
posisi ke-" << posisi + 1 << " dalam kalimat." << endl;
    } else {
        cout << "Huruf \"" << huruf << "\" tidak ditemukan
dalam kalimat." << endl;
    }
}
```



```
        return 0;
    }
```

## OUTPUT

```
Masukkan kalimat: Ahmadan Syaridin
Masukkan huruf yang ingin dicari: m
Huruf "m" ditemukan pada posisi ke-12 dalam kalimat.
```

## DESKRIPSI PROGRAM

Program tersebut membaca sebuah kalimat dari pengguna, mengurutkannya, dan kemudian menggunakan algoritma pencarian biner untuk mencari huruf yang dimasukkan oleh pengguna. Jika huruf ditemukan, program mencetak posisi huruf tersebut dalam kalimat yang telah diurutkan; jika tidak, program mencetak pesan bahwa huruf tersebut tidak ditemukan.

- b. Buatlah sebuah program yang dapat menghitung banyaknya huruf vocal dalam sebuah kalimat!**

## SORCE CODE

```
#include <iostream>
#include <string>
#include <cctype> // Untuk fungsi isalpha dan tolower

using namespace std;

// Fungsi untuk menghitung jumlah huruf vokal dalam sebuah kalimat
int hitungVokal(const string &kalimat)
{
    int jumlahVokal = 0;
    for (char huruf : kalimat)
    {
        // Mengonversi huruf menjadi huruf kecil
        char lowerHuruf = tolower(huruf);
        // Memeriksa apakah huruf merupakan huruf vokal
        if (lowerHuruf == 'a' || lowerHuruf == 'e' || lowerHuruf ==
'i' || lowerHuruf == 'o' || lowerHuruf == 'u')
        {
            jumlahVokal++;
        }
    }
    return jumlahVokal;
}

int main()
{
    string kalimat;
    // Input kalimat dari pengguna
    cout << "Masukkan sebuah kalimat: ";
    getline(cin, kalimat);

    // Menghitung jumlah huruf vokal dalam kalimat
    int jumlahVokal = hitungVokal(kalimat);
```

```

        // Menampilkan hasil
        cout << "Jumlah huruf vokal dalam kalimat adalah: " <<
        jumlahVokal << endl;

        return 0;
    }

```

## OUTPUT

```

Masukkan sebuah kalimat: AHMADAN
Jumlah huruf vokal dalam kalimat adalah: 3

```

## DESKRIPSI PROGRAM

Program ini menerima input kalimat dari pengguna dan menghitung jumlah huruf vokal di dalamnya. Dengan mengonversi setiap huruf menjadi huruf kecil, program memeriksa apakah huruf tersebut termasuk dalam lima huruf vokal bahasa Inggris ('a', 'e', 'i', 'o', 'u'). Jika ya, program menambahkan ke penghitung huruf vokal. Akhirnya, program menampilkan jumlah total huruf vokal yang ditemukan dalam kalimat tersebut.

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

## SOURCE CODE

```

#include <iostream>
using namespace std;

// Fungsi untuk mencari jumlah kemunculan suatu angka dalam
// array dengan Sequential Search
int hitungAngka(const int data[], int ukuran, int angka) {
    int jumlah = 0;
    for (int i = 0; i < ukuran; ++i) {
        if (data[i] == angka) {
            jumlah++;
        }
    }
    return jumlah;
}

int main() {
    const int ukuran = 10;
    int data[ukuran] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int angkaYangDicari = 4;

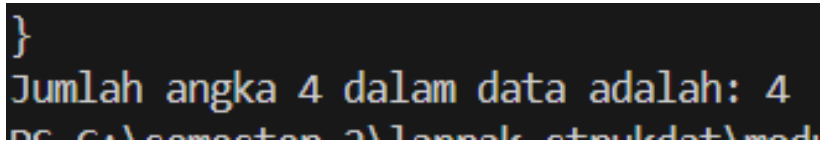
    // Menghitung jumlah kemunculan angka 4 dalam data
    // menggunakan Sequential Search
    int jumlahAngka4 = hitungAngka(data, ukuran,
    angkaYangDicari);

    // Menampilkan hasil
    cout << "Jumlah angka " << angkaYangDicari << " dalam data
    adalah: " << jumlahAngka4 << endl;

    return 0;
}

```

## SCREENSHOOT PROGRAM



```
}  
Jumlah angka 4 dalam data adalah: 4  
D5 C:\semester 2\lengkap\strukdat\modu
```

## DESKRIPSI PROGRAM

Program ini menggunakan metode pencarian berurutan (Sequential Search) untuk mencari jumlah kemunculan angka tertentu dalam sebuah array integer. Array tersebut telah diinisialisasi sebelumnya dengan nilai-nilai acak. Dalam contoh ini, angka yang dicari adalah 4. Setelah pencarian selesai, program menghitung jumlah kemunculan angka tersebut dalam array dan menampilkan hasilnya ke layar.

## **BAB V**

### **E. KESIMPULAN**

Modul 8 tentang Algoritma Searching membahas dua algoritma pencarian utama: Sequential Search dan Binary Search. Sequential Search adalah metode sederhana yang memeriksa setiap elemen satu per satu, cocok untuk data yang belum terurut, meskipun kurang efisien untuk dataset besar dengan kompleksitas waktu  $O(n)$ . Di sisi lain, Binary Search bekerja dengan membagi data menjadi dua bagian dan menawarkan efisiensi lebih tinggi dengan kompleksitas waktu  $O(\log n)$ , membuatnya lebih efektif untuk dataset besar yang terurut. Modul ini mencakup contoh implementasi kedua algoritma dalam bahasa C++, termasuk pencarian elemen dalam array dan penggunaan Selection Sort sebelum pencarian biner. Latihan mandiri diberikan untuk memperdalam pemahaman, seperti mencari huruf dalam kalimat dan menghitung jumlah kemunculan angka dalam array. Referensi utama modul ini adalah buku "Data Structures and Algorithms Made Easy" oleh Narasimha Karumanchi, yang memberikan pemahaman komprehensif tentang teori dan aplikasi praktis algoritma pencarian dalam pemrograman.

## **BAB VI**

### **F. DAFTAR PUSTAKA**

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