

**LAPORAN PRAKTIKUM**

**MODUL 8**

**ALGORITMA SEARCHING**



**Disusun Oleh:**  
Riyon Aryono : **2211102241**

**Dosen**  
Muhamad Azrino Gustalika

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**  
**FAKULTAS INFORMATIKA**  
**INSTITUT TEKNOLOGI TELKOM**  
**PURWOKERTO**  
**2023**

# **BAB I**

## **TUJUAN PRAKTIKUM**

### **A. Tujuan Praktikum**

1. Mahasiswa mampu menjelaskan definisi dan konsep dari Algoritma Searching
2. Mahasiswa mampu menerapkan Algoritma Searching kedalam pemograman

## **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:

#### **1. 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

#### **2. 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

## BAB III

### LATIHAN & TUGAS

#### 1. Guided

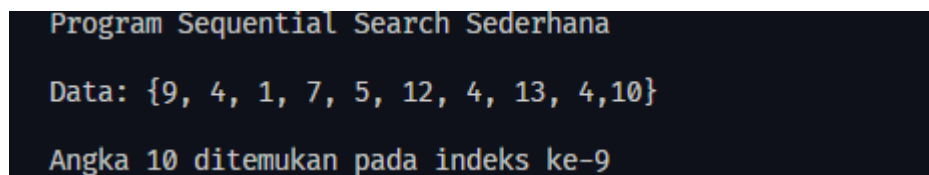
- Demo Sequential Search

*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;

    for (i = 0; i < n; i++){
        if (data[i] == cari) {
            ketemu = true;
            break;
        }
    }
    cout << "Program 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*

A screenshot of a terminal window showing the output of the C++ program. The text is displayed in a monospaced font on a dark background. The output consists of three lines: "Program Sequential Search Sederhana", "Data: {9, 4, 1, 7, 5, 12, 4, 13, 4,10}", and "Angka 10 ditemukan pada indeks ke-9".

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

Penjelasan

Kode di atas adalah implementasi konsep Sequential search program sequential search akan melakukan looping sebanyak n sampai data yang di

cari ketemu disini kita mencari angka 10 maka akan di cari dari array indeks pertama sampai akhir

- Demo 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;
}

```

## Output

```

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

```

## Penjelasan

Kode diatas adalah contoh dari implementasi algoritma binary search, berbeda dengan algoritma sequential search yang mengecek data sampai ke n yang dilakukan binary search pertama kali adalah mengurutkan data nya terlebih dahulu baru kemduain mencari data yang ingin dicari

## 2. Unguided

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

*Source Code*

```
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

int binarySearch(const char arr[], int low, int high, char target) {
    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (tolower(arr[mid]) == tolower(target))
            return mid;
        else if (arr[mid] < target)
            low = mid + 1;
        else
            high = mid - 1;
    }

    return -1;
}

void searchChar(const char arr[], int length, char target) {
    int index = binarySearch(arr, 0, length - 1, target);

    if (index != -1) {
        cout << "Huruf ditemukan pada indeks: " << index << endl;

        int left = index - 1;
        while (left >= 0 && tolower(arr[left]) == tolower(target)) {
            cout << "Huruf ditemukan pada indeks: " << left << endl;
            left--;
        }

        int right = index + 1;
```



```

        while (right < length && tolower(arr[right]) == tolower(target)) {
            cout << "Huruf ditemukan pada indeks: " << right << endl;
            right++;
        }
    }
    else {
        cout << "Huruf tidak ditemukan." << endl;
    }
}

int main() {
    const int MAX_SIZE = 100;
    char sentence[MAX_SIZE];
    char target;

    cout << "Masukkan kalimat: ";
    cin.getline(sentence, MAX_SIZE);

    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> target;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    searchChar(sentence, strlen(sentence), target);

    return 0;
}

```

### *Output*

```

Masukkan kalimat: Riyon Aryono
Masukkan huruf yang ingin dicari: A
Huruf ditemukan pada indeks: 6

```

### *Penjelasan*

Program diatas adalah contoh implementasi dari algoritma binary search memungkinkan kita memasukkan sebuah kalimat dan huruf yang ingin dicari, kemudian mencari huruf tersebut dalam kalimat menggunakan algoritma Binary Search. Jika huruf ditemukan, program akan mencetak indeks pertama di mana huruf tersebut ditemukan, serta indeks tambahan jika ada huruf target yang berulang. Jika huruf tidak ditemukan, program akan mencetak pesan "Huruf tidak ditemukan."

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

*Source Code*

```
#include <iostream>
#include <cstring>
#include <cctype>

using namespace std;

int countVowels(const char* sentence) {
    int count = 0;
    int length = strlen(sentence);

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

    return count;
}

int main() {
    const int MAX_SIZE = 100;
    char sentence[MAX_SIZE];

    cout << "Masukkan kalimat: ";
    cin.getline(sentence, MAX_SIZE);

    int vowelCount = countVowels(sentence);

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

    return 0;
}
```

Output

```
Masukkan kalimat: Institut Teknologi Telkom Purwokerto
Jumlah huruf vokal dalam kalimat: 13
```

Penjelasan

Pada fungsi `countVowels`: Fungsi ini digunakan untuk menghitung jumlah huruf vokal dalam kalimat. Fungsi menerima parameter `sentence` yang merupakan pointer ke array karakter yang berisi kalimat. Program menginisialisasi variabel `count` sebagai penghitung jumlah huruf vokal. Kemudian, program melakukan iterasi melalui setiap karakter dalam kalimat menggunakan loop `for`. Di dalam loop ini, program menggunakan fungsi `tolower` untuk mengubah huruf menjadi huruf kecil dan memeriksa apakah huruf tersebut adalah huruf vokal (a, e, i, o, atau u). Jika huruf adalah huruf vokal, program meningkatkan nilai `count`. Pada akhirnya, fungsi mengembalikan nilai `count` yang merupakan jumlah huruf vokal dalam kalimat.

- 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 sequentialSearch(int arr[], int size, int target) {
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            count++;
        }
    }
    return count;
}

int main() {
    int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int size = sizeof(data) / sizeof(data[0]);
    int target = 4;

    int count = sequentialSearch(data, size, target);

    cout << "Jumlah kemunculan angka " << target << " adalah: " << count
    << endl;
```

```
    return 0;  
}
```

#### Output

```
Jumlah kemunculan angka 4 adalah: 4
```

#### Penjelasan

Pada program diatas adalah contoh implementasi dari algoritma sequential search pada fungsi `sequentialSearch`: Ini adalah fungsi yang mengimplementasikan algoritma Sequential Search. Fungsi ini menerima array `arr`, ukuran array `size`, dan angka target yang ingin dicari `target`. Program melakukan iterasi melalui setiap elemen array menggunakan loop `for`. Di setiap iterasi, program memeriksa apakah elemen array saat ini sama dengan angka target. Jika sama, program meningkatkan nilai `count` yang merupakan penghitung jumlah kemunculan angka target. Pada akhirnya, fungsi mengembalikan nilai `count`.

## **BAB IV**

### **KESIMPULAN**

Pemilihan algoritma tidak dapat sembarang dilakukan karena dapat mempengaruhi kompleksitas waktu dari sebuah program pada. Jika program data yang relatif kecil Algoritma Sequential Search cocok untuk dipilih namun jika jumlah data besar maka Algoritma Binary Search lah yang cocok untuk digunakan.