

**LAPORAN PRAKTIKUM**

**MODUL 8**

**ALGORITMA SEARCHING**



**Disusun oleh:**  
**Nabila Shasya Sabrina**  
**NIM: 2311102039**

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

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**PURWOKERTO**

**2024**

# **BAB I**

## **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**

### **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.

#### **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) 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

## BAB III

### GUIDED

#### 1. Guided 1

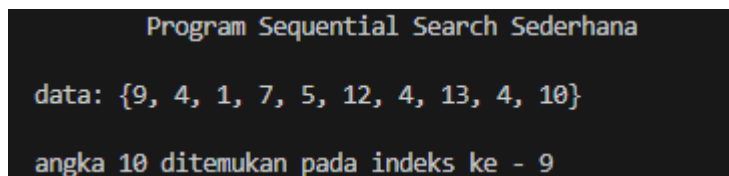
##### Source code:

```
#include <iostream>
using namespace std;
int main()
{
    int n = 10;
    int arr[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 (arr[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:



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

##### Deskripsi program:

Program ini adalah implementasi sederhana dari pencarian sekuensial (sequential search) dalam bahasa C++. Program dimulai dengan mendeklarasikan sebuah array berukuran 10 yang berisi angka-angka tertentu. Program kemudian mencari angka 10 dalam array tersebut. Jika angka tersebut ditemukan, program akan menampilkan indeks (posisi) di mana angka tersebut ditemukan. Jika tidak ditemukan, program akan memberikan pesan bahwa angka

tersebut tidak ada dalam data. Hasil dari pencarian ditampilkan di layar menggunakan perintah `cout`.

## 2. Guided 2

### Source code:

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

using namespace std;

int arr[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 (arr[j] < arr[min]) {
                min = j;
            }
        }
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}

void binarysearch() {
    // searching
    int awal, akhir, tengah, b_flag = 0;
    awal = 0;
    akhir = 6; // Corrected the value from 7 to 6
    while (b_flag == 0 && awal <= akhir) {
        tengah = (awal + akhir) / 2;
        if (arr[tengah] == cari) {
            b_flag = 1;
            break;
        } else if (arr[tengah] < cari) {
            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";
    }
}

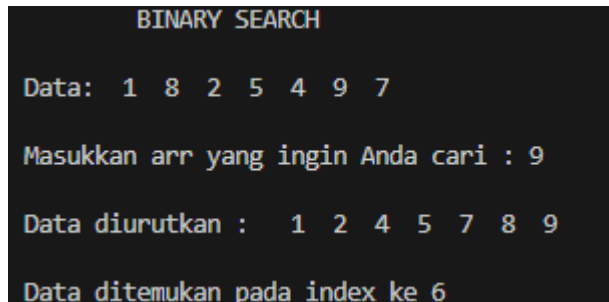
int main() {
    cout << "\tBINARY SEARCH" << endl;
    cout << "\nData:";
    // tampilkan arr awal
    for (int x = 0; x < 7; x++) {
```

```

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

```

### Screenshot Program:



```

BINARY SEARCH

Data: 1 8 2 5 4 9 7

Masukkan arr yang ingin Anda cari : 9

Data diurutkan : 1 2 4 5 7 8 9

Data ditemukan pada index ke 6

```

### Deskripsi program:

Program ini mengimplementasikan metode pencarian biner (binary search). Program dimulai dengan mendeklarasikan sebuah array berukuran 7 yang berisi angka-angka tertentu. Pertama, program mengurutkan array tersebut menggunakan algoritma selection sort. Setelah array diurutkan, pengguna diminta untuk memasukkan angka yang ingin dicari dalam array tersebut.

Setelah itu, program menjalankan algoritma pencarian biner untuk menemukan angka yang dimasukkan oleh pengguna. Pencarian biner dilakukan dengan membagi array menjadi dua bagian secara berulang sampai angka yang dicari ditemukan atau sampai tidak ada lagi bagian yang bisa dibagi. Jika angka ditemukan, program akan menampilkan indeks (posisi) di mana angka tersebut ditemukan. Jika tidak, program akan menampilkan pesan bahwa angka tersebut tidak ada dalam data. Seluruh proses dan hasil ditampilkan di layar menggunakan perintah `cout`.

## BAB IV

### 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 <algorithm>
#include <string>
#include <cstring>

using namespace std;

void binary_search(const char arr[], int size, char cari) {
    int awal = 0;
    int akhir = size - 1;
    bool found = false;

    while (awal <= akhir) {
        int tengah = (awal + akhir) / 2;
        if (arr[tengah] == cari) {
            found = true;
            cout << "\nHuruf '" << cari << "' ditemukan pada indeks ke-" << tengah << endl;
            break;
        } else if (arr[tengah] < cari) {
            awal = tengah + 1;
        } else {
            akhir = tengah - 1;
        }
    }

    if (!found) {
        cout << "\nHuruf '" << cari << "' tidak ditemukan dalam kalimat." << endl;
    }
}

int main() {
    string kalimat = "Semangat";
    char cari;

    cout << "Kata: Semangat" << endl;
    kalimat.erase(remove(kalimat.begin(), kalimat.end(), ' '),
        kalimat.end());

    int size = kalimat.length();
    char* array = new char[size];
    strncpy(array, kalimat.c_str(), size);

    cout << "Masukkan huruf yang ingin Anda cari: ";
    cin >> cari;

    // Panggil fungsi binary_search
    binary_search(array, size, cari);

    // Menghapus alokasi dinamis
    delete[] array;
```



```
    return 0;
}
```

### Screenshot Output:

```
Kata: Semangat
Masukkan huruf yang ingin Anda cari: t
Huruf 't' ditemukan pada indeks ke-7
```

### Deskripsi Program:

Program ini adalah sebuah program C++ yang menggunakan metode pencarian biner (binary search) untuk mencari sebuah huruf dalam sebuah kalimat yang telah ditentukan di dalam program. Program dimulai dengan mendefinisikan sebuah fungsi `binary_search` yang mengimplementasikan algoritma pencarian biner. Fungsi ini menerima tiga parameter: sebuah array karakter, ukuran array, dan huruf yang ingin dicari. Program kemudian menginisialisasi sebuah string kalimat dengan kata "Semangat". Setelah itu, program menampilkan kata yang akan dicari hurufnya, yaitu "Semangat". Selanjutnya, program menghapus spasi dari kalimat menggunakan fungsi `erase` dan `remove`. Setelah itu, program menghitung ukuran kalimat dan mengalokasikan array karakter dinamis untuk menyimpan kalimat yang telah dihilangkan spasi. Pengguna diminta untuk memasukkan huruf yang ingin dicari, kemudian program memanggil fungsi `binary_search` untuk mencari huruf tersebut dalam kalimat. Jika huruf ditemukan, program akan menampilkan indeks (posisi) huruf tersebut dalam kalimat. Jika tidak ditemukan, program akan memberikan pesan bahwa huruf tidak ada dalam kalimat. Setelah selesai, program akan menghapus alokasi memori yang digunakan untuk array karakter.

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

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

using namespace std;

bool isVowel(char c) {
    return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c
        == 'u' ||
        c == 'A' || c == 'E' || c == 'I' || c == 'O' || c
        == 'U');
}

int countVowels(const string& kalimat) {
    int count = 0;

    for (char c : kalimat) {
        if (isVowel(c)) {
            count++;
        }
    }
}
```

```

    }

    return count;
}

int main() {
    string kalimat;

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

    int jumlahVokal = countVowels(kalimat);

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

    return 0;
}

```

### Screenshot Output:

```

Masukkan sebuah kalimat: afzaal rabani
Jumlah huruf vokal dalam kalimat adalah: 6

```

### Deskripsi Program:

Program ini adalah sebuah program yang menghitung jumlah huruf vokal dalam sebuah kalimat yang dimasukkan oleh pengguna. Program dimulai dengan mendefinisikan fungsi `isVowel`, yang mengembalikan `true` jika karakter yang diberikan adalah huruf vokal (baik huruf kecil maupun huruf besar), dan `false` jika tidak. Kemudian, fungsi `countVowels` menghitung jumlah huruf vokal dalam sebuah string yang diberikan dengan melakukan iterasi melalui setiap karakter dalam string tersebut dan menggunakan fungsi `isVowel` untuk menentukan apakah karakter tersebut merupakan huruf vokal. Program utama kemudian meminta pengguna untuk memasukkan sebuah kalimat menggunakan `getline`, kemudian memanggil fungsi `countVowels` untuk menghitung jumlah huruf vokal dalam kalimat yang dimasukkan. Hasilnya kemudian ditampilkan di layar. Program ini menggunakan fungsi-fungsi dari pustaka standar C++ seperti `iostream`, `string`, dan `algorithm`, serta fungsi-fungsi manipulasi string dan struktur kontrol untuk menghitung jumlah huruf vokal dan berinteraksi dengan pengguna.

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

```

#include <iostream>

using namespace std;

int sequentialSearchCount(const 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() {
        const int size = 10;
        int data[size] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
        int target = 4;

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

        cout << "Banyaknya angka 4 dalam data adalah: " << count <<
            endl;

        return 0;
    }
```

### Screenshot Output:



Banyaknya angka 4 dalam data adalah: 4

### Deskripsi Program:

Program tersebut adalah sebuah program yang menghitung berapa kali angka 4 muncul dalam data yang telah ditentukan. Program dimulai dengan mendefinisikan sebuah fungsi `sequentialSearchCount`, yang melakukan pencarian sekuensial dalam data dan menghitung jumlah kemunculan angka 4. Fungsi ini menerima array data, ukuran array, dan angka target yang ingin dicari sebagai parameter. Program utama kemudian menginisialisasi array data dengan nilai-nilai yang telah ditentukan, dan memanggil fungsi `sequentialSearchCount` untuk menghitung jumlah kemunculan angka 4 dalam data tersebut. Hasilnya kemudian ditampilkan di layar. Program ini menggunakan struktur kontrol dan manipulasi array dalam bahasa C++ untuk melakukan pencarian sekuensial dan penghitungan kemunculan angka 4.

## **BAB V**

### **KESIMPULAN**

Pada praktikum struktur data dan algoritma dengan modul algoritma searching, kami mempelajari berbagai teknik pencarian data dalam sebuah kumpulan data. Algoritma searching penting karena sering digunakan dalam pengembangan perangkat lunak untuk menemukan elemen tertentu dalam suatu dataset. Dalam modul ini, kami mempelajari dua algoritma pencarian utama: sequential search (pencarian sekuensial) dan binary search (pencarian biner). Sequential search adalah metode pencarian sederhana yang dilakukan dengan mengurutkan data secara berurutan dan memeriksa setiap elemen satu per satu. Sementara itu, binary search adalah algoritma yang memanfaatkan fakta bahwa data sudah diurutkan untuk mencari elemen dengan membagi dataset menjadi setengah setiap iterasi. Dari praktikum ini, kami belajar bagaimana mengimplementasikan kedua algoritma tersebut dalam bahasa pemrograman C++ dan memahami kelebihan dan kelemahan masing-masing algoritma. Kesimpulannya, pemahaman tentang algoritma searching penting dalam pengembangan perangkat lunak karena membantu dalam memilih dan menerapkan metode pencarian yang efisien tergantung pada sifat data yang dimiliki.

## **DAFTAR PUSTAKA**

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