

LAPORAN PRAKTIKUM

MODUL VIII ALGORITMA SEARCHING



**Disusun oleh:
Ilhan Sahal Mansiz
2311102029**

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

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

BAB I

TUJUAN PRAKTIKUM

- 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.

BAB II

DASAR TEORI

Algoritma searching adalah teknik untuk menemukan elemen tertentu dalam kumpulan data. Dua algoritma searching yang umum digunakan adalah Binary Search dan Sequential Search. Kedua algoritma ini memiliki karakteristik dan kegunaan yang berbeda, tergantung pada struktur data dan kebutuhan efisiensi pencarian.

1. Binary Search

Binary search adalah metode pencarian yang sangat efisien untuk data yang terurut. Algoritma ini bekerja dengan prinsip divide-and-conquer, di mana kumpulan data dibagi menjadi dua bagian secara berulang-ulang hingga elemen yang dicari ditemukan atau kumpulan data habis.

Proses:

- 1.** Tentukan elemen tengah dari kumpulan data.
- 2.** Bandingkan elemen tengah dengan elemen yang dicari:
 - Jika elemen tengah adalah elemen yang dicari, pencarian selesai.
 - Jika elemen yang dicari lebih kecil dari elemen tengah, ulangi pencarian pada setengah bagian kiri.
 - Jika elemen yang dicari lebih besar dari elemen tengah, ulangi pencarian pada setengah bagian kanan.

3. Ulangi langkah-langkah di atas hingga elemen ditemukan atau ruang pencarian habis.

Karakteristik:

- Kompleksitas Waktu: $O(\log n)$, di mana n adalah jumlah elemen dalam kumpulan data.
- Keuntungan: Sangat efisien untuk data yang besar dan terurut.
- Kekurangan: Membutuhkan data yang sudah terurut sebelum pencarian dilakukan.

Binary search sangat berguna dalam aplikasi yang membutuhkan pencarian cepat dalam kumpulan data besar, seperti dalam database dan sistem penyimpanan yang diindeks.

2. Sequential Search

Sequential search, sering disebut juga linear search, adalah metode pencarian yang paling sederhana. Algoritma ini memeriksa setiap elemen dalam kumpulan data satu per satu, mulai dari elemen pertama hingga elemen terakhir, hingga elemen yang dicari ditemukan atau kumpulan data habis.

Proses:

1. Mulai dari elemen pertama.
2. Bandingkan setiap elemen dengan elemen yang dicari:
 - Jika elemen yang dicari ditemukan, pencarian selesai.
 - Jika tidak, lanjutkan ke elemen berikutnya.
3. Ulangi hingga elemen ditemukan atau akhir kumpulan data tercapai.

Karakteristik:

- Kompleksitas Waktu: $O(n)$, di mana n adalah jumlah elemen dalam kumpulan data.

- Keuntungan: Tidak memerlukan data yang terurut dan mudah diimplementasikan.
- Kekurangan: Tidak efisien untuk kumpulan data yang besar.

Sequential search berguna dalam situasi di mana data tidak terurut atau kumpulan data relatif kecil sehingga efisiensi tidak menjadi masalah utama. Algoritma ini juga fleksibel dan dapat diterapkan pada berbagai jenis struktur data, termasuk array, linked list, dan struktur data dinamis lainnya.

BAB III

GUIDED

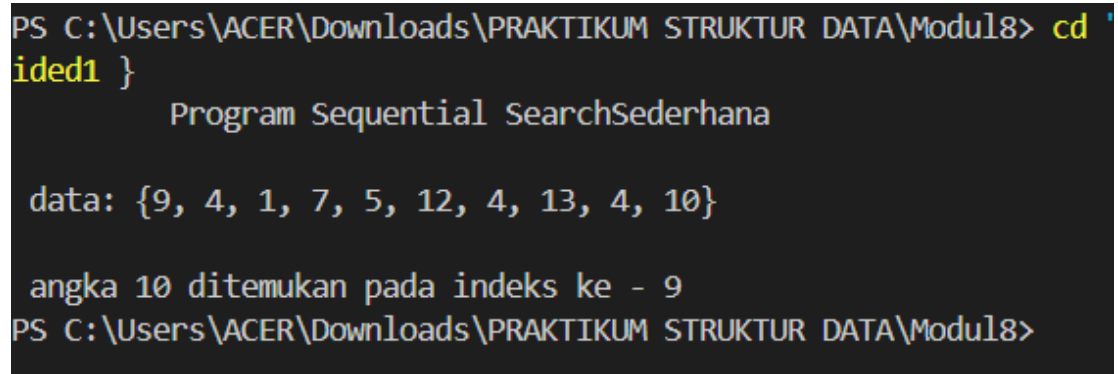
1. Guided 1

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 << "\t Program Sequential SearchSederhana\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



```
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul8> cd '
ided1 }
        Program Sequential SearchSederhana

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

angka 10 ditemukan pada indeks ke - 9
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul8>
```

Deskripsi program

Program di atas adalah implementasi sederhana dari algoritma Sequential Search untuk mencari elemen tertentu dalam array. Program dimulai dengan mendefinisikan sebuah array 'data' berisi 10 elemen integer dan sebuah variabel 'cari' dengan nilai 10, yang akan dicari dalam array tersebut. Algoritma Sequential Search berjalan melalui setiap elemen dalam array dari indeks pertama hingga terakhir. Jika elemen yang dicari ditemukan, variabel 'ketemu' diatur menjadi 'true', dan pencarian dihentikan. Setelah pencarian selesai, program mencetak hasil pencarian: jika elemen ditemukan, program menampilkan indeks di mana elemen ditemukan; jika tidak, program menampilkan pesan bahwa elemen tidak ditemukan dalam array.

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 << "\narr ditemukan pada index ke " << tengah <<
endl;
} else {
    cout << "\narr tidak ditemukan\n";
}
}

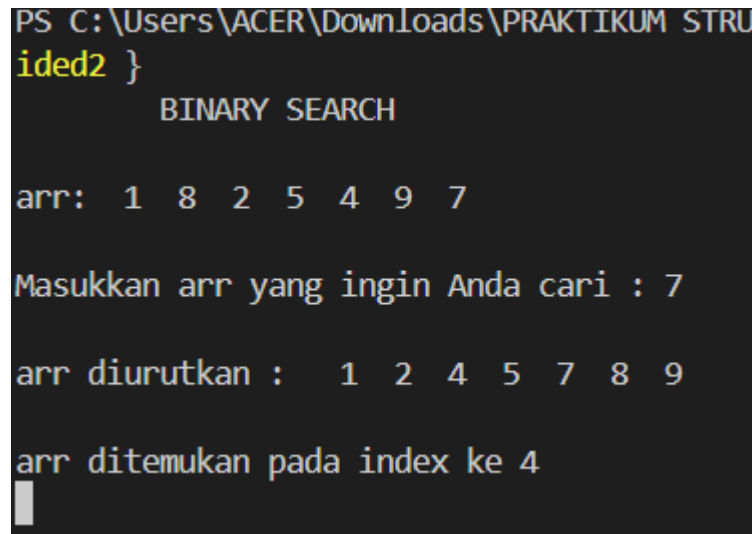
int main() {
    cout << "\tBINARY SEARCH" << endl;
    cout << "\narr:";
    // tampilkan arr awal
    for (int x = 0; x < 7; x++) {
        cout << setw(3) << arr[x];
    }
    cout << endl;
    cout << "\nMasukkan arr yang ingin Anda cari : ";
    cin >> cari;
    cout << "\narr diurutkan : ";
    // 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;  
}
```

Screenshoot program



```
PS C:\Users\ACER\Downloads\PRAKTIKUM STRU  
ided2 }  
      BINARY SEARCH  
  
arr:  1  8  2  5  4  9  7  
  
Masukkan arr yang ingin Anda cari : 7  
  
arr diurutkan :  1  2  4  5  7  8  9  
  
arr ditemukan pada index ke 4  
|
```

Deskripsi program

Program di atas mengimplementasikan algoritma Binary Search untuk mencari elemen tertentu dalam array yang telah diurutkan menggunakan algoritma Selection Sort. Program dimulai dengan mendefinisikan array 'arr' berisi tujuh elemen integer dan variabel 'cari' untuk menyimpan nilai yang akan dicari. Fungsi 'selection_sort' mengurutkan array 'arr' secara ascending. Setelah array diurutkan, fungsi 'binarysearch' melakukan pencarian elemen 'cari' dalam array yang telah diurutkan. Jika elemen ditemukan, program mencetak indeks tempat elemen ditemukan; jika tidak, program mencetak pesan bahwa elemen tidak ditemukan. Program juga mencetak array sebelum dan sesudah pengurutan untuk memberikan visualisasi perubahan data.

LATIHAN KELAS - 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 <cctype>

using namespace std;

bool binary_search(const string &kalimat, char cari)
{
    int awal = 0, akhir = kalimat.length() - 1;
    while (awal <= akhir)
    {
        int tengah = (awal + akhir) / 2;
        if (kalimat[tengah] == cari)
        {
            return true;
        }
        else if (kalimat[tengah] < cari)
        {
            awal = tengah + 1;
        }
        else
        {
            akhir = tengah - 1;
        }
    }
    return false;
}
```

```
int main()
{
    string kalimat;
    char cari;
    cout << "Masukan Kalimat : ";
    getline(cin, kalimat);
    cout << "Masukan huruf yang ingin anda cari : ";
    cin >> cari;

    cari = tolower(cari);
    transform(kalimat.begin(), kalimat.end(), kalimat.begin(),
::tolower);
    sort(kalimat.begin(), kalimat.end());

    bool ditemukan = binary_search(kalimat, cari);
    if (ditemukan)
    {
        cout << "Huruf " << cari << " ditemukan dalam kalimat."
<< endl;
    }
    else
    {
        cout << "Huruf " << cari << " tidak ditemukan dalam
kalimat." << endl;
    }

    return 0;
} //2311102029 ILHAN SAHAL MANSIZ
```

Screenshot Program

```
Masukan Kalimat : Ilhan suka jpop
Masukan huruf yang ingin anda cari : p
Huruf p ditemukan dalam kalimat.
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul8>
```

```
Masukan Kalimat : ilhan suka jpop
Masukan huruf yang ingin anda cari : z
Huruf z tidak ditemukan dalam kalimat.
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul8>
```

Deskripsi Program

Program di atas adalah implementasi algoritma Binary Search untuk mencari sebuah karakter dalam sebuah string setelah string tersebut diubah menjadi huruf kecil dan diurutkan. Proses dimulai dengan meminta pengguna memasukkan sebuah kalimat dan karakter yang ingin dicari dalam kalimat tersebut. Pengguna diminta untuk memasukkan kalimat menggunakan fungsi getline untuk memastikan seluruh kalimat, termasuk spasi, terbaca dengan benar. Selanjutnya, pengguna diminta memasukkan satu karakter yang ingin dicari dalam kalimat tersebut.

Setelah mendapatkan input dari pengguna, karakter yang dicari dan seluruh karakter dalam kalimat diubah menjadi huruf kecil menggunakan fungsi tolower dan transform. Hal ini dilakukan untuk memastikan pencarian tidak peka terhadap huruf besar atau kecil. Kalimat kemudian diurutkan secara alfabetis menggunakan fungsi sort dari STL (Standard Template Library). Pengurutan ini penting karena algoritma binary search hanya dapat bekerja dengan benar pada data yang sudah terurut.

Dengan kalimat yang sudah diurutkan, algoritma binary search kemudian digunakan untuk mencari karakter yang diinginkan. Algoritma ini bekerja dengan membagi kalimat menjadi dua bagian berulang kali dan membandingkan karakter tengah dengan karakter yang dicari. Jika karakter ditemukan, program mencetak pesan bahwa karakter tersebut ditemukan dalam kalimat. Jika tidak, program mencetak pesan bahwa karakter tidak ditemukan. Program ini menunjukkan kombinasi penggunaan fungsi transformasi dan algoritma searching yang efisien dalam C++.

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

Source Code

```
#include <iostream>
#include <algorithm>
using namespace std;

int hurufVokal(const string &kalimat)
{
    int count = 0;
    string vokal = "aiueo";
    string lowercaseKalimat = kalimat;
    transform(lowercaseKalimat.begin(),
lowercaseKalimat.end(), lowercaseKalimat.begin(),
::tolower);
    for (char c : lowercaseKalimat)
    {
        if (vokal.find(c) != string::npos)
        {
            count++;
        }
    }
    return count;
}

int main()
{
    string kalimat;
    cout << "Masukan Kalimat : ";
    getline(cin, kalimat);
    int jumlahVokal = hurufVokal(kalimat);
    cout << "Jumlah huruf vokal dalam kalimat: " <<
jumlahVokal << endl;

    return 0;
} // ILHAN SAHAL MANSIZ // 2311102029
```

Screenshot Program

```
Masukan Kalimat : lo penyuka sawanohiroyuki kita teman  
Jumlah huruf vokal dalam kalimat: 15  
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul8>
```

Deskripsi Program

Program di atas bertujuan untuk menghitung jumlah huruf vokal dalam sebuah kalimat yang diberikan oleh pengguna. Proses dimulai dengan mendefinisikan fungsi 'hurufVokal' yang menerima sebuah string sebagai parameter dan mengembalikan jumlah huruf vokal dalam string tersebut. Fungsi ini pertama-tama mendefinisikan string 'vokal' yang berisi huruf vokal dalam bahasa Indonesia (a, i, u, e, o). Kemudian, string kalimat diubah menjadi huruf kecil seluruhnya untuk memastikan bahwa pencarian huruf vokal tidak peka terhadap huruf besar atau kecil.

Transformasi huruf dalam kalimat menjadi huruf kecil dilakukan menggunakan fungsi 'transform' dari STL (Standard Template Library). Selanjutnya, program melakukan iterasi melalui setiap karakter dalam kalimat yang telah diubah menjadi huruf kecil. Dalam setiap iterasi, program memeriksa apakah karakter tersebut termasuk dalam string 'vokal'. Pemeriksaan ini dilakukan dengan menggunakan fungsi 'find', yang mencari keberadaan karakter dalam string 'vokal'. Jika karakter tersebut ditemukan (tidak sama dengan 'string::npos'), maka variabel penghitung 'count' ditambahkan satu.

Fungsi 'hurufVokal' mengembalikan nilai dari variabel 'count' yang merupakan jumlah huruf vokal dalam kalimat. Di dalam fungsi 'main', program meminta pengguna untuk memasukkan sebuah kalimat menggunakan fungsi 'getline' untuk membaca seluruh kalimat termasuk spasi. Kemudian, fungsi 'hurufVokal' dipanggil dengan kalimat tersebut sebagai argumennya. Hasil dari fungsi ini, yaitu jumlah huruf vokal dalam kalimat, disimpan dalam variabel 'jumlahVokal' dan ditampilkan ke layar. Program ini menunjukkan penggunaan transformasi string dan pencarian karakter secara efisien menggunakan STL dalam C++.

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

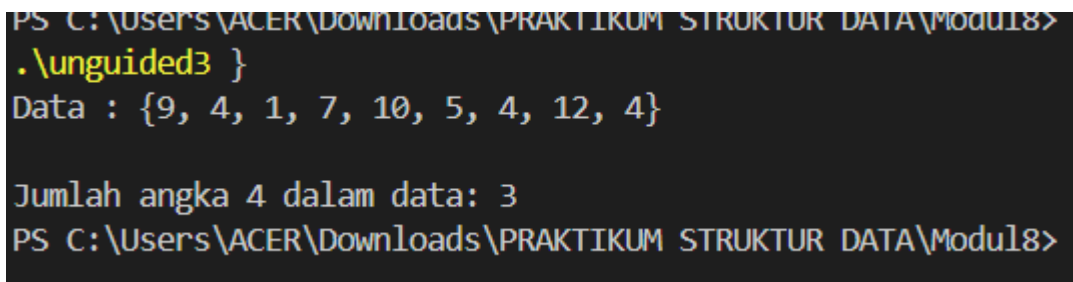
Source Code

```
#include <iostream>
using namespace std;

int main()
{
    const int n = 10;
    int *data = new int[n]{9, 4, 1, 7, 10, 5, 4, 12, 4};
    int cari = 4;
    int count = 0;
    // algorithm sequential search
    for (int i = 0; i <n; i++)
    {
        if (data[i] == cari)
        {
            count++;
        }
    }
    cout << "Data : {9, 4, 1, 7, 10, 5, 4, 12, 4}" << endl;
    cout << "\nJumlah angka " << cari << " dalam data: " <<
count << endl;

    return 0;
} // ILHAN SAHAL MANSIZ // 2311102029
```

Screenshot Program



```
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul8>
.\unguided3 }
Data : {9, 4, 1, 7, 10, 5, 4, 12, 4}

Jumlah angka 4 dalam data: 3
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul8>
```

Deskripsi Program

Program di atas bertujuan untuk menghitung jumlah kemunculan suatu nilai tertentu dalam sebuah array menggunakan algoritma Sequential Search. Dalam program ini, sebuah array data berisi sepuluh elemen integer diinisialisasi secara dinamis menggunakan operator new. Nilai yang akan dicari dalam array adalah 4, dan

variabel count digunakan untuk menghitung jumlah kemunculan nilai tersebut. Algoritma Sequential Search diterapkan dengan melakukan iterasi melalui setiap elemen array dan memeriksa apakah elemen tersebut sama dengan nilai yang dicari. Jika ditemukan kecocokan, variabel count akan ditambahkan satu.

Setelah pencarian selesai, program menampilkan isi array dan jumlah kemunculan nilai yang dicari. Program ini menunjukkan penggunaan dasar dari algoritma Sequential Search untuk menghitung frekuensi kemunculan suatu nilai dalam array. Algoritma ini bekerja dengan efisiensi $O(n)$, di mana n adalah jumlah elemen dalam array. Penggunaan array dinamis memungkinkan fleksibilitas dalam pengelolaan memori, meskipun pada kasus ini, array statis juga bisa digunakan karena ukuran array telah diketahui sebelumnya. Program ini menekankan konsep iterasi dan perbandingan elemen dalam konteks pencarian data.

BAB IV

KESIMPULAN

Modul 8 tentang algoritma searching memberikan pemahaman mendalam tentang dua algoritma searching yang fundamental: Sequential Search dan Binary Search. Dalam Sequential Search, setiap elemen dalam struktur data diperiksa satu per satu hingga elemen yang dicari ditemukan atau sampai akhir dari struktur data tercapai. Algoritma ini sederhana dan mudah diimplementasikan, namun memiliki kompleksitas waktu $O(n)$, di mana n adalah jumlah elemen dalam struktur data. Binary Search, di sisi lain, adalah algoritma yang lebih efisien karena memanfaatkan sifat data yang terurut. Dengan membagi struktur data menjadi dua bagian secara berulang dan membandingkan elemen tengah dengan elemen yang dicari, Binary Search dapat mencapai kompleksitas waktu $O(\log n)$, yang jauh lebih efisien daripada Sequential Search untuk data yang besar.

Selain mempelajari kedua algoritma tersebut, modul ini juga membahas berbagai aspek terkait implementasi, kompleksitas, dan aplikasi dari masing-masing algoritma. Sequential Search, meskipun sederhana, dapat berguna dalam situasi di mana data tidak terurut atau ketika kumpulan data relatif kecil. Di sisi lain, Binary Search sangat efektif untuk data yang besar dan terurut, seperti dalam database yang membutuhkan akses cepat dan efisien terhadap data. Pemahaman yang baik tentang kedua algoritma ini memungkinkan pengembang untuk memilih metode pencarian yang paling sesuai dengan kebutuhan spesifik aplikasi mereka.

Kesimpulannya, modul ini memberikan landasan yang kuat dalam memahami dan menerapkan algoritma searching dalam pemrograman. Dengan memahami prinsip-prinsip dasar Sequential Search dan Binary Search, serta memahami kelebihan dan kekurangan masing-masing, pengembang dapat membuat keputusan yang bijak dalam merancang dan mengoptimalkan pencarian data dalam program mereka. Hal ini menjadi penting karena efisiensi pencarian data dapat memiliki dampak signifikan terhadap kinerja dan responsivitas aplikasi secara keseluruhan.

BAB V

REFERENSI

- <https://mikirinkode.com/program-penerapan-algoritma-searching-c/>
- <https://kelasprogrammer.com/contoh-program-searching-c-array/>
- <https://www.freecodecamp.org/news/binary-search-in-c-algorithm-example/>