

LAPORAN PRAKTIKUM

MODUL IIX ALGORITMA SEARCHING



Disusun oleh:
Andika Indra Prastawa
NIM: 2311102033

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

- 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

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.

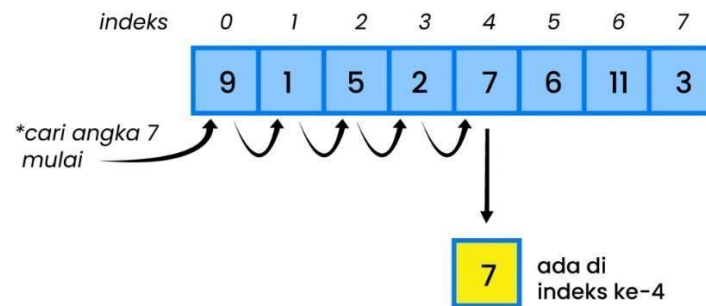
Algoritma pencarian berurutan dapat dituliskan sebagai berikut :

- 1) $i \leftarrow 0$
- 2) $ketemu \leftarrow false$
- 3) Selama (tidak ketemu) dan ($i \leq N$) kerjakan baris 4

- 4) Jika ($\text{Data}[i] = x$) maka $\text{ketemu} \leftarrow \text{true}$, jika tidak $i \leftarrow i + 1$
- 5) Jika (ketemu) maka i adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

Contoh dari Sequential Search, yaitu:

$\text{Int } A[8] = \{9, 1, 5, 2, 7, 6, 11, 3\}$



Gambar 1. Ilustrasi Sequential Search

Misalkan, dari data di atas angka yang akan dicari adalah angka 7 dalam array A, maka proses yang akan terjadi yaitu:

- Pencarian dimulai pada index ke-0 yaitu angka 9, kemudian dicocokkan dengan angka yang akan dicari, jika tidak sama maka pencarian akan dilanjutkan ke index selanjutnya.
- Pada index ke-1, yaitu angka 1, juga bukan angka yang dicari, maka pencarian akan dilanjutkan pada index selanjutnya.
- Pada index ke-2 dan index ke-3 yaitu angka 5 dan 2, juga bukan angka yang dicari, sehingga pencarian dilanjutkan pada index selanjutnya.
- Pada index ke-4 yaitu angka 7 dan ternyata angka 7 merupakan angka yang dicari, sehingga pencarian akan dihentikan dan proses selesai.

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),

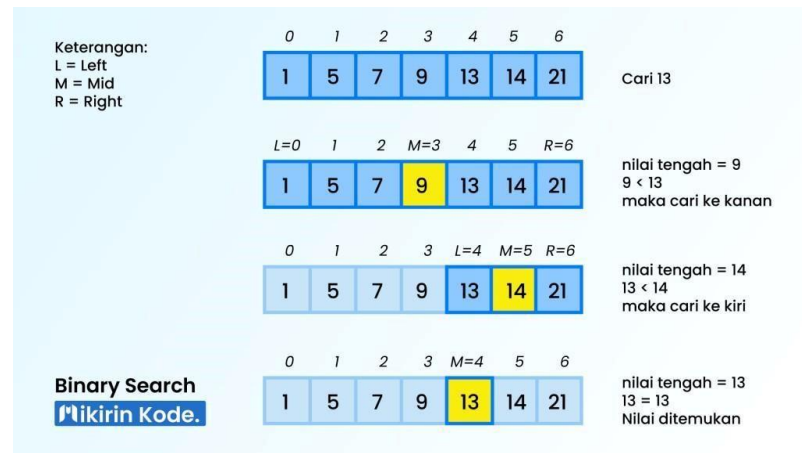
untuk setiap tahap pencarian. Dalam penerapannya algoritma ini sering digabungkan dengan algoritma sorting karena data yang akan digunakan harus sudah terurut terlebih dahulu. Konsep Binary Search:

- 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.

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

GUIDED

1. Guided 1

Source code

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

int main()
{
    int n = 10;
    int data[] = {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;
        }
    }

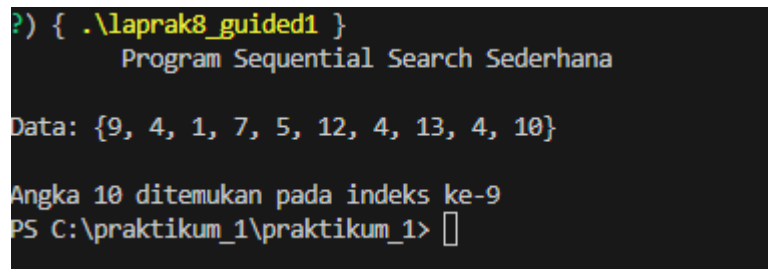
    cout << "\tProgram Sequential Search Sederhana\n" << endl;
    cout << "Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}" << endl;

    if (ketemu)
    {
        cout << "\nAngka " << cari << " ditemukan pada indeks ke-" << i << endl;
    }
}
```

```
        else
        {
            cout << cari << " tidak dapat ditemukan pada data." <<
endl;
        }

        return 0;
    }
}
```

Screenshoot program



```
? { .\laprak8_guided1 }
    Program Sequential Search Sederhana

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

Angka 10 ditemukan pada indeks ke-9
PS C:\praktikum_1\praktikum_1>
```

Deskripsi program

Program ini mendeklarasikan sebuah array data yang berisi 10 elemen integer dan mencari angka tertentu yang disimpan dalam variabel cari. Dengan menggunakan perulangan for, program memeriksa setiap elemen dalam array untuk melihat apakah ada yang cocok dengan nilai yang dicari. Jika ditemukan, variabel boolean ketemu diatur menjadi true dan perulangan berhenti. Program kemudian menampilkan hasil pencarian, yaitu indeks di mana angka ditemukan atau pesan bahwa angka tidak ditemukan. Dalam contoh ini, program mencari angka 10 dalam array yang diberikan.

2. 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)
    {
```

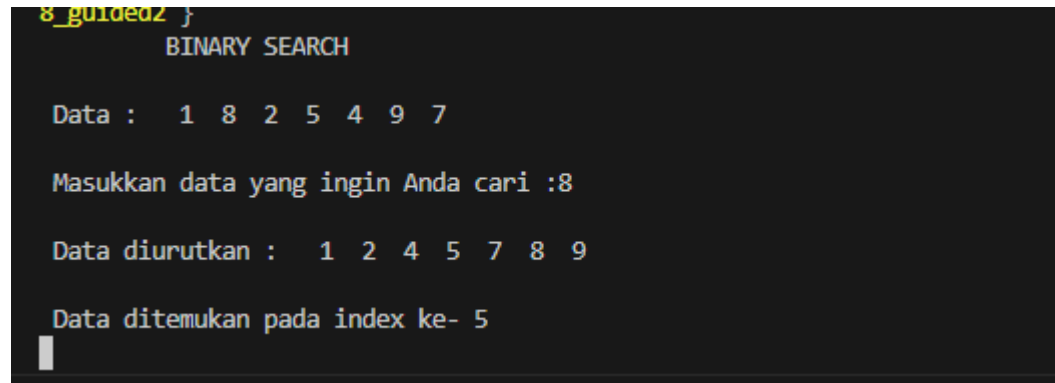
```

        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;  
}
```

Screenshoot program



```
8_guided2 }  
BINARY SEARCH  
  
Data :  1 8 2 5 4 9 7  
  
Masukkan data yang ingin Anda cari :8  
  
Data diurutkan :  1 2 4 5 7 8 9  
  
Data ditemukan pada index ke- 5
```

Deskripsi program

Program ini pertama kali mendeklarasikan array data1 berisi 7 elemen integer dan variabel cari yang akan digunakan untuk menyimpan angka yang dicari oleh pengguna. Setelah menampilkan data awal, program meminta pengguna untuk memasukkan angka yang ingin dicari. Program kemudian mengurutkan array menggunakan fungsi selection_sort, menampilkan data yang sudah diurutkan, dan melakukan pencarian biner dengan fungsi binarysearch. Jika angka yang dicari ditemukan, program menampilkan indeks di mana angka tersebut berada; jika tidak ditemukan, program menampilkan pesan bahwa data tidak ditemukan. Fungsi selection_sort mengurutkan array dengan cara mencari elemen terkecil dari sisa array dan menukarnya dengan elemen saat ini. Fungsi binarysearch melakukan pencarian biner pada array yang sudah diurutkan untuk menemukan posisi elemen yang dicari.

LATIHAN KELAS - UNGUIDED

1. Unguided

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;

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;
        }
        else if (str[mid] < target)
        {
            left = mid + 1;
        }
        else
        {
            right = mid - 1;
        }
    }
    return -1;
}
```

```
int main()
{
    string kalimat;
    char huruf;

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

    sort(kalimat.begin(), kalimat.end());

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

    int posisi = binarySearch(kalimat, huruf);

    if (posisi != -1)
    {
        cout << "Huruf \"" << huruf << "\" ditemukan pada posisi " << posisi + 1 << " dari kalimat." << endl;
    }
    else
    {
        cout << "Huruf \"" << huruf << "\" tidak ditemukan dalam kalimat." << endl;
    }

    return 0;
}
```

Screenshoot program

```
PS C:\praktikum_1\praktikum_1> cd "c:\praktikum_1\praktikum_1"
Masukkan kalimat: andika
Masukkan huruf yang ingin dicari: a
Huruf "a" ditemukan pada posisi 1 dari kalimat.
PS C:\praktikum_1\praktikum_1> 
```

Deskripsi program

Program mengambil input kalimat dari pengguna dan menyimpannya dalam variabel kalimat. Kemudian, kalimat tersebut diurutkan menggunakan fungsi sort dari C++ Standard Library. Setelah itu, program meminta pengguna untuk memasukkan huruf yang ingin dicari dalam kalimat. Fungsi binarySearch kemudian digunakan untuk mencari posisi huruf tersebut dalam kalimat yang sudah diurutkan. Jika huruf ditemukan, program menampilkan posisi huruf dalam kalimat; jika tidak ditemukan, program menampilkan pesan bahwa huruf tidak ditemukan dalam kalimat.

2. Unguided

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

Source code

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

int hitungVokal(const string &kalimat)
{
    int jumlahVokal = 0;
```

```

        for (char huruf : kalimat)
        {

            char lowerHuruf = tolower(huruf);

            if (lowerHuruf == 'a' || lowerHuruf == 'e' || lowerHuruf
== 'i' || lowerHuruf == 'o' || lowerHuruf == 'u')
            {
                jumlahVokal++;
            }
        }
        return jumlahVokal;
    }
int main()
{
    string kalimat;

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

    int jumlahVokal = hitungVokal(kalimat);

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

    return 0;
}

```

Screenshoot program

```

PS C:\praktikum 1\praktikum 1> cd "c:\praktikum_1\praktikum
Masukkan sebuah kalimat: andika indra prastawa
Jumlah huruf vokal dalam kalimat adalah: 8
PS C:\praktikum_1\praktikum_1> 

```

Deskripsi program

program meminta pengguna untuk memasukkan sebuah kalimat dan menyimpannya dalam variabel kalimat. Fungsi hitungVokal kemudian digunakan untuk menghitung jumlah huruf vokal dalam kalimat tersebut. Fungsi ini bekerja dengan cara mengiterasi setiap karakter dalam kalimat, mengonversinya menjadi huruf kecil menggunakan fungsi tolower, dan memeriksa apakah karakter tersebut adalah salah satu dari huruf vokal ('a', 'e', 'i', 'o', 'u'). Jika iya, maka variabel penghitung jumlahVokal ditingkatkan. Setelah proses perhitungan selesai, program menampilkan jumlah huruf vokal yang ditemukan dalam kalimat.

3. Unguided

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 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;

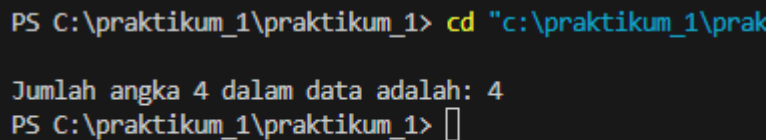
    int    jumlahAngka4    =    hitungAngka(data,    ukuran,
angkaYangDicari);

    cout << "Jumlah angka 4 dalam data adalah: " << jumlahAngka4
<< endl;

    return 0;
}

```

Screenshoot program



```

PS C:\praktikum_1\praktikum_1> cd "c:\praktikum_1\prak
Jumlah angka 4 dalam data adalah: 4
PS C:\praktikum_1\praktikum_1> 

```

Deskripsi program

Program mendeklarasikan sebuah array data yang berisi 10 elemen integer dan variabel `angkaYangDicari` yang menyimpan angka yang akan dicari, yaitu 4. Fungsi `hitungAngka` digunakan untuk menghitung jumlah kemunculan angka tersebut dalam array. Fungsi ini bekerja dengan mengiterasi setiap elemen dalam array dan meningkatkan variabel penghitung jumlah setiap kali elemen yang sesuai ditemukan. Setelah perhitungan selesai, program menampilkan jumlah kemunculan angka yang dicari dalam array. Dalam contoh ini, program menghitung berapa kali angka 4 muncul dalam array data dan menampilkan hasilnya.

BAB IV

KESIMPULAN

Modul 8 tentang Algoritma Pencarian ini mengulas dua algoritma pencarian utama, yaitu Sequential Search dan Binary Search. Sequential Search adalah metode pencarian sederhana yang memeriksa setiap elemen satu per satu, cocok untuk data yang belum terurut, namun kurang efisien untuk dataset besar dengan waktu pencarian $O(n)$. Sebaliknya, Binary Search, yang memerlukan data terurut, bekerja dengan membagi data menjadi dua bagian, menawarkan efisiensi lebih tinggi dengan waktu pencarian $O(\log n)$, sehingga lebih efektif untuk dataset besar. Modul ini menyediakan contoh implementasi kedua algoritma dalam bahasa C++, menunjukkan cara mencari elemen dalam array serta penggunaan Selection Sort sebelum pencarian biner. Selain itu, terdapat latihan mandiri untuk memperdalam pemahaman, seperti mencari huruf dalam kalimat dan menghitung jumlah kemunculan angka dalam array. Buku "Data Structures and Algorithms Made Easy" oleh Narasimha Karumanchi digunakan sebagai referensi utama modul ini, memberikan pemahaman mendalam tentang teori dan aplikasi praktis algoritma pencarian dalam pemrograman.

BAB VI

DAFTAR PUSTAKA

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