

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VIII
ALGORITMA SEARCHING**



Disusun Oleh :

NAMA : Ahmad Titana Nanda Pramudya

NIM : 2311102042

Dosen

Wahyu Andi Saputra, S.pd., M,Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. DASAR TEORI

Algoritma Search adalah serangkaian langkah atau instruksi yang digunakan untuk mencari elemen atau informasi tertentu di dalam suatu dataset. Tujuannya adalah untuk menemukan posisi atau keberadaan elemen yang dicari. Dalam pemrograman, algoritma search menjadi salah satu teknik penting dalam menyelesaikan berbagai masalah.

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 teratur. 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) $ketemu \leftarrow \text{false}$
- 3) Selama (tidak $ketemu$) dan $(i \leq N)$ kerjakan baris 4
- 4) Jika $(Data[i] = x)$ maka $ketemu \leftarrow \text{true}$, jika tidak $i \leftarrow i + 1$
- 5) Jika ($ketemu$) maka i adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

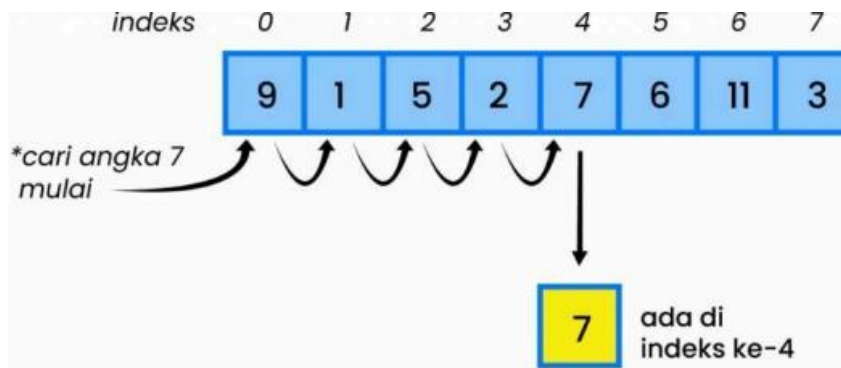
Di bawah ini merupakan fungsi untuk mencari data menggunakan pencarian sekuensial.

```

int SequentialSearch (int x)
{
    int i = 0;
    bool ketemu = false;
    while ((!ketemu) && (i < Max)){
        if (Data[i] == x)
            ketemu = true;
        else
            i++;
    }
    if (ketemu)
        return i;
    else
        return -1;
}

```

Fungsi diatas akan mengembalikan indeks dari data yang dicari. Apabila data tidak ditemukan maka fungsi diatas akan mengembalikan nilai -1. Contoh dari Sequential Search, yaitu: 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.

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:

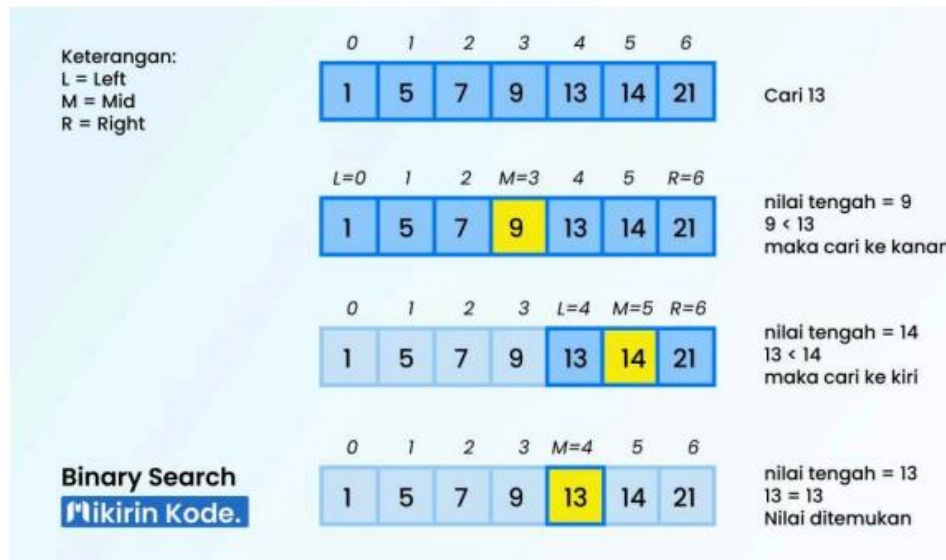
- Data diambil dari posisi 1 sampai posisi akhir N.
- Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
- Selanjutnya data yang dicari akan dibandingkan dengan data yang berada di posisi tengah, apakah lebih besar atau lebih kecil.
- Apabila data yang dicari lebih besar dari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

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, didapatkanlah 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.

Jenis-Jenis Algoritma Search

1. Linear Search

Linear Search, atau pencarian linear, adalah algoritma pencarian yang paling sederhana. Ini bekerja dengan cara memeriksa setiap elemen dalam kumpulan data secara berurutan hingga elemen yang dicari ditemukan atau sampai seluruh kumpulan data telah diperiksa.

Contoh Program:

```
def linear_search(arr, target):  
    for i in range(len(arr)):  
        if arr[i] == target:  
            return i  
    return -1
```

2. Depth First Search (DFS)

Depth First Search (DFS) adalah algoritma pencarian yang digunakan dalam struktur data seperti graf. Ia mengikuti jalur sejauh mungkin sebelum kembali dan mengeksplorasi cabang lain.

Contoh Program DFS:

```
class Graph:  
    def __init__(self):  
        self.graph = {}  
  
    def add_edge(self, node, neighbor):  
        if node not in self.graph:  
            self.graph[node] = []  
        self.graph[node].append(neighbor)  
  
    def dfs(graph, start, visited=[]):  
        visited.append(start)  
        for neighbor in graph[start]:  
            if neighbor not in visited:  
                dfs(graph, neighbor, visited)
```

3. Binary Search

Pencarian Binary adalah algoritma pencarian yang efisien digunakan untuk mencari elemen dalam kumpulan data yang sudah terurut. Algoritma ini membagi data menjadi dua bagian dan membandingkan elemen tengah dengan elemen yang dicari.

Contoh Program Pencarian Binary:

```
def binary_search(arr, target):  
    left, right = 0, len(arr) - 1  
    while left <= right:  
        mid = (left + right) // 2  
        if arr[mid] == target:  
            return mid  
        elif arr[mid] < target:  
            left = mid + 1  
        else:  
            right = mid - 1  
    return -1
```

4. Jump Search

Pencarian Jump adalah algoritma pencarian yang efisien digunakan untuk mencari elemen dalam kumpulan data yang sudah terurut. Ia melompati beberapa langkah sekaligus dalam pencarian, memungkinkan efisiensi yang lebih baik daripada pencarian linear.

Contoh Program pencarian Jump:

```
import math  
def jump_search(arr, target):  
    n = len(arr)  
    step = int(math.sqrt(n))  
    prev = 0  
    while arr[min(step, n)-1] < target:  
        prev = step
```

```

    step += int(math.sqrt(n))

    if prev >= n:
        return -1

    while arr[prev] < target:
        prev += 1

    if prev == min(step, n):
        return -1

    if arr[prev] == target:
        return prev

    return -1

```

B. Guided

Guided 1

Sourcode :



Screenshots Output :

```

gexxe=C:\msys64\usr\bin\gdb.exe --interpreter=python3
Program Sequential Search Sederhana

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

angka 10 ditemukan pada indeks ke-9
PS D:\contoh>

```

Nama : Ahmad Titana Nanda Pramudya
 Nim : 2311102042
 Kelas: IF-11-B

Ln 4, Col 1 | 68 characters | 100% | Window UTF

Deskripsi:

Program ini menggunakan algoritma Sequential Search untuk mencari angka **10** dalam array **data**. Jika angka ditemukan, program akan menampilkan indeks di mana angka tersebut ditemukan. Jika tidak ditemukan, program akan menyatakan bahwa angka tersebut tidak ada dalam data.

Guided 2

Sour code:

```
#include <iostream>
#include <iomanip>
using namespace std;
// Deklarasi array dan variabel untuk pencarian
int arrayData[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;
void selection_sort(int arr[], int n)
{
    int temp, min;
    for (int i = 0; i < n - 1; i++)
    {
        min = i;
        for (int j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[min])
            {
                min = j;
            }
        }
        // Tukar elemen
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}

void binary_search(int arr[], int n, int target)
{
    int awal = 0, akhir = n - 1, tengah, b_flag = 0;
    while (b_flag == 0 && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (arr[tengah] == target)
        {
            b_flag = 1;

            break;
        }
        else if (arr[tengah] < target)
        {
            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 awal: ";
    // Tampilkan data awal
    for (int x = 0; x < 7; x++)
    {
        cout << setw(3) << arrayData[x];
    }
    cout << endl;
    cout << "\nMasukkan data yang ingin Anda cari: ";
    cin >> cari;
    // Urutkan data dengan selection sort
    selection_sort(arrayData, 7);
    cout << "\nData diurutkan: ";
    // Tampilkan data setelah diurutkan
    for (int x = 0; x < 7; x++)
    {
        cout << setw(3) << arrayData[x];
    }
    cout << endl;
    // Lakukan binary search
    binary_search(arrayData, 7, cari);
    return 0;
}

```

Screenshot Output :

The screenshot shows the output of a C++ program in a terminal window. The output displays the initial data array [1, 8, 2, 5, 4, 9, 7], the user input '3', the sorted array [1, 2, 4, 5, 7, 8, 9], and the message 'Data tidak ditemukan' (Data not found). The terminal prompt is 'PS D:\contoh>'. To the right of the terminal is a form with the following text:

Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B

Below the form, the status bar of the application shows 'Ln 3, Col 15 | 68 characters | 100% | Window | UTF-8'.

Deskripsi:

Program ini menggabungkan dua algoritma: Selection Sort untuk mengurutkan data dan Binary Search untuk mencari nilai dalam data yang sudah diurutkan. Setelah data diurutkan, pengguna dapat mencari nilai tertentu dengan cepat menggunakan Binary Search.

C. Unguided

Unguided 1

Sourcode :

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

using namespace std;

string sentence;
char c;

void toLower() {
    string temp;
    for (int i = 0; i < sentence.length(); i++) {
        temp += tolower(sentence[i]);
    }
    sentence = temp;
}

void selection_sort()
{
    int min, i, j;
    char temp;
```

```

toLower();
for (i = 0; i < sentence.length(); i++)
{
    min = i;
    for (j = i + 1; j < sentence.length(); j++)
    {
        if (sentence[j] < sentence[min])
        {
            min = j;
        }
    }
    temp = sentence[i];
    sentence[i] = sentence[min];
    sentence[min] = temp;
}
}

void binarysearch()
{
    int awal, akhir, tengah, b_flag = 0;
    awal = 0;
    akhir = sentence.length();
    while (b_flag == 0 && awal <= akhir)
    {
        tengah = (awal + akhir) / 2;
        if (sentence[tengah] == c)
        {
            b_flag = 1;
            break;

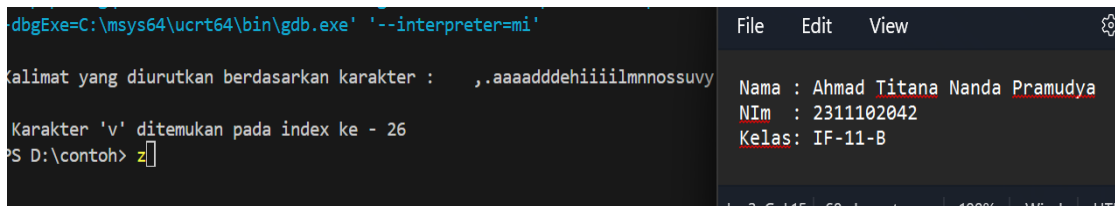
```

```

    }
    else if (sentence[tengah] < c)
        awal = tengah + 1;
    else
        akhir = tengah - 1;
    }
    if (b_flag == 1)
        cout << "\n Karakter " << c << " ditemukan pada index ke - " << tengah <<
endl;
    else
        cout << "\nData tidak ditemukan\n";
    }
int main()
{
    cout << "\t BINARY SEARCH " << endl;
    cout << "Masukkan kalimat : ";
    getline(cin, sentence);
    cout << "\nMasukkan karakter yang ingin Anda cari : ";
    cin >> c;
    c = tolower(c);
    cout << "\nKalimat yang diurutkan berdasarkan karakter : ";
    selection_sort();
    for (int x = 0; x < sentence.length(); x++)
        cout << sentence[x];
    cout << endl;
    binarysearch();
    return EXIT_SUCCESS;
}

```

Screenshot Output:



The screenshot shows a debugger window with a command prompt on the left and a sidebar on the right. The command prompt displays the following text:

```
dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'  
kalimat yang diurutkan berdasarkan karakter : ,.aaaadddehiiiiilmnnossuvy  
Karakter 'v' ditemukan pada index ke - 26  
PS D:\contoh> z
```

The sidebar on the right contains the following information:

```
File Edit View  
Nama : Ahmad Titana Nanda Pramudya  
NIm : 2311102042  
Kelas: IF-11-B
```

Deskripsi:

Program di atas merupakan implementasi sederhana untuk melakukan pencarian biner (binary search) pada sebuah string yang diurutkan. mengurutkan kalimat tersebut berdasarkan karakter, dan kemudian mencari karakter tertentu dalam kalimat yang sudah diurutkan menggunakan algoritma pencarian biner.

Unguded 2

Sourcode :

```
#include <iostream>  
using namespace std;  
int main()  
{  
    string s;  
    int count = 0;  
  
    cout << "\t JUMLAH HURUF VOKAL" << endl;  
    cout << "\nMasukkan kalimat : ";  
    getline(cin, s);  
  
    for (int i = 0; i < s.length(); i++) {  
        char c = tolower(s[i]);  
        if (c == 'a' || c == 'i' || c == 'u' || c == 'e' || c == 'o') count++;  
    }
```

```

    }

    cout << "\nKalimat yang dimasukkan memiliki " << count << " huruf vokal.";

    return 0;
}

```

Screenshot Output :

```

JUMLAH HURUF VOKAL

Masukkan kalimat : AHmad pintar membuat drama sedih

Kalimat yang dimasukkan memiliki 11 huruf vokal.
PS D:\contoh>

```

Deskripsi :

Program ini sebuah program sederhana dalam bahasa C++ yang menghitung jumlah huruf vokal dalam sebuah kalimat yang dimasukkan oleh pengguna.

Unguided 3

Sour code :

```

#include <iostream>

using namespace std;

int arr[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
int arrLength = sizeof(arr)/sizeof(arr[0]);
int cari;

int seqSearch() {
    int count = 0;
    for (int i = 0; i < arrLength; i++)
    {

```

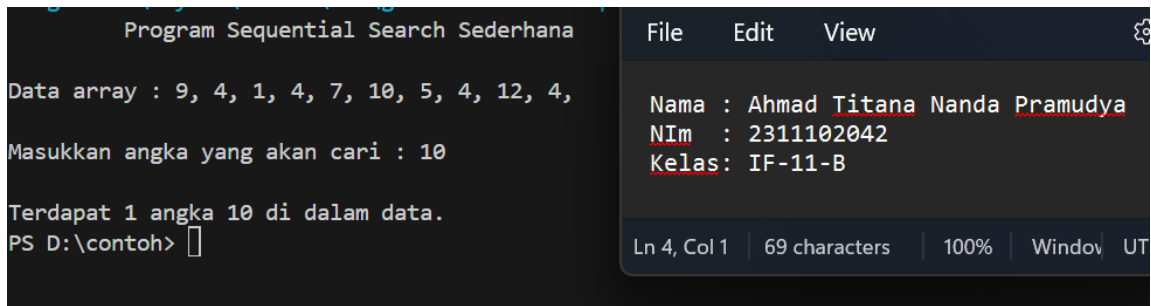
```
        if (arr[i] == cari)
        {
            count++;
        }
    }
    return count;
}

int main()
{
    // algoritma Sequential Search

    cout << "\t Program Sequential Search Sederhana\n" << endl;
    cout << "Data array : ";
    for (int x : arr) {
        cout << x << ", ";
    }
    cout << "\n\nMasukkan angka yang akan cari : ";
    cin >> cari;

    cout << "\nTerdapat " << seqSearch() << " angka " << cari << " di dalam data.";
    return 0;
}
```


Screenshot Output :



```
Program Sequential Search Sederhana
Data array : 9, 4, 1, 4, 7, 10, 5, 4, 12, 4,
Masukkan angka yang akan cari : 10
Terdapat 1 angka 10 di dalam data.
PS D:\contoh>
```

Deskripsi :

Program di atas merupakan implementasi sederhana dari algoritma pencarian sekuensial (sequential search) untuk menghitung kemunculan suatu angka dalam sebuah array

D. Kesimpulan

Algoritma Search adalah serangkaian langkah atau instruksi yang digunakan untuk mencari elemen atau informasi tertentu di dalam suatu dataset. Tujuannya adalah untuk menemukan posisi atau keberadaan elemen yang dicari. Dalam pemrograman, algoritma search menjadi salah satu teknik penting dalam menyelesaikan berbagai masalah.

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. 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. 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.

E. Referensi

[1] UMSU

<https://fikti.umsu.ac.id/algoritma-search-pencarian-pengertian-jenis-dan-contoh-programmnya/>

[2]Media.com

<https://medium.com/@imanshu822/binary-search-and-its-powerful-applications-39ae7d7bca69>