

MODUL 8

ALGORITMA SEARCHING

A. 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.

B. 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 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.

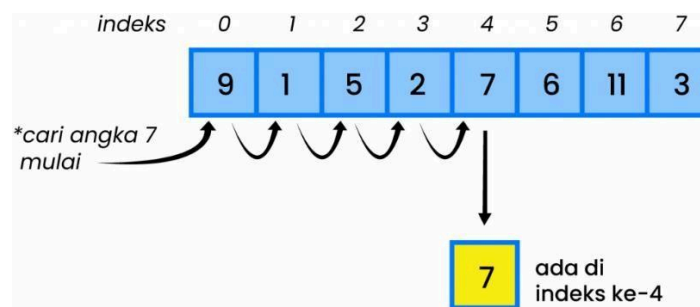
Di bawah ini merupakan fungsi untuk mencari data menggunakan pencarian sekuensial.

```
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:

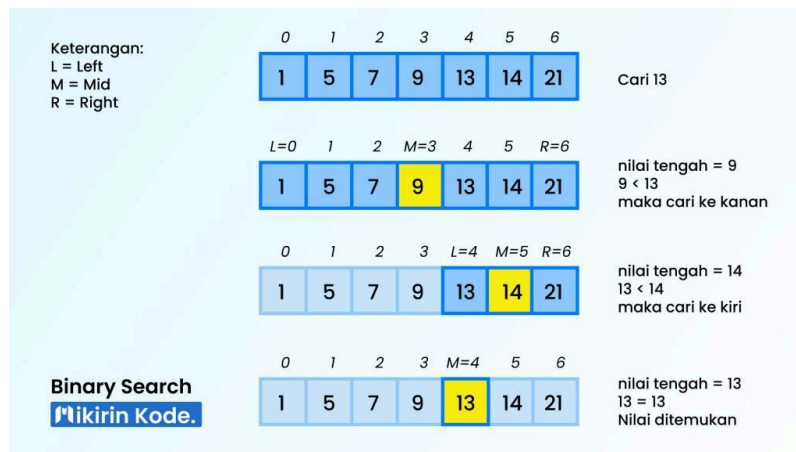
- Data diambil dari posisi 1 sampai posisi akhir N.
- Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
- Selanjutnya data yang dicari akan dibandingkan dengan data yang berada di posisi tengah, apakah lebih besar atau lebih kecil.
- Apabila data yang dicari lebih besar dari 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) $L \leq 0$

- 2) $R \leftarrow N - 1$
- 3) ketemu \leftarrow false
- 4) Selama $(L \leq R)$ dan (tidak ketemu) kerjakan baris 5 sampai dengan 8
- 5) $m \leftarrow (L + R) / 2$
- 6) Jika $(Data[m] = x)$ maka ketemu \leftarrow true
- 7) Jika $(x < Data[m])$ maka $R \leftarrow m - 1$
- 8) Jika $(x > Data[m])$ 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, 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.

C. GUIDED

1. Buatlah sebuah project dengan menggunakan sequential search sederhana untuk melakukan pencarian data.

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 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;
}
```

Output:

```
Program Sequential Search Sederhana

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

angka 10 ditemukan pada indeks ke-9

Process returned 0 (0x0)   execution time : 0.050 s
Press any key to continue.
```

2. Buatlah sebuah project untuk melakukan pencarian data dengan menggunakan 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;
        }
    }
}
```

```

        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];

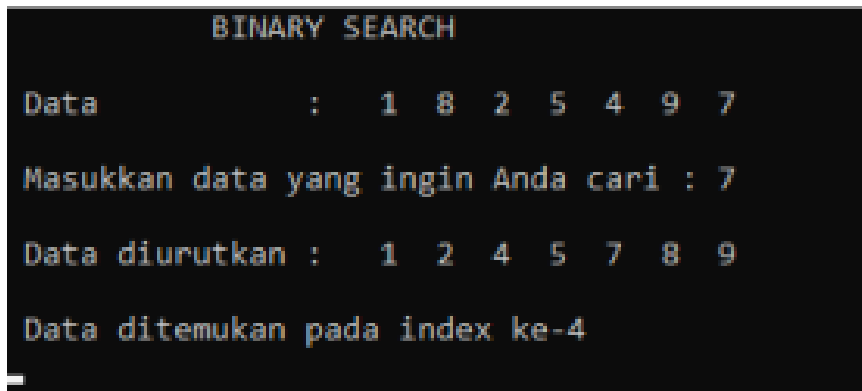
    cout<<endl;

    binarysearch();

    _getche();
    return EXIT_SUCCESS;
}

```

Output:



```

        BINARY SEARCH

Data          :   1   8   2   5   4   9   7

Masukkan data yang ingin Anda cari : 7

Data diurutkan :   1   2   4   5   7   8   9

Data ditemukan pada index ke-4

```

D. UNGUIDED

1. Buatlah sebuah program untuk mencari sebuah huruf pada sebuah kalimat yang sudah di input dengan menggunakan Binary Search!
2. Buatlah sebuah program yang dapat menghitung banyaknya huruf vocal dalam sebuah kalimat!
3. Diketahui data = 9, 4, 1, 4, 7, 10, 5, 4, 12, 4. Hitunglah berapa banyak angka 4 dengan menggunakan algoritma Sequential Search!

E. DAFTAR PUSTAKA

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