BAB V
TEKNIK SEARCHING

A. TUJUAN

1. Mahasiswa mampu mengetahui dan memahami teknik searching
2. Mahasiswa mampu memahami teknik Linear/Sequential Search
3. Mahasiswa mampu memahami teknik Binary Search
4. Mahasiswa mampu mengimplementasikan berbagai jenis metode searching berdasarkan kasus yang diberikan menggunakan bahasa pemrograman C++.

B. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. IDE C++ (Visual Studio Code)

C. KESEHATAN DAN KESELAMATAN KERJA

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana Panjang dan kemeja
6. Gunakan kacamata anti radiasi layar

D. TEORI

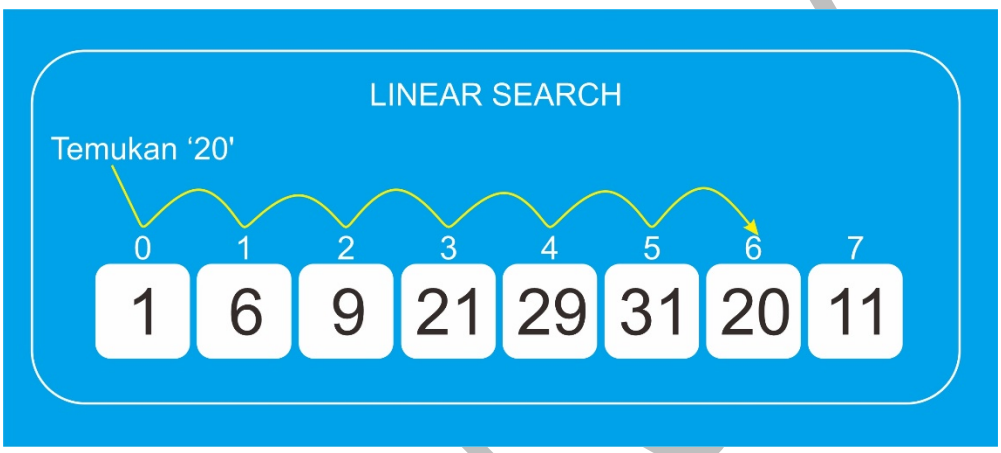
Searching adalah mencari data yang dibutuhkan. Searching dalam pemrograman bisa dilakukan untuk mencari data yang ada di dalam memory computer. Dalam kehidupan sehari-hari kita juga sering melakukan kegiatan searching seperti mencari data/informasi yang ada dalam internet. Ada beberapa metode yang dapat digunakan untuk searching, 2 diantaranya:

- Linear/ Sequential Search
- Binary Search

1. Linear Search

Teknik pencarian data pada array yang paling mudah adalah dengan cara linear/sequential search, dimana data dalam array dibaca 1 demi satu, diurutkan dari index terkecil ke index terbesar, maupun sebaliknya.

Linear search sebaiknya digunakan untuk pencarian data dalam jumlah kecil, dan tidak sering. Jika data banyak, sebaiknya data diurutkan, dan algoritma lain digunakan (misalnya binary search), atau gunakan struktur data khusus untuk pencarian (binary search tree, hash table, dsb). Algoritma pencarian linear sangat sederhana: Untuk setiap elemen di list, jika elemen itu cocok, hentikan pencarian (ketemu), jika tidak cocok, lihat elemen berikutnya, hingga akhir list. Jika akhir list dicapai dan tidak ada yang cocok, berarti elemen tidak ditemukan.



Gambar 4.1. Linear Search

Algoritma:

```
1. Mulai.
2. Tentukan data (a) yang akan dicari.
3. Tentukan panjang array data (J).
4. Tentukan counter j=0.
5. Untuk setiap elemen dalam array, lakukan Langkah
   berikut:
      5.1. Bandingkan data (a) dengan data dalam array
           indeks ke-j.
      5.2. Jika data (a) sama dengan data array indeks
           ke-j, lanjut ke langkah ke 6.
      5.3. Jika data (a) tidak sama dengan data array
           indeks ke-j, maka nilai j ditambah 1.
      5.4. Jika j<=(j-1), kembali ke langkah 5.1 jika
           tidak lanjut ke langkah 7.
```

```
6. Cetak "Elemen a ditemukan pada array indeks ke-j" dan
   lanjut ke langkah 8
7. Cetak "Elemen a tidak dapat ditemukan".
8. Selesai
```

Model algoritma lainnya dapat dilihat sebagai berikut (Mohanty dkk, 2021):

```
n ➔ Boundary of the list
item ➔ angka yang dicari
data ➔ array linear

Step-1      i=0;
Step-2      Repeat while i<=n
            If(item=data[i])
            Print "searching is successful"
            Exit
            Else
            Print "searching is unsuccessful"
Step-3      Exit
```

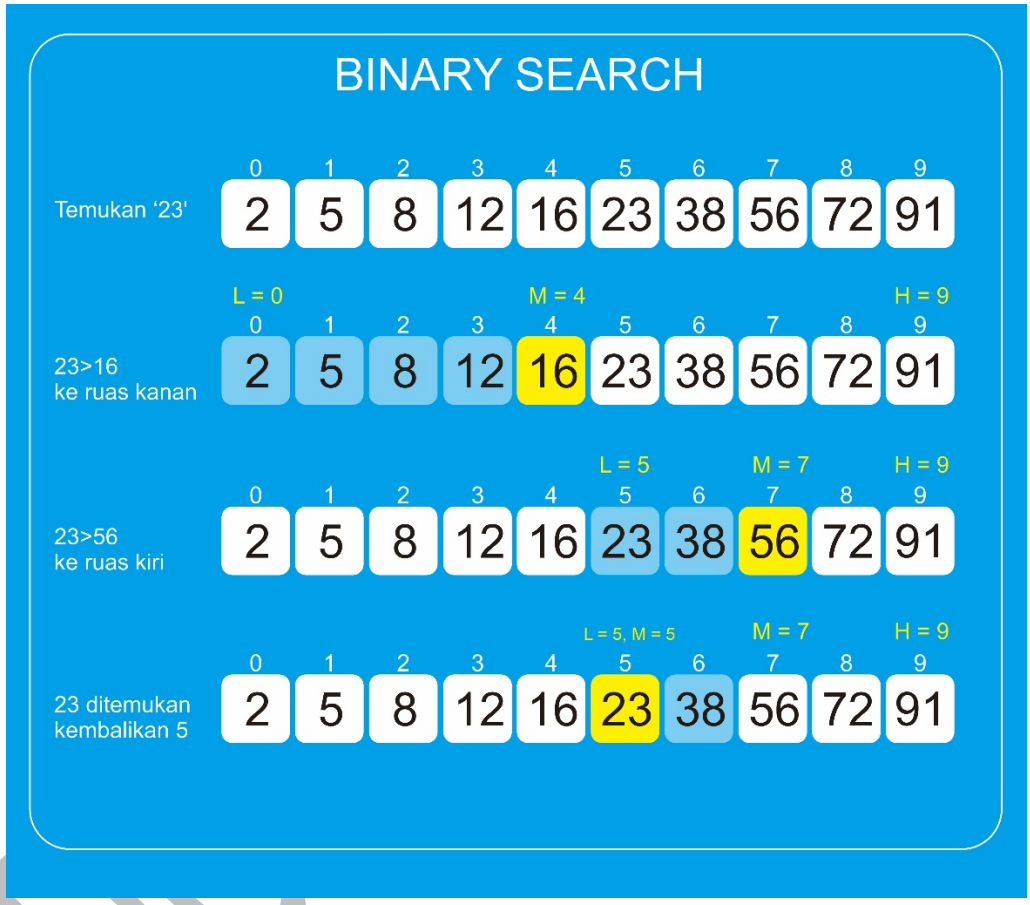
Dimana *n* adalah jumlah elemen, *item* adalah nilai yang dicari dan *data* adalah sebuah array linear dimana nilai-nilai tersimpan. Pada langkah pertama nilai *i* diinisialisasi dengan nilai 0. Kemudian langkah kedua nilai *i*dibandingkan dengan nilai n. Jika nilai *i* lebih kecil atau sama dengan *n* maka lakukan pengecekan pada *data* indeks ke-i. Jika nilai *data* indeks ke-*i* nilainya sama dengan nilai item maka tampilkan "Pencarian berhasil" namun jika tidak, tambah nilai *i* dan bandingkan nilainya dengan indeks berikutnya. Perulangan dilakukan hingga nilai ditemukan. Jika nilai tidak ditemukan maka tampilkan "Pencarian gagal". Program selesai.

Pada Teknik pencarian linear search jumlah perbandingan yang dilakukan agar pencarian berhasil tergantung pada dimana posisi nilai kunci yang dicari berada. Jika nilai yang dicari berada pada indeks pertama maka hanya diperlukan sekali perbandingan.

2. Binary search

Pencarian biner adalah algoritma pencarian cepat yang bekerja berdasarkan prinsip divide and conquer. Agar algoritma ini berfungsi dengan baik, kumpulan data harus dalam bentuk yang telah diurutkan. Metode pencarian Binary yaitu mencari data dengan melakukan pengelompokan

array menjadi beberapa bagian. Binary Search hanya dapat diimplementasikan pada data yang telah terurut baik ascending maupun descending dalam suatu array. Pencarian dilakukan dengan langsung menebak apakah data yang dicari berada ditengah-tengah data yang lainnya, kemudiaan membandingkan data yang ditengah dengan data yang dicari, apabila sama maka dapat dikatakan data ditemukan, namun apabila data yang dicari ternyata lebih kecil daripada elemen tengah, maka pencarian dilakukan dari bagian tengah ke bawah.



Gambar 4.2. Binary Search

Algoritma:

```
1. Mulai.
2. Tentukan nilai (x) yang akan dicari.
3. Tentukan panjang array (n).
4. jika panjang ruas array saat ini sama dengan atau
   lebih besar dari 1 maka lanjut ke langkah 5, jika
   tidak lanjut ke langkah 10.
```

```
5. Bagi array menjadi dua ruas dan cari indeks (j)
   paling tengah dari array (mid).
6. Jika elemen indeks (j) tengah (mid) dari ruas saat
   ini sama dengan nilai yang dicari, maka lanjut ke
   langkah 9
7. Jika elemen indeks (j) tidak sama dengan data yang
   dicari dan nilai yang dicari lebih besar dari elemen
   indeks (j) maka lanjutkan pencarian ke ruas kanan
   (indeks lebih besar), kembali ke langkah 4.
8. Jika elemen indeks (j) tidak sama data yang dicari
   dan jika data yang dicari nilainya lebih kecil maka
   lanjutkan pencarian ke ruas kiri (indeks lebih
   kecil), kembali ke langkah 4.
9. Cetak "Elemen a ditemukan pada array indeks ke-j"
   lalu lanjut ke langkah 11.
10.     Cetak "Elemen a tidak dapat ditemukan".
11.     Selesai
```

Model algoritma lainnya dapat dilihat sebagai berikut (Mohanty dkk, 2021):

```
Step-1     low=0, up = n-1
Step-2     Repeat while low<=up
                mid = int(low+up)/2
           if (no=arr[mid])
                print "Searched element is found"
                exit
           if (no<arr[mid])then
                up = mid - 1
           else
                low = mid+1
Step-3     print "Searched element is not found"
Step-4     Exit
```

Pada algoritma di atas *low* diinisialisasi dengan nilai 0 dan *up* diinisialisasi dengan banyak elemen (*n*) dikurangi 1. Selama nilai *low* masih lebih kecil atau sama dengan *up* maka cari nilai *mid* dengan membagi dua hasil penjumlahan *low* dan *up*, hasilnya dibulatkan ke bawah. Jika nilai yang dicari

(*no*) sama dengan nilai elemen array indeks ke-*mid* maka tampilkan "Searched element is found" lalu ke langkah ke 4. Namun jika elemen yang dicari (*no*) nilainya lebih kecil dari array indeks ke-*mid* maka ganti nilai *up* dengan *mid* dikurangi 1. Selain itu *low* diisi dengan nilai *mid* ditambah 1. Jika nilai yang dicari tidak ditemukan maka tampilkan "Searched element is not found" lalu keluar.

E. PRAKTIKUM

Praktikum 5.1. Linear search

1. Ketikkan kode berikut pada Main.cpp

```
#include <iostream>
#include "Cari.h"
using namespace std;

int main(void)
{
    int arr[] = { 5, 3, 50, 10, 40 };
    int x = 50;
    int n = sizeof(arr) / sizeof(arr[0]);

    Cari cariNilai;
    int hasilPencarian = cariNilai.linearSearch(arr, n, x);

    (hasilPencarian == -1)?
        cout<<"Elemen tidak ditemukan di dalam array"
        :cout<<"Elemen ditemukan pada indeks ke-" <<hasilPencarian;

    return 0;
}
```

2. Buat file baru bernama "Cari.cpp" dan "Cari.h" pada folder src
3. Pada file Cari.h, ketikkan kode berikut.

```
#include <iostream>
class Cari
{
public:
int linearSearch(int arr[], int n, int x);

};
```

4. Pada file Cari.cpp, ketikkan kode berikut.

```
#include <iostream>
```

```
#include "Cari.h"
using namespace std;

int Cari::linearSearch(int arr[], int n, int x){
    int i;
    for (i = 0; i < n; i++){
        if (arr[i] == x){
            return i;
        }
    }
    return -1;
};
```

5. Jalankan kode program (Ctrl+Shit+B)

Praktikum 5.2. Binary search

1. Ketikkan kode berikut pada Main.cpp

```
#include <iostream>
#include "Cari.h"
using namespace std;

int main(void)
{
    int arr[] = {2,3,4,5};
    int x = 9;
    int n = sizeof(arr) / sizeof(arr[0]);

    Cari cariNilai;
    int hasil = cariNilai.binarySearch(arr, 0, n - 1, x);

    (hasil == -1) ?
      cout << "Elemen tidak ditemukan di dalam array"
      : cout << "Elemen ditemukan pada array indeks ke- " << hasil;
    return 0;
}
```

2. Buat file baru bernama “Cari.cpp” dan “Cari.h” pada folder src
3. Pada file Cari.h, ketikkan kode berikut.

```
#include <iostream>

class Cari{
public:
```

```
int binarySearch(int arr[], int l, int r, int x);
};
```

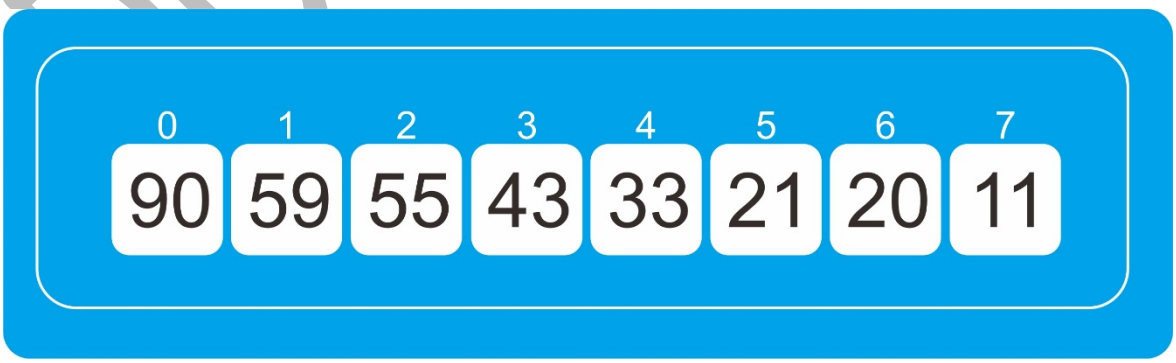
4. Pada file Cari.cpp, ketikkan kode berikut.

```
#include <iostream>
#include "Cari.h"
using namespace std;

int Cari::binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;

        if (arr[mid] == x)
            return mid;

        if (arr[mid] > x)
        return binarySearch(arr, l, mid - 1, x);
        return binarySearch(arr, mid + 1, r, x);
    }
    return -1;
}
```

5. Jalankan kode program (Ctrl+Shit+B)

F. LATIHAN

1. Perhatikan array berikut



Terdapat sebuah array yang menampung nilai berurutan secara descending (dari besar ke kecil). Terapkanlah tekni pencarian Binary search pada array tersebut.

Adapun syarat programnya sebagai berikut:

- Gunakan pemanggilan fungsi didalam class untuk melakukan pencarian.
- File class dibuat terpisah.

Kunci: manfaatkan percabangan, perulangan, dan array.

CATATAN