JOBSHEET VI
SEARCHING

6.1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Menjelaskan mengenai algoritma Searching.
2. Membuat dan mendeklarasikan struktur algoritma Searching.
3. Menerapkan dan mengimplementasikan algoritma Searching.

6.2. Searching / Pencarian Menggunakan Agoritma Sequential Search

Perhatikan diagram class Mahasiswa di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program class Mahasiswa.

Mahasiswa
Nim: int nama: String umur: int ipk: double
Mahasiswa(ni:int, n: String, u: int, i: double) tampil(): void

Berdasarkan class diagram di atas, akan dibuat class Mahasiswa yang berfunsi untuk membuat objek mahasiswa yang akan dimasukan ke dalam sebuah array. Terdapat sebuah konstruktor berparameter dan juga fungsi tampil() untuk menampilkan semua attribute yang ada.

PencarianMhs
listMhs: Mahasiswa[5] idx: int
tambah(mhs: Mahasiswa): void tampil(): void FindSeqSearch(int cari): int Tampilpoisisi(int x,int pos): void TampilData(int x,int pos) :void

Selanjutnya class diagram di atas merupakan representasi dari sebuah class yang berfungsi untuk melakukan operasi-operasi dari objek array mahasiswa, misalkan untuk menambahkan objek mahasiswa, menampilkan semua data mahasiswa, untuk melakukan pencarian berdasarkan NIM menggunakan algoritma Sequential Search, menampilkan posisi dari data yang dicari, serta menampilkan data mahasiswa yang dicari.

6.2.1. Langkah-langkah Percobaan Sequential Search

1. Buatlah Project baru pada Netbeans dengan nama **`TestSearching`**

2. Kemuadian buat packages baru dengan nama **minggu7**.
3. Buat class **Mahasiswa**, kemudian deklarasikan atribut berikut ini:

```
public class Mahasiswa {
    int nim;
    String nama;
    int umur;
    double ipk;
```

4. Buatlah konstruktor dengan nama **Mahasiswa** dengan parameter (**int ni, String n, int u, double i**) kemudian Isi konstruktor tersebut dengan kode berikut!

```
Mahasiswa(int ni, String n, int u, double i){
    nim = ni;
    nama = n;
    umur = u;
    ipk = i;
}
```

5. Buatlah method **tampil** bertipe void.

```
void tampil(){
    System.out.println("Nim = " + nim);
    System.out.println("Nama = " + nama);
    System.out.println("Umur = " + umur);
    System.out.println("IPK = " + ipk);
}
```

6. Buat class baru dengan nama **PencarianMhs** seperti di bawah ini!

```
public class PencarianMhs {
    Mahasiswa listMHs[] = new Mahasiswa[5];
    int idx;

}
```

7. Tambahkan method **tambah()** di dalam class tersebut! Method **tambah()** digunakan untuk menambahkan objek dari class Mahasiswa ke dalam atribut listMhs.

```
void tambah(Mahasiswa m) {
    if(idx < listMHs.length){
        listMHs[idx] = m;
        idx ++;
    } else {
        System.out.println("Data sudah penuh !!");
    }
}
```

8. Tambahkan method **`tampil()`** di dalam class **`PencarianMhs`**! Method **`tampil()`** digunakan untuk menampilkan semua data mahasiswa-mahasiswa yang ada di dalam class tersebut! Perhatikan penggunaan sintaks for yang agak berbeda dengan for yang telah dipelajari sebelumnya, meskipun secara konsep sebenarnya mirip.

```
void tampil(){
    for(Mahasiswa m : listMHs){
        m.tampil();
        System.out.println("------------------------");
    }
}
```

9. Tambahkan method **`FindSeqSearch`** bertipe integer dengan parameter **`cari`** bertipe integer. Kemudian Deklarasikan isi method **`FindSeqSearch`** dengan algoritma pencarian data menggunakan teknik sequential searching.

```
public int FindSeqSearch(int cari) {
    int posisi = -1;
    for (int j = 0; j < listMHs.length; j++) {
        if (listMHs[j].nim==cari) {
            posisi = j;
            break;
        }
    }
    return posisi;
}
```

10. Buatlah method **`Tampilposisi`** bertipe void dan Deklarasikan isi dari method **`Tampilposisi`**.

```
public void Tampilpoisisi(int x,int pos)
{
    if (pos!= -1) {
        System.out.println("data : " + x + "ditemukan pada indeks " + pos);

    } else {
        System.out.println("data " + x + "tidak ditemukan");
    }
}
```

11. Buatlah method **`TampilData`** bertipe void dan Deklarasikan isi dari method **`TampilData.`**

```
public void TampilData(int x,int pos)
{
    if (pos!= -1) {
        System.out.println("Nim\t : " + x );
        System.out.println("Nama\t : "+listMHs[pos].nama);
        System.out.println("Umur\t : "+listMHs[pos].umur);
        System.out.println("IPK\t : "+listMHs[pos].ipk);
    } else {
        System.out.println("data " + x + "tidak ditemukan");
    }
}
```

12. Buatlah class baru dengan nama **`MahasiswaMain`** tambahkan method **`main`** seperti pada gambar berikut!

```
public class MahasiswaMain {
    public static void main(String[] args) {

    }
}
```

13. Di dalam method **`main()`**, buatlah sebuah objek PencarianMhs dan buatlah 5 objek mahasiswa kemudian tambahkan semua objek mahasiswa tersebut dengan memanggil fungsi tambah pada objek PencarianMhs.

```
Scanner s = new Scanner(System.in);
Scanner sl = new Scanner(System.in);

PencarianMhs data = new PencarianMhs();
int jumMhs = 5;

System.out.println("-----------------------------------------------");
System.out.println("Masukkan data mahasiswa secara Urut dari Nim Terke
for(int i = 0; i < jumMhs; i++){
    System.out.println("------------------");
    System.out.print("Nim\t: ");
    int nim = s.nextInt();
    System.out.print("Nama\t  : ");
    String nama = sl.nextLine();
    System.out.print("Umur\t  : ");
    int umur = s.nextInt();
    System.out.print("IPK\t  : ");
    double ipk = s.nextDouble();

    Mahasiswa m = new Mahasiswa(nim, nama, umur, ipk);
    data.tambah(m);
}
```

14. Panggil method **`tampil()`** untuk melihat semua data yang telah dimasukan.

```
    System.out.println("-----------------------------------------------");
    System.out.println("Data keseluruhan Mahasiswa : ");
    data.tampil();
```

15. Untuk melakukan pencarian berdasarkan NIM mahasiswa. Buatlah variable **`cari`** yang dapat menampung masukan dari keyboard lalu panggil method **`FindSeqSearch`** dengan isi parameternya adalah variable cari.

```
System.out.println("_____________________________________________");
System.out.println("_____________________________________________");
System.out.println("Pencarian Data : ");
System.out.println("Masukkan Nim Mahasiswa yang dicari: ");
System.out.print("NIM : ");
int cari =  s.nextInt();
System.out.println("menggunakan sequential Search");
int posisi = data.FindSeqSearch(cari);
```

16. Lakukan pemanggilan method **`Tampilposisi`** dari class **`PencarianMhs.`**

```
data.Tampilpoisisi(cari, posisi);
```

17. Lakukan pemanggilan method **`TampilData`** dari class **`PencarianMhs.`**

```
data.TampilData(cari, posisi);
```

18. Jalankan dan amati hasilnya.

6.2.2. Verifikasi Hasil Percobaan

Cocokkan hasil kode program anda dengan gambar berikut ini.

```
Masukkan data mahasiswa secara Urut dari Nim Terkecil :
------------------
Nim     : 2017
Nama      : Dewi Lestari
Umur      : 23
IPK       : 3.5
------------------
Nim     : 2018
Nama      : Sinta Sanjaya
Umur      : 22
IPK       : 4
------------------
Nim     : 2019
Nama      : Danang Adi
Umur      : 22
IPK       : 3.7
------------------
Nim     : 2020
Nama      : Budi Prakarsa
Umur      : 20
IPK       : 2.9
------------------
Nim     : 2021
Nama      : Vania Siti
Umur      : 20
IPK       : 3.0

-----------------------------------------------
Data keseluruhan Mahasiswa :
Nim = 2017
Nama = Dewi Lestari
Umur = 23
IPK = 3.5
-----------------------
Nim = 2018
Nama = Sinta Sanjaya
Umur = 22
IPK = 4.0
-----------------------
Nim = 2019
Nama = Danang Adi
Umur = 22
IPK = 3.7
-----------------------
Nim = 2020
Nama = Budi Prakarsa
Umur = 20
IPK = 2.9
-----------------------
Nim = 2021
Nama = Vania Siti
Umur = 20
IPK = 3.0
-----------------------
```

```
Pencarian Data :
Masukkan Nim Mahasiswa yang dicari:
NIM : 2018
menggunakan sequential Search
data : 2018ditemukan pada indeks 1
Nim      : 2018
Nama     : Sinta Sanjaya
Umur     : 22
IPK      : 4.0
```

6.2.3. Pertanyaan

1. Jelaskan perbedaan metod **`TampilData`** dan **`Tampilposisi`** pada class PencarianMhs
2. Jelaskan fungsi **`break`** pada kode program dibawah ini!

```
if (listMHs[j].nim==cari) {
    posisi = j;
    break;
}
```

3. Jika Data Nim yang dimasukkan tidak terurut dari kecil ke besar. Apakah program masih dapat berjalan? Apakah hasil yang dikeluarkan benar? Mengapa demikian!

6.3. Searching / Pencarian Menggunakan Binary Search

6.3.1. Langkah-langkah Percobaan Binary Search

1. Pada percobaan 6.2.1 (sequential search) tambahkan method **`FindBinarySearch`** bertipe integer pada class **`PencarianMhs`**. Kemudian Deklarasikan isi method **`FindBinarySearch`** dengan algoritma pencarian data menggunakan teknik binary searching.

```
public int FindBinarySearch(int cari, int left, int right) {
    int mid;
    if (right >= left) {
        mid = (left + right) / 2;
        if (cari == listMHs[mid].nim) {
            return (mid);
        } else if (listMHs[mid].nim > cari) {
            return FindBinarySearch(cari, left, mid - 1);
        } else {
            return FindBinarySearch(cari, mid + 1, right);
        }
    }
    return -1;
}
```

2. Panggil method **`FindBinarySearch`** terdapat pada class **`PencarianMhs`** di kelas **`Mahasiswamain.`** Kemudia panggil **`method tampilposisi`** dan **`tampilData`**

```
System.out.println("==============================");
System.out.println("menggunakan binary Search");
posisi = data.FindBinarySearch(cari, 0, jumMhs - 1);
data.Tampilpoisisi(cari, posisi);
data.TampilData(cari, posisi);
```

3. Jalankan dan amati hasilnya.

6.3.2. Verifikasi Hasil Percobaan

Cocokkan hasil kode program anda dengan gambar berikut ini.

```
Masukkan data mahasiswa secara Urut dari Nim Terkecil :
------------------
Nim     : 2017
Nama      : Dewi Lestari
Umur      : 23
IPK       : 3.5
------------------
Nim     : 2018
Nama      : Sinta Sanjaya
Umur      : 22
IPK       : 4
------------------
Nim     : 2019
Nama      : Danang Adi
Umur      : 22
IPK       : 3.7
------------------
Nim     : 2020
Nama      : Budi Prakarsa
Umur      : 20
IPK       : 2.9
------------------
Nim     : 2021
Nama      : Vania Siti
Umur      : 20
IPK       : 3.0
```

```
-------------------------------------------
Data keseluruhan Mahasiswa :
Nim = 2017
Nama = Dewi Lestari
Umur = 23
IPK = 3.5
-----------------------
Nim = 2018
Nama = Sinta Sanjaya
Umur = 22
IPK = 4.0
-----------------------
Nim = 2019
Nama = Danang Adi
Umur = 22
IPK = 3.7
-----------------------
Nim = 2020
Nama = Budi Prakarsa
Umur = 20
IPK = 2.9
-----------------------
Nim = 2021
Nama = Vania Siti
Umur = 20
IPK = 3.0
-----------------------
```

```
Pencarian Data :
Masukkan Nim Mahasiswa yang dicari:
NIM : 2018
menggunakan sequential Search
data : 2018ditemukan pada indeks 1
Nim      : 2018
Nama     : Sinta Sanjaya
Umur     : 22
IPK      : 4.0
================================
menggunakan binary Search
data : 2018ditemukan pada indeks 1
Nim      : 2018
Nama     : Sinta Sanjaya
Umur     : 22
IPK      : 4.0
```

6.3.3. Pertanyaan

1. Tunjukkan pada kode program yang mana proses divide dijalankan!
2. Tunjukkan pada kode program yang mana proses conquer dijalankan!
3. Jika data Nim yang dimasukkan tidak urut. Apakah program masih dapat berjalan? Mengapa demikian!

4. Jika Nim yang dimasukkan dari NIM terbesar ke terkecil (missal : 20215, 20214, 20212, 20211, 20210) dan elemen yang dicari adalah 20210. Bagaimana hasil dari binary search? Apakah sesuai? Jika tidak sesuai maka ubahlah kode program binary seach agar hasilnya sesuai
5. Modifikasilah program diatas yang mana jumlah mahasiswa yang di inputkan sesuai dengan masukan dari keyboard.

6.4. Percobaan Pengayaan Divide and Conquer

6.4.1. Langkah-langkah Percobaan Merge Sort

1 Buatlah Package baru pada NetBeans dengan nama **`MergeSortTest`**

2 Tambahkan class **`MergeSorting`** pada package tersebut

3 Pada class **`MergeSorting`** buatlah method **`mergeSort`** yang menerima parameter data array yang akan diurutkan

```
public void mergeSort(int[] data) {
```

4 Buatlah method **`merge`** untuk melakukan proses penggabungan data dari bagian kiri dan kanan.

```
private void merge(int data[], int left, int middle, int right) {
```

5 Implementasikan proses **`merge`** sebagai berikut.

```
public void merge(int data[], int left, int middle, int right) {
    int[] temp = new int[data.length];
    for (int i = left; i <= right; i++) {
        temp[i] = data[i];
    }
    int a = left;
    int b = middle + 1;
    int c = left;

     //membandingkan setiap bagian
     while (a <= middle && b <= right) {
         if (temp[a] <= temp[b]) {
             data[c] = temp[a];
             a++;
         } else {
             data[c] = temp[b];
             b++;
         }
         c++;
     }
     int s = middle - a;
     for (int i = 0; i <= s; i++) {
         data[c + i] = temp[a + i];
     }
 }
```

6 Buatlah method **`sort`**

```
private void sort(int data[], int left, int right) {
```

7 Implementasikan kode berikut pada method **sort**

```
//membagi menjadi 2 bagian dan dibagi kembali hingga tidak dapat dibagi kembali
private void sort(int data[], int left, int right) {
    if (left < right) {
        int middle = (left + right) / 2;
        sort(data, left, middle);
        sort(data, middle + 1, right);
        merge(data, left, middle, right);
    }
}
```

8 Pada method **mergeSort**, panggil method **sort** dengan parameter data yang ingin diurutkan serta range data awal sampai dengan akhir.

9 Tambahkan method **printArray**

```
public void printArray(int arr[]){
    int n= arr.length;
    for (int i=0; i<n;i++)
    {
        System.out.print(arr[i]+" ");
    }
    System.out.println();
}
```

10 Sebagai langkah terakhir, deklarasikan data yang akan diurutkan kemudian panggil proses sorting pada class SortMain

```
class SortMain {

    public static void main(String[] args) {
        int data[] ={10,40,30,50,70,20,100,90};
        System.out.println("sorting dengan merge sort");
        MergeSorting mSort= new MergeSorting();
        System.out.println("data awal");
        mSort.printArray(data);
        mSort.mergeSort(data);
        System.out.println("setelah diurutkan");
        mSort.printArray(data);
    }

}
```

1.

6.4.2. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
sorting dengan merge sort
data awal
10 40 30 50 70 20 100 90
setelah diurutkan
10 20 30 40 50 70 90 100
BUILD SUCCESSFUL (total time: 1 second)
```

6.5. Latihan Praktikum

1. Modifikasi percobaan searching diatas dengan ketentuan berikut ini
   - Sebelum dilakukan searching dengan binary search data harus dilakukan pengurutan dengan menggunakan algoritma sorting (pilih salah satu algoritma sorting dari pertemuan sebelumnya)

2. Modifikasi percobaan searching diatas dengan ketentuan berikut ini
   - Pencarian dilakukan berdasarkan Nama Mahasiswa (gunakan Algoritma Sequential Search)
   - Jika terdapat nama yang sama? Bagaimana keluaran dari kode program ketika pencarian dilakukan pada nama yang sama!

3. Terdapat sebuah data array 2 dimensi sebagai berikut:

Indeks	0	1	2	3	4
0	45	78	7	200	80
1	90	1	17	100	50
2	21	2	40	18	65

   Berdasarkan data di atas buatlah program untuk melakukan pencarian data pada array dua dimensi tersebut, dimana data yang dicari di inputkan melalui keyboard (menggunakan sequential search) !

4. Terdapat sebuah data array 1 dimensi sebagai berikut

0	1	2	3	4	5	6	7	8	9
12	17	2	1	70	50	90	17	2	90

   Buatlah program untuk mengurutkan array tersebut (boleh memilih metode pengurutan) selanjutnya lakukan pencarian dan mencetak isi array yang nilainya terbesar, dan mencetak ada

berapa buah nilai terbesar tersebut serta berada dilokasi mana saja nilai terbesar tersebut! (menggunakan binary search)