



# **MODUL PRAKTEK LOGIKA DAN ALGORITMA**



**Di susun Oleh :  
UNIT PENGEMBANGAN AKADEMIK**

**UNIVERSITAS BINA SARANA INFORMATIKA  
PROGRAM STUDI SISTEM INFORMASI  
KAMPUS JAKARTA**

**2021**

## KATA PENGANTAR

Puji syukur penulis panjatkan kehadiran Allah SWT, yang telah memberikan rahmat dan hidayahnya sehingga modul praktek matakuliah Logika dan Algoritma ini dapat terselesaikan dengan baik. Selanjutnya modul ini disusun untuk memberikan gambaran bagi mahasiswa yang mempelajari Logika dan Algoritma karena modul ini disertai contoh kasus, sehingga lebih memudahkan dalam memahami materi melalui soal studi kasus

Tak lupa penulis mengucapkan banyak terima kasih kepada semua pihak yang telah membantu dengan tenaga dan pikirannya, terima kasih juga kepada rekan-rekan instruktur, dosen dan semuanya yang tidak bisa disebutkan satu persatu, yang selalu mendukung penulis sehingga modul ini sehingga dapat selesai sesuai yang kita inginkan semua.

Penulis menyadari masih banyak kekurangan dalam penyusunan modul ini. Untuk itu saran dan kritik yang membangun sangat penulis harapkan guna perbaikan dan pengembangan modul ini ke depan.

Akhir kata penulis berharap semoga modul praktek matakuliah Logika dan Algoritma ini dapat dipergunakan sebaik-baiknya dan dapat dijadikan referensi untuk mahasiswa umum yang ingin mempelajari materi tersebut.

Jakarta, Januari 2021

**Tim Penyusun**  
**Unit Pengembangan Akademik**  
**Program Studi Sistem Informasi**

## DAFTAR ISI

|   |     |
|---|-----|
| KATA PENGANTAR .....  | ii  |
| DAFTAR ISI.....   | iii |
| Minggu Ke- 1 Pengertian Dasar Logika dan Algoritma.....     | 1   |
| Minggu Ke- 2 Konsep Algoritma dan Tipe Data.....            | 4   |
| Minggu Ke- 3 Flowchart.....                                 | 8   |
| Minggu Ke- 4 Struktur Branching .....                       | 11  |
| Minggu Ke- 5 Looping .....                                  | 15  |
| Minggu Ke- 6 Rekursif .....                                 | 21  |
| Minggu Ke- 7 QUIZ .....                                     | 24  |
| Minggu Ke- 8 UTS .....                                      | 25  |
| Minggu Ke-10 Metode Divide & Conquer.....                   | 31  |
| Minggu Ke-11 Tehnik Searching.....                          | 37  |
| Minggu Ke-12 Metode Greedy .....                            | 45  |
| Minggu Ke-13 Penyelesaian Algoritma Pemrograman Greedy..... | 45  |
| Minggu Ke-14 Pewarnaan .....                                | 51  |
| Minggu Ke-15 QUIZ .....                                     | 54  |
| Minggu Ke-16 UAS.....                                       | 55  |
| DAFTAR PUSTAKA .....  | 56  |

# Minggu Ke- 1

## Pengertian Dasar Logika dan Algoritma

### Deskripsi

Membahas tentang logika dan algoritma, program dan bahasa pemrograman, tahapan analisa algoritma dan sifat-sifat algoritma

### Tujuan Pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan algoritma pada kegiatan sehari-hari
2. Menjelaskan program dan bahasa pemrograman
3. Menjelaskan tahapan analisa algoritma
4. Menjelaskan sifat-sifat algoritma

### Pengertian Logika dan Algoritma

Logika: pertama kali diperkenal oleh Aristoteles (384 – 322 SM)

Algoritma: diperkenalkan oleh Ahli Matematika yang bernama Abu Ja'far Muhammad Ibnu Musa Al Khawarizmi. Seorang ilmuwan Persia yang menulis kitab al Jabr w'al muqabala (*rules of restoration and reduction*) sekitar tahun 825 M

Definisi Logika adalah Penalaran atau bentuk pemikiran, atau Ilmu yang memberikan prinsip-prinsip yang harus diikuti agar dapat berfikir valid menurut aturan yang berlaku.

Definisi Algoritma adalah Langkah - langkah yang dilakukan agar solusi masalah dapat diperoleh atau suatu prosedur yang merupakan urutan langkah-langkah yg berintegrasi atau suatu metode khusus yang digunakan untuk menyelesaikan suatu masalah yang nyata.

Program adalah kumpulan intruksi-instruksi yang diberikan kepada komputer untuk melaksanakan suatu tugas atau pekerjaan. Dalam membuat program dibutuhkan bahasa pemrograman.

Bahasa pemrograman adalah bahasa komputer yang digunakan dalam menulis program

Berdasarkan kedekatan bahasa pemrograman dikelompokkan menjadi 2 macam yaitu:

1. Bahasa tingkat rendah  
Bahasa yang dirancang agar setiap instruksinya langsung dikerjakan oleh komputer, tanpa harus melalui penerjemah. Contoh: bahasa mesin (sekumpulan kode biner (0 dan 1))
2. Bahasa tingkat tinggi  
Bahasa jenis ini membuat program menjadi lebih mudah dipahami. Contoh: *Pascal, Cobol, Fortran, Basic, Prolog, C, C++, PHP, Java, Python*

### Tahapan Analisa Algoritma

1. Bagaimana merencanakan suatu algoritma.  
Dengan menentukan model atau desain untuk menyelesaikan suatu masalah sebagai sebuah solusi, sehingga akan banyak terdapat variasi model yang diambil yang terbaik.
2. Bagaimana menyatakan suatu algoritma

Menentukan model algoritma yang digunakan untuk membuat barisan secara urut agar mendapatkan solusi masalah. Model algoritma tersebut dapat dinyatakan dengan *pseudocode* atau *flowchart*.

- a. Pseudocode (bahasa semu)  
Merupakan bentuk informal untuk mendeskripsikan algoritma yang mengikuti struktur bahasa pemrograman tertentu.
- b. Flowchart (Diagram Alir)  
Penggambaran algoritma secara diagram yang menggambarkan alur susunan logika dari suatu masalah.
3. Bagaimana validitas suatu algoritma.  
Validitas suatu algoritma dengan didapatkan solusi sebagai penyelesaian dari masalah
4. Bagaimana Menganalisa suatu Algoritma.  
Analisa algoritma dengan melihat waktu tempuh dan jumlah memori yang digunakan
5. Bagaimana Menguji Program dari suatu Algoritma.  
Algoritma tersebut diimplementasikan kedalam bahasa pemrograman misal: Python. Proses uji algoritma tersebut dengan dua tahap yaitu:
  - a. Fase Debugging yaitu fase dari proses program eksekusi yang akan melakukan koreksi terhadap kesalahan.
  - b. Fase Profilling yaitu yaitu fase yang akan bekerja jika program tersebut sudah benar (telah melewati fase debugging).

Sifat-sifat Algoritma

1. Banyaknya Langkah Instruksi Harus Berhingga,
2. Langkah atau Instruksi harus Jelas,
3. Proses harus Jelas dan mempunyai batasan,
4. Input dan Output harus mempunyai Batasan,
5. Efektifitas,
6. Adanya Batasan Ruang Lingkup

### Latihan

1. Buatlah sebuah rancangan program dengan menggunakan Pseduocode untuk mencari luas bangun geometris yang terdiri dari :
  - a. Bujur Sangkar
  - b. Segitiga
  - c. Persegi panjang

Langkah-langkah :

1. Buat input pilihan perhitungan bangun geometris yang diinginkan
  - baca pilih perhitungan yaitu
    - Jika a pilihan untuk bujur sangkar
    - Jika b pilihan untuk segitiga
    - Jika c pilihan untuk persegi panjang
2. Buat proses perhitungan untuk masing-masing bangun geometris, sebagai berikut :
  - a. bujur sangkar
    - baca sisi
    - hitung luas
    - $L = s \times s$
  - b. segitiga
    - baca alas dan tinggi

- hitung luas  
 $L = \frac{1}{2} \times a \times t$
- c. persegi panjang
  - baca panjang dan lebar
  - hitung luas  
 $L = p \times l$
- 3. Buat output dari perhitungan luas bangun geometris yang telah dipilih sebagai berikut
  - cetak luas

**Hasil Praktek : Logika dari perhitungan mencari luas bujur sangkar, segitiga dan persegi panjang**

## Minggu Ke- 2

### Konsep Algoritma dan Tipe Data

#### Deskripsi

Membahas tentang konsep algoritma yang terdiri dari algoritma peubah, algoritma pertukaran dan analisa algoritma

#### Tujuan pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan algoritma peubah
2. Menjelaskan algoritma pertukaran
3. Menjelaskan analisa algoritma

#### 1. ALGORITMA PE-UBAH

Adalah Variabel yang nilainya BUKAN konstanta (selalu berubah – sesuai dengan kondisi Variabel terkini)

Sintaks :  $P = Q$   
Algoritma :  $P \leftarrow Q$   
Arti : Bahwa Nilai P diberi harga Nilai Q  
Nilai P akan SAMA DENGAN nilai Q, & Nilai Q TETAP

#### 2. ALGORITMA PERTUKARAN

Berfungsi mempertukarkan masing-masing isi Variabel sedemikian sehingga nilai dari tiap Variabel akan berubah/bertukar.

Contoh Soal:

1. Diketahui  $P=0$ ,  $Q=5$  dan  $R=10$ .  
Diberikan Algoritma  $P=Q, Q=R$ , mk Nilai  $P, Q, R$  sekarang?
2. Diketahui Algoritma  $P=10, P=P+1$  dan  $Q = P$   
Berapakan Nilai  $P$  dan  $Q$  ? .....
3. Diketahui 3 variabel peubah  $P, Q$  dan  $R$ . Agar isi  $Q$  ditaruh di  $P$ , isi  $R$  ditaruh di  $Q$  dan isi  $P$  ditaruh di  $R$ , maka Algoritma yang dapat ditulis adalah : .....
4. Diketahui 2 peubah  $K = 10$  dan  $L = 20$ . Buat Algoritma untuk mempertukarkan isi  $K$  dan  $L$ . : .....

Tipe Data Pemrograman Python.

Tipe data pada pemrograman Python dapat dilihat pada tabel 1 dibawah ini

Tabel 1. Tipe Data Python

| Tipe Data | Keterangan   |
|-----------|--|
| Boolean   | Mempunyai dua nilai yaitu true bernilai 1 dan false bernilai 0                 |
| String    | Terdiri dari karakter/kalimat berupa huruf, angka, dll (diapit tanda “ atau ‘) |
| Integer   | Menyatakan bilangan bulat  |
| Float     | Menyatakan bilangan yang mempunyai koma  |
| Complex   | Menyatakan pasangan angka real dan imajiner                                    |
| List      | Data untaian yang menyimpan berbagai tipe data, isinya dapat berubah-ubah      |
| Tuple     | Data untaian yang menyimpan berbagai tipe data, tapi isinya tidak              |

|             |   |
|-------------|---|
|             | dapat berubah-ubah  |
| Hexadecimal | Menyatakan bilangan dalam format heksa  |
| Dictionary  | Data untaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai |

Contoh tipe data pada Python

#tipe data Boolean

```
print(True)
```

Hasilnya: True

#tipe data String

```
print("Belajar Python menyenangkan...")
```

Hasilnya: Belajar Python menyenangkan...

#tipe data Integer

```
print(20)
```

Hasilnya: 20

#tipe data Float

```
print(3.14)
```

Hasilnya: 3.14

#tipe data Complex

```
print(5j)
```

Hasilnya 5j

A. Tipe data list adalah array yang berisi kumpulan tipe yang tidak sejenis.

Contoh tipe data list:

#tipe data list

```
kata = ["Belajar", "Python", "di", "Kampus UBSI"]
```

```
angka = [10, 50, 100, 1000]
```

```
campur = ["Belajar", 100, 7.99, True]
```

#cetak

```
print(kata)
```

```
print(angka)
```

```
print(campur)
```

Hasil Running:

```
['Belajar', 'Python', 'di', 'Kampus UBSI']
```

```
[10, 50, 100, 1000]
```

```
['Belajar', 100, 7.99, True]
```

B. Tipe Data Tuple hampir sama dengan list, perbedaanya anggotanya tidak bisa diubah setelah dideklarasikan. Tuple menggunakan kurung biasa dan dipisahkan dengan koma untuk anggota

Contoh tipe data tuple:

#tipe data tuple

```
kata = ("Belajar", "Python", "di", "Kampus UBSI")
```

```
angka = (10, 50, 100, 1000)
```

```
campur = ("Belajar", 100, 7.99, True)
```



```
#cetak
print(kata)
print(angka)
print(campur)
```

Hasil Running:

```
('Belajar', 'Python', 'di', 'Kampus UBSI')
(10, 50, 100, 1000)
('Belajar', 100, 7.99, True)
```

C. Tipe data dictionary

Bentuk umum tipe data dictionary pada pemrograman python:

```
Nama_variabel = {"key1": "value1", "key2": "value2", "key3": "value3" }
```

Contoh tipe data dictionary:

```
#Tipe data dictionary
```

```
data = {1:"Belajar",
        2: ["C++", "Python"],
        "Di Kampus": "UBSI",
        "menyerah": False,
        "Tahun": 2021}
```

```
print(data)
```

Hasil Running:

```
{1: 'Belajar', 2: ['C++', 'Python'], 'Di Kampus': 'UBSI', 'menyerah': False, 'Tahun': 2021}
```

Tabel 2. Operator Aritmatika Pada Python

| Operator | Keterangan                                  |
|----------|---|
| +        | Penjumlahan                                 |
| -        | Pengurangan                                 |
| *        | Perkalian                                   |
| /        | Pembagian                                   |
| //       | Pembagian bulat, hasil pembagian tanpa koma |
| %        | Modulus (sisanya)                           |
| **       | pemangkatan                                 |

Contoh Operator Aritmatika

```
>>> 1+2
3
>>> 8-12
-4
>>> 4*5
20
>>> 42/7
6.0
>>> 9%2
1
>>> 5**2
25
```

Tabel 3. Operator Perbandingan

| Operator | Keterangan              |
|----------|-------------------------|
| >        | Lebih besar dari        |
| <        | Lebih kecil dari        |
| ==       | Sama dengan             |
| !=       | Tidak sama dengan       |
| <=       | Lebih kecil sama dengan |
| >=       | Lebih besar sama dengan |

Contoh:

```
>>> 10>5
True
>>> 8<6
False
>>> 10==10
True
>>> 5!=6
True
>>> 6<=6
True
>>> 8>=3
True
```

#### D. Menggabungkan Nilai String

Pada Pemrograman Python untuk Untuk menggabungkan nilai string pada program adalah sebagai berikut:

```
#Penggabungan dua string
kata1 = "Belajar Bahasa Pemrograman Python "
kata2 = "Sangat Menyenangkan"
print("Kata1: ",kata1)
Print("Kata2: ",kata2)
#kata pertama dan kedua digabungkan
gabung = kata1 + kata2
print("Hasil Penggabungan kata1 dan kata2")
print(gabung)
```

Hasil Running:

Belajar Bahasa Pemrograman Python Sangat Menyenangkan

#### Latihan

Tentukan apa hasil numerik dari ekspresi relasi dan logika dibawah ini. Diberikan nilai A = 10; B = 5 ; C = 3 ; K = 8; L = 6; M = 2

1.  $D = (4 + 2 > A \ \& \ B - 2 > 3 + 2 \mid B + 2 \leq 6 + 2)$
2.  $D = K + 5 < M \mid (C * M < L \ \& \ 2 * M - L > 0)$
3.  $D = L + 5 < M \mid C * K < L \ \& \ 2 * K - L > 0$
4.  $D = A * 4 \leq 3 * M + B$
5.  $D = K + 10 > A \ \& \ L - 2 > 4 * C$

## Minggu Ke- 3

### Flowchart

#### Deskripsi

Membahas tentang pembuatan flowchart dari algoritma dan pembuatan program dari flowchart yang ada


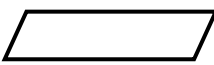

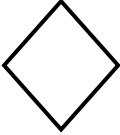
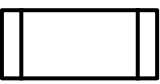

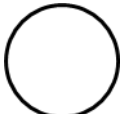
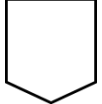
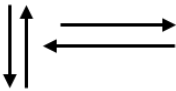
#### Tujuan Pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Membuat flowchart dari algoritma
2. Membuat program berdasarkan flowchart

Diagram Alur adalah suatu diagram yang menggambarkan susunan logika suatu program.

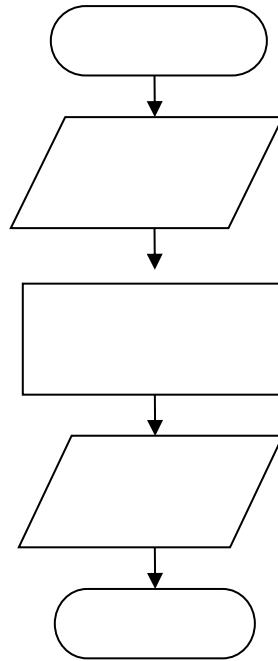
Tabel 1. Simbol-simbol Flowchart, sebagai berikut:

| Simbol  | Keterangan   |
|---|--|
|    | Proses/prosesing, satu atau beberapa himpunan penugasan yang akan dilaksanakan secara berurutan.   |
|    | Input / Output data yg akan dibaca & dimasukkan ke dalam memori komputer dari suatu alat input   |
|  | Terminal, berfungsi sebagai awal (berisi 'Start') sebagai akhir (berisi 'End') dari suatu proses alur.   |
|  | Decision (kotak keputusan) berfungsi utk memutuskan arah/percabangan yg diambil sesuai dgn kondisi yg dipenuhi, yaitu Benar/Salah. (dibahas dalam struktur branching). |
|  | Subroutine digunakan untuk menjalankan proses suatu bagian (sub program) atau prosedur.  |
|  | Preparation digunakan untuk pemberian harga awal   |
|  | On page Connector/penghubung, digunakan untuk menghubungkan diagram alur yang terputus dimana bagian tersebut masih berada pada halaman yang sama                      |
|  | Off page Connector, Untuk menghubungkan sambungan dari bagian flowchart yang terputus dimana sambungannya berada pada halaman lain                                     |
|  | Flowline, menunjukkan bagian arah instruksi dijalankan   |

Flowchart terdiri dari tiga struktur

1. Struktur *Sequence* / Struktur Sederhana  
Digunakan untuk program yang instruksinya sequential atau urutan.

Bentuk Flowchartnya:

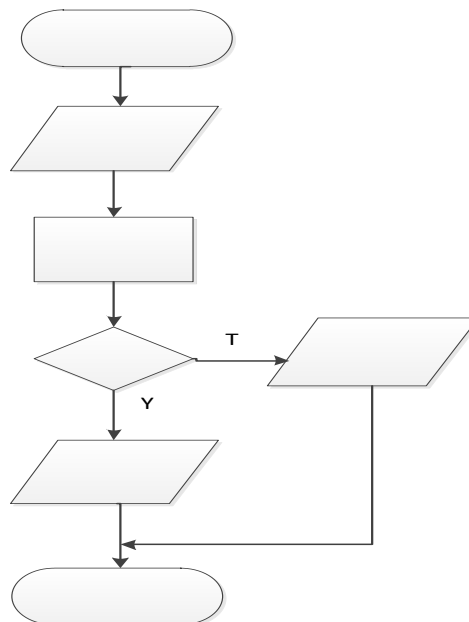


Gambar 1. Flowchart Struktur Sequence/Sederhana

## 2. Struktur Branching

Digunakan untuk program yang menggunakan pemilihan atau penyeleksian kondisi.

Bentuk Flowchartnya

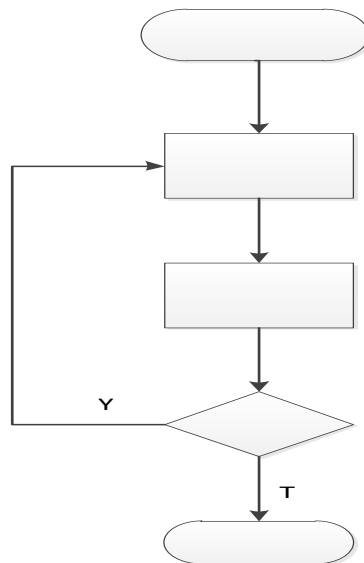


Gambar 2. Flowchart Struktur Branching

## 3. Stuktur Looping

Digunakan untuk program yang instruksinya akan dieksekusi berulang-ulang.

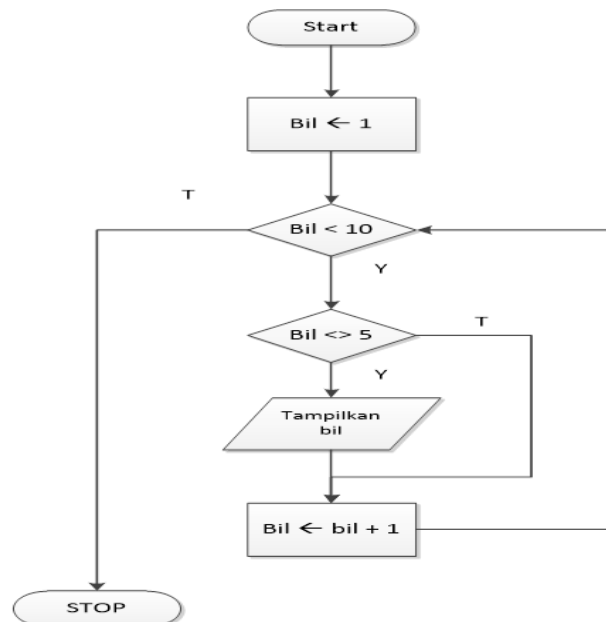
Bentuk Flowchartnya:



Gambar 3. Flowchart Struktur Looping

### Latihan

1. Diketahui empat bilangan 20, 60, 40, dan 100. Buatlah flowchart/diagram alir untuk mendapatkan nilai terbesar diantara keempat bilangan tersebut
2. Buatlah flowchart/diagram alir untuk mengitung jumlah suku pada deret angka berikut:
  - a.  $S = 1 + 3 + 5 + 7 + 9 + 11$
  - b.  $S = 2 + 5 + 10 + 17 + 26 + 37$
3. Buatlah algoritma dari flowchart ini:



**Hasil Praktek : Flowchart bilangan terbesar, faktorial dan algoritma**

## Minggu Ke- 4

### Struktur Branching

#### Deskripsi

Membahas tentang konsep struktur branching bersyarat if, if else, if elif else dan nested if

#### Tujuan pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan struktur percabangan if
2. Menjelaskan struktur percabangan if else
3. Menjelaskan struktur percabangan if elif else
4. Struktur Percabangan nested if

Struktur Percabangan dalam pemrograman python, yaitu:

1. Struktur Percabangan if adalah struktur percabangan yang digunakan untuk satu pilihan keputusan. Jika kondisi True/benar maka statement dikerjakan, Jika kondisi False/salah maka statement tidak dikerjakan.

Tabel 1. Percabangan If

| Bentuk Umum                              | Flowchartnya  |
|--|---|
| <pre>if kondisi:<br/>    statement</pre> | <pre>graph TD; Entry(( )) --&gt; Kondisi{Kondisi}; Kondisi -- True --&gt; Statement[Statement]; Kondisi -- False --&gt; Exit(( )); Statement --&gt; Exit;</pre> |

2. Struktur Percabangan if ... else adalah percabangan yang akan menyeleksi kondisi jika bernilai True/benar maka statement1 dijalankan, jika kondisi bernilai False/salah maka statement2 dijalankan.

Tabel 2. Percabangan If else

| Bentuk Umum  | Flowchartnya  |
|--|---|
| <pre> if kondisi:     statement1 else:     statement2                     </pre> | <pre> graph TD     Entry(( )) --&gt; Kondisi{Kondisi}     Kondisi -- True --&gt; S1[Statement 1]     Kondisi -- False --&gt; S2[Statement 2]     S1 --&gt; Exit(( ))     S2 --&gt; Exit     Exit --&gt; Exit                     </pre> |

3. Struktur Percabangan if ... elif ... else  
 Digunakan untuk menguji lebih dari 2 kondisi, bila kondisi1 benar maka statement1 dikerjakan, bila salah menuju ke kondisi2 . Bila kondisi2 benar maka statement2 dikerjakan, jika salah maka statemen3 dikerjakan.

Tabel 3. Percabangan if elif else

| Bentuk Umum   | Flowchartnya   |
|---|--|
| <pre> if kondisi1:     statement1 elif kondisi2:     statement2 else:     statement3                     </pre> | <pre> graph TD     Entry(( )) --&gt; K1{Kondisi1}     K1 -- True --&gt; S1[Statement 1]     K1 -- False --&gt; K2{Kondisi2}     K2 -- True --&gt; S2[Statement 2]     K2 -- False --&gt; S3[Statement 3]     S1 --&gt; Exit(( ))     S2 --&gt; Exit     S3 --&gt; Exit     Exit --&gt; Exit                     </pre> |

4. Struktur Percabangan nested if adalah suatu kondisi if didalam kondisi if.

Tabel 4. Percabangan Nested if

| Bentuk Umum   | Flowchartnya   |
|---|--|
| <pre> if kondisi1:     if kondisi 1.1:         statement 1.1     elif kondisi 1.2:         statement 1.2     else:         statement 1.3 elif kondisi2:     if kondisi 2.1:         statement 2.1     elif kondisi 2.2:         statement 2.2     else:         statement 2.3 else:     statement3         </pre> | <pre> graph TD     Start(( )) --&gt; K1{Kondisi 1}     K1 -- True --&gt; K11{Kondisi 1.1}     K1 -- False --&gt; K2{Kondisi 2}     K11 -- True --&gt; S11[Statement 1.1]     K11 -- False --&gt; K12{Kondisi 1.2}     K12 -- True --&gt; S12[Statement 1.2]     K12 -- False --&gt; S13[Statement 1.3]     K2 -- True --&gt; K21{Kondisi 2.1}     K2 -- False --&gt; S3[Statement 3]     K21 -- True --&gt; S21[Statement 2.1]     K21 -- False --&gt; K22{Kondisi 2.2}     K22 -- True --&gt; S22[Statement 2.2]     K22 -- False --&gt; S23[Statement 2.3]     S11 --&gt; Join(( ))     S12 --&gt; Join     S13 --&gt; Join     S21 --&gt; Join     S22 --&gt; Join     S23 --&gt; Join     S3 --&gt; Join     Join --&gt; End(( ))         </pre> |

### Kasus:

Buatlah sebuah rancangan program transaksi penjualan dengan menggunakan flowchart, dengan ketentuan sebagai berikut :

- Baca input yang terdiri dari nama pembeli, kode paket dan jumlah beli
- Proses perhitungan transaksi sebagai berikut :
  - Gunakan fungsi percabangan untuk menentukan nama dan harga  
 Jika kode SP31 maka nama paket SmartPhone Murah seri 31 , harga 1000000  
 Jika kode SP32 maka nama paket SmartPhone seri 32 , harga 1800000  
 Jika kode SP33 maka nama paket SmartPhone seri 33 , harga 2500000
  - Gunakan fungsi percabangan untuk menentukan Bonus  
 Jika jumlah bayar melebihi 3000000 maka bonusnya 1 buah batre cadangan  
 Jika jumlah bayar melebihi 6000000 maka bonusnya 1 buah charge cadangan  
 Selain itu tidak mendapatkan bonus apapun
  - Gunakan Fungsi Looping/perulangan untuk mengulang transaksi.
- Buat output dari perhitungan transaksi penjualan sebagai berikut
  - cetak judul struk LuckyShell Distributor SmartPhone
  - cetak Jln. Masjid Attaufik



- cetak Struk Pembayaran
- cetak Nama Pembeli
- cetak Nama Paket
- cetak Jumlah Beli
- cetak Harga Satuan
- cetak Total Bayar
- cetak Potongan
- cetak Jumlah Bayar
- cetak Uang Bayar
- cetak Kembali
- cetak Bonus

**Hasil Praktek : Flowchart dan program transaksi penjualan**

## Minggu Ke- 5

### Looping

#### Deskripsi

Membahas tentang instruksi looping (perulangan) yaitu statement for, while dan nested while

#### Tujuan pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan perulangan for
2. Menjelaskan perulangan while
3. Menjelaskan perulangan nested while/for

Pengertian Looping:

Instruksi pengulangan (repetition) adalah instruksi yang dapat mengulangi melaksanakan sederetan instruksi lain berulang kali sesuai dengan persyaratan yang ditentukan.

Struktur instruksi perulangan pada dasarnya terdiri atas:

1. Kondisi perulangan. Suatu kondisi yang harus dipenuhi agar perulangan dapat terjadi
2. Badan (*body*) perulangan. Deretan instruksi yang akan diulang-ulang pelaksanaannya
3. Pencacah (*counter*) perulangan. Suatu variabel yang nilainya harus berubah agar perulangan dapat terjadi dan pada akhirnya membatasi jumlah perulangan yang dapat dilaksanakan

Bentuk Perulangan pada Python, yaitu:

1. Perulangan for: Perulangan yang mengerjakan “bagian pernyataan yang sama” secara berulang-ulang berdasarkan syarat atau kondisi yang ditentukan.

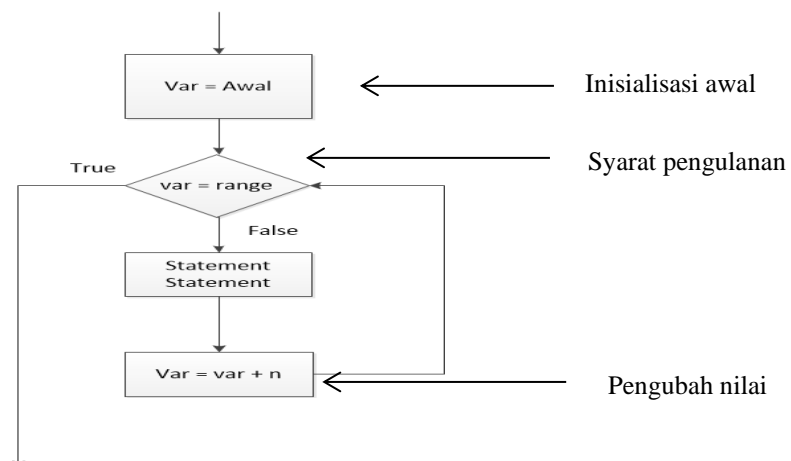
Bentuk Umum:

For variabel in range :

Statements

Variabel adalah sebagai nilai awal. Fungsi range() sebagai counter pada perulangan for.

Flowchart Perulangan For



Gambar 1. Flowchart Perulangan for

Contoh Perulangan for untuk mencetak bilangan 1 sampai 5

```
for i in range(5):  
    print(i+1)
```

Hasil Running:

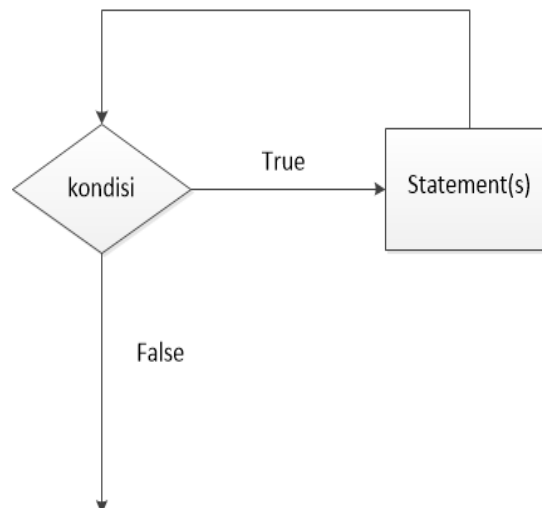
1  
2  
3  
4  
5

2. Perulangan While: Perulangan yang mengerjakan perintah selama kondisinya bernilai benar.

Bentuk Umum:

```
while kondisi:  
    statement(s)
```

Flowchart perulangan while:



Gambar 2. Perulangan While

Penjelasan perulangan while:

- 1) Ada instruksi yang berkaitan dengan kondisi sebelum masuk ke while sehingga kondisi ini benar (terpenuhi) dan pengulangan bisa dilaksanakan.
- 2) Ada suatu instruksi di antara instruksi-instruksi yang diulang yang mengubah nilai variabel perulangan agar pada saat kondisi perulangan tidak terpenuhi sehingga perulangan berhenti.

Contoh algoritma perulangan while untuk mencetak bilangan 1 sampai 15

Algoritma Perulangan\_while

{mencetak angka 1 hingga 15}

Deklarasi

angka =1

Deskripsi

```
while angka <= 15:
```

```
    cetak angka
```

```
    angka ← angka + 1
```

Program mencetak angka 1 sampai 15:

```
# Perulangan While
```

```
angka = 1
```

```
while angka <= 15:
```

```
    print ("Bilangan ke-", angka)
```

```
    angka = angka + 1
```

```
print ("Terima Kasih")
```

Hasil Running:

Bilangan ke-: 1

Bilangan ke-: 2

Bilangan ke-: 3

Bilangan ke-: 4

Bilangan ke-: 5

Bilangan ke-: 6

Bilangan ke-: 7

Bilangan ke-: 8

Bilangan ke-: 9

Bilangan ke-: 10

Bilangan ke-: 11

Bilangan ke-: 12

Bilangan ke-: 13

Bilangan ke-: 14

Bilangan ke-: 15

Terima Kasih

Contoh algoritma while\_mencetak angka menurun

```
#mencetak angka 10 sampai angka 1
```

```
Deklarasi
```

```
bil = 10
```

```
Deskripsi
```

```
while bil > 0:
```

```
    cetak bil
```

```
    bil = bil - 1
```

Program mencetak angka 10 sampai 1

```
#Perulangan While
```

```
#Mencetak bilangan 10 sampai 1
```

```
bil = 10
```

```
while bil > 0:
```

```
    print(bil)
```

```
    bil = bil - 1
```

```
print ("Hasil Mencetak Bilangan Secara menurun")
```

**Hasil Running:**

10

9

8

7

6

5  
4  
3  
2  
1

Hasil Mencetak Bilangan Secara menurun

Didalam perulangan ada perintah yang dapat melakukan kontrol terhadap perulangan yaitu break dan continue.

- 1) Perintah Break berfungsi untuk keluar dari suatu loop for atau while.

Bentuk Umum:

```
.....  
.....  
break  
.....  
.....
```

Contoh program akan keluar setelah mencetak sampai angka 6

#Perintah break pada perulangan for

#Program akan keluar setelah mencetak angka sampai 6 karena perintah break

bil = 6

```
for i in range(0,10):
```

```
    print(i)
```

```
    if i is bil:
```

```
        break
```

Hasil Running:

0  
1  
2  
3  
4  
5  
6

Program keluar setelah mencetak angka 6 karena instruksi break

**Note:**

Looping akan dikerjakan terus sampai dipaksa keluar oleh instruksi **break**;

- 2) Perintah Continue berfungsi untuk melakukan pengulangan mulai dari awal lagi

Penerapan perintah continue pada Python

#penggunaan continue pada while

bil = 0

pilihan = 'y'

```
while (pilihan != 'n'):
```

```
    bil = int(input("Masukkan bilangan dibawah 50: "))
```

```
    if (bil > 50):
```

```
        print("Bilangan melebihi angka 50, Silahkan diulangi.")
```

```
        continue
```

```
    print("Pangkat dua dari bilangan ini adalah: ",bil*bil)
```

```
    pilihan = input("Apakah Anda ingin mengulang kembali (y/n)? ")
```

Hasil Running:

Masukkan bilangan dibawah 50: 20  
Pangkat dua dari bilangan ini adalah: 400  
Apakah Anda ingin mengulang kembali (y/n)? y  
Masukkan bilangan dibawah 50: 36  
Pangkat dua dari bilangan ini adalah: 1296  
Apakah Anda ingin mengulang kembali (y/n)? y  
Masukkan bilangan dibawah 50: 70  
Bilangan melebihi angka 50, Silahkan diulangi.  
Masukkan bilangan dibawah 50: 25  
Pangkat dua dari bilangan ini adalah: 625  
Apakah Anda ingin mengulang kembali (y/n)? n

3. Loop bersarang (Nested loop): Perulangan di dalam perulangan  
Bentuk Umum nested while:

```
While kondisi:
    while kondisi:
        statement(s)
    statement(s)
```

Contoh Nested while mencetak bilangan prima antara 1 sampai 50

```
i = 2
while(i < 50):
    j = 2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print (i, "adalah Bilangan Prima")
    i = i + 1
```

Hasil Running:

2 adalah Bilangan Prima  
3 adalah Bilangan Prima  
5 adalah Bilangan Prima  
7 adalah Bilangan Prima  
11 adalah Bilangan Prima  
13 adalah Bilangan Prima  
17 adalah Bilangan Prima  
19 adalah Bilangan Prima  
23 adalah Bilangan Prima  
29 adalah Bilangan Prima  
31 adalah Bilangan Prima  
37 adalah Bilangan Prima  
41 adalah Bilangan Prima  
43 adalah Bilangan Prima  
47 adalah Bilangan Prima

### Kasus:

1. Buatlah program untuk menampilkan baris bilangan sesuai perulangannya sebagai berikut:

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ADMIN\AppData\Local\Programs\Python\Python37-32\Modul Python\nested for1.py
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7 7
8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9
10 10 10 10 10 10 10 10 10 10
>>> |
```

2. Buatlah program untuk menampilkan simbol asterisk sesuai perulangannya sebagai berikut:

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ADMIN\AppData\Local\Programs\Python\Python37-32\Modul Python\nested for1.py
*
* *
* * *
* * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
>>> |
```

3. Buatlah program untuk menampilkan simbol asterisk sesuai perulangannya sebagai berikut:

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ADMIN\AppData\Local\Programs\Python\Python37-32\Modul Python\nested while.py
1
3 3
5 5 5
7 7 7 7
9 9 9 9 9
>>> |
```

**Hasil Praktek : Memahami konsep perulangan pada Pemrograman Python**

## Minggu Ke- 6

### Rekursif

#### Deskripsi

Membahas tentang struktur rekursif seperti fungsi pangkat, faktorial, fibonancy dan menara hanoi

#### Tujuan Pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan teknik rekursif untuk fungsi pangkat
2. Menjelaskan teknik rekursif untuk faktorial
3. Menjelaskan teknik rekursif untuk fibonancy
4. Menjelaskan teknik rekursif untuk menara hanoi

Rekursif adalah suatu proses yang bisa memanggil dirinya sendiri.

Contoh konsep penggunaan Rekursif

Masalah : Memotong Roti tawar tipis-tipis sampai habis

Algoritma :

1. Jika roti sudah habis atau potongannya sudah paling tipis maka pemotongan roti selesai.
2. Jika roti masih bisa dipotong, potong tipis dari tepi roti tersebut, lalu lakukan prosedur 1 dan 2 untuk sisa potongannya.

Contoh Fungsi Rekrusif, antara lain:

1. Fungsi pangkat

Menghitung 10 pangkat n dengan menggunakan konsep rekursif.

Secara Notasi pemrograman dapat ditulis :

$$10^0 = 1 \quad \dots\dots\dots(1)$$

$$10^n = 10 * 10^{n-1} \quad \dots\dots\dots(2)$$

Contoh :

$$10^3 = 10 * 10^2$$

$$10^2 = 10 * 10^1$$

$$10^1 = 10 * 10^0$$

$$10^0 = 1$$

2. Faktorial

$$0! = 1$$

$$N! = N * (N-1)! \quad \text{Untuk } N > 0$$

Scr notasi pemrograman dapat ditulis sebagai :

$$\text{FAKT}(0) = 1 \quad \dots\dots\dots(1)$$

$$\text{FAKT}(N) = N * \text{FAKT}(N-1) \dots\dots\dots(2)$$

Contoh :

$$\text{FAKT}(5) = 5 * \text{FAKT}(4)$$

$$\text{FAKT}(4) = 4 * \text{FAKT}(3)$$

$$\text{FAKT}(3) = 3 * \text{FAKT}(2)$$

$$\text{FAKT}(2) = 2 * \text{FAKT}(1)$$

$$\text{FAKT}(1) = 1 * \text{FAKT}(0)$$

**Nilai Awal**



Contoh program faktorial

#Listing Program Faktorial

```
a = input('masukkan bilangan bulat: ')
def faktorial(x):
    if x==1:
        return 1
    elif x==0:
        return 1
    else:
        return (x * faktorial(x-1))
bil = int(a)
print('faktorial %s' %bil, 'adalah %s' % faktorial(bil))
```

**Hasil Running:**

masukkan bilangan bulat: 5  
faktorial 5 adalah 120  
masukkan bilangan bulat: 6  
faktorial 6 adalah 720  
masukkan bilangan bulat: 7  
faktorial 7 adalah 5040

3. Fibonancy

Deret Fibonancy : 0,1,1,2,3,5,8,13,.....

Secara notasi pemrograman dapat ditulis sebagai :

Fibo (1) = 0            & Fibo (2) = 1            ..... (1)

Fibo (N) = Fibo (N-1) + Fibo (N-2) ..... (2)

Contoh :

Fibo(5) = Fibo(4) + Fibo(3)

    Fibo(4) = Fibo(3) + Fibo(2)

        Fibo(3) = Fibo(2) + Fibo(1)

                  └───┘  
                  Nilai Awal

Contoh deret fibonancy

#Program Deret Fibonancy

fibonacci = int(input("Masukkan jumlah Deretnya: "))

#fibonacci1, fibonacci2

n1, n2 = 0, 1

count = 0

#cek fibonacci

if fibonacci <= 0:

    print("Silakan Masukkan bilangan Positif")

elif fibonacci == 1:

    print("Deret Urut Fibonacci ", fibonacci, ":")

    print(n1)

else:

    print("Deret urut Fibonacci :")

    while count < fibonacci:

        print(n1)

```
nth = n1 + n2
#update values
n1 = n2
n2 = nth
count +=1
```

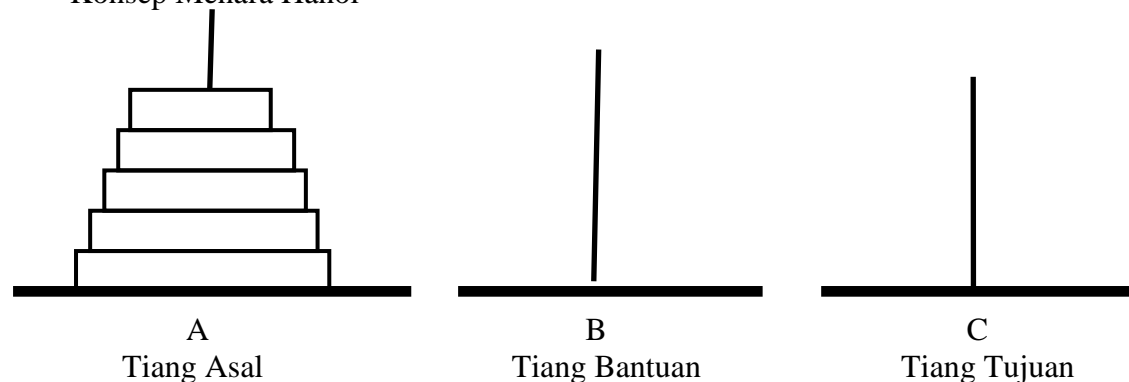
**Hasil Running:**

Masukkan jumlah Deretnya: 8

Deret urut Fibonacci :

0  
1  
1  
2  
3  
5  
8  
13

4. Menara Hanoi  
Konsep Menara Hanoi



**Penjelasannya:**

- Jika  $n=1$ , maka langsung pindahkan saja piringan dr tiang A ke tiang C & selesai.
- Pindahkan  $n-1$  piringan yg paling atas dr tiang A ke tiang B.
- Pindahkan piringan ke  $n$  (piringan terakhir) dr tiang A ketiang C
- Pindahkan  $n-1$  piringan dari tiang B ke tiang C.

**Kasus:**

- Buatlah algoritma dan program untuk Faktorial
- Buatlah algoritma dan program untuk pemangkatan

**Hasil Praktek : Memahami konsep dan logika rekursif dengan penggunaan bahasa Python**

## **Minggu Ke- 7**

### **QUIZ**

#### **Deskripsi**

Membahas tentang review materi dalam quiz

#### **Tujuan Pembelajaran**

Setelah melakukan bagian ini mahasiswa mampu menjelaskan materi dari pertemuan 1 sampai pertemuan 6

**Minggu Ke- 8**

**UTS**

# UJIAN TENGAH SEMESTER

## Minggu Ke- 9

### List Dan Matriks

#### Deskripsi

Membahas tentang list dan matriks

#### Tujuan Pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan List
2. Menjelaskan Matriks

List merupakan variabel yang menyimpan lebih dari satu data didalamnya. List menyerupai array dalam pemrograman lainnya (Jud, 2017).

List dapat menyimpan banyak data dalam 1(satu) variable dan diakses menggunakan indeks.

Inisialisasi List menggunakan kurung siku [ dan ]

List dapat menyimpan data seragam maupun tidak seragam. (jenis tipe data seperti string, integer, float, double, Boolean, object dll)

#### Contoh:

|   |   |                     |     |
|---|---|---------------------|-----|
| 1 | 2 | Sistem<br>Informasi | 1.2 |
| 0 | 1 | 3                   | 4   |

Nomor index list selalu dimulai dari nol (0). Penggunaan nomor index adalah untuk mengambil isi(item) dari list

NamaVar=[val1,val2,val3,...]

data=[1, 2, "Sistem Informasi", 1.2]

#### Contoh Program List dengan tipe data yang berbeda

```
data=[1,2,"Sistem Informasi","Komputer Akuntansi",1.2,0.3]
print(data[:2])
print(data[:4])
print(data[len(data)-3:])
```

#### Hasil program:

```
[1, 2]
[1, 2, 'Sistem Informasi', 'Komputer Akuntansi']
['Komputer Akuntansi', 1.2, 0.3]
```

#### Mengindex LIST Pada Python

Tabel 1. Index List Pada Python

| Metode   | Deskripsi                           |
|----------|-------------------------------------|
| append() | Menambah elemen pada index terakhir |
| clear()  | Mengosongkan list                   |
| len()    | Menghitung panjang list             |

|          |   |
|----------|---|
| del      | Menghapus list atau element               |
| insert() | Menambah element pada index tertentu      |
| pop()    | Menghapus element pada index tertentu     |
| remove() | Menghapus 1 element dengan nilai tertentu |

Contoh Penggunaan mengindex LIST Pada Python

append(item) menambahkan item dari belakang

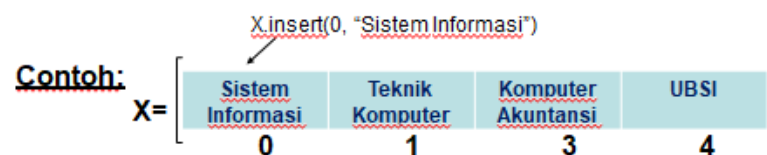


```
#list append
A = ([ 1, 2, "Sistem Informasi"])
#Tambahkan UBSI
A.append("UBSI")
print(A)
```

Hasil Program:

[1, 2, 'Sistem Informasi', 'UBSI']

insert(index, item) menambahkan item dari indeks tertentu

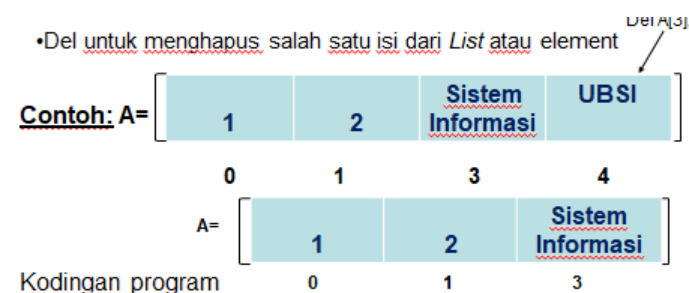


Kodingan program

```
#menambahkan data
X = ["Komputer Akuntansi","Teknik Komputer"]
X.insert(0,"Sistem Informasi")
print(X)
```

Hasil Program

['Sistem Informasi', 'Komputer Akuntansi', 'Teknik Komputer']



```
p = ["sistem informasi", "komputer akuntasni", "Teknik Komputer"]  
del p[1]  
print(p)
```

Hasil Program:

**['sistem informasi', 'Teknik Komputer']**

### OPERASI LIST

Ada beberapa operasi yang bisa dilakukan List, diantaranya:

Penggabungan (+)

Perkalian (\*)

```
# Beberapa list program studi  
list_prodi = [  
    "sistem informasi", "komputer akuntasni", "Teknik Komputer" ]  
# list warna buah  
kode_prodi = [12, 11, 13]  
]  
# Penggabungannya  
semua_prodi = list_prodi + kode_prodi  
print (semua_prodi)
```

### Hasil Program

```
['sistem informasi', 'komputer akuntasni', 'Teknik Komputer', 12, 11, 13]
```

Contoh: Perkalian (\*)

```
# Data nama program studi  
list_prodi = [  
    "Sistem Informasi", "Komputer Akuntansi", "Teknik Komputer"]  
# ulangi sebanyak 3x  
ulangi = 3  
prodi = list_prodi * ulangi  
print (prodi)
```

### Hasil Program

```
['Sistem Informasi', 'Komputer Akuntansi', 'Teknik Komputer', 'Sistem Informasi',  
'Komputer Akuntansi', 'Teknik Komputer', 'Sistem Informasi', 'Komputer Akuntansi',  
'Teknik Komputer']
```

### DIMENSI DALAM LIST

Dalam list mengenal 2 dimensi yaitu:

1. List Satu Dimensi
2. List Dua Dimensi

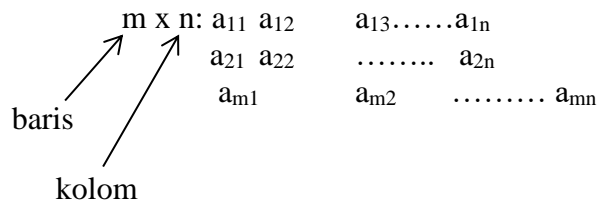
#### List Satu Dimensi

Sebuah variabel menyimpan sekumpulan data yang memiliki tipe data sama atau berbeda dan elemen yang akan diakses hanya melalui 1 indeks atau subskrip





Contoh:



Matrik ordo  $3 \times 3 =$

|   |   |   |
|---|---|---|
| 2 | 1 | 2 |
| 3 | 0 | 1 |
| 2 | 0 | 0 |

Penjelasan matrik  $3 \times 3 =$

|              |              |              |
|--------------|--------------|--------------|
| $a_{11} = 2$ | $a_{21} = 3$ | $a_{31} = 2$ |
| $a_{12} = 1$ | $a_{22} = 0$ | $a_{32} = 0$ |
| $a_{13} = 2$ | $a_{23} = 1$ | $a_{33} = 0$ |

Contoh: Diberikan matriks A sebagai berikut:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perintah pokok yang digunakan pada pengisian matriks A adalah  $A[i,j] = 1$ , , jika  $i \leq j$  ,  $A[i,j] = 0$ , jika  $i > j$

### Kasus:

1. Buat perintah pokok pada pengisian matriks B dibawah ini

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

2. Buatlah program untuk penjumlahkan dua buah matriks dengan ordo  $3 \times 3$  dengan menggunakan pemrograman Python.
3. Buatlah program untuk pengurangan dua buah matriks dengan ordo  $2 \times 2$  dengan menggunakan pemrograman Python.

## Minggu Ke- 10

### Metode Divide & Conquer

#### Deskripsi

Membahas tentang konsep sorting (selection sort, bubble sort, merge sort, quick sort dan insertion)

#### Tujuan pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan metode merge sort
2. Menjelaskan metode quick sort
3. Menjelaskan metode binary search
4. Menjelaskan Teknik D and C

Pengertian divide adalah Memilah data nilai elemen–elemen dari rangkaian data menjadi dua bagian dan mengulangi pemilahan hingga satu elemen terdiri maksimal dua nilai (Sonita & Nurtaneo, 2015).

Pengertian conquer adalah Mengurutkan masing-masing data nilai elemen (Sonita & Nurtaneo, 2015).

Prinsip dasar metode divide dan conquer adalah:

- a) Membagi  $n$  input menjadi  $k$  subset input yang berbeda ( $1 < k \leq n$ ).
- b)  $k$  subset input tersebut akan terdapat  $k$  subproblem.
- c) Setiap subproblem mempunyai solusi menjadi  $k$  subsolusi.
- d) dari  $k$  subsolusi akan mendapatkan solusi yang optimal

Pengertian Sorting:

Proses pengaturan sederetan data ke dalam suatu urutan atau susunan urutan tertentu. Data yang diurutkan dapat berupa data bilangan, data karakter maupun data string (Sitorus, 2015).

Teknik D and C

Menggunakan teknik Rekursif yang membagi masalah menjadi dua atau lebih submasalah dengan ukuran yang sama. Masalah umum untuk teknik ini seperti pengurutan, perkalian.

Metode D AND C, diantaranya:

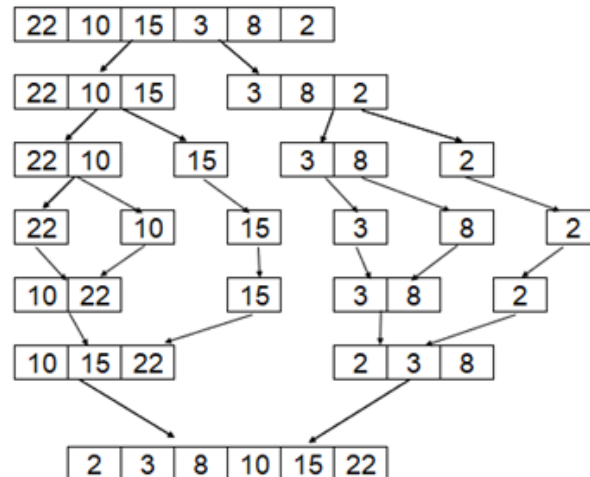
1. Merge Sort
2. Quick Sort
3. Binary Search
4. Teknik D and C

1. Merge Sort Menggabungkan dua array yang sudah terurut (Utami, 2017)

Prinsip kerjanya:

- 1) Kelompokkan deret bilangan kedalam 2 bagian, 4 bagian, 8 bagian, .....dst  
→  $(2^n)$
- 2) Urutkan secara langsung bilangan dalam kelompok tersebut.
- 3) Lakukan langkah diatas untuk kondisi bilangan yang lain sampai didapatkan urutan yang optimal.

Contoh :



Contoh Program Merget Sort:

```

def mergeSort(X):
    print("Bilangan diurutkan ",X)
    if len(X)>1:
        mid = len(X)//2
        lefthalf = X[:mid]
        righthalf = X[mid:]
        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=j=k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                X[k]=lefthalf[i]
                i=i+1
            else:
                X[k]=righthalf[j]
                j=j+1
            k=k+1
        while i < len(lefthalf):
            X[k]=lefthalf[i]
            i=i+1
            k=k+1
        while j < len(righthalf):
            X[k]=righthalf[j]
            j=j+1
            k=k+1
        print("Merging ",X)
  
```

```
X = [22,10,15,3,8,2]
mergeSort(X)
print(X)
```

Hasilnya:

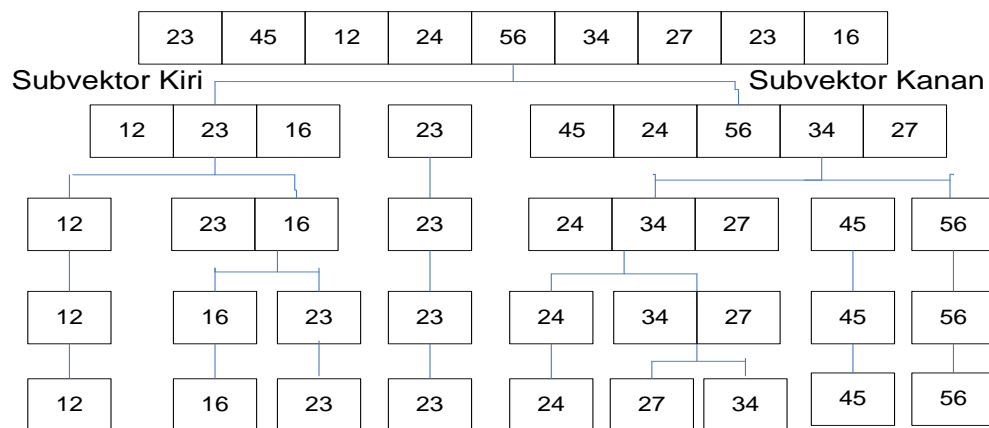
```
Bilangan diurutkan [22, 10, 15, 3, 8, 2]
Bilangan diurutkan [22, 10, 15]
Bilangan diurutkan [22]
Bilangan diurutkan [10, 15]
Bilangan diurutkan [10]
Bilangan diurutkan [15]
Merging [10, 15]
Bilangan diurutkan [3, 8, 2]
Bilangan diurutkan [3]
Bilangan diurutkan [8, 2]
Bilangan diurutkan [8]
Bilangan diurutkan [2]
Merging [2, 3, 8]
[2, 3, 8, 10, 15, 22]
```

2. Quick Sort Merupakan metode sort tercepat, Quicksort diperkenalkan oleh C.A.R. Hoare. Quicksort partition exchange sort, karena konsepnya membuat bagian-bagian, dan sort dilakukan perbagian. Pada algoritma quick sort, pemilihan pivot merupakan hal yang menentukan apakah algoritma quicksort tersebut akan memberikan performa terbaik atau terburuk (Nugraheny, 2018).

Misal ada N elemen dalam keadaan urut turun, adalah mungkin untuk mengurutkan N elemen tersebut dengan  $N/2$  kali, yakni pertama kali menukarkan elemen paling kiri dengan paling kanan, kemudian secara bertahap menuju ke elemen yang ada di tengah. Tetapi hal ini hanya bisa dilakukan jika tahu pasti bahwa urutannya adalah urut turun.

Secara garis besar metode ini dijelaskan sebagai berikut, misal: akan mengurutkan vektor A yang mempunyai N elemen. Pilih sembarang dari vektor tersebut, biasanya elemen pertama misalnya X. Kemudian semua elemen tersebut disusun dengan menempatkan X pada posisi J sedemikian rupa sehingga elemen ke 1 sampai ke  $j-1$  mempunyai nilai lebih kecil dari X dan elemen ke  $J+1$  sampai ke N mempunyai nilai lebih besar dari X. Dengan demikian mempunyai dua buah subvektor, subvektor pertama nilai elemennya lebih kecil dari X, subvektor kedua nilai elemennya lebih besar dari X. Pada langkah berikutnya, proses diatas diulang pada kedua subvektor, sehingga akan mempunyai empat subvektor. Proses diatas diulang pada setiap subvektor sehingga seluruh vektor semua elemennya menjadi terurutkan.

Contoh:



Gambar 1. Ilustrasi Metode Quick Sort

## Binary Search

### Binary Search (Untuk data yang sudah terurut)

Digunakan mencari sebuah data pada himpunan data-data yang tersusun secara urut, yaitu data yang telah diurutkan dari besar ke kecil/sebaliknya. Proses dilaksanakan pertama kali pada bagian tengah dari elemen himpunan, jika data yang dicari ternyata  $<$  elemen bagian atasnya, maka pencarian dilakukan dari bagian tengah ke bawah

### Algoritma Binary Search

1. Low = 1 , High = N
2. Ketika Low  $\leq$  High Maka kerjakan langkah No .3, Jika tidak Maka kerjakan langkah No.7
3. Tentukan Nilai Tengah dengan rumus  
(Low + High) Div 2
4. Jika  $X <$  Nilai Tengah, Maka High = Mid -1, Ulangi langkah 1
5. Jika  $X >$  Nilai Tengah, Maka Low = Mid +1, Ulangi langkah 1
6. Jika  $X =$  Nilai Tengah, Maka Nilai Tengah = Nilai yang dicari
7. Jika  $X >$  High Maka Pencarian GAGAL

Contoh:

Data A = { 1, 3, 9, 11, 15, 22, 29, 31, 48 }

Dicari 3

### Langkah Pencariannya:

Langkah 1: Low = 1 dan High = 9

Langkah 2: Low  $\leq$  High (jika YA ke L-3, jika TDK ke L-7)

Langkah 3: Mid = (1+10) div 2 = 5 yaitu 15

Langkah 4: 3 < 15, maka High = 5 - 1 = 4 yaitu 11

Langkah 1: Low = 1 dan High = 4

Langkah 2: Low  $\leq$  High

Langkah 3: Mid = (1+4) div 2 = 2 yaitu 3

Langkah 4: 3 < 3, ke langkah 5

Langkah 5: 3 > 3, ke langkah 6

Langkah 6: 3 = 3 (Pencarian berhasil)

### Teknik D AND C

Dengan Prinsip Dasar Metode Devide & akan dapat dipecahkan suatu permasalahan proses Searching elemen Max&Min dengan teknik D and C

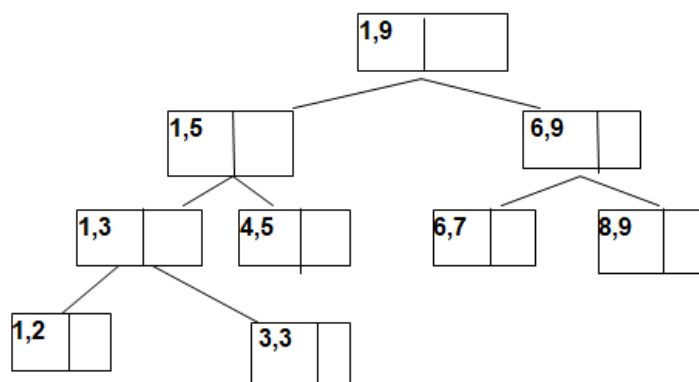
#### Contoh :

Tentukan elemen MaxMin suatu array A yang terdiri dari 9 bilangan :

A[1] = 22,                      A[4] = -8,                      A[7] = 17  
 A[2] = 13,                      A[5] = 15,                      A[8] = 31  
 A[3] = -5,                      A[6] = 60,                      A[9] = 47

#### Penyelesaian Teknik D and C

A = { 22, 13, -5, -8, 15, 60, 17, 31, 47 }

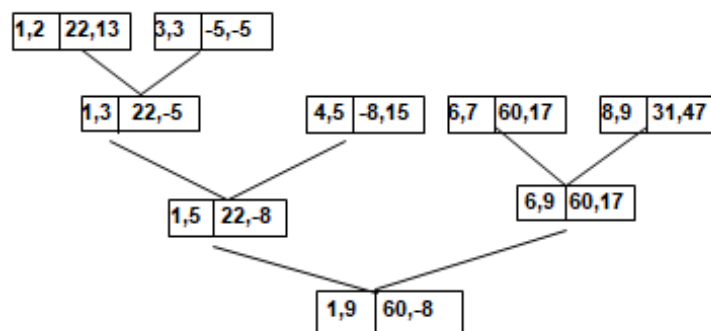


#### Penyelesaian Teknik D AND C

Lalu *Proses tree call* dr setiap elemen yg ditunjuk pada bagan tree tersebut diatas. Dengan cara, membalik terlebih dahulu posisi tree dari bawah ke atas. Lalu mengisinya dengan elemen-elemennya sesuai dengan bagan tree. Perhatikan bagan *tree call* ini :

A = { 22, 13, -5, -8, 15, 60, 17, 31, 47 }

Posisi penggabungan → Max, Min



**Kasus**

Suatu array A terdiri dari 10 elemen yaitu 210, 285, 179, 652, 351, 423, 861, 254, 450 dan 520. Buatlah pengurutannya menggunakan merge sort

**Hasil Praktek: teknik sorting (menggunakan metode merge sort)**

## Minggu Ke- 11

### METODE SORTING

#### Deskripsi

Membahas tentang metode sorting selection sort, bubble sort, dan insertion sort

#### Tujuan pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

Menjelaskan metode sorting dengan selection sort

Menjelaskan metode sorting dengan bubble sort

Menjelaskan metode sorting dengan insertion sort

#### 1. Pengertian Sorting

Proses pengaturan sederetan data ke dalam suatu urutan atau susunan urutan tertentu. Data yang diurutkan dapat berupa data bilangan, data karakter maupun data string (Sitorus, 2015).

#### 2. Macam-Macam Metode Sorting:

1. Selection Sort
2. Bubble Sort
3. Insertion Sort

Hal yang mempengaruhi Kecepatan Algoritma Sorting: Jumlah Operasi Perbandingan & Jumlah Operasi pemindahan Data

1. Selection Sort adalah Teknik pengurutan dengan cara pemilihan elemen atau proses kerja dengan memilih elemen data terkecil untuk kemudian dibandingkan & ditukarkan dengan elemen pada data awal, dst s/d seluruh elemen sehingga menghasilkan pola data yang telah disort.

Prinsip Kerjanya adalah:

- 1) Pengecekan dimulai data ke-1 sampai dengan data ke-n
- 2) Tentukan bilangan dengan Index terkecil dari data bilangan tersebut
- 3) Tukar bilangan dengan Index terkecil tersebut dengan bilangan pertama ( $I = 1$ ) dari data bilangan tersebut
- 4) Lakukan langkah 2 dan 3 untuk bilangan berikutnya ( $I = I + 1$ ) sampai didapatkan urutan yang optimal.

**Contoh :**            **22      10      15      3      8      2**

#### **Iterasi 1:**

|           |   |                        |    |    |   |   |           |
|-----------|---|------------------------|----|----|---|---|-----------|
| Langkah 1 | : | 22                     | 10 | 15 | 3 | 8 | 2         |
| Langkah 2 | : | 22                     | 10 | 15 | 3 | 8 | <b>2</b>  |
| Langkah 3 | : | 2                      | 10 | 15 | 3 | 8 | <b>22</b> |
| Langkah 4 | : | Ulangi langkah 2 dan 3 |    |    |   |   |           |

#### **Iterasi 2**

|           |   |   |    |    |          |   |    |
|-----------|---|---|----|----|----------|---|----|
| Langkah 1 | : | 2 | 10 | 15 | 3        | 8 | 22 |
| Langkah 2 | : | 2 | 10 | 15 | <b>3</b> | 8 | 22 |



Langkah 3 : 2 3 15 10 8 22  
 Langkah 4 : Ulangi langkah 2 dan 3

### **Iterasi 3**

Langkah 1 : 2 3 15 10 8 22  
 Langkah 2 : 2 3 15 10 **8** 22  
 Langkah 3 : 2 3 **8** 10 15 22  
 Langkah 4 : Ulangi langkah 2 dan 3

### **Iterasi 4**

Langkah 1 : 2 3 8 10 15 22  
 Langkah 2 : 2 3 8 **10** 15 22  
 Langkah 3 : 2 3 8 **10** 15 22  
 Langkah 4 : Ulangi langkah 2 dan 3

### **Iterasi 5**

Langkah 1 : 2 3 8 10 15 22  
 Langkah 2 : 2 3 8 10 **15** 22  
 Langkah 3 : 2 3 8 10 **15** 22  
 Langkah 4 : Ulangi langkah 2 dan 3

### **Iterasi 6**

Langkah 1 : 2 3 8 10 15 22  
 Langkah 2 : 2 3 8 10 15 **22**  
 Langkah 3 : 2 3 8 10 15 **22**  
 Langkah 4 : Ulangi langkah 2 dan 3

Contoh Program Selection sort:

```
def SelectionSort(val):
    for i in range(len(val)-1,0,-1):
        Max=0
        for l in range(1,i+1):
            if val[l]>val[Max]:
                Max = l
        temp = val[i]
        val[i] = val[Max]
        val[Max] = temp
```

```
Angka = [22,10,15,3,8,2]
SelectionSort(Angka)
print(Angka)
```

Hasilnya: **[2, 3, 8, 10, 15, 22]**

2. Bubble Sort adalah Metode pengurutan dengan membandingkan data nilai elemen yang sekarang dengan data nilai elemen-elemen berikutnya. Pembandingan elemen dapat dimulai dari awal atau mulai dari paling akhir. Apabila elemen yang sekarang lebih besar (untuk urut menaik) atau lebih kecil (untuk urut menaik) dari elemen berikutnya, maka posisinya ditukar, tapi jika tidak maka posisinya tetap (Harumy et al., 2016).

Prinsip Kerjanya adalah

- 1) Pengecekan mulai dari data ke-1 sampai data ke-n
- 2) Bandingkan data ke-n dengan data sebelumnya (n-1)
- 3) Jika lebih kecil maka pindahkan bilangan tersebut dengan bilangan yang ada didepannya (sebelumnya) satu persatu (n-1,n-2,n-3,...dst)
- 4) Jika lebih besar maka tidak terjadi pemindahan
- 5) Ulangi langkah 2 dan 3 s/d sort optimal.

**Contoh : 22      10      15      3      8      2**

**Iterasi 1:**

|           |   |                        |    |    |   |   |   |
|-----------|---|------------------------|----|----|---|---|---|
| Langkah 1 | : | 22                     | 10 | 15 | 3 | 8 | 2 |
| Langkah 2 | : | 22                     | 10 | 15 | 3 | 8 | 2 |
| Langkah 3 | : | 22                     | 10 | 15 | 3 | 2 | 8 |
| Langkah 4 | : | Ulangi langkah 2 dan 3 |    |    |   |   |   |

Hasil iterasi 1 :    2      22      10      15      3      8

**Iterasi 2**

|           |   |   |    |    |    |   |   |
|-----------|---|---|----|----|----|---|---|
| Langkah 1 | : | 2 | 22 | 10 | 15 | 3 | 8 |
| Langkah 2 | : | 2 | 22 | 10 | 15 | 3 | 8 |

Note:  $8 > 3$ , maka 8 tidak pindah, untuk selanjutnya bandingkan data sebelumnya yaitu 3.

|           |   |                        |   |    |    |    |   |
|-----------|---|------------------------|---|----|----|----|---|
| Langkah 3 | : | 2                      | 3 | 22 | 10 | 15 | 8 |
| Langkah 4 | : | Ulangi langkah 2 dan 3 |   |    |    |    |   |

Lakukan Iterasi selanjutnya sampai iterasi ke- 6

**Iterasi 3**

|             |   |    |    |    |    |    |
|-------------|---|----|----|----|----|----|
| Langkah 1 : | 2 | 22 | 10 | 15 | 3  | 8  |
| Langkah 2 : | 2 | 3  | 22 | 10 | 15 | 8  |
| Langkah 3 : | 2 | 3  | 8  | 22 | 10 | 15 |
| Langkah 4 : | 2 | 3  | 8  | 22 | 10 | 15 |

**Iterasi 4**

|            |   |    |    |    |    |    |
|------------|---|----|----|----|----|----|
| Langkah 1: | 2 | 22 | 10 | 15 | 3  | 8  |
| Langkah 2: | 2 | 3  | 22 | 10 | 15 | 8  |
| Langkah 3: | 2 | 3  | 8  | 22 | 10 | 15 |
| Langkah 4: | 2 | 3  | 8  | 22 | 10 | 15 |
| Langkah 5: | 2 | 3  | 8  | 10 | 22 | 15 |

Contoh Program Bubble Sort:

```
def BubbleSort(X):
    for i in range(len(X)-1,0,-1):
        Max=0
        for l in range(1,i+1):
            if X[l]>X[Max]:
                Max = l
        temp = X[i]
        X[i] = X[Max]
        X[Max] = temp
```

```
Hasil = [22,10,15,3,8,2]
BubbleSort(Hasil)
print(Hasil)
```

Hasilnya: **[2, 3, 8, 10, 15, 22]**

3. Insertion Sort Pengurutan data yang membandingkan data dengan dua elemen data pertama, kemudian membandingkan elemen-elemen data yang sudah diurutkan, kemudian perbandingan antara data tersebut akan terus diulang hingga tidak ada elemen data yang tersisa (Rahayuningsih, 2016). Mirip dengan cara **mengurutkan** kartu, perlembar yang diambil & **disisipkan** (insert) ke tempat yang seharusnya.

Prinsip kerjanya:

- 1) Pengecekan mulai dari data ke-1 sampai data ke-n
- 2) Bandingkan data ke-I ( I = data ke-2 s/d data ke-n )
- 3) Bandingkan data ke-I tersebut dengan data sebelumnya (I-1), Jika lebih kecil maka data tersebut dapat disisipkan ke data awal sesuai dengan posisi yang seharusnya
- 4) Lakukan langkah 2 dan 3 untuk bilangan berikutnya (I=I+1) sampai didapatkan urutan yang optimal.

Contoh :            22      10      15      3      8      2

#### Iterasi 1

Langkah 1:        22      10      15      3      8      2

Langkah 2:       22      10      15      3      8      2

Langkah 3:       10      22      15      3      8      2

Langkah 4:        Ulangi langkah 2 dan 3

#### Iterasi 2

Langkah 1:       10      22      15      3      8      2

Langkah 2:       10      22      15      3      8      2

Langkah 3:       10      15      22      3      8      2

Langkah 4:        Ulangi langkah 2 dan 3

Lakukan Iterasi selanjutnya sampai iterasi ke- 6

Catatan : Setiap ada pemindahan, maka elemen yang sudah ada akan di-insert sehingga akan bergeser ke belakang.

Contoh Program Insertion Sort:

```
def InsertionSort(val):
    for index in range(1,len(val)):
        a = val[index]
        b = index
        while b>0 and val[b-1]>a:
            val[b]=val[b-1]
            b = b-1
        val[b]=a
```

Angka = [22,10,15,3,8,2]

```
InsertionSort(Angka)
print(Angka)
```

Hasilnya: `[2, 3, 8, 10, 15, 22]`

### Teknik Searching

Pengertian Teknik Searching adalah teknik dalam memilih dan menyeleksi sebuah elemen dari beberapa elemen yang ada.

Teknik Pencarian (*searching*) dibagi dua, yaitu

1. Teknik Pencarian Tunggal

- a) Teknik Sequential Search/Linier Search (untuk data yang belum terurut/yang sudah terurut)

Pencarian yang dimulai dari record-1 diteruskan ke record selanjutnya yaitu record-2, ke-3,..., sampai diperoleh isi record sama dengan informasi yang dicari (Nilai X).

Algoritmanya:

- Tentukan  $I = 1$
- Ketika Nilai  $(I) \neq X$  Maka Tambahkan  $I = I + 1$
- Ulangi langkah No. 2 sampai Nilai  $(I) = X$
- Jika Nilai  $(I) = N + 1$  Maka Cetak “Pencarian Gagal” selain itu Cetak “Pencarian Sukses “

Contoh1:

Data A = { 10, 4, 9, 1, 15, 7 }

Dicari **15**

**Langkah pencariannya:**

Langkah 1:  $A[1] = 10$

Langkah 2:  $10 \neq 15$ , maka  $A[2] = 4$

Langkah 3: ulangi langkah 2

Langkah 2:  $4 \neq 15$ , maka  $A[3] = 9$

Langkah 2:  $9 \neq 15$ , maka  $A[4] = 1$

Langkah 2:  $1 \neq 15$ , maka  $A[5] = 15$

Langkah 2:  $15 = 15$

Langkah 4: “Pencarian Sukses”

Contoh2:

Apabila ditemukan kondisi:

Nilai  $(i) = N + 1$ , maka pencarian tidak ditemukan atau **gagal**.

Dikarenakan jumlah elemen adalah N,  $N + 1$  artinya data yang dicari bukan merupakan elemen data dari N.

2. Teknik StraitMaxMin

- Menentukan/mencari elemen max&min. Pada Himpunan yang berbentuk array linear.
- Waktu tempuh/*time complexity* yang digunakan untuk menyelesaikan pencarian hingga mendapatkan solusi yang optimal terbagi atas **best case**, **average case** dan **worst case**.

Algoritma untuk mencari elemen MaxMin dalam Python

```
def STRAITMAXMIN(A,n):  
    max = A[0]  
    min = A[0]  
    for i in range(1,n):  
        if A[i] > max:  
            max = A[i]  
        elif A[i] < min:  
            min = A[i]  
    print("Max =", max, ", Min =", min)
```

### Best Case

Keadaan yang tercapai jika elemen pada himpunan A disusun secara *increasing* (menaik).

Dengan perbandingan waktu  $n - 1$  kali satuan operasi.

### Contoh :

Terdapat himpunan A yang berisi 4 buah bilangan telah disusun secara *increasing* dengan  $A[0]=2$ ,  $A[1]=4$ ,  $A[2]=5$ ,  $A[3]=10$ .

Tentukan/cari Bilangan Max&Min serta jumlah operasi perbandingan yang dilakukan.

### Penyelesaian Best Case:

Untuk masalah tersebut dapat digunakan procedure STRAITMAXMIN yang menghasilkan bilangan Min=2 & bilangan Max=10,

Operasi perbandingan data mencari bilangan MaxMin dari himpunan tersebut ( $n-1$ ) = 3 kali operasi perbandingan.

### Penyelesaian Best Case detail:

Contoh:  $A = \{ 2, 4, 5, 10 \}$

Operasi Perbandingan :  $4 - 1 = 3$

### STRAITMAXMIN(A,n)

```
max = min = 2  
FOR i ← 1 TO 3  
    Saat i=1 Apakah  $4 > 2$  maka max = 4 dan kembali ke FOR  
    Saat i=2 Apakah  $5 > 4$  maka max = 5 dan kembali ke FOR  
    Saat i=3 Apakah  $10 > 5$  maka max = 10  
Maka Nilai MIN=2 dan MAX=10
```

### WORST CASE

Terjadi jika elemen dalam himpunan disusun secara *decreasing* (menurun).

Dengan Operasi perbandingan sebanyak  $2(n-1)$  kali satuan operasi.

**Contoh :**

Mencari elemen MaxMin & jumlah oprasi perbandingan yang dilakukan terhadap himpunan A yang disusun *decreasing*.

A[0]=80, A[1]=21, A[2]=6, A[3]=-10

**Penyelesaian Worst Case**

Untuk masalah tersebut dengan proses STRAITMAXMIN adalah elemen max=80 & elemen min=-10,

Operasi perbandingan untuk elemen Maxmin tersebut adalah  $2(4-1) = 6$  kali satuan operasi.

Penyelesaian Worst Case Detail:

Contoh: A = { 80, 21, 6, -10 }

Operasi Perbandingan :  $2(4 - 1) = 6$

STRAITMAXMIN(A,n)

Max = min = 80

For i  $\leftarrow$  1 to 3

Saat i=1      Apakah 21 > 80 maka

                 ELSE Apakah 21 < 80 maka    min = 21

Saat i=2      Apakah 6 > 80 maka

                 ELSE Apakah 6 < 21 maka    min = 6

Saat i=3      Apakah -10 > 80 maka

                 ELSE Apakah -10 < 6 maka    min = -10

Maka Nilai MIN=-10 dan MAX=80

**AVERAGE CASE**

Jika pencarian elemen MaxMin dilakukan pada elemen dalam himpunan yang tersusun secara acak (tidak decreasing/tidak increasing).

Jumlah oprasi Perbandingan yang dilakukan adalah rata-rata waktu tempuh *best case* & *worst case*, yaitu  $\frac{1}{2} [ (n-1) + 2(n-1) ] = ( \frac{3n}{2} - 1 )$  kali.

**Contoh:**

Pada himpunan A yg berisi { 5,-4, 9,7 } dilakukan pencarian elemen max & min dengan menggunakan proses STRAITMAXMIN.

Berapa elemen maxmin yang didapatkan & jumlah oprasi perbandingan yang dilakukan.

Penyelesaian:

Elemen max=9, & elemen min=-4.

Jumlah operasi perbandingan adalah  $( 3 \cdot \frac{4}{2} - 1 ) = 5$  kali satuan operasi.

**KESIMPULAN METODE SORTING DAN SEARCHING**

1. Bubble sorting membutuhkan waktu komputasi paling lama.
2. Insertion sort dan Selection sort memiliki kompleksitas yang sama dengan Bubble sort, tetapi waktunya lebih cepat.
3. Teknik searching dapat digunakan untuk pencarian data terurut ataupun belum terurut

**Kasus1**

Terdapat angka sebagai berikut: 80 , 45, 21, 100 , 23, 67, 43, 20, 90, 99, 46, 75, 73, 29  
Buatlah algoritma untuk mencari angka 29 dengan teknik linear search

**Hasil Praktek : Analisis algoritma dalam searching menggunakan linier search**

## Minggu Ke- 12

### Metode Greedy

#### Deskripsi

Membahas tentang metode greedy (optimal on tape storage dan knapsack problem) untuk menyelesaikan masalah

#### Tujuan Pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan tentang metode greedy optimal on tape storage untuk menyelesaikan masalah
2. Menjelaskan tentang metode greedy knapsack problem untuk menyelesaikan masalah

#### 1. Metode Greedy

*Greedy* diambil dari bahasa inggris berarti rakus, tamak, loba, serakah. Prinsip *greedy*: “*Take What You Can Get Now!*”. Algoritma *greedy* membentuk solusi langkah per langkah (*step by step*).

Greedy adalah strategi pencarian untuk masalah optimasi berbasis prinsip: pada setiap tahap, pilih solusi paling baik. Dengan harapan, semua tahapan ini akan menemukan solusi terbaik untuk masalah tersebut. Algoritma greedy termasuk sederhana dan tidak rumit (Santosa and Ai, 2017).

Untuk mendapatkan solusi optimal dari permasalahan yang mempunyai dua kriteria yaitu:

1. Fungsi Tujuan/Utama
2. Nilai pembatas (*constrain*)

#### 2. Proses Kerja Metode Greedy:

Untuk menyelesaikan suatu permasalahan dengan  $n$  input data yang terdiri dari beberapa fungsi pembatas & 1 fungsi tujuan yang diselesaikan dengan memilih beberapa solusi yang mungkin (*feasible solution/feasible sets*), yaitu bila telah memenuhi fungsi tujuan/obyektif.

Contoh permasalahan optimasi:

Diberikan uang senilai A. Tukar A dengan koin-koin uang yang ada. Berapa jumlah minimum koin yang diperlukan untuk penukaran tersebut?

**Contoh 1:** tersedia banyak koin 1, 5, 10, 25

Uang senilai A = 32 dapat ditukar dengan banyak cara berikut:

- a)  $32 = 1 + 1 + \dots + 1$  (32 koin)
- b)  $32 = 5 + 5 + 5 + 5 + 10 + 1 + 1$  (7 koin)
- c)  $32 = 10 + 10 + 10 + 1 + 1$  (5 koin)
- ... dst

Minimum:  $32 = 25 + 5 + 1 + 1$  (4 koin)

Metode GREEDY digunakan dalam penyelesaian masalah-masalah :

#### 1. Optimal On Tape Storage Problem

Permasalahan bagaimana mengoptimalkan storage/memory dalam komputer agar data yg disimpan dapat termuat dengan optimal. Misalkan terdapat  $n$



program yang akan disimpan didalam pita (*tape*). Pita tsb mempunyai panjang maks. sebesar  $L$ , masing-masing program yang akan disimpan mempunyai panjang  $L_1, L_2, L_3, \dots, L_n$ . Cara penyimpanan adalah penyimpanan secara terurut (*sequential*).

|                         |                         |                         |            |                         |
|-------------------------|-------------------------|-------------------------|------------|-------------------------|
| <b><math>L_1</math></b> | <b><math>L_2</math></b> | <b><math>L_3</math></b> | <b>...</b> | <b><math>L_n</math></b> |
|-------------------------|-------------------------|-------------------------|------------|-------------------------|

Persoalan = Bagaimana susunan penyimpanan program<sup>2</sup> tersebut sehingga  $L_1 + L_2 + L_3 + \dots + L_n = L$  ?

Contoh soal:

Misal terdapat 3 buah program ( $n=3$ ) yg masing-masing mempunyai panjang program  $(I_1, I_2, I_3) = (5, 10, 3)$ . Tentukan urutan penyimpanannya secara berurutan (*sequential*) agar optimal....!

Penyelesaiannya:

Dari 3 program tersebut akan didapat 6 buah kemungkinan order, yang didapat dari nilai faktorial  $3 \rightarrow 3!$  (ingat faktorial  $n!$ ).

Tabel1. Ordering Optimal On Tape Storage Problem

| <b>Ordering</b> | <b>Panjang</b> | <b>D(I)</b>                   | <b>MRT</b>   |
|-----------------|----------------|-------------------------------|--------------|
| 1,2,3           | 5,10,3         | $5 + (5+10) + (5+10+3) = 38$  | $38/3=12,66$ |
| 1,3,2           | 5,3,10         | $5 + (5+3) + (5+3+10) = 31$   | $31/3=10,33$ |
| 2,1,3           | 10,5,3         | $10 + (10+5) + (10+5+3) = 43$ | $43/3=14,33$ |
| 2,3,1           | 10,3,5         | $10 + (10+3) + (10+3+5) = 41$ | $41/3=13,66$ |
| 3,1,2           | 3,5,10         | $3 + (3+5) + (3+5+10) = 29$   | $29/3=9,66$  |
| 3,2,1           | 3,10,5         | $3 + (3+10) + (3+10+5) = 34$  | $34/3=11,33$ |

Dari tabel tersebut, didapat susunan/order yang optimal, sbb :

- 1) susunan pertama untuk program ke tiga
- 2) susunan kedua untuk program kesatu
- 3) susunan ketiga untuk program kedua

## 2. Knapsack Problem

- Knapsack adalah tas atau karung
- Karung digunakan memuat objek, tentunya tidak semua objek dapat ditampung di dalam karung.
- Karung hanya dapat menyimpan beberapa objek dengan total ukurannya (*weight*) lebih kecil atau sama dengan ukuran kapasitas karung.

Permasalahan pada knapsack problem adalah bagaimana obyek-obyek tersebut dimuat/dimasukan kedalam ransel (*knapsack*) yang mempunyai kapasitas  $\max=M$ , sehingga timbul permasalahan sbb:

- Bagaimana memilih obyek yang akan dimuat dari  $n$  obyek yang ada sehingga nilai obyek termuat jumlahnya sesuai dgn kapasitas ( $\leq M$ )
- Jika semua obyek harus dimuat kedalam ransel maka berapa bagian dari setiap obyek yang ada dapat dimuat kedalam ransel sedemikian sehingga nilai kum. maks. & sesuai dgn kapasitas ransel?

**Kasus:**

Terdapat 3 buah program ( $n=4$ ) yg masing-masing mempunyai panjang program. ( $I_1, I_2, I_3, I_4$ )=(4, 8, 15, 5). Tentukan urutan penyimpanannya secara berurutan (*sequential*) agar optimal....!

**Hasil Praktek : Analisis algoritma dalam searching menggunakan metode greedy**

## Minggu Ke- 13

### Penyelesaian Dengan Algoritma Pemrograman Greedy

#### Deskripsi

Membahas tentang penyelesaian dengan algoritma pemrograman greedy

#### Tujuan pembelajaran

Setelah melakukan bagian ini mahasiswa mampu:

1. Menjelaskan penyelesaian dengan algoritma pemrograman greedy
2. Menjelaskan model graph pada metode greedy untuk travelling salesman problem
3. Menjelaskan model graph pada metode greedy untuk shortest path problem
4. Menjelaskan model graph pada metode greedy untuk minimum spanning tree

Penyelesaian dengan Algoritma Pemrograman Greedy

Algoritma GREEDY KNAPSACK

PROCEDURE GREEDY\_KNAPSACK (W, x, n)

float W[n], x[n], M, isi;

int i, n;

x(1 : 1)  $\leftarrow$  0 ; isi  $\leftarrow$  M ;

FOR i  $\leftarrow$  1 TO n

{

IF W(i) > isi

EXIT

ENDIF

x[i]  $\leftarrow$  1

isi  $\leftarrow$  isi - W[i]

}

IF i  $\leq$  n;

x[i]  $\leftarrow$  isi / W[i]

ENDIF

END\_GREEDY\_KNAPSACK

Contoh:

Diketahui bahwa kapasitas  $M = 20\text{kg}$ , dengan jumlah barang  $n=3$ . Berat  $W_i$  masing<sup>2</sup> barang =  $(W_1, W_2, W_3) = (18, 15, 10)$ . Nilai  $P_i$  masing<sup>2</sup> barang =  $(P_1, P_2, P_3) = (25, 24, 15)$ . Lakukan pengurutan secara tdk naik terhadap hasil  $P_i/W_i$ , misalnya :

$P_1/W_1 \rightarrow 25/18 = 1,39$  menjadi urutan ke 3

$P_2/W_2 \rightarrow 24/15 = 1,60$  menjadi urutan ke 1

$P_3/W_3 \rightarrow 15/10 = 1.50$  menjadi urutan ke 2

Sehingga m'hasilk' pola urutan data yg baru, yaitu  $W_1, W_2, W_3 \rightarrow 15, 10, 18$  dan  $P_1, P_2, P_3 \rightarrow 24, 15, 25$

Lalu data<sup>2</sup> tersebut diinputkan pada Algoritma Greedy, terjadi proses :

$x(1:n) \leftarrow 0$  ; isi  $\leftarrow 20$  ;  $i = 1$

$W(i) > \text{isi} ? \rightarrow 15 > 20 ? \rightarrow$  kondisi SALAH

$x(1) = 1 \rightarrow$  b'arti bhw brg tsb dpt dimuat seluruhnya.

Isi =  $20 - 15 \rightarrow$  kapasitas ransel b'kurang dengan

sisa 5kg  $i = 2$

$W(2) > \text{isi} \text{ ??} \rightarrow 10 > 5 \text{ ??} \rightarrow \text{kondisi BENAR}$

$x(2) = 5/10 = 1/2 \rightarrow \text{benda 10kg hanya dpt dimuat } 1/2 \text{ bagian yaitu 5 kg.}$

$i = 3$

Endif  $\rightarrow$  diakhiri krn ransel sdh penuh (max = 20kg)

Profit nilai yang didapat adalah :  $P1 + P2 + P3$  yaitu:

$$24.1 + 15.1/2 + 18.0 = 24 + 7.5 = 31.5$$

Penyelesaiannya:

$x(1:n) \leftarrow 0$  ; isi  $\leftarrow 20$  ;  $i = 1$

FOR  $i \leftarrow 1$  TO 3

Saat  $i=1$  APK  $W[1] > \text{isi} \text{?} \rightarrow 15 > 20 \text{?}$

$x[1] \leftarrow 1$  (barang dapat dimuat seluruhnya)

isi =  $20 - 15 = 5$  (sisa kapasitas 5kg)

Saat  $i=2$  APK  $W[2] > \text{isi} \text{?} \rightarrow 10 > 5 \text{?} \rightarrow \text{exit}$

APK  $i \leq n \text{?} \rightarrow 2 \leq 3 \text{?}$

$x[2] = \text{isi}/W[2] = 5/10 = 1/2$  benda 10kg dimuat  $1/2$  bag = 5

ENDIF diakhiri karena ransel sudah penuh (max = 20kg)

Profit nilai :  $P1 + P2 + P3$  yaitu:

$$24(1) + 15(1/2) + 18(0) = 24 + 7,5 = 31,5$$

## 2. PROBLEMA DAN MODEL GRAPH DALAM METODE GREEDY

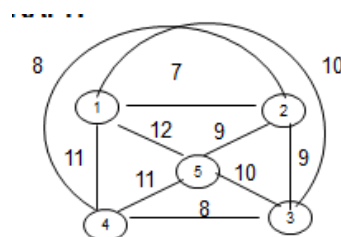
### 1) TRAVELLING SALESMAN

Untuk menentukan waktu perjalanan seorang salesman seminimal mungkin.

Permasalahan:

Setiap minggu sekali, seorang petugas kantor telepon berkeliling untuk mengumpulkan coin-coin pada telepon umum yang dipasang diberbagai tempat. Berangkat dari kantornya, ia mendatangi satu demi satu telepon umum tersebut dan akhirnya kembali ke kantor lagi. Masalahnya ia menginginkan suatu rute perjalanan dengan waktu minimal.

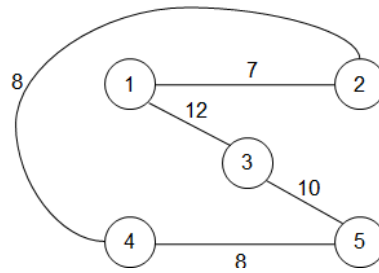
MODEL GRAPH



Misalnya: Kantor pusat adalah simpul 1 dan misalnya ada 4 telepon umum, yang kita nyatakan sebagai simpul 2, 3, 4 dan 5 dan bilangan pada tiap-tiap ruas menunjukan waktu ( dalam menit ) perjalanan antara 2 simpul .

Langkah penyelesaian:

1. Dimulai dari simpul yang diibaratkan sebagai kantor pusat yaitu simpul 1
2. Dari simpul 1 pilih ruas yang memiliki waktu yang minimal.
3. Lakukan terus pada simpul-simpul yang lainnya tepat satu kali yang nantinya Graph akan membentuk Graph tertutup karena perjalanan akan kembali ke kantor pusat.
4. Problema diatas menghasilkan waktu minimalnya adalah 45 menit dan diperoleh perjalanan sbb :



- 2) Minimum Spanning Tree adalah mencari minimum biaya (*cost*) spanning tree dari setiap ruas (*edge*) graph yang membentuk pohon (*tree*).

Solusi dari permasalahan ini:

1. Dengan memilih ruas suatu graph yang memenuhi kriteria dari optimisasi yang menghasilkan biaya minimum.
2. Penambahan dari setiap ruas pada seluruh ruas yang membentuk graph akan menghasilkan nilai/biaya yang kecil (*minimum cost*).

- 3) Shortest Path Problem adalah permasalahan menghitung jalur terpendek dari sebuah graph

Kriteria untuk permasalahan Shortest Path problem tersebut :

- Setiap ruas pada graph harus mempunyai nilai (*label graph*)
- Setiap ruas pada graph tidak harus terhubung (*unconnected*)
- Setiap ruas pada graph tersebut harus mempunyai arah (graph berarah).

#### Kasus1:

Terdapat sebuah kapal dengan kapasitas 180 Ton, Akan memuat 6 buah barang masing-masing adalah : Gula pasir 50 Ton dengan harga 100 Juta, Gula merah 60 Ton dengan harga 80 Juta dan Gula batu 70 Ton dengan harga 90 Juta. Beras 50 Ton dengan harga 150 Juta, Terigu 20 ton dengan harga 40 Juta, Minyak goreng 60 Ton dengan harga 200 Juta. Dengan metode Algoritma Greedy Tentukan barang apa saja yang dimuat truk dengan harga yang paling mahal

#### Kasus2:

Diberikan n buah objek dan sebuah knapsack dengan kapasitas tertentu (K). Setiap objek memiliki property bobot w(weight) dan keuntungan p(profit). Objektif persoalan adalah bagaimana memilih objek-objek yang dimasukkan ke dalam knapsack sehingga tidak melebihi kapasitas knapsack namun memaksimalkan total keuntungan yang diperoleh.” Tinjau persoalan 0/1 Knapsack dengan n = 4. Misalkan objek-objek tersebut kita beri nomor 1, 2, 3, dan 4. Properti setiap objek i dan kapasitas knapsack adalah sebagai berikut

w1 = 2; p1 = 20

w2 = 5; p1 = 30

w3 = 10; p1 = 50

w4 = 5; p1 = 10

Kapasitas knapsack W = 16, pecahkan permasalahan knapsack diatas dengan metode greedy

**Hasil Praktek : Analisis algoritma dalam knapsack problem dengan metode greedy**

## Minggu Ke- 14

### Pewarnaan

#### Deskripsi

Membahas tentang pewarnaan atau coloring untuk memecahkan masalah

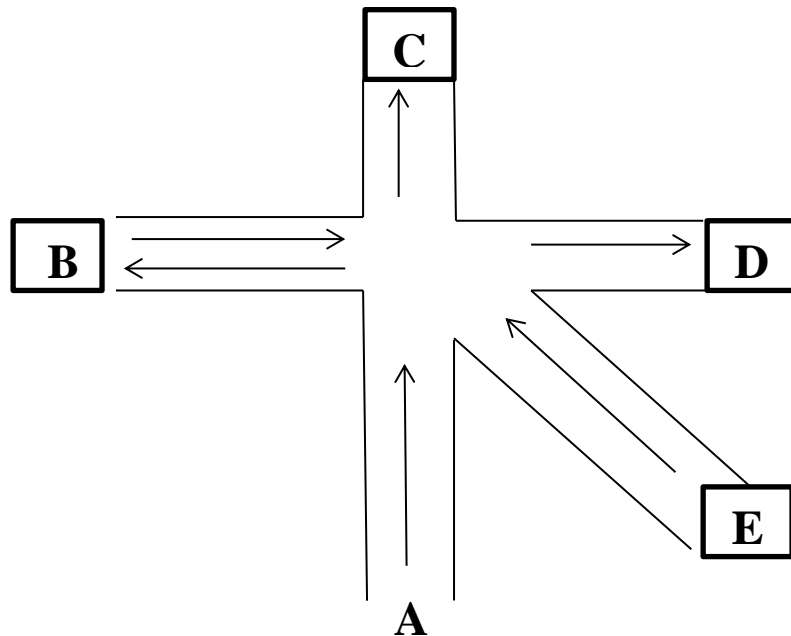
#### Tujuan Pembelajaran

Setelah melakukan bagian ini mahasiswa mampu menjelaskan tentang konsep greedy pada pewarnaan untuk menyelesaikan masalah

#### Pewarnaan (*Coloring*)

Problema pemberian warna kepada semua simpul, sedemikian sehingga 2 simpul yang berdampingan ( ada ruas menghubungkan ke dua simpul tersebut ) mempunyai warna yang berbeda. Banyak warna yang dipergunakan, diminta seminimal mungkin.

Contoh:



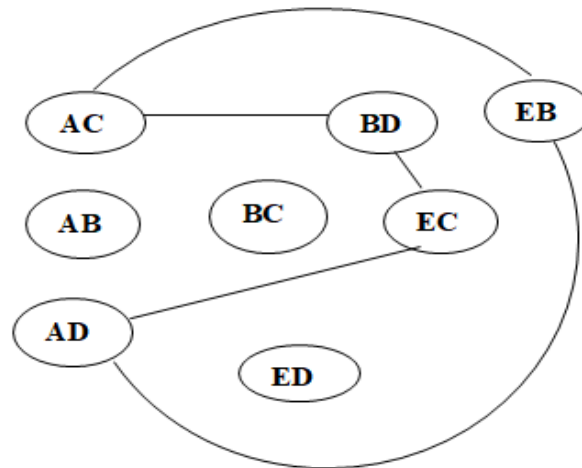
Gambar 1. Pola lalu lintas

#### Permasalahan:

Menentukan pola lampu lalu lintas dengan jumlah fase minimal seperti pada gambar 1 diatas, setiap fase tidak ada perjalanan yang saling melintas . Perjalanan yang diperbolehkan adalah : A ke B, A ke C, A ke D, B ke C, B ke D, E ke B, E ke C dan E ke D.

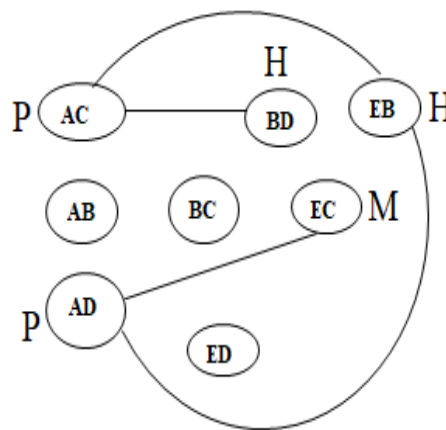
#### Langkah-langkah penyelesaian masalah :

1. Tentukan simpul dari perjalanan yang diperbolehkan (untuk peletakan simpulnya bebas)
2. Tentukan ruas untuk menghubungkan 2 simpul yg menyatakan 2 perjalanan yg saling melintas



Gambar 2. Pola Penyelesaian Masalah

3. Beri warna pada setiap simpul dengan warna-warna baru.
  - Bila Simpul berdampingan maka berilah warna lain.
  - Bila simpul tidak berdampingan maka berilah warna yang sama



Gambar 3. Pemberian warna pada simpul

4. Kita lihat Bahwa simpul AB , BC dan ED tidak dihubungkan oleh suatu ruas jadi untuk simpul tersebut tidak pernah melintas perjalanan-perjalanan lain dan simpul tersebut selalu berlaku lampu hijau
5. Tentukan pembagian masing –masing simpul yang sudah diberikan warna.  
Putih = (AC, AD )  
Hitam = (BD, EB )  
Merah = (EC)

**Catatan :**

Pembagian simpul berdasarkan simpul yang tidak langsung berhubungan seminimal mungkin (BISA DILAKUKAN DENGAN BEBERAPA KEMUNGKINAN)

6. Dari langkah ke 5 diperoleh 3 fase, sehingga bisa kita simpulkan keseluruhan situasi dan hasilnya dapat dinyatakan sebagai berikut:

Fase 1:

|       |                    |
|-------|--------------------|
| HIJAU | AC, AD, AB, BC, ED |
| MERAH | BD, EB, EC         |

Fase 2:

|       |                    |
|-------|--------------------|
| HIJAU | BD, EB, AB, BC, ED |
| MERAH | AC, AD, EC         |

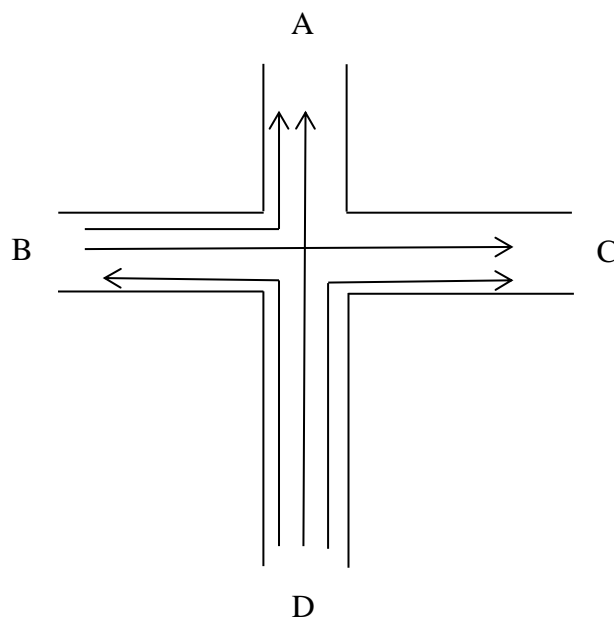
Fase 3:

|       |                |
|-------|----------------|
| HIJAU | EC, AB, BC, ED |
| MERAH | AC, AD, BD, EB |

**Kasus:**

Buatlah pola lalu lintas agar perjalanan dapat dilakukan dengan aman dengan jumlah Fase minimal

Perjalanan yang diperbolehkan adalah D ke B, D ke A, D ke C, B ke A, B ke C



**Hasil Praktek : Analisis algoritma dalam pewarnaan dengan metode greedy**



## **Minggu Ke- 15**

### **QUIZ**

#### **Deskripsi**

Membahas tentang review materi dari pertemuan 9 sampai pertemuan 14

#### **Tujuan pembelajaran**

Setelah melakukan bagian ini mahasiswa mampu untuk memahami materi pertemuan 9 sampai pertemuan 14 dengan menjawab soal

**Minggu Ke- 16**

**UAS**

# UJIAN AKHIR SEMESTER

## DAFTAR PUSTAKA

1. Kadir, Abdul. 2019. Logika Pemrograman Python. Elex Media Komputindo. Jakarta
2. Jubilee Enterprise. 2017. Otodidak Pemrograman Python. Elex Media Komputindo. Jakarta
3. Supardi, Yuniar. 2020. Semua Bisa Menjadi Programmer Python Case Study. Elex Media Komputindo. Jakarta
4. Dwi Yuniarti, Wenty. 2019. Dasar-dasar Pemrograman Dengan Python. Deepublish Publisher. Yogyakarta
5. Swastika, Windra. 2018. Pengantar Algoritma dan Penerapannya pada Python. Ma Chung Press, Malang
6. Harumy, T.Hendry Febriana, dkk. 2016. Belajar Dasar Algoritma dan Pemrograman C++. Deepublish. Yogyakarta
7. A.S, Rosa. 2018. Logika Algoritma dan Pemrograman Dasar. Modula. Bandung.
8. Munir, Rinaldi. 2016. Algoritma dan Pemrograman Dalam Bahasa Pascal, C Dan C++ Edisi Keenam. Informatika Bandung.
9. Sjukani, Moh. 2013. Algoritma (Algoritma & Struktur Data 1) dengan C, C++ dan Java Edisi 8. Mitra Wacana Media. Jakarta.
10. Jud. (2017). *Mastering Python*. CV Jubilee Solusi Enterprise.
11. Sitorus, lamhot. (2015). *Algoritma dan Pemrograman*. CV. Andi Offset.
12. Ramadhani, Cipta. 2015. Dasar Algoritma & Struktur Data dengan Bahasa Java. Andi Publisher. Yogyakarta.
13. Kadir, Abdul. 2012. Algoritma & Pemrograman Menggunakan Java. Andi Offset. Yogyakarta
14. Sriyadi; Nurhasanah; Baidawi Taufik. 2018. Sistem Pakar Diagnosa Penyakit Ikan Nila (*Oreochromis Niloticus*) Berbasis Web Menggunakan Metode Forward Chaining. PARADIGMA. Volume XX No. 2 September 2018 P-ISSN 1410-5063 p.123-128.
15. Indriyani, Fintri. Irfiani, Eni. 2016. Sistem Pakar Diagnosa Keguguran Pada Ibu Hamil. Konferensi Nasionak Ilmu Sosial & Teknologi (KNIST) Maret 2016 p.254-258.
16. Aristi, G. (2015) 'Perbandingan Penyelesaian Knapsack Problem Secara

Matematika, Kriteria Greedy Dan Algoritma Greedy', *Jurnal Technoper*, 1, pp. 3–4.

17. Sonita, A., & Nurtaneo, F. (2015). Analisis Perbandingan Algoritma Bubble Sort, Merge Sort, Dan Quick Sort Dalam Proses Pengurutan Kombinasi Angka Dan Huruf. *Jurnal Pseudocode*, II(September), 75–80.  
<https://ejournal.unib.ac.id/index.php/pseudocode/article/view/887>