

LAPORAN PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
MODUL 3 : COLLECTIONS, ARRAYS, AND LINKED STRUCTURES



Disusun Oleh :
MHD. FARHAN LUBIS
L200220277
F

TEKNIK INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA
TAHUN 2024

Daftar Isi

Daftar Isi.....	2
1.11 Soal – Soal Mahasiswa.....	3
1. Terkait array dua dimensi, kita akan membuat tipe data sebuah matrix yang berisi angka-angka. 3	
• Kode Program.....	3
• Screenshot hasil praktikum.....	5
2. Terkait matrix dan list comprehension, buatlah (dengan memanfaatkan list comprehension) fungsi-fungsi	5
• Kode Program.....	5
• Screenshot hasil praktikum.....	6
3. Terkait linked list, buatlah fungsi untuk	6
• Kode Program.....	6
• Screenshot hasil praktikum.....	9
4. Terkait doubly linked list, buatlah fungsi untuk	9
• Kode Program.....	9
• Screenshot hasil praktikum.....	11

1.11 Soal – Soal Mahasiswa

1. Terkait array dua dimensi, kita akan membuat tipe data sebuah matrix yang berisi angka-angka.

Untuk itu buat fungsi-fungsi berikut ini

- Untuk memastikan bahwa isi dan ukuran matrix-nya konsisten (karena tiap anggota dari list-luar-nya bisa saja mempunyai ukuran yang berbeda-beda. dan bahkan bisa saja berbeda tipe!)
- Untuk mengambil ukuran matrix-nya
- Untuk menjumlahkan dua matrix (pastikan ukurannya sesuai)
- Untuk mengalikan dua matrix (pastikan ukurannya sesuai)
- Untuk menghitung determinan sebuah matrix bujursangkar

- **Kode Program**

```
class ArrayMatrix:
    def __init__(self, matriks):
        self.matriks = matriks

    def cek_konsistensi(self):
        valid_types = (int, float)
        for baris in self.matriks:
            for elemen in baris:
                if type(elemen) not in valid_types:
                    return False
        return all(len(baris) == len(self.matriks[0]) for
        baris in self.matriks)

    def dapatkan_ukuran(self):
        baris = len(self.matriks)
        kolom = len(self.matriks[0]) if self.matriks else
0
        return baris, kolom

    def tambah(self, matriks_lain):
        if not self.cek_konsistensi() or not
matriks_lain.cek_konsistensi():
            print("Ukuran matriks tidak konsisten")
        hasil = []
        for i in range(len(self.matriks)):
            baris_hasil = []
            for j in range(len(self.matriks[0])):
                baris_hasil.append(self.matriks[i][j] +
matriks_lain.matriks[i][j])
            hasil.append(baris_hasil)
        return ArrayMatrix(hasil)

    def kali(self, matriks_lain):
        if len(self.matriks[0]) !=
len(matriks_lain.matriks):
            print("Ukuran matriks tidak kompatibel untuk
perkalian")

        hasil = []
        for i in range(len(self.matriks)):
            baris_hasil = []
```

```

        for j in range(len(matriks_lain.matriks[0])):
            jumlah = 0
            for k in range(len(self.matriks[0])):
                jumlah += self.matriks[i][k] *
matriks_lain.matriks[k][j]
            baris_hasil.append(jumlah)
        hasil.append(baris_hasil)
        return ArrayMatrix(hasil)

    def determinan(self):
        n = len(self.matriks)
        if n == 1:
            return self.matriks[0][0]
        if n == 2:
            return self.matriks[0][0] * self.matriks[1][1]
        - self.matriks[0][1] * self.matriks[1][0]
        hasil = 0
        tanda = 1
        for c in range(len(self.matriks[0])):
            submatriks = [baris[:c] + baris[c+1:] for
baris in self.matriks[1:]]
            subdeterminan =
ArrayMatrix(submatriks).determinan()
            hasil += tanda * self.matriks[0][c] *
subdeterminan
            tanda *= -1
        return hasil

matriks1 = ArrayMatrix([[1.9, 2, 3.8], [4, 5.3, 6], [7.2,
8, 9.6]])
matriks2 = ArrayMatrix([[7,8.5, 9], [4.1, 5, 6.8], [1,
2.7, 3]])

print(matriks1.cek_konsistensi())
print(matriks1.dapatkan_ukuran())

hasil_tambah = matriks1.tambah(matriks2)
print(hasil_tambah.matriks)

hasil_kali = matriks1.kali(matriks2)
print(hasil_kali.matriks)
print(matriks1.determinan())

print("\nProgram Completed!\n\n--- By L200220277 ---")

```

Kode 3.1 program array dua dimensi

PENJELASAN:

Method "cek_konsistensi" digunakan untuk mengecek konsistensi ukuran matriks agar setiap baris memiliki jumlah isi yang sama. Method ini melakukan iterasi setiap baris dalam matriks dan mengecek panjangnya. Method ini memastikan isi matriks bertipe data numerik (int atau float) dan kemudian menggunakan function "all" untuk mengecek apakah panjang semua baris sama dengan panjang baris pertama dalam matriks. Jika berbeda, maka mengembalikan False yang artinya tidak konsisten. Jika sama, maka mengembalikan True, artinya matriks konsisten.

Method "dapatkan_ukuran" digunakan untuk mengembalikan jumlah baris dan kolom

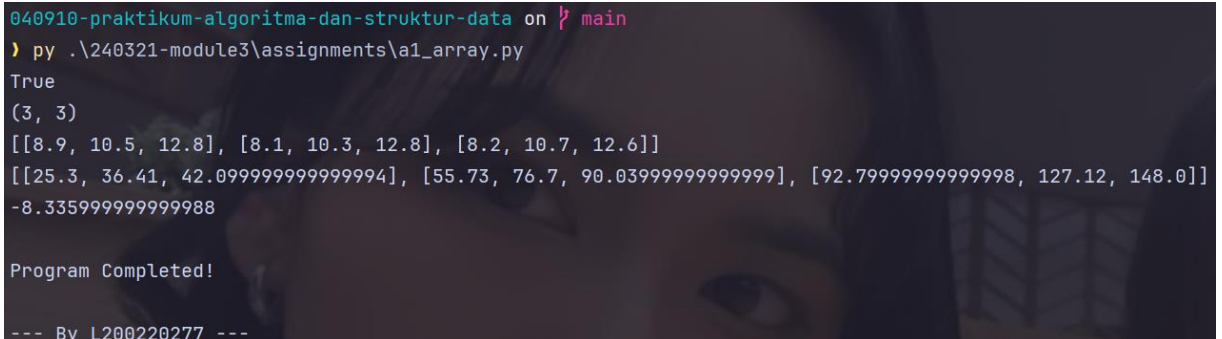
dalam matriks berupa tuple. Method ini menghitung panjang matriks untuk mendapatkan jumlah baris, dan panjang baris pertama untuk mendapatkan jumlah kolomnya. Jika matriks kosong, maka jumlah kolom jadi 0.

Method "tambah" digunakan untuk melakukan penambahan antara dua matriks. Diawal, method ini mengecek konsistensi kedua matriksnya. Kemudian, Mendeklarasikan matriks hasil. lalu, dengan nested loop isinya ditambahkan dari matriks input dan matriks lainnya, dengan hasilnya disimpan dalam matriks hasil. Akhirnya, matriks hasil dikembalikan.

Method "kali" digunakan untuk melakukan perkalian antara dua matriks. method ini mengecek konsistensi kedua matriks dengan memastikan jumlah kolom dalam matriks pertama sama dengan jumlah baris dalam matriks kedua. Kemudian, Mendeklarasikan matriks kosong. Dengan nested loop setiap isi dari matriks hasil akan di hitung dengan menjumlahkan hasil perkalian isi-isi yang sesuai dari baris matriks pertama dengan kolom matriks kedua. Terakhir, matriks hasil dikembalikan.

Method "determinan" digunakan untuk menghitung determinan dari matriks. Jika matriks berukuran 1x1, maka nilai isi itu determinannya. Jika matriks 2x2, determinannya dihitung menggunakan rumus determinan matriks 2x2. Untuk matriks berukuran lebih dari itu, method ini menggunakan metode ekspansi kofaktor dengan rekursi untuk menghitung determinannya. Dengan iterasi setiap isi dalam baris pertama matriks, dan menghitung determinan submatriksnya yang didapatkan dengan menghapus baris dan kolom yang sesuai dari isi tersebut. Terakhir, hasilnya ditambahkan dengan memperhitungkan tanda yang sesuai.

- **Screenshot hasil praktikum**



```
040910-praktikum-algoritma-dan-struktur-data on main
> py .\240321-module3\assignments\a1_array.py
True
(3, 3)
[[8.9, 10.5, 12.8], [8.1, 10.3, 12.8], [8.2, 10.7, 12.6]]
[[25.3, 36.41, 42.099999999999994], [55.73, 76.7, 90.03999999999999], [92.79999999999998, 127.12, 148.0]]
-8.335999999999998

Program Completed!

--- By L200220277 ---
```

Gambar 3.1 output a1_array.py

2. Terkait matrix dan list comprehension, buatlah (dengan memanfaatkan list comprehension) fungsi-fungsi

- Untuk membangkitkan matrix berisi nol semua, dengan diberikan ukurannya. Pemanggilan: `buatNol(m,n)` dan `buatNol(m)`. Pemanggilan dengan cara terakhir akan memberikan matrix bujursangkar ukuran $m \times m$.
- Untuk membangkitkan matrix identitas, dengan diberikan ukurannya. Pemanggilan: `buatIdentitas(m)`

- **Kode Program**

```
class MatrixListComprehension(object):
    def buat_nol(self, m, n=None):
        if n is None:
```

```

        n = m
        return [[0 for _ in range(n)] for _ in range(m)]
    def buat_identitas(self, m):
        return [[1 if i == j else 0 for j in range(m)] for
i in range(m)]

matriks = MatrixListComprehension()

matriksmxn = matriks.buat_nol(8, 5)
print(matriksmxn)

matriksm = matriks.buat_nol(4)
print(matriksm)

matriksid = matriks.buat_identitas(5)
print(matriksid)

print("\nProgram Completed!\n\n--- By L200220277 ---")

```

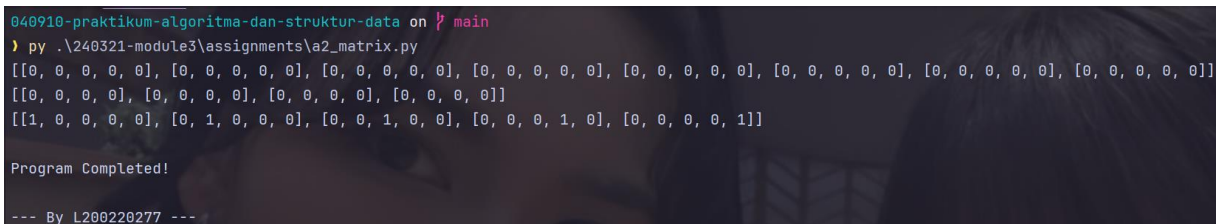
Kode 3.2 program matrix dan list comprehension

PENJELASAN:

Method "buat_nol" dimulai dengan mengecek apakah argumen kedua, "n", ada. Jika tidak, "n" dimasukkan nilai "m". Lalu, list comprehension digunakan untuk membuat matriks berukuran "m"x"n" yang berisi elemen nol. Setiap baris dalam matriks dibuat dengan menggunakan list comprehension untuk membuat list yang berisi "n" elemen nol, dan lalu list ini diulang sebanyak "m" kali.

Method "buat_identitas" juga menggunakan list comprehension untuk membuat matriks identitas berukuran "m"x"m". Setiap elemen pada diagonal matriksnya akan bernilai 1, sedangkan elemen lainnya akan bernilai 0. Hasilnya kemudian dikembalikan menjadi matriks.

• Screenshot hasil praktikum



```

040910-praktikum-algoritma-dan-struktur-data on 1 main
> py .\240321-module3\assignments\A2_matrix.py
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
[[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]

Program Completed!

--- By L200220277 ---

```

Gambar 3.2 output a2_matrix.py

3. Terkait linked list, buatlah fungsi untuk

- mencari data yang isinya tertentu: cari(head,yang dicari)
- menambah suatu simpul di awal: tambahDepan(head)
- menambah suatu simpul di akhir: tambahAkhir(head)
- menyisipkan suatu simpul di mana saja: tambah(head,posisi)
- menghapus suatu simpul di awal, di akhir, atau di mana saja: hapus(posisi)

• Kode Program

```

class Node:
    def __init__(self, data=None):

```

```

        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def cari(self, target):
        current = self.head
        while current:
            if current.data == target:
                return True
            current = current.next
        return False

    def tambah_depan(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node

    def tambah_akhir(self, data):
        if not self.head:
            self.tambah_depan(data)
            return
        last = self.head
        while last.next:
            last = last.next
        last.next = Node(data)

    def tambah(self, data, posisi):
        if posisi == 0:
            self.tambah_depan(data)
            return
        new_node = Node(data)
        prev = None
        current = self.head
        for _ in range(posisi):
            if current is None:
                print("Posisi melebihi panjang linked
list")
            prev = current
            current = current.next
        prev.next = new_node
        new_node.next = current

    def hapus(self, posisi):
        if posisi == 0:
            self.head = self.head.next
            return
        prev = None
        current = self.head
        for _ in range(posisi):
            if current is None:
                print("Posisi melebihi panjang linked
list")

```

```

        prev = current
        current = current.next
        prev.next = current.next

    # Method tambahan untuk mempermudah melihat hasil
    def cetak(self):
        current = self.head
        while current:
            print(current.data, end=" ")
            current = current.next
        print()

ll = LinkedList()

ll.tambah_depan(9)
ll.tambah_depan(5.7)
ll.tambah_depan(2)
ll.cetak()

ll.tambah_akhir(4)
ll.cetak()

ll.tambah(3.5, 3)
ll.cetak()

print(ll.cari(9))
print(ll.cari(5))
ll.cetak()

ll.hapus(2)
ll.cetak()

print("\nProgram Completed!\n\n--- By L200220277 ---")

```

Kode 3.3 program linked list

PENJELASAN:

Method cari digunakan untuk mencari apakah suatu node tertentu ada dalam linked list. Method ini melakukan iterasi setiap node dalam linked list dan mengecek apakah data pada node tersebut sama dengan target yang dicari. Jika ditemukan, akan mengembalikan True; jika tidak ditemukan, mengembalikan False.

Method tambah_depan digunakan untuk menambahkan node baru di depan linked list. Node baru dibuat dengan data yang diberikan, dan node itu dijadikan head baru dan mendeklarasikan next node baru menjadi head sebelumnya.

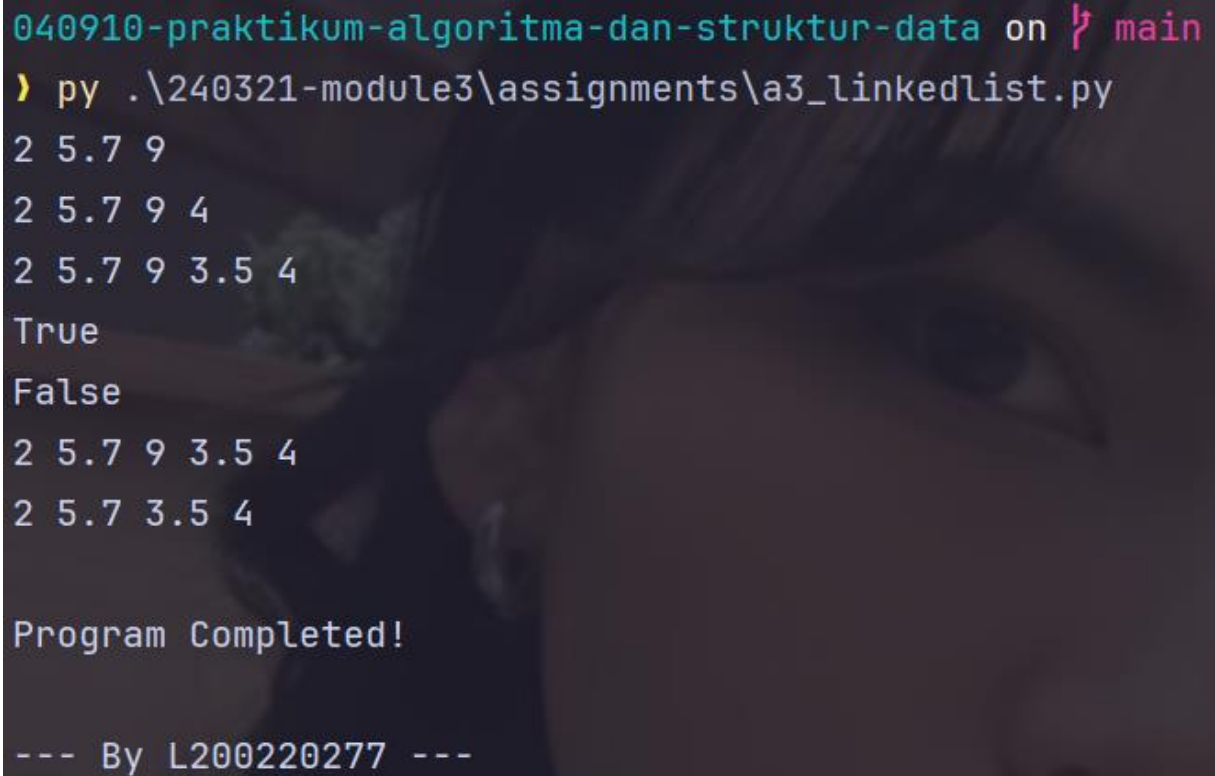
Method tambah_akhir digunakan untuk menambahkan node baru di akhir linked list. Jika linked list kosong, maka node tersebut ditambahkan menggunakan method tambah_depan. Jika tidak, dilakukan iterasi linked list untuk dapat node terakhir, lalu menetapkan next dari node terakhir ke node baru.

Method tambah digunakan untuk menambahkan node baru pada posisi tertentu dalam Linked list. Jika posisinya 0, maka node ditambahkan menggunakan method tambah_depan. Jika tidak, dilakukan iterasi linked list untuk mendapat posisi yang sesuai, lalu memasukkan node baru di antara node sebelumnya dan node pada posisi tersebut.

Method hapus digunakan untuk menghapus node pada posisi tertentu dalam linked list.

Jika posisinya 0, maka head diubah menjadi node kedua menggunakan next. Jika tidak, dilakukan iterasi linked list untuk mendapat posisi yang sesuai, lalu menghubungkan node sebelumnya dengan node setelahnya serta menghapus node dari linked list.

- **Screenshot hasil praktikum**



```
040910-praktikum-algoritma-dan-struktur-data on main
> py .\240321-module3\assignments\a3_linkedlist.py
2 5.7 9
2 5.7 9 4
2 5.7 9 3.5 4
True
False
2 5.7 9 3.5 4
2 5.7 3.5 4

Program Completed!

--- By L200220277 ---
```

Gambar 3.3 output a3_linkedlist.py

4. Terkait doubly linked list, buatlah fungsi untuk

- mengunjungi dan mencetak data tiap simpul dari depan dan dari belakang
- menambah suatu simpul di awal
- menambah suatu simpul di akhir

- **Kode Program**

```
class Node:
    def __init__(self, data=None):
        self.data = data
        self.prev = None
        self.next = None

class DoublyLinkedList:
    def __init__(self):
        self.head = None
        self.tail = None

    def kunjungi_maju(self):
        current = self.head
        while current:
            print(current.data, end=" ")
```

```

        current = current.next

    def kunjungi_mundur(self):
        current = self.tail
        while current:
            print(current.data, end=" ")
            current = current.prev

    def tambah_depan(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = self.tail = new_node
        else:
            new_node.next = self.head
            self.head.prev = new_node
            self.head = new_node

    def tambah_akhir(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = self.tail = new_node
        else:
            new_node.prev = self.tail
            self.tail.next = new_node
            self.tail = new_node

dll = DoublyLinkedList()

dll.tambah_depan(8)
dll.tambah_depan(3)
dll.tambah_akhir(10)
dll.kunjungi_maju()
print("")
dll.tambah_depan(7.5)
dll.kunjungi_mundur()

print("\nProgram Completed!\n\n--- By L200220277 ---")

```

Kode 3.4 program doubly linked list

PENJELASAN:

Method "kunjungi_maju" digunakan untuk menelusuri linked list dari depan ke belakang. Dimulai dari head, method ini melakukan iterasi setiap node, mencetak data setiap node, kemudian berlanjut ke node berikutnya. Proses ini berlanjut sampai akhir linked list yaitu node tail.

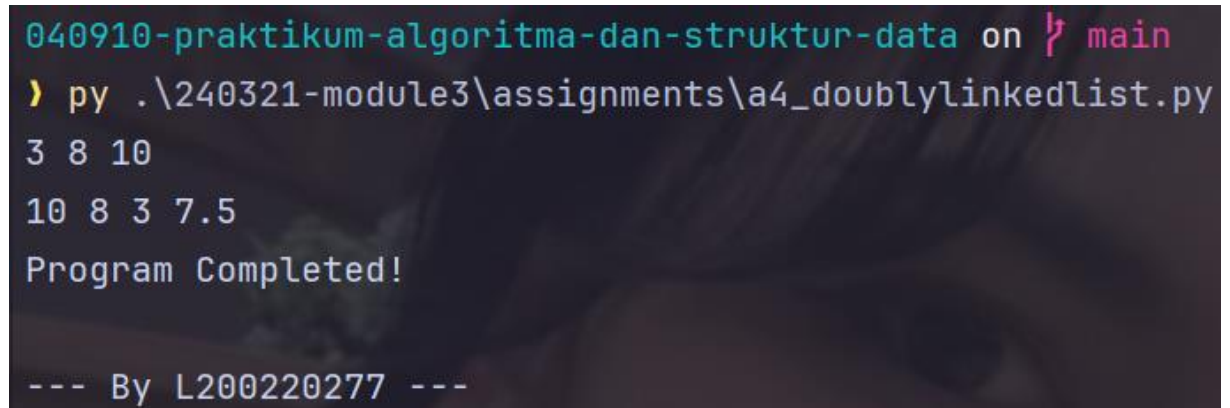
Method "kunjungi_mundur" digunakan untuk menelusuri linked list dari belakang ke depan. Dimulai dari tail, method ini melakukan iterasi mundur setiap node, mencetak data setiap node, dan kemudian berlanjut ke node sebelumnya. Proses ini berlanjut sampai awal linked list, yaitu node head.

Method "tambah_depan" digunakan untuk menambahkan node baru ke depan linked list. Jika linked list kosong, node baru dijadikan head dan tail. Jika tidak, node baru dibuat dari data yang di berikan, dan dijadikan head baru. pointer "prev" dan "next" disesuaikan dengan node baru dan node yang sebelumnya menjadi head.

Method "tambah_akhir" digunakan untuk menambahkan node baru ke akhir linked list. Jika linked list kosong, node baru dijadikan head dan tail. Jika tidak, node baru dibuat

dengan data yang diberikan, dan dijadikan tail baru. pointer "prev" dan "next" disesuaikan dengan node baru dan node yang sebelumnya menjadi tail.

- **Screenshot hasil praktikum**



```
040910-praktikum-algoritma-dan-struktur-data on main
> py .\240321-module3\assignments\a4_doublylinkedlist.py
3 8 10
10 8 3 7.5
Program Completed!

--- By L200220277 ---
```

Gambar 3.4 output *a4_doublylinkedlist.py*