1 Konsep

Jelaskan secara lengkap konsep linked list berikut ini :

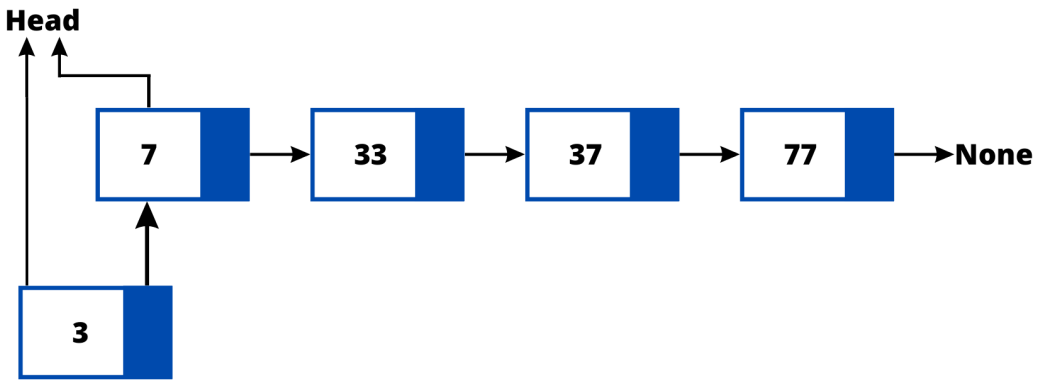
1. Definisi linked list, kelebihan linked list, class yang dibutuhkan untuk membuat linked list
2. Penambahan data dari awal node di dalam linked, jika perlu ditambahkan gambar proses penambahandata (seperti lecture notes yang saya upload) buatan kalian sendiri, dan method yang diperlukan
3. Traversal linked list - jika perlu ditambahkan ilustrasi dengan gambar

1 Konsep

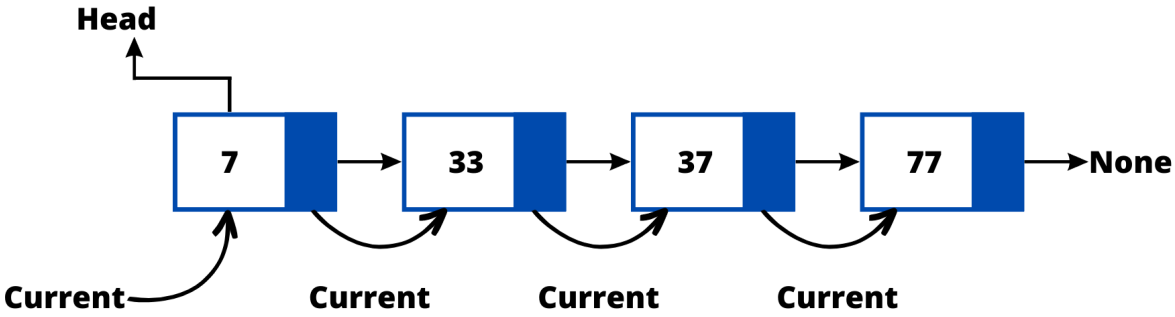
1.
   - Linked list adalah struktur data yang terdiri dari sekumpulan node yang terhubung satu sama lain melalui pointer. Setiap node dalam linked list menyimpan data dan juga pointer ke node berikutnya.
   - Kelebihan :
     1. Ukuran yang dinamis : linked list dapat memperbesar atau memperkecil ukuranya sesuai dengan kebutuhan
     2. Pemakaian Memori yang Efisien : linked list menggunakan ruang memori yang efisien untuk menyimpan element-elementnya karena ruang memori hanya dialokasikan saat dibutuhkan untuk element baru
     3. Modifikasi awal cepat : Operasi-operasi yang dilakukan pada awal linked list seperti penghapusan atau penambahan element di awal dapat dilakukan dengan cepat
   - Class yang dibutuhkan untuk membuat linked list :
     1. Node : class node digunakan untuk mengambarkan node/simpul dalam linked list yang setiap node menyimpan data dan pointer ke simpul berikutnya.
     2. Linked list : class yang digunakan untuk mengatur dan mengelola linked list yang memiliki method-method untuk menghapus,menambah,mencari,dll
2. Konsep penambahan data dari awal linked list menggunakan tiga langkah :
   - Data yang akan di tambahkan harus dalam bentuk node sehingga perlu membuat node baru dengan data yang akan di tambahkan.
   - Atur pointer/next node baru ke node pertama dalam linked list yang disebut dengan head
   - Menjadikan node baru sebagai node pertama (head)

```
def addFront(self,item):
      newNode = Node(item)
```

```
newNode.next = self.head
self.head  = newNode
```



3. Konsep traversal dalam linked list adalah proses mengunjungi setiap node dalam linked list secara berurutan mulai dari pertama hingga akhir. tujuan dari traversal adalah untuk mengakses dan memodifikasi data pada setiap node dalam linked list.



2 Implementasi

2.1 Traversal - Menampilkan data

Tambahkan method pada class LinkedList yang sudah dibuat untuk menampilkan data-data yang terdapat pada linked list, seperti contoh pada Gambar 1

```
myList=LinkedList()
```

```
1 myList.addFront(12)
2 myList.addFront(30)
3 myList.addFront(25)
4 myList.addFront(7)
5 myList.addFront(19)
6 myList.display()
```

```
[  19 7 25 30 12  ]
```

Gambar 1: Display Linked List

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

    def getItem(self):
        return self.item

    def getNext(self):
        return self.next

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

    def addFront(self,item):
        newNode = Node(item)
        newNode.next = self.head
        self.head  = newNode

# 2.2 Traversal - Menampilkan Data
    def display(self):
        current = self.head
        dis = "[ "
```

```
        while current != None :
            dis += str(current.getItem()) + " "
            current = current.getNext()
        print( dis + ']')

# 2.2 Traversal - Menambahkan Data
    def addRear(self,item):
        newNode = Node(item)
        if self.head == None:
            self.head  = newNode
        else:
            current = self.head
            while current.getNext() != None :
                current = current.getNext()
            current.next = newNode

# 2.3 Traversal - Searching
    def search(self,find):
        current = self.head
        if current == None :
            print(f'Data {find} tidak diketemukan')
        else:
            i=0
            temp=0
            while current != None :
                if current.getItem() == find :
                    print(f'{find} berada di Node ke- {i} ')
                    temp += 1
                i+=1
                current = current.getNext()
            if temp == 0 :
                print(f'Data {find} tidak diketemukan')
```

```
# 2.1 Traversal - Menampilkan Data

myList = LinkedList()
myList.addFront(12)
myList.addFront(30)
myList.addFront(25)
myList.addFront(7)
myList.addFront(19)
myList.display()
```

```
    [ 19 7 25 30 12 ]
```

2.2 Traversal - Menambah data

Tambahkan method pada class LinkedList yang sudah dibuat untuk menambahkan data (node), sedemikian hingga node baru tersebut terletak di posisi paling belakang dari linked list, seperti contoh pada Gambar 2

```
myList=LinkedList()
```

```
myList.addFront(12)
myList.addFront(30)
myList.addFront(25)
myList.addFront(7)
myList.addFront(19)
myList.addRear(8)
myList.addRear(45)
myList.display()
```

```
[  19 7 25 30 12 8 45  ]
```

```
myList=LinkedList()
```

```
myList.addRear(41)
myList.addRear(20)
myList.addRear(7)
myList.display()
myList.addFront(10)
myList.addFront(55)
myList.display()
myList.addRear(99)
myList.display()

```

```
[  41 20 7  ]
[  55 10 41 20 7  ]
[  55 10 41 20 7 99  ]
```

Gambar 2: Penambahan Node di belakang Linked List

```
# 2.2 Traversal-Menambah Data
myList = LinkedList()
```

```
myList.addFront(12)
myList.addFront(30)
myList.addFront(25)
myList.addFront(7)
myList.addFront(19)
myList.addRear(8)
```

```
7 myList.addRear(45)
8 myList.display()
```

```
    [ 19 7 25 30 12 8 45 ]
```

```
1 myList = LinkedList()
```

```
1 myList.addRear(41)
2 myList.addRear(20)
3 myList.addRear(7)
4 myList.display()
5 myList.addFront(10)
6 myList.addFront(55)
7 myList.display()
8 myList.addRear(99)
9 myList.display()
```

```
    [ 41 20 7 ]
    [ 55 10 41 20 7 ]
    [ 55 10 41 20 7 99 ]
```

2.3 Traversal - Searching

Tambahkan method searching pada class LinkedList untuk pencarian data, dan jika data diketemukan, tampilkan data berada di node keberapa dari linked list (node awal adalah node ke-0). Output dari method ini dapat dilihat pada Gambar 3

```
myList=LinkedList()
```

```
myList.addRear(41)
myList.addRear(20)
myList.addRear(7)
myList.addFront(10)
myList.addFront(55)
myList.addRear(99)
myList.addFront(41)
myList.display()
myList.search(41)
```

```
[  41 55 10 41 20 7 99  ]
41 berada di Node ke- 0
41 berada di Node ke- 3
```

```
myList.display()
myList.search(10)
myList.search(87)
```

```
[  41 55 10 41 20 7 99  ]
10 berada di Node ke- 2
Data  87  tidak diketemukan
```

Gambar 3:

Searching Data

```
1 # 2.3 Traversal - Searching
2 myList = LinkedList()
```

```
1 myList.addRear(41)
2 myList.addRear(20)
3 myList.addRear(7)
4 myList.addFront(10)
5 myList.addFront(55)
6 myList.addRear(99)
7 myList.addFront(41)
8 myList.display()
9 myList.search(41)
```

```
[ 41 55 10 41 20 7 99 ]
41 berada di Node ke- 0
41 berada di Node ke- 3
```

```
1 myList.display()
```

✓ 0s completed at 12:48 PM