LAPORAN PRAKTIKUM
SINGLE AND DOUBLE LINKED LIST
STRUKTUR DATA

Oleh:
DANUARY BIMA HAMMAM MAYFALAH
24091397007

Program Studi D4 Manajemen Informatika
Fakultas Vokasi
Universitas Negeri Surabaya
2024

1. PENDAHULUAN

A. Definisi

1. Linked link list
   Single Linked List adalah struktur data di mana setiap node hanya memiliki satu pointer yang menunjuk ke node berikutnya. Struktur ini membentuk rantai satu arah, di mana navigasi hanya dapat dilakukan dari awal ke akhir.
2. Double link list
   Double link list adalah struktur data yang terdiri dari sekumpulan elemen yang disebut node, di mana setiap node memiliki dua referensi atau pointer: satu menunjuk ke node sebelumnya (prev) dan satu lagi menunjuk ke node berikutnya (next). Ini memungkinkan traversal (penelusuran) dalam kedua arah, baik maju maupun mundur.

2. Tujuan Praktikum

1. Mempelajari apa itu linked list, termasuk jenis-jenisnya single linked list, double linked list.
2. Mengimplementasikan linked list dalam bahasa pemrograman tertentu untuk memahami cara kerja internalnya.
3. Mengembangkan keterampilan dalam manipulasi data, seperti menyisipkan elemen di posisi tertentu, menghapus elemen, dan membalikkan linked list.
4. Memahami kompleksitas waktu dan ruang dari operasi yang dilakukan pada linked list.

3. Latihan

a. Source Code

```
1   class Node:
2       def __init__(self, value):
3           self.value = value
4           self.next = None
5           self.prev = None
6
7   class DoubleLinkedList:
8       def __init__(self, value):
9           new_node = Node(value)
10          self.head = new_node
11          self.tail = new_node
12          self.length = 1
```

b. Fungsi append dalam Linked List digunakan untuk menambahkan elemen baru di akhir (tail) dari daftar

```
14      def append(self, value):
15          new_node = Node(value)
16          if self.length == 0:
17              self.head = new_node
18              self.tail = new_node
19          else:
20              new_node.prev = self.tail
21              self.tail.next = new_node
22              self.tail = new_node
23          self.length += 1
```

c. Fungsi pop dalam Linked List digunakan untuk menghapus elemen terakhir (tail) dari daftar dan mengembalikan nilainya

```
25      def pop(self):
26          if self.length == 0:
27              return None
28          temp = self.tail
29          if self.length == 1:
30              self.head = None
31              self.tail = None
32          else:
33              self.tail = temp.prev
34              self.tail.next = None
35              temp.prev = None
36          self.length -= 1
37          return temp.value
```

d. Fungsi prepend dalam Linked List digunakan untuk menambahkan elemen di awal (head) daftar

```
39      def prepend(self, value):
40          new_node = Node(value)
41          if self.length == 0:
42              self.head = new_node
43              self.tail = new_node
44          else:
45              new_node.next = self.head
46              self.head.prev = new_node
47              self.head = new_node
48          self.length += 1
```

e. Fungsi insert dalam Linked List digunakan untuk menyisipkan elemen (node) ke dalam posisi tertentu di dalam daftar

```
    def insert(self, index, value):
        if index < 0 or index > self.length:
            return False
        if index == 0:
            self.prepend(value)
            return True
        if index == self.length:
            self.append(value)
            return True
        new_node = Node(value)
        temp = self.head
        for _ in range(index - 1):
            temp = temp.next
        new_node.next = temp.next
        new_node.prev = temp
        if temp.next:
            temp.next.prev = new_node
        temp.next = new_node
        self.length += 1
        return True
```

f. Fungsi remove dalam Linked List digunakan untuk menghapus elemen (node) dari posisi tertentu dalam daftar

```
    def remove(self, index):
        if index < 0 or index >= self.length:
            return None
        if index == 0:
            removed_node = self.head
            self.head = self.head.next
            if self.head:
                self.head.prev = None
            self.length -= 1
            return removed_node.value
        temp = self.head
        for _ in range(index - 1):
            temp = temp.next
        removed_node = temp.next
        temp.next = removed_node.next
        if removed_node.next:
            removed_node.next.prev = temp
        self.length -= 1
        return removed_node.value
```

g. Tampilkan hasil

```
    def print_list(self):
        temp = self.head
        while temp:
            print(temp.value)
            temp = temp.next
        print("None")
```

4. Tugas

1. Buatlah program Notasi Big O dibawah ini:
   a. O(2n)
   b. $O(n+n^2)$

2. Buatlah program Double Linked List untuk mencetak list dari “NIM” dan “Nama Mahasiswa”, dimana program tersebut harus memuat fungsi dibawah ini:
   a) Fungsi Append
   b) Fungsi Pop
   c) Fungsi Prepend
   d) Fungsi Insert
   e) Fungsi Remove

Jawaban

1. A. O(2n)

Source Code

```
# Nomer 1 A
def print_item(n):
    for i in range(n):
        print(f"Loop 1: {i}")

    for j in range(n):
        print(f"Loop 2: {j}")

print_item(5)
```

Output

```
Loop 1: 0
Loop 1: 1
Loop 1: 2
Loop 1: 3
Loop 1: 4
Loop 2: 0
Loop 2: 1
Loop 2: 2
Loop 2: 3
Loop 2: 4
```

Penjelasan code

```
def print_item(n):
```

fungsi bernama print_item yang menerima satu parameter, yaitu **n**. Parameter ini akan menentukan berapa kali loop akan dijalankan.

```
    for i in range(n):
        print(f"Loop 1: {i}")
```

for yang akan berjalan dari 0 hingga (n-1). Fungsi **range(n)** menghasilkan urutan angka dari 0 hingga (n-1).

print(f"Loop 1: {i}"):

Di dalam loop, kita mencetak string yang menunjukkan nilai **i** saat ini. **f"Loop 1: {i}"** adalah f-string yang memungkinkan kita menyisipkan nilai **i** ke dalam string.

```
6       for j in range(n):
7           print(f"Loop 2: {j}")
```

for j in range(n)::
Ini adalah loop kedua yang juga berjalan dari 0 hingga (n-1), mirip dengan loop pertama, tetapi menggunakan variabel **j**.
print(f"Loop 2: {j}"):
Di dalam loop ini, kita mencetak string yang menunjukkan nilai **j** saat ini.

```
9   print_item(5)
```

Memanggil Print_item dengan n=5, yang berarti kedua loop akan mencetak angka dari 0 hingga 4 dua kali.

2. O(n+n2)

Source code

```
11  def Print_item(n):
12      for i in range(n):
13          print(i)
14      for i in range(n):
15          for j in range(n):
16              print(i, j)
17
18  Print_item(5)
```

Output

```
0
1
2
3
4
0 0
0 1
0 2
0 3
0 4
1 0
1 1
1 2
1 3
1 4
2 0
2 1
2 2
2 3
2 4
3 0
3 1
3 2
3 3
3 4
4 0
4 1
4 2
4 3
4 4
```

Penjelasan

```
11    def Print_item(n):
```

fungsi bernama print_item yang menerima satu parameter, yaitu **n**. Parameter ini akan menentukan berapa kali loop akan dijalankan.

```
12        for i in range(n):
13            print(i)
```

Loop pertama.

Mencetak nilai iterasi pada loop pertama.

```
14        for i in range(n):
15            for j in range(n):
```

Loop kedua

Loop bersarang untuk setiap nilai I, loop j akan berjalan dari 0 untuk setiap I..

```
16                print(i, j)
```

Mencetak nilai I dan j dengan bentuk pasangan.

```
18    Print_item(5)
```

Memanggil Print_item dengan n=5.

2.Source code

a) Fungsi append

```
13        def append(self, nim, nama):
14            new_node = Node(nim, nama)
15            if not self.head:
16                self.head = new_node
17                self.tail = new_node
18            else:
19                self.tail.next = new_node
20                new_node.prev = self.tail
21                self.tail = new_node
```

b) Fungsi pop

```
23        def pop(self):
24            if not self.tail:
25                return None
26            popped_node = self.tail
27            if self.tail.prev:
28                self.tail = self.tail.prev
29                self.tail.next = None
30            else:
31                self.head = None
32                self.tail = None
33            return popped_node
```

c) Fungsi prepend

```
35      def prepend(self, nim, nama):
36          new_node = Node(nim, nama)
37          if not self.head:
38              self.head = new_node
39              self.tail = new_node
40          else:
41              new_node.next = self.head
42              self.head.prev = new_node
43              self.head = new_node
```

d) Fungsi insert

```
45      def insert(self, index, nim, nama):
46          if index == 0:
47              self.prepend(nim, nama)
48              return
49          new_node = Node(nim, nama)
50          current = self.head
51          for _ in range(index - 1):
52              if current is None:
53                  raise IndexError("Index out of bounds")
54              current = current.next
55          new_node.next = current.next
56          new_node.prev = current
57          if current.next:
58              current.next.prev = new_node
59          current.next = new_node
60          if new_node.next is None:
61              self.tail = new_node
```

e) Fungsi Remove

```
63      def remove(self, nim):
64          current = self.head
65          while current:
66              if current.nim == nim:
67                  if current.prev:
68                      current.prev.next = current.next
69                  if current.next:
70                      current.next.prev = current.prev
71                  if current == self.head:
72                      self.head = current.next
73                  if current == self.tail:
74                      self.tail = current.prev
75                  return current
76              current = current.next
77          return None
```

f) Tampilkan Hasil

```
79      def print_list(self):
80          current = self.head
81          while current:
82              print(f"NIM: {current.nim}, Nama: {current.nama}")
83              current = current.next
```

g) Contoh penggunaan

```
dll = DoubleLinkedList()
dll.append("123", "Bima")
dll.append("456", "Andre")
dll.prepend("789", "Dava")
dll.insert(1, "101", "Alvin")
print("\nList setelah append, prepend, dan insert:")
dll.print_list()
print("\nList setelah pop:")
dll.print_list()
print("\nList setelah Remove:")
dll.remove("456")
dll.print_list()
```

h) Output

```
List setelah append, prepend, dan insert:
NIM: 789, Nama: Dava
NIM: 101, Nama: Alvin
NIM: 123, Nama: Bima
NIM: 456, Nama: Andre

List setelah pop:
NIM: 789, Nama: Dava
NIM: 101, Nama: Alvin
NIM: 123, Nama: Bima
NIM: 456, Nama: Andre

List setelah Remove:
NIM: 789, Nama: Dava
NIM: 101, Nama: Alvin
NIM: 123, Nama: Bima
```

Kesimpulan

Double Linked List adalah struktur data yang fleksibel dan efisien untuk operasi penyisipan dan penghapusan, tetapi memerlukan lebih banyak memori dan lebih kompleks untuk diimplementasikan dibandingkan dengan struktur data lainnya. Analisis Big O menunjukkan bahwa operasi tertentu dapat dilakukan dengan efisien (O(1)), sementara yang lain memerlukan waktu yang lebih lama (O(n)). Pemilihan antara Double Linked List dan struktur data lainnya harus didasarkan pada kebutuhan spesifik aplikasi dan jenis operasi yang paling sering dilakukan.