Algoritma dan Struktur Data 2

Modul 5
Linked List

Disusun oleh:

Dwi Intan Af'idah, S.T., M.Kom

PROGRAM STUDI TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA
TAHUN AJARAN 2020/2021

Daftar Isi

1 SINGLE & DOUBLE LINKED LIST

1.1 Target Pembelajaran

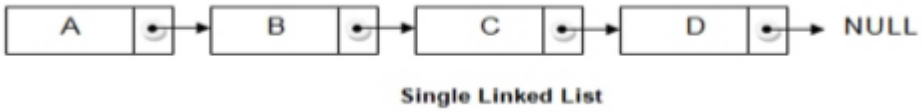
1. Menjelaskan mengenai *Single & Double Linked List*
2. Membuat abstraksi atau ilustrasi dari tipe data *Single & Double Linked List*
3. Mampu menerapkan operasi *Single Linked List Non Circular & Single Linked List Circular*
4. Mampu menerapkan operasi *Double Linked List Non Circular & Double Linked List Circular*

1.2 Dasar Teori

Struktur data array memang sederhana namun unsur-unsur pada array terkait rapat sehingga proses menggeser data di dalam array memerlukan waktu yang lebih lama. Selain itu array juga terbatas ukuran, sehingga jika ukuran diset terlalu besar namun yang dipakai sedikit akan mubazir space. Sebaliknya jika data yang dimasukkan terlalu banyak, maka bisa saja melebihi ukuran. Oleh karena itu terdapat struktur data yang fleksibel dari segi ukuran dan kaitan antar elemen, yaitu linked list.

Linked list merupakan salah satu bentuk struktur data yang berisi kumpulan data yang tersusun secara sekuensial, saling bersambungan, dinamis dan terbatas. Suatu ***linked list*** adalah satu simpul **(node)** yang dikaitkan dengan simpul yang lain.

Struktur **linked list** berupa rangkaian elemen saling berkait dimana setiap elemen dihubungkan elemen lain melalui **pointer**. **Pointer** adalah alamat elemen. Penggunaan **pointer** untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logik. Terdapat tempat yang disediakan pada satu area memori tertentu untuk menyimpan setiap **node** memiliki **pointer** yang menunjuk ke simpul berikutnya sehingga terbentuk satu untaian.



Gambar 1. Ilustrasi Linked List

Operasi-operasi Linked List

- **Insert**

  Istilah Insert berarti menambahkan sebuah simpul baru ke dalam suatu linked list.

- **IsEmpty**

  Fungsi ini menentukan apakah linked list kosong atau tidak.

- **Find First**

  Fungsi ini mencari elemen pertama dari linked list.

- **Find Next**

  Fungsi ini mencari elemen sesudah elemen yang ditunjuk now.

- **Retrieve**

  Fungsi ini mengambil elemen yang ditunjuk oleh now. Elemen tersebut lalu dikembalikan oleh fungsi.

- **Update**

  Fungsi ini mengubah elemen yang ditunjuk oleh now dengan isi dari sesuatu.

- **Delete Now**

  Fungsi ini menghapus elemen yang ditunjuk oleh now. Jika yang dihapus adalah elemen pertama dari linked list (head), head akan berpindah ke elemen berikutnya.

- **Delete Head**

  Fungsi ini menghapus elemen yang ditunjuk head. Head berpindah ke elemen sesudahnya.

- **Clear**

  Fungsi ini menghapus linked list yang sudah ada. Fungsi ini wajib dilakukan bila anda ingin mengakhiri program yang menggunakan linked list. Jika anda melakukannya, data-data yang dialokasikan ke memori pada program sebelumnya akan tetap tertinggal di dalam memori.

Tipe-tipe linked list dijelaskan pada tabel 1.

Tabel 1. Tipe Linked List

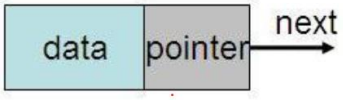
Tipe linked list	*Non Cicular / Circular*
1. Single Linked List	1.1 *Single Linked List Non Circular / Linear*
	1.2 *Single Linked List Circular*
2. Double Linked List	2.1 *Double Linked List Non Circular/ Linear*
	2.2 *Double Linked List Circular*

1.2.1 Single Linked List

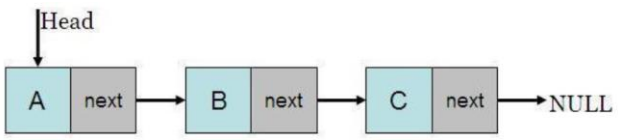
Single linked list adalah apabila hanya ada satu pointer yang menghubungkan setiap node (satu arah "next").

1.2.1.1 *Single Linked List Non Circular /* Linear *Single Linked List*

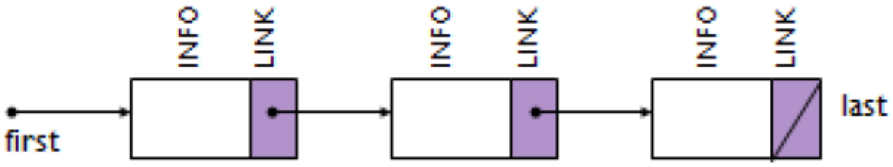
Linear single linked list terdiri dari node yang dihubungkan oleh satu reference link menunjuk ke node yang lain secara satu arah. Linear single linked list memiliki 2 atribut yaitu node first dan node last. Elemen awal diakses oleh node first dan elemen akhir diakses oleh node last. Ilustrasinya seperti gambar di bawah.



Gambar 1. Sebuah node pada Single Linked List

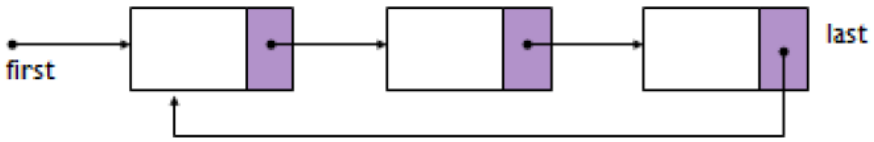


Gambar 2. Single Linked List



1.2.1.2 *Single Linked List Circular*

Circular single linked list mirip seperti linear single linked list. Bedanya jika node last pada linear menunjuk ke null, namun pada circular node last akan selalu menunjuk ke first. Sehingga linked list seakan-akan menjadi bentuk circular/lingkaran. Kondisi lainnya tidak berbeda jauh dengan linked list.



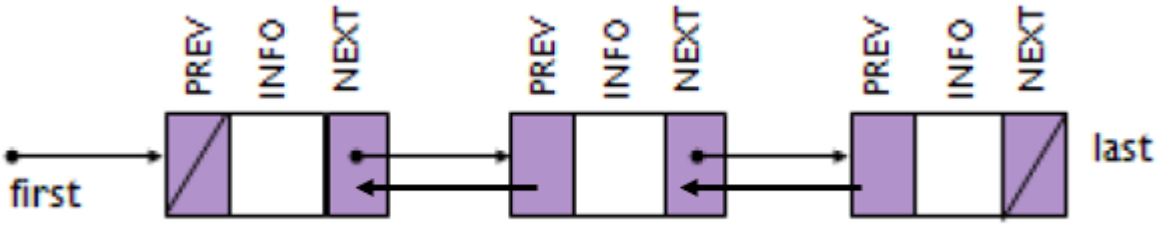
1.2.2 Double Linked List

Double Linked List adalah elemen-elemen yang dihubungkan dengan dua pointer dalam satu elemen dan list dapat melintas baik di depan atau belakang. Elemen double linked list terdiri dari tiga bagian:

1. Bagian data informasi
2. Pointer next yang menunjuk ke elemen berikutnya
3. Pointer prev yang menunjuk ke elemen sebelumnya

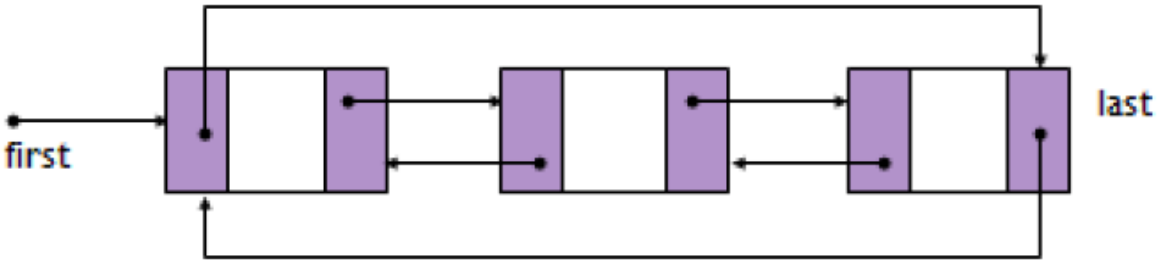
1.2.2.1 *Double Linked List Non Circular / Linear Double Linked List*

Linear double linked list terdiri dari node yang dihubungkan oleh dua reference link menunjuk ke node sebelumnya dan node sesudahnya. Linear double linked list memiliki 2 atribut yaitu node start dan node end. Elemen awal diakses oleh node start dan elemen akhir diakses oleh node end. Ilustrasinya seperti pada gambar.



1.2.2.1 *Double Linked List Circular*

Circular double linked list mirip seperti linear double linked list. Bedanya jika node last pada linear menunjuk ke null, namun pada circular node last akan selalu menunjuk ke first. Sehingga linked list seakan-akan menjadi bentuk circular/lingkaran. Kondisi lainnya tidak berbeda jauh dengan linked list yang linear double. Ilustrasinya seperti gambar di bawah.



1.3 Latihan

1.3.1 *Single Linked List Non Circular*

1. Membuat **Class Node**

```
1
2   package Slinkedlist;
3
4   public class Node {
5       protected int data;
6       protected Node link;
7
8       /* Constructor */
9       public Node()
10      {
11      link = null;
12      data = 0;
13      }
14
15      /* Constructor */
16      public Node(int d,Node n)
17      {
18      data = d;
19      link = n;
20      }
21
22      /* Function to set link to next Node */
23      public void setLink(Node n)
24      {
25      link = n;
26      }
27
28      /* Function to set data to current Node */
29      public void setData(int d)
30      {
31      data = d;
32      }
33
34      /* Function to get link to next node */
35      public Node getLink()
36      {
37      return link;
38      }
39
40      /* Function to get data from current Node */
41      public int getData()
42      {
43      return data;
44      }
45  }
```

2. Membuat **Class SinglyLinkedList**

```
public class SinglyLinkedList {
    protected Node first;
    protected Node last;
    public int size;

    /* Constructor */
    public SinglyLinkedList()
    {
    first = null;
    last = null;
    size = 0;
    }

    //method untuk mengecek apakah linked list kosong atau tidak
    public boolean isEmpty()
    {
    return first == null;
    }

    //method untuk mengembalikan ukuran linked list sekarang
    public int getSize()
    {
    return size;
    }

    //method untuk memasukkan node di awal linked list
    public void insertAwal(int val)
    {

    //buat satu node baru
    Node nptr = new Node(val, null);
    if(first == null) //jika linked list masih kosong
    {
        first = nptr;
        last = first;
    }

    else
    {
        nptr.setLink(first);
        first = nptr;
    }
    //tambah ukuran linked list
    size++;
    }

    //method untuk memasukkan node di akhir linked list
    public void insertAkhir(int val)
    {
    //buat satu node baru
    Node nptr = new Node(val, null);
    if(first == null) //jika linked list masih kosong
    {
        first = nptr;
```

```
            last = first;
        }
        else
        {
            last.setLink(nptr);
            last = nptr;
        }
        //tambah ukuran linked list
        size++;
        }

    //method untuk memasukkan node di posisi tertentu
    public void insertAtPos(int val, int pos)
    {
        //buat satu node baru
        Node nptr = new Node(val, null);
        if(pos > size)
            System.out.println("Posisi melebihi batas linked list");
        else if(pos == 1)
            insertAwal(val);
        else if(pos == size)
            insertAkhir(val);
        else
        {
            Node ptr = first;
            pos = pos - 1;
            for(int i=1; i<size; i++)
            {
                if(i == pos) //ketemu posisi
                {
                    Node tmp = ptr.getLink();
                    ptr.setLink(nptr);
                    nptr.setLink(tmp);
                    break;
                }

                ptr = ptr.getLink();
                }
                //tambah ukuran linked list
                size++;
            }
        }

    //method untuk menghapus node di awal linked list
    public void deleteAwal()
    {
        first = first.getLink();
        size--; //kurangi ukuran linked list
    }

    //method untuk menghapus node di akhir linked list
    public void deleteAkhir()
    {
        Node temp = first;
        for(int i=1; i<size-1; i++)
        temp = temp.getLink();
```

```
    last = temp;
    last.setLink(null);
    size--; //kurangi ukuran linked list
}

//method untuk menghapus node pada posisi tertentu
public void deleteAtPos(int pos)
{
    if(pos > size)
        System.out.println("Posisi node melebihi ukuran");
    else if(pos == 1)
        this.deleteAwal();
    else if(pos == size)
        this.deleteAkhir();
    else
    {
        Node ptr = first;
        pos--;
        for(int i=1; i<=pos; i++)
        {
            if(i == pos)
            {
                Node temp = ptr.getLink();
                temp = temp.getLink();
                ptr.setLink(temp);
                break;
            }
            ptr = ptr.getLink();
        }

        size--;
    }
}

//method untuk menampilkan semua unsur dalam linked list
public void display()
{
    Node ptr = first;
    while(true)
    {
        if(ptr == null)
        break;
        System.out.print(ptr.getData() + "->");
        ptr = ptr.getLink();
    }
    System.out.println();
}
}
```

3. Membuat **Class SinglyLinkedListApp**

```
1
2   package Slinkedlist;
3
4   public class SinglyLinkedListApp {
5       public static void main(String[] ar)
6       {
7           SinglyLinkedList lk = new SinglyLinkedList();
8           lk.insertAwal(10);
9           lk.display();
10          lk.insertAwal(20);
11          lk.display();
12          lk.insertAkhir(30);
13          lk.display();
14          lk.insertAwal(40);
15          lk.display();
16          lk.insertAtPos(50, 2);
17          lk.display();
18          lk.insertAkhir(60);
19          lk.display();
20          lk.deleteAtPos(2);
21          lk.display();
22      }
23  }
24
```

1.3.2 *Double Linked List Non Circular*

1. Membuat **Class Node**

```
1   package Dlinkedlist;
2
3   public class NodeDoubly
4   {
5   protected int data;
6   protected NodeDoubly next, prev;
7
8   /* Constructor */
9   public NodeDoubly()
10  {
11  next = null;
12  prev = null;
13  data = 0;
14  }
15
16  /* Constructor */
17  public NodeDoubly(int d, NodeDoubly n, NodeDoubly p)
18  {
19  data = d;
20  next = n;
21  prev = p;
22  }
23
```

```
24  /* Function to set link to next node */
25  public void setLinkNext(NodeDoubly n)
26  {
27  next = n;
28  }
29
30  /* Function to set link to previous node */
31  public void setLinkPrev(NodeDoubly p)
32  {
33  prev = p;
34  }
35
36  /* Funtion to get link to next node */
37  public NodeDoubly getLinkNext()
38  {
39  return next;
40  }
41
42  /* Function to get link to previous node */
43  public NodeDoubly getLinkPrev()
44  {
45  return prev;
46  }
```

2. Membuat **DoublyLinkedList**

```
public class DoublyLinkedList {
  protected NodeDoubly first;
    protected NodeDoubly last;
    public int size;

    /* Constructor */
    public DoublyLinkedList()
    {
    first = null;
    last = null;
    size = 0;
    }

    //method untuk mengecek apakah linked list kosong
    public boolean isEmpty()
    {
    return first == null;
    }

    //method untuk mengembalikan ukuran linked list
    public int getSize()
    {
    return size;
    }

    //method untuk menambah elemen di awal linked list
    public void insertAwal(int val)
    {
    NodeDoubly nptr = new NodeDoubly(val, null, null);
    if(first == null)
    {
        first = nptr;
        last = first;
    }
    else
    {
```

```
        first.setLinkPrev(nptr);
        nptr.setLinkNext(first);
        first = nptr;
    }
    size++;
    }

    //method untuk menambah elemen di akhir linked list
    public void insertAkhir(int val)
    {
    NodeDoubly nptr = new NodeDoubly(val, null, null);
    if(first == null)
    {
        first = nptr;
        last = first;
    }
    else
    {
    nptr.setLinkPrev(last);
    last.setLinkNext(nptr);
    last = nptr;
    }
    size++;
    }

    //method untuk menambah elemen di sekitar tengah
    public void insertAtPos(int val , int pos)
    {
    NodeDoubly nptr = new NodeDoubly(val, null, null);
    if (pos > size)
        System.out.println("Posisi melebihi batas linked list");
    else if (pos == 1)
        insertAwal(val);
    else if (pos == size)
        insertAkhir(val);
    else
    {
```

```
    NodeDoubly ptr = first;
    for (int i = 2; i <= size; i++)
    {
        if (i == pos)
        {
            NodeDoubly tmp = ptr.getLinkNext();
            ptr.setLinkNext(nptr);
            nptr.setLinkPrev(ptr);
            nptr.setLinkNext(tmp);
            tmp.setLinkPrev(nptr);
        }
            ptr = ptr.getLinkNext();
    }
    size++ ;
}
}
```

3. Membuat DoublyLinkedListApp

```
1
2   package Dlinkedlist;
3
4   public class DoublyLinkedListApp {
5       public static void main(String[] ar)
6       {
7       DoublyLinkedList dl = new DoublyLinkedList();
8       dl.insertAwal(10);
9       dl.display();
10      dl.insertAwal(20);
11      dl.display();
12      dl.insertAkhir(30);
13      dl.display();
14      dl.insertAtPos(50, 3);
15      dl.display();
16      dl.deleteAtPos(2);
17      dl.display();
18      }
19  }
20
```

1.4 Tugas Praktikum 5

1. Buatlah kode program *Single Linked List Non Circular* dengan *output* seperti di bawah ini

```
run-single:
100->
100->200->
50->100->200->
50->80->100->200->
80->100->200->
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Buatlah rangkuman mengenai perbedaan:

Tipe linked list	*Non Cicular / Circular*
1. Single Linked List	1.1 *Single Linked List Non Circular / Linear*
	1.2 *Single Linked List Circular*
2. Double Linked List	2.1 *Double Linked List Non Circular/ Linear*
	2.2 *Double Linked List Circular*