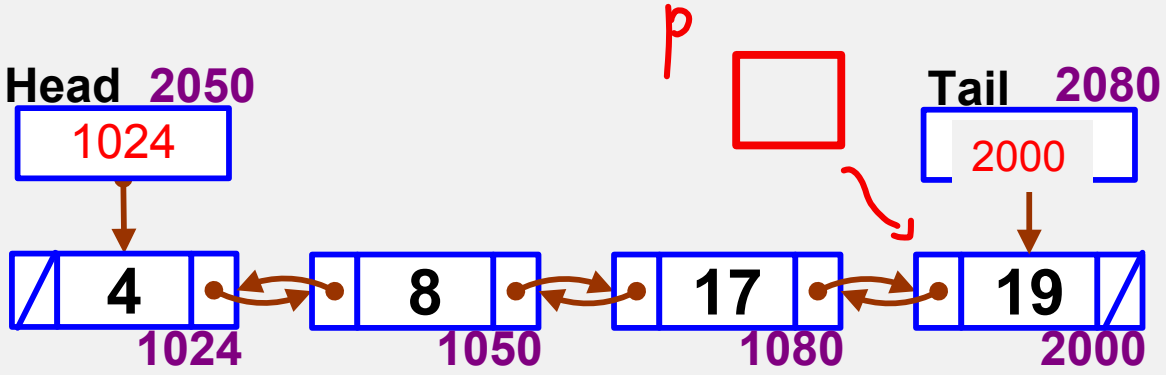
2.4 Doubly Linked Lists

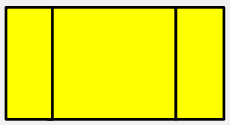
The link list that add extra field to the data structure, containing a pointer to the previous cell.




```
struct record
{    int data;
     struct record *next;
     struct record *prev;
};
```

ไป head → tail

กลับ tail → head



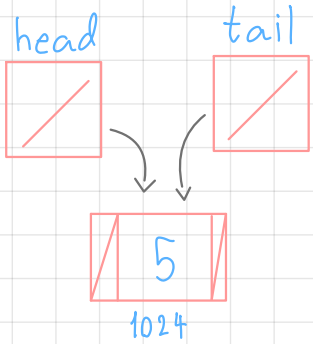
```
p = tail ;
while( p != NULL) {
    cout << p -> value;
    p = p -> next ;
}
```

2.4 Doubly Linked Lists

*

1. tail เป็นตัวแปร Global

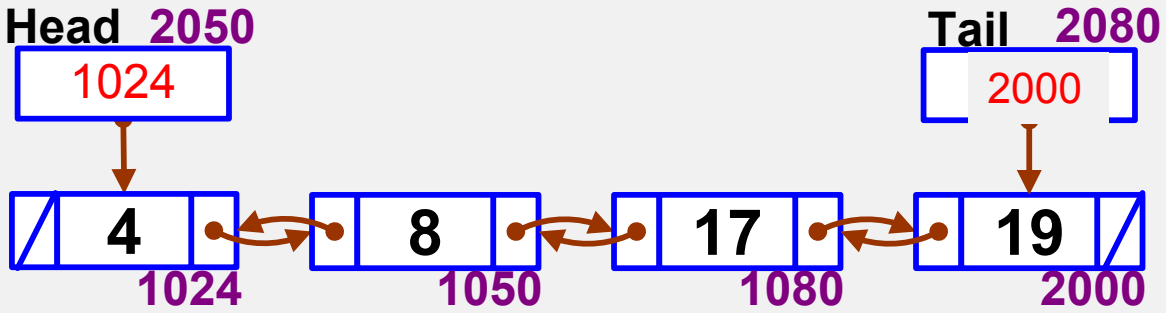
2. tail




```
head = new struct record
head -> value = data;
head -> prev = NULL;
head -> next = NULL;
tail = head
```

2.4.1 Insertion (Doubly Linklist)

1. Insert while no data in list
2. Insert first
3. Insert last
4. Insert middle



โครงโปรแกรมในการ insert

```
if(ยังไม่มีข้อมูล ?)
{     ....
}
else
 {    สร้าง node เตรียมไว้
      if ( insert ด้านหน้า ? )
      {
            ......
      }  else if( insert ด้านหลัง ? )
                {
                        ....
                }
                 else   //ตรงกลาง
                {   หาตำแหน่ง
                        ............
                }
}
```

```
if(head==NULL)
{     ....
}
else
{     สร้าง node เตรียมไว้
      if( data ....  head->value?)
      {
            ......
       }  else if(                          )
                {
                        ....
                }
                 else
                {   หาตำแหน่ง
                        ............
                }
}
```

```
                    .....
struct record
{       int value;
        struct record *prev;
        struct record *next;
};
struct record *tail=NULL;

struct record *insert(struct record *head,int data)
{       if(head==NULL)
        {       head=new struct record;
                head->value=data;
                head->next=head->prev=NULL;
                tail=head;
        }
        return head;
}
```

```
int main()
{           struct record *head=NULL;

            head=insert(head,8);
            cout << head->value<< endl;
            cout << tail->value<< endl;
}
```

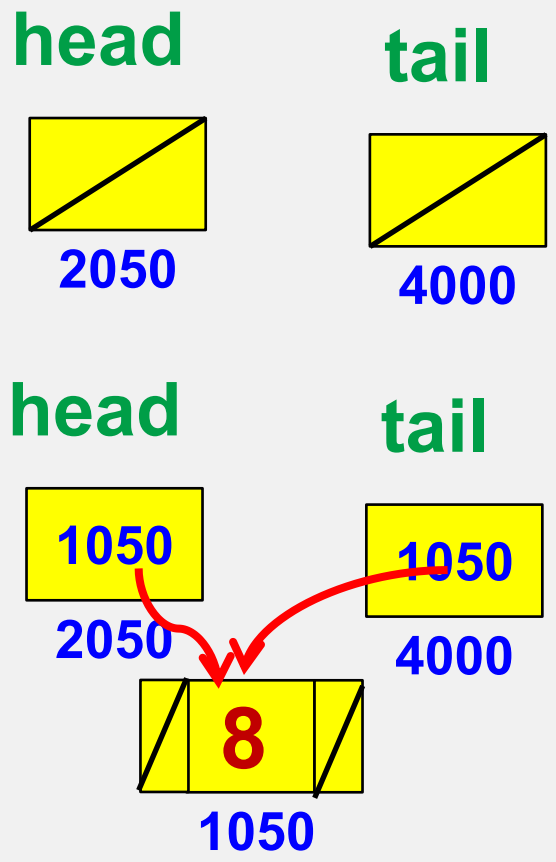
1. Insert while no data in list

8

struct record *insert(struct record *head, int data)

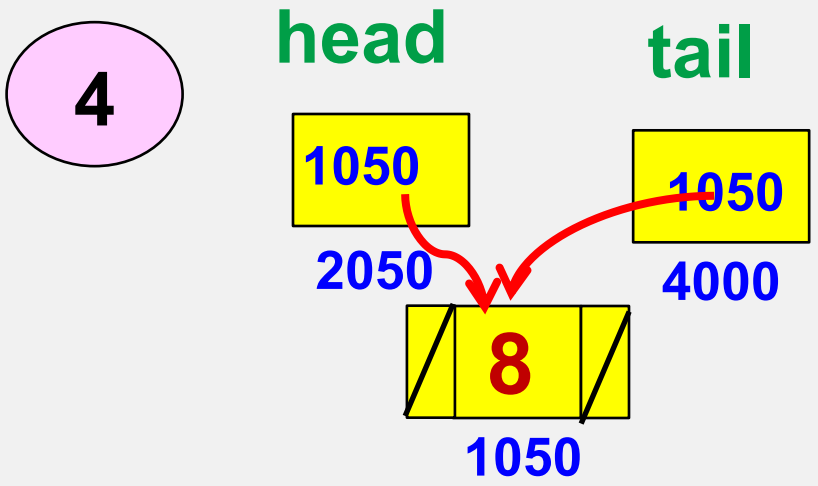
.... ประกาศตัวแปรเอง...

```
1. if( head == NULL)
2. {     head=new struct rec;
3.       head->value= data;
4.       head->next=NULL;
5.       head->prev=NULL;
6.      tail=head;
7.  }
8.  return head;
```



2. มีข้อมูลแล้วจะ Insert first

```
struct record *insert(struct record *head, int data)
.... ประกาศตัวแปรเอง...
1. if( head == NULL)
2. {     head=new struct rec;
3.       head->value= data;
4.       head->next=NULL;
5.       head->prev=NULL;
6.      tail=head;
7.  }
```

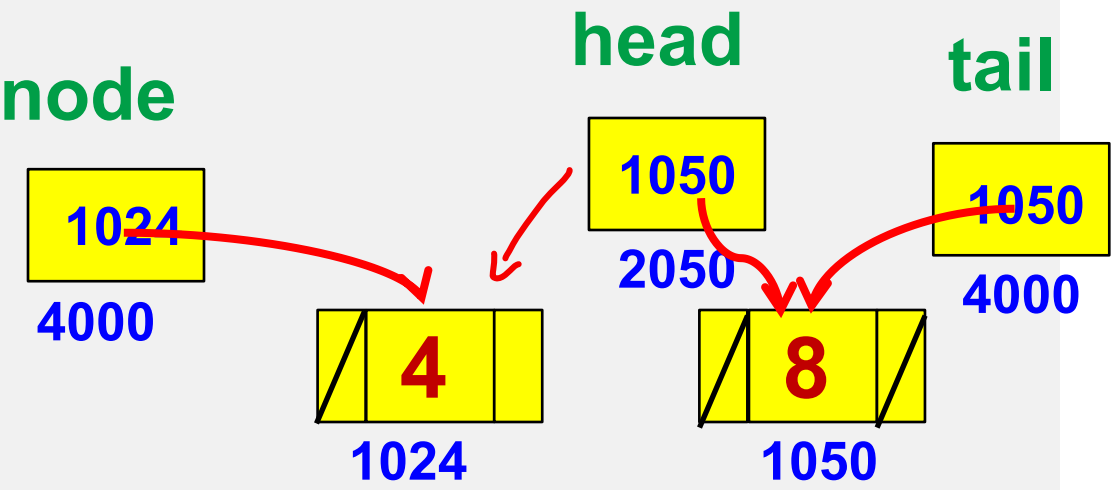


2. มีข้อมูลแล้วจะ Insert first

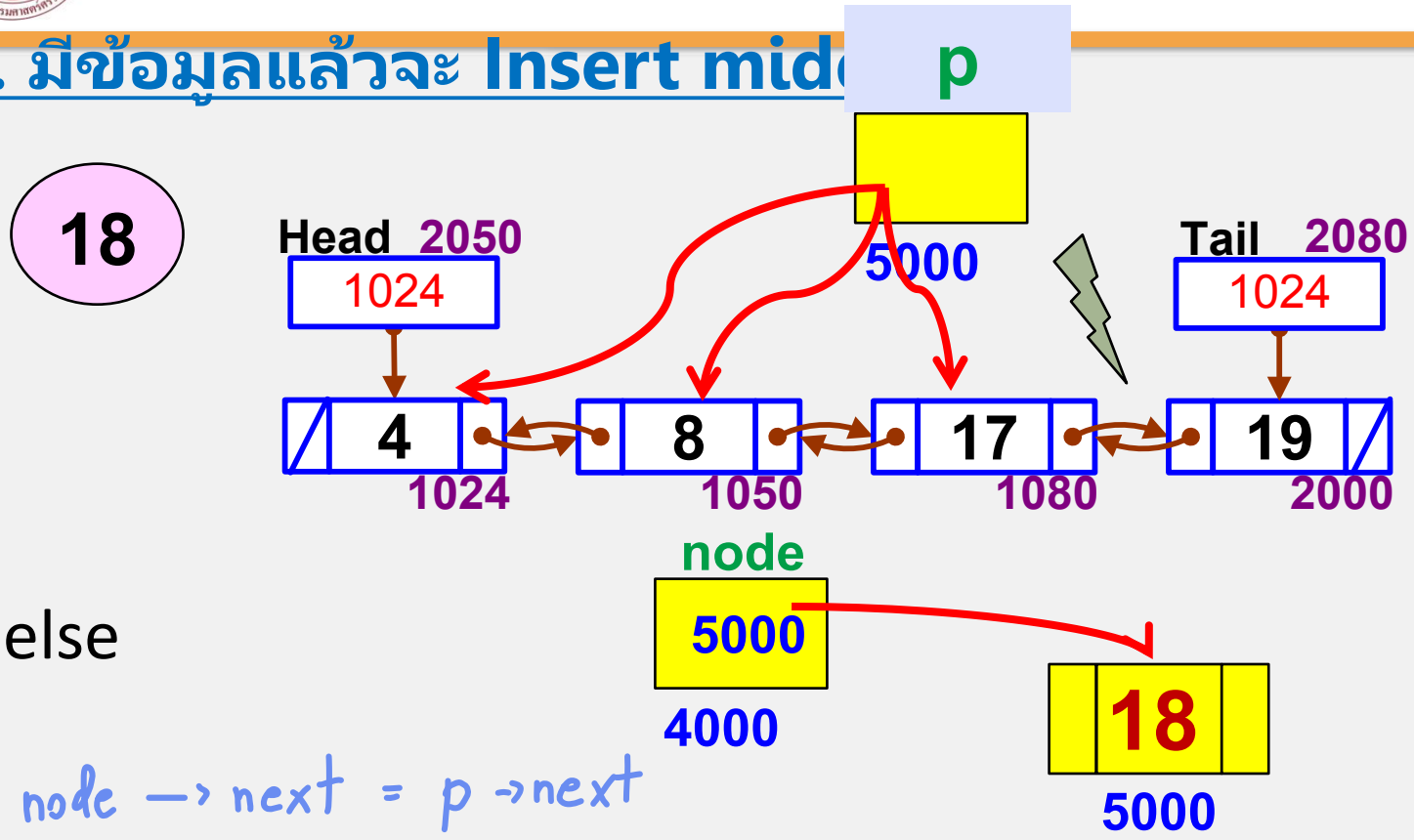
8. else
9. { **node = สร้าง node เก็บข้อมูลใหม่**
10. if (data <= temp->value)
{

4

```
node -> next = head;
head -> prev = node;
head -> prev = NULL;
head = node;
```



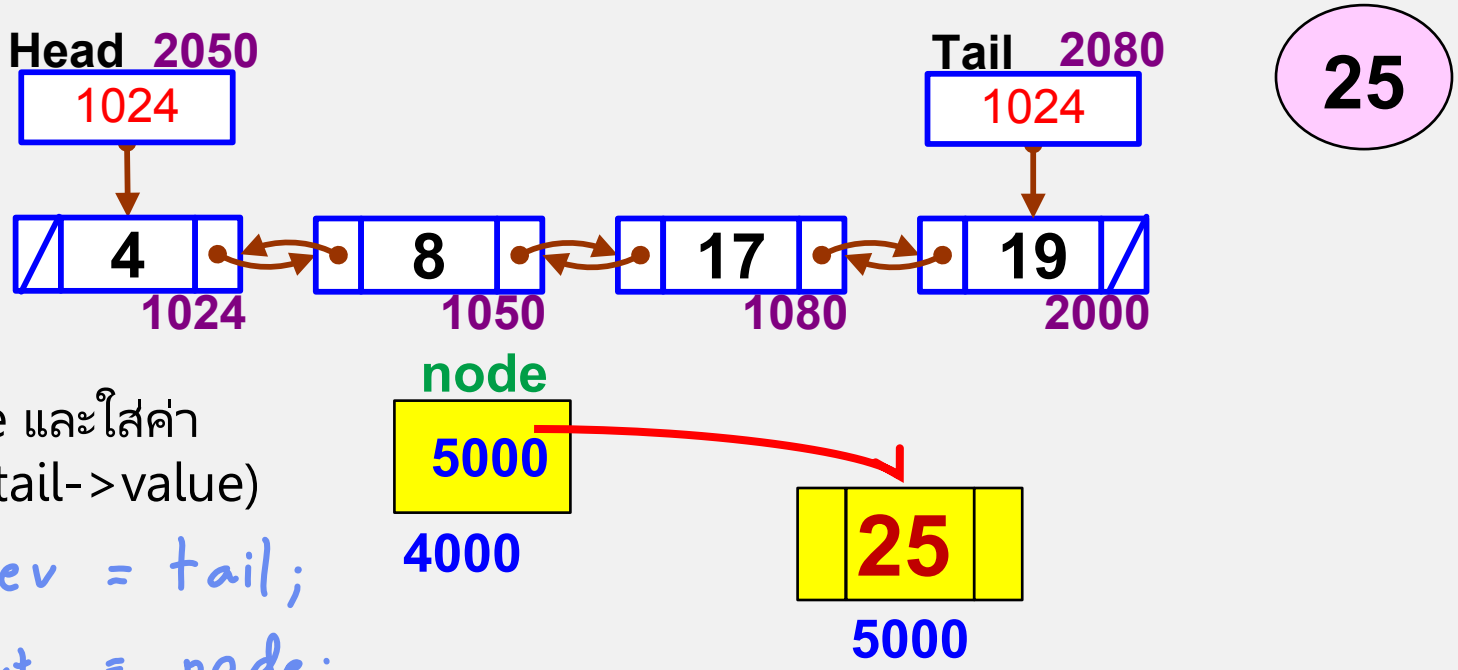
3. มีข้อมูลแล้วจะ Insert mid



else

```
node -> next = p -> next
p -> next -> prev = node;
p -> next = node;
node -> prev = p;
```

4. มีข้อมูลแล้วจะ Insert last



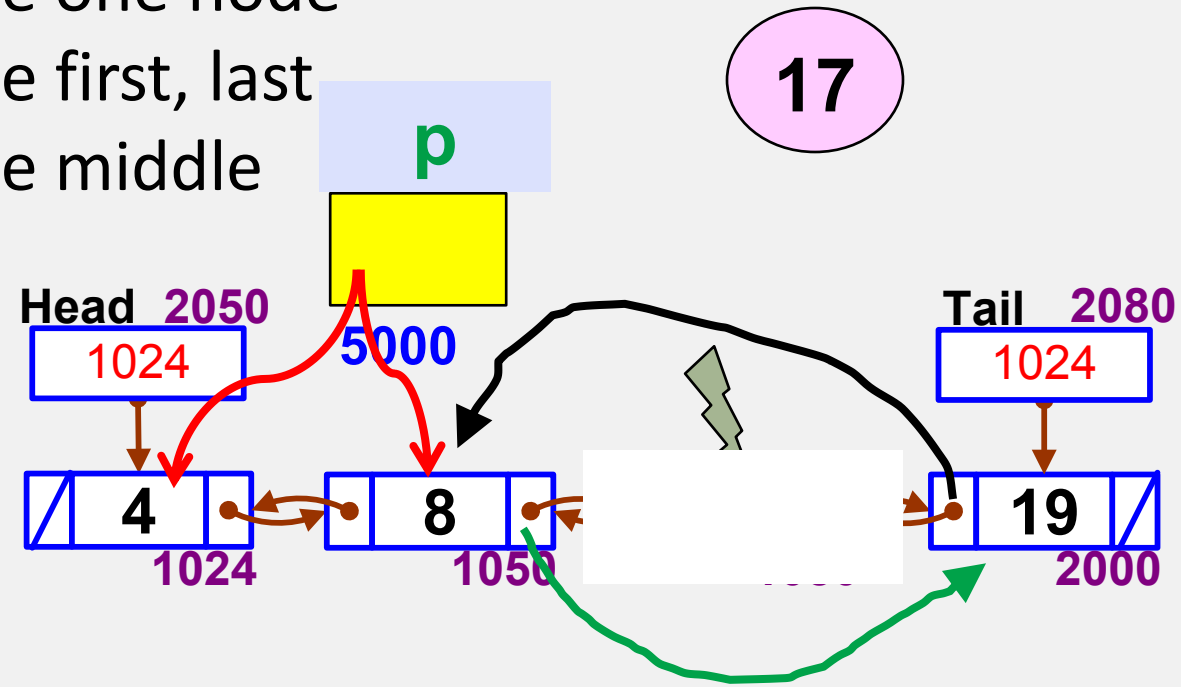
สร้าง node และใส่ค่า

if(data > tail->value)

```
node -> prev = tail;
tail -> next = node;
node -> next = NULL;
tail = node ;
```

2.4.2 Delete (Doubly Linklist)

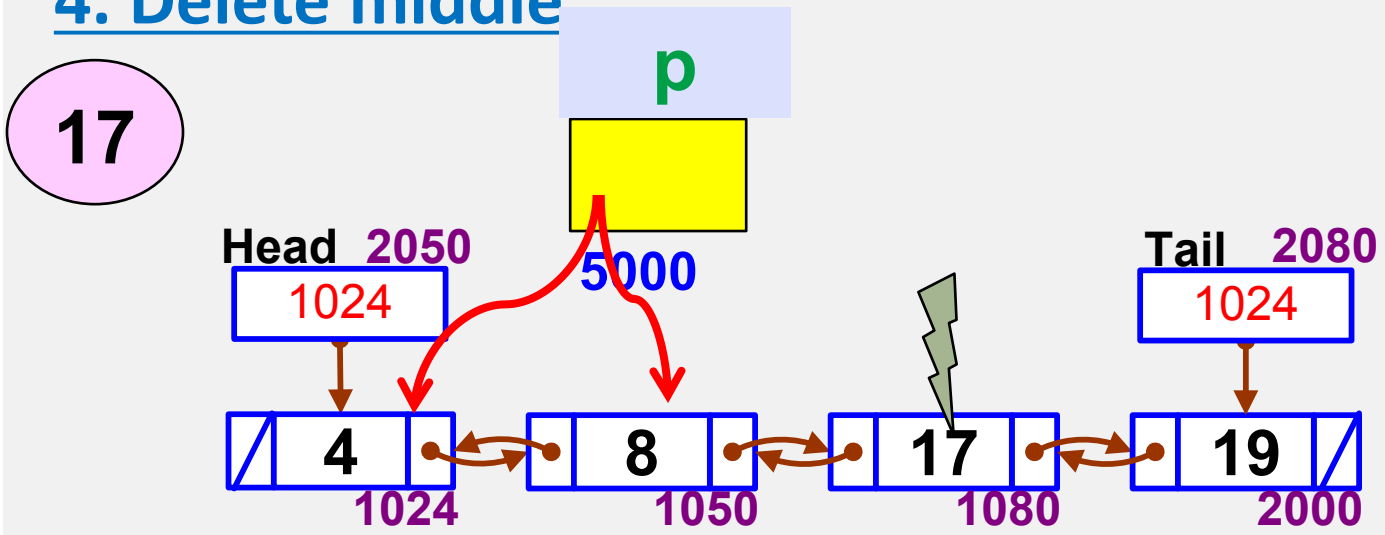
1. Delete while no data in list
2. Delete one node
3. Delete first, last
4. Delete middle



โครงโปรแกรมในการ delete

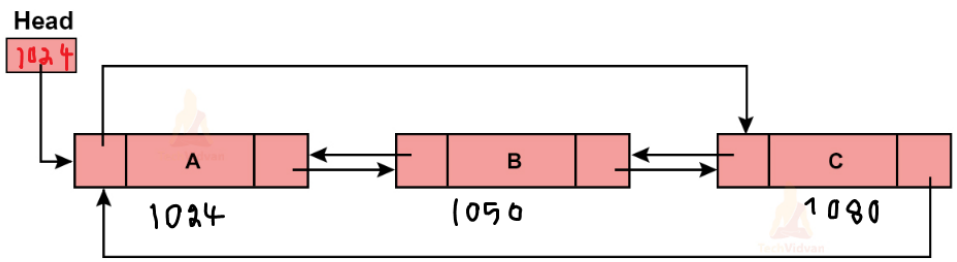
```
if(มี node เดียว ?)
{     ....
}
else
 {    if ( ลบโหนดแรก ? )
      {
            ......
      }  else if(ลบโหนดท้าย? )
                {
                        ....
                }
                 else   //ตรงกลาง
                {   หาตำแหน่ง
                        ............
                }
}
```

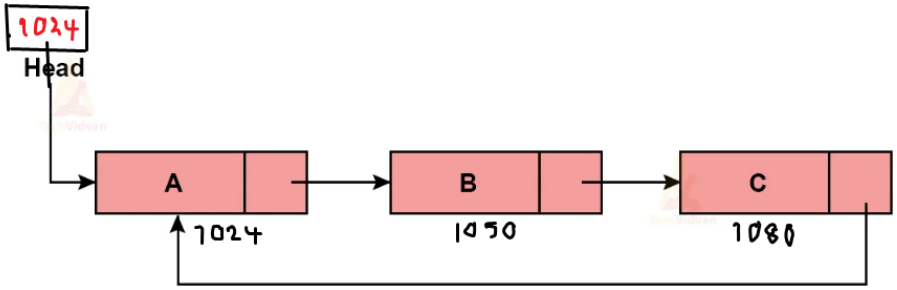
4. Delete middle

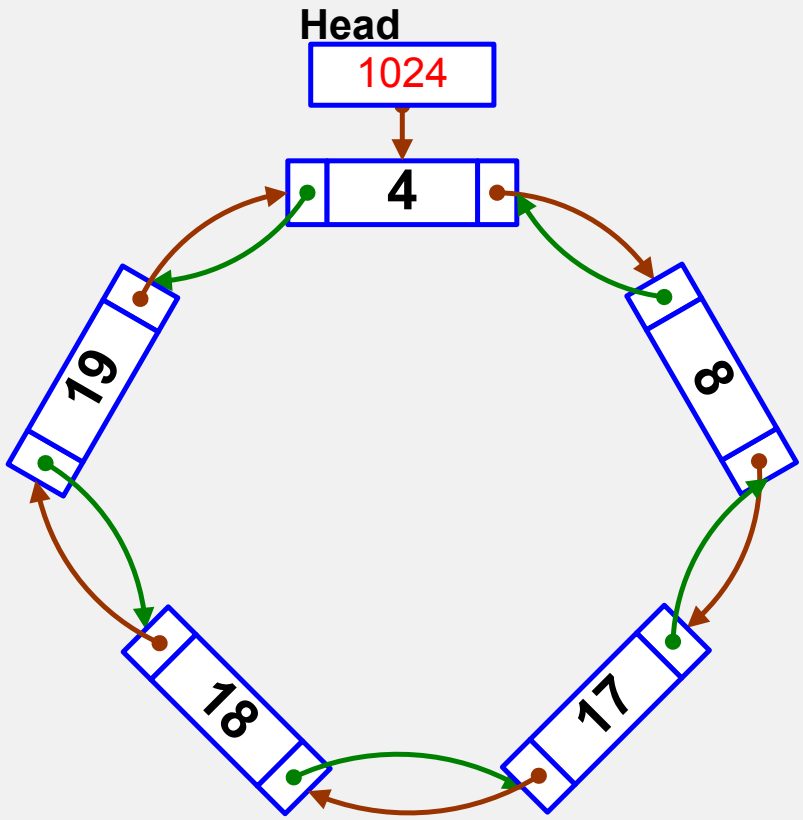


2.5 Circularly doubly linked lists

A popular convention is to have the last cell keep pointer back to the first. This can be done with or without a header (If the header present, the last cell point to it.)





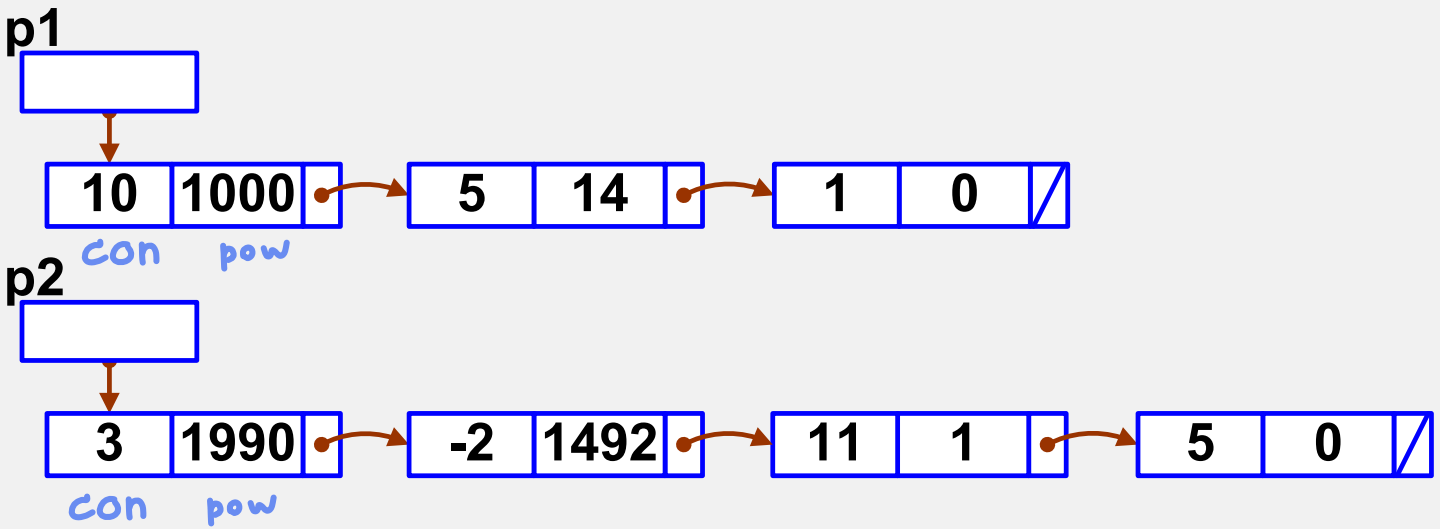

Head
1024
4
19
8
18
17

2.6 Examples

2.6.1 The polynomial ADT

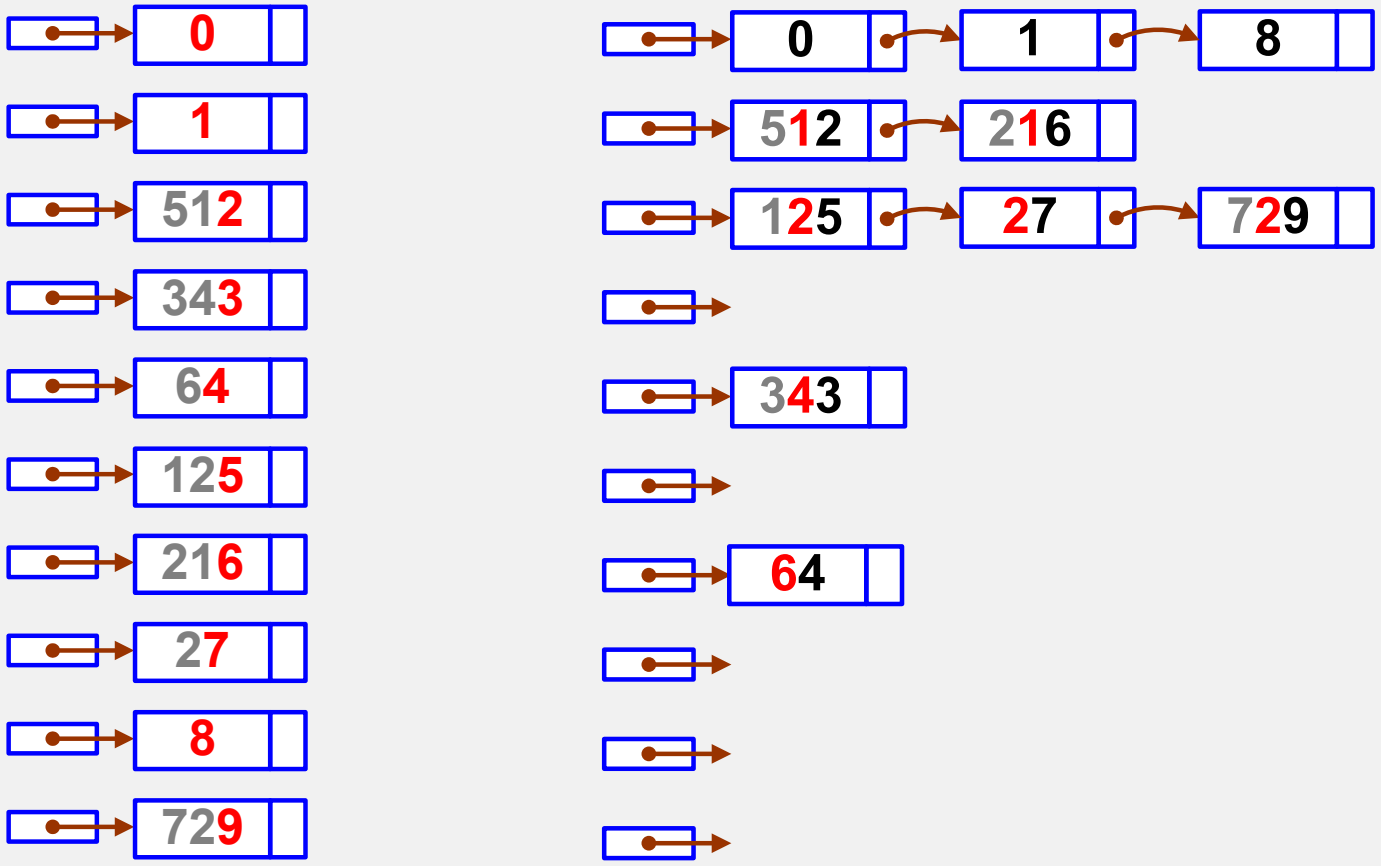
p1(x) = 10x^1000 + 5x^14 + 1

p2(x) = 3x^1990 - 2x^1492 + 11x + 5



2.6.2 Radix Sort

Input 64, 8, 216, 512, 27, 729, 0, 1, 343, 125



2.6.3 Multilists