DoublyLinkedList

May 15, 2025

[19]:
```
class DLinkedList:
    class DListNode:
        def __init__(self,p=None,data=0,n=None):
            self.Prev=p
            self.Data=data
            self.Next=n
    def __init__(self):
        self.head=None
        self.size=0

    def IsEmpty(self):
        return self.head==None
        #return self.size==0
    def show(self):
        tmp=self.head
        if tmp==None:
            print('list is empty')
        else:
            while tmp:
                print(tmp.Data,end='<=>')
                tmp=tmp.Next
            print()
    def Add2FirstByValue(self,x):
        node=self.DListNode(None,x,None)
        if not self.IsEmpty():
            node.Next=self.head
            self.head.Prev=node
        self.head=node
        self.size+=1

    def Add2FirstByRef(self,node):
        node.Next=self.head
        node.Prev=None
        if not self.IsEmpty():
            self.head.Prev=node
        self.head=node
        self.size+=1
```

```
    def Add2LastByValue(self,x):
        if self.IsEmpty():
            self.Add2FirstByValue(x)
        else:
            node=self.DListNode(None,x,None)
            tmp=self.head
            while tmp.Next!=None:
                tmp=tmp.Next
            tmp.Next=node
            node.Prev=tmp
            self.size+=1
    def Add2LastByRef(self,node):
        pass

    def SearchByVal(self,x):
        tmp=self.head
        while tmp!=None:
            if tmp.Data==x:
                break
            tmp=tmp.Next

        if tmp==None:
            return 0,None
        else:
            return 1,tmp

    def SearchByRef(self,node):
        tmp=self.head
        while tmp!=None:
            if tmp==node:
                break
            tmp=tmp.Next

        if tmp==None:
            return 0,None
        else:
            return 1,tmp
    def AddNodeAfterNode1(self,x,y):
        flag,search_node=self.SearchByVal(x)
        if flag==1:
            node=self.DListNode(search_node,y,search_node.Next)
            if search_node.Next:
                search_node.Next.Prev=node
            search_node.Next=node
            self.size+=1
            return 1,node
```

```
        else:
            return 0,None
    def AddNodeAfterNode2(self,x,node):
        flag,search_node=self.SearchByVal(x)
        if flag==1:
            node.Next=search_node.Next
            node.Prev=search_node
            if search_node.Next:
                search_node.Next.Prev=node
            search_node.Next=node
            self.size+=1
            return 1,node
        else:
            return 0,None
    def AddNodeAfterNode3(self,tmp,y):
        flag,search_node=self.SearchByRef(tmp)
        if flag==1:
            node=self.DListNode(search_node,y,search_node.Next)
            if search_node.Next:
                search_node.Next.Prev=node
            search_node.Next=node
            self.size+=1
            return 1,node
        else:
            return 0,None

    def AddNodeAfterNode4(self,node1,node2):
        flag,search_node=self.SearchByRef(node1)
        if flag==1:
            #node=self.DListNode(search_node,y,search_node.Next)
            node2.Next=search_node.Next
            node2.Prev=search_node
            if search_node.Next:
                search_node.Next.Prev=node
            search_node.Next=node
            self.size+=1
            return 1,node
        else:
            return 0,None
    def AddNodeBeforeNode1(self,x,y):
        flag,search_node=self.SearchByVal(x)
        if flag==1:
            node=self.DListNode(search_node.Prev,y,search_node)
            if search_node.Prev:
                search_node.Prev.Next=node
            else:
                self.head=node
```

```
            search_node.Prev=node
            self.size+=1
            return 1,node
        else:
            return 0,None
    def AddNodeBeforeNode2(self,x,tmp):
        pass
    def AddNodeBeforeNode3(self,tmp,y):
        pass
    def AddNodeBeforeNode4(self,node1,node2):
        pass

    def RemoveFirstNode(self):
        if not self.IsEmpty():
            tmp=self.head
            self.head=self.head.Next
            if self.head:
                self.head.Prev=None
            self.size-=1
            return 1,tmp
        else:
            return 0,None

dlist=DLinkedList()
# for i in reversed(range(10)):
#     dlist.Add2FirstByValue(i)
#     dlist.show()

for i in reversed(range(10)):
    dlist.Add2LastByValue(i)
    dlist.show()

# if dlist.IsEmpty():
#     print('dlist is empty')
# else:
#     print('#node=' ,dlist.size)
# flag,search_node=dlist.SearchByVal(40)
# if flag==1:
#     print(search_node.Prev.Data, search_node.Data,search_node.Next.Data)
# else:
#     print('not exist')

# dlist.AddNodeAfterNode1(9,90)
# dlist.show()
# dlist.AddNodeAfterNode1(4,40)
# dlist.show()
# dlist.AddNodeAfterNode1(0,10)
```

```
# dlist.show()
# flag,tmp=dlist.AddNodeAfterNode1(50,500)
# if flag==1:
#     dlist.show()
# else:
#     print(50,'not exist')

flag,tmp=dlist.AddNodeBeforeNode1(9,90)
dlist.show()
dlist.AddNodeBeforeNode1(4,40)
dlist.show()
dlist.AddNodeBeforeNode1(0,10)
dlist.show()
flag,tmp=dlist.AddNodeBeforeNode1(50,500)
# if flag==1:
#     dlist.show()
# else:
#     print(50,'not exist')
while not dlist.IsEmpty():
    dlist.RemoveFirstNode()
    dlist.show()
```

```
9<=>
9<=>8<=>
9<=>8<=>7<=>
9<=>8<=>7<=>6<=>
9<=>8<=>7<=>6<=>5<=>
9<=>8<=>7<=>6<=>5<=>4<=>
9<=>8<=>7<=>6<=>5<=>4<=>3<=>
9<=>8<=>7<=>6<=>5<=>4<=>3<=>2<=>
9<=>8<=>7<=>6<=>5<=>4<=>3<=>2<=>1<=>
9<=>8<=>7<=>6<=>5<=>4<=>3<=>2<=>1<=>0<=>
90<=>9<=>8<=>7<=>6<=>5<=>4<=>3<=>2<=>1<=>0<=>
90<=>9<=>8<=>7<=>6<=>5<=>40<=>4<=>3<=>2<=>1<=>0<=>
90<=>9<=>8<=>7<=>6<=>5<=>40<=>4<=>3<=>2<=>1<=>10<=>0<=>
9<=>8<=>7<=>6<=>5<=>40<=>4<=>3<=>2<=>1<=>10<=>0<=>
8<=>7<=>6<=>5<=>40<=>4<=>3<=>2<=>1<=>10<=>0<=>
7<=>6<=>5<=>40<=>4<=>3<=>2<=>1<=>10<=>0<=>
6<=>5<=>40<=>4<=>3<=>2<=>1<=>10<=>0<=>
5<=>40<=>4<=>3<=>2<=>1<=>10<=>0<=>
40<=>4<=>3<=>2<=>1<=>10<=>0<=>
4<=>3<=>2<=>1<=>10<=>0<=>
3<=>2<=>1<=>10<=>0<=>
2<=>1<=>10<=>0<=>
1<=>10<=>0<=>
10<=>0<=>
0<=>
```

```
list is empty
```

[ ]:

[ ]: