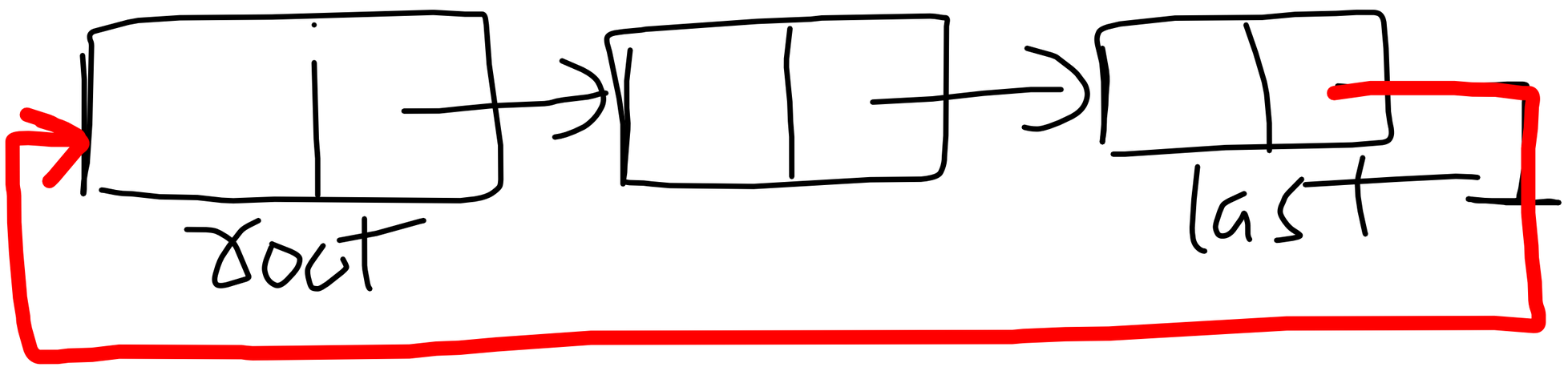
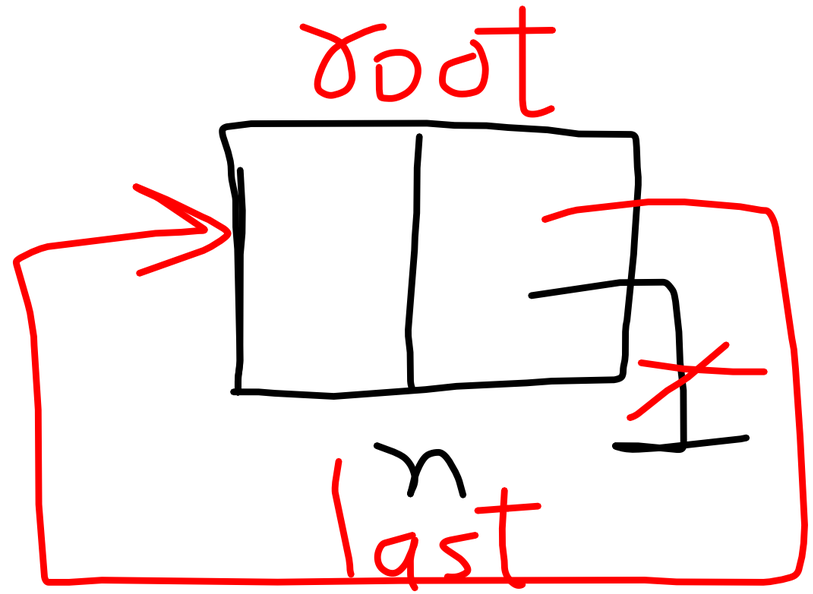
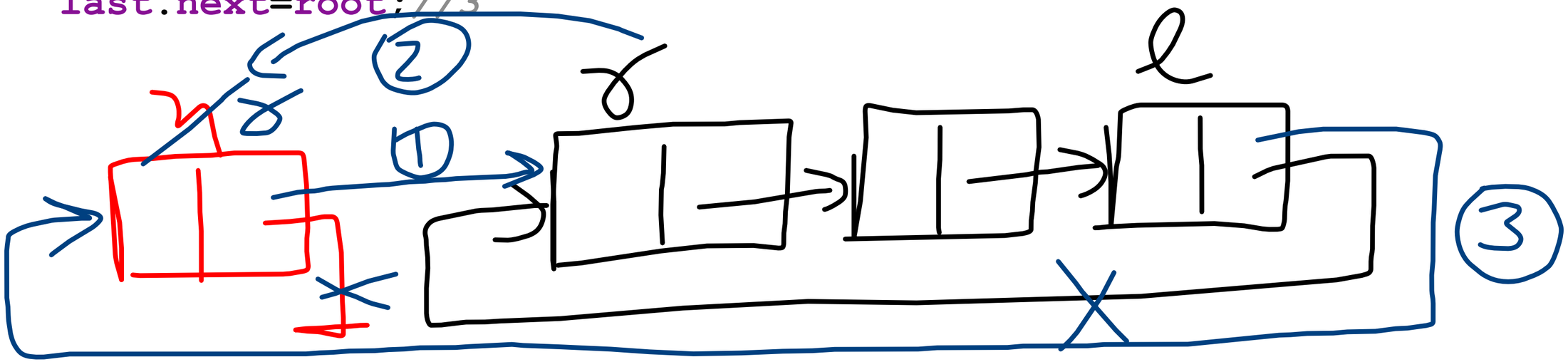
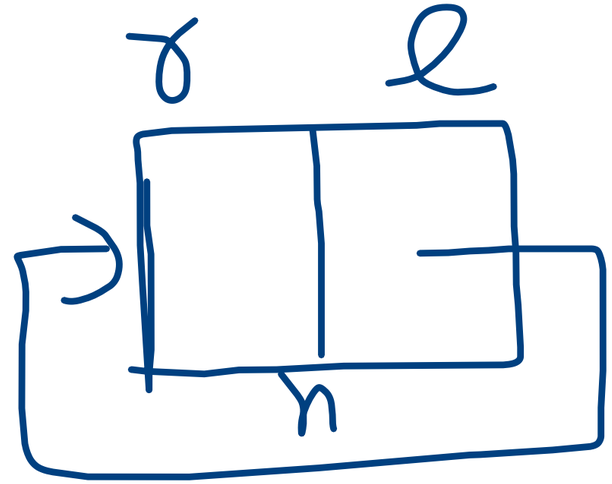

root
last

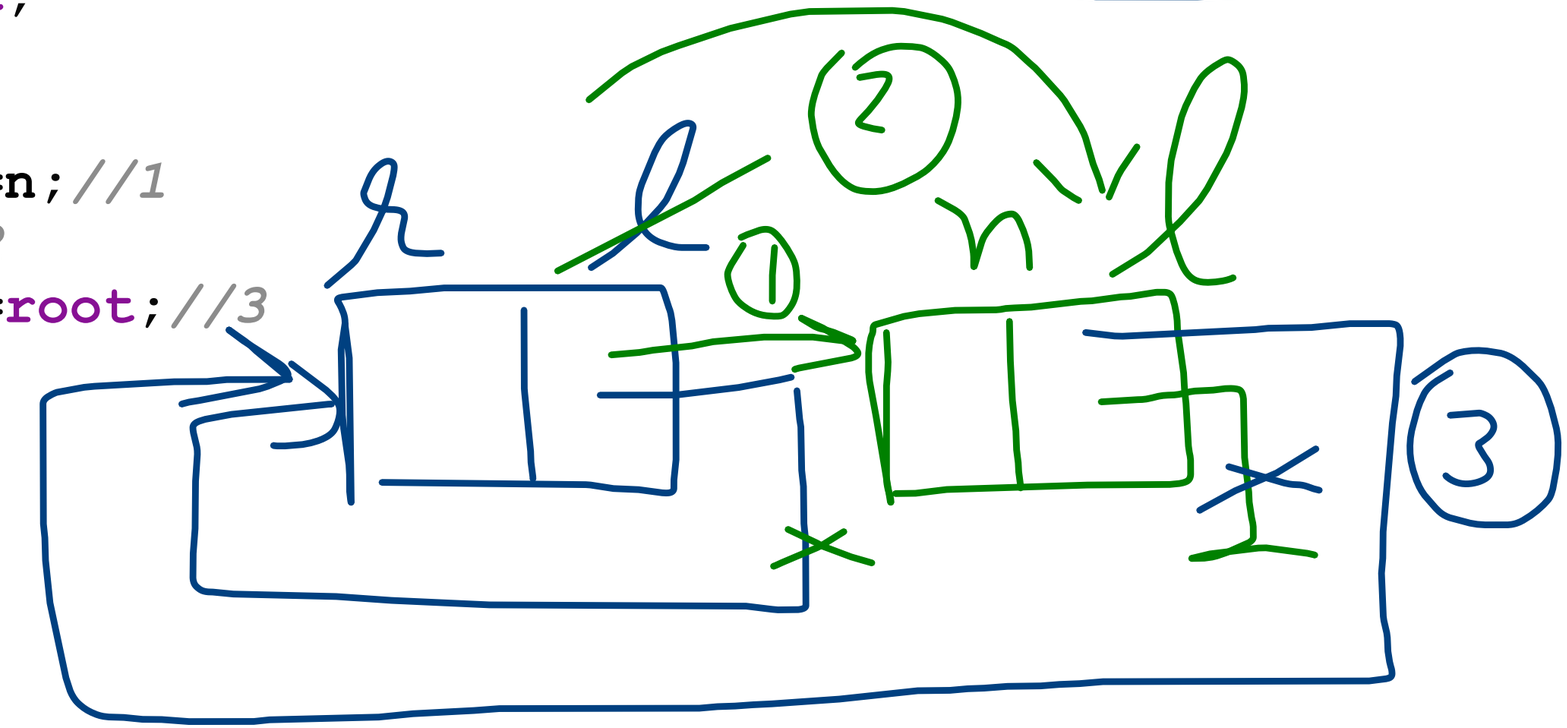
```java
void insert_left(int data) {
    Node n = new Node(data);//created node
    if (root == null)//no root
    {
        root = last=n;//1st becomes root
        last.next=root;
    }
    else {
        n.next = root;//1
        root = n;//2
        last.next=root;//3
    }
}
```



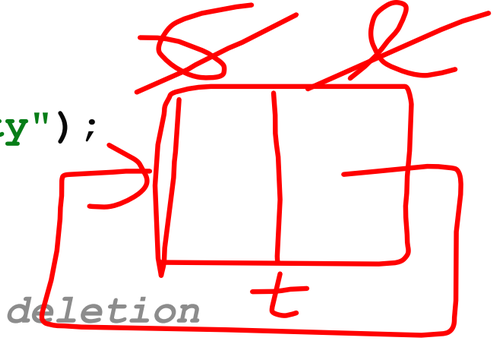


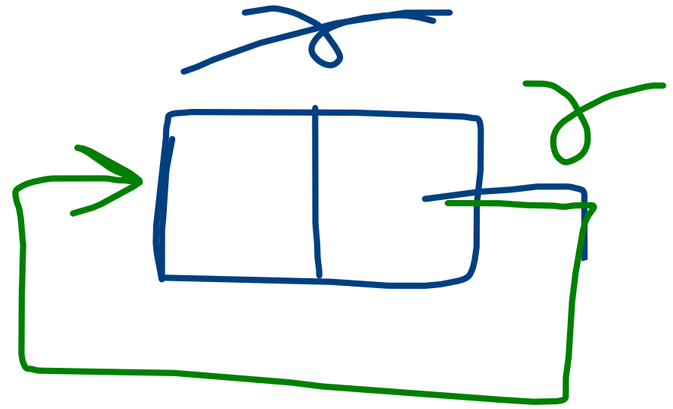
```
void insert_right(int data) {
    Node n = new Node(data);//created node
    if (root == null)//no root
    {
        root = last=n;//1st becomes root
        last.next=root;
    }
    else {
            last.next=n;//1
            last=n;//2
            last.next=root;//3
    }
}
```

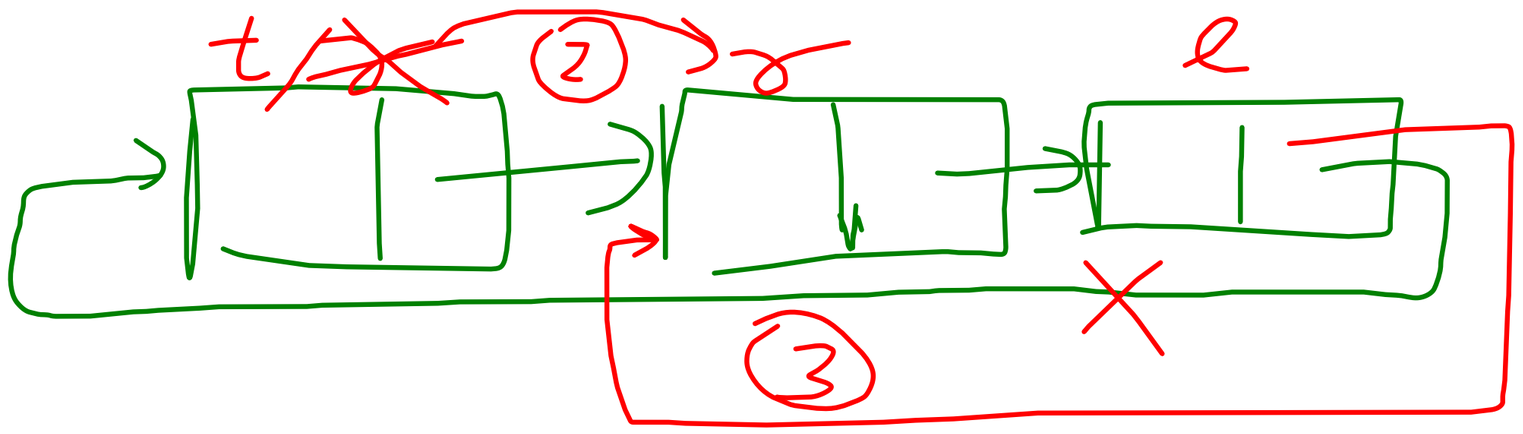




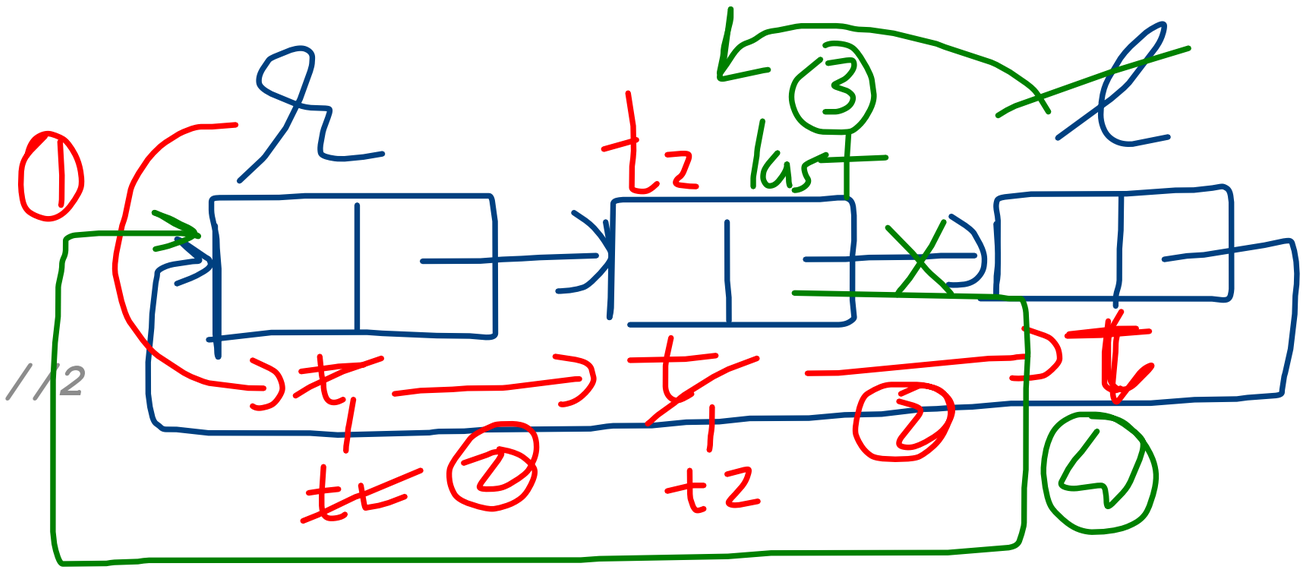
```
void delete_left() {
    if (root == null)//no root
        System.out.println("List is empty");
    else {
        Node t = root;//1
        if (root == last)//single node
            root = last = null;//manual deletion
        else {
            root = root.next;//2
            last.next = root;//3
    }
            System.out.println("Deleted:" + t.data);//3 response message of
deletion
        }
}
```





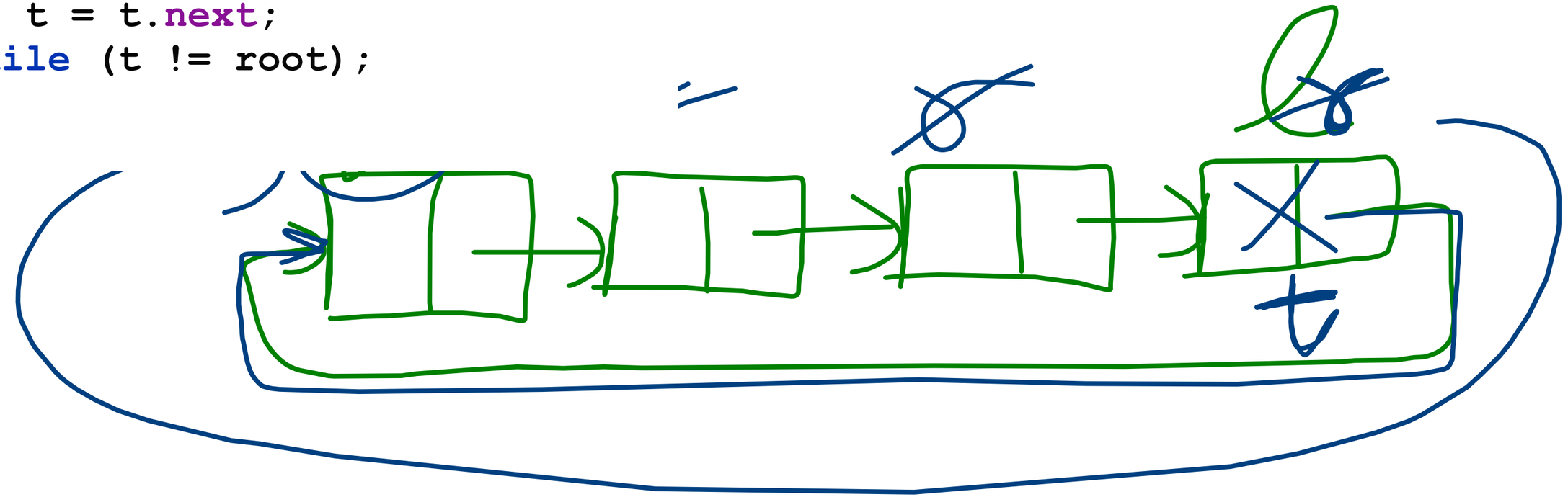


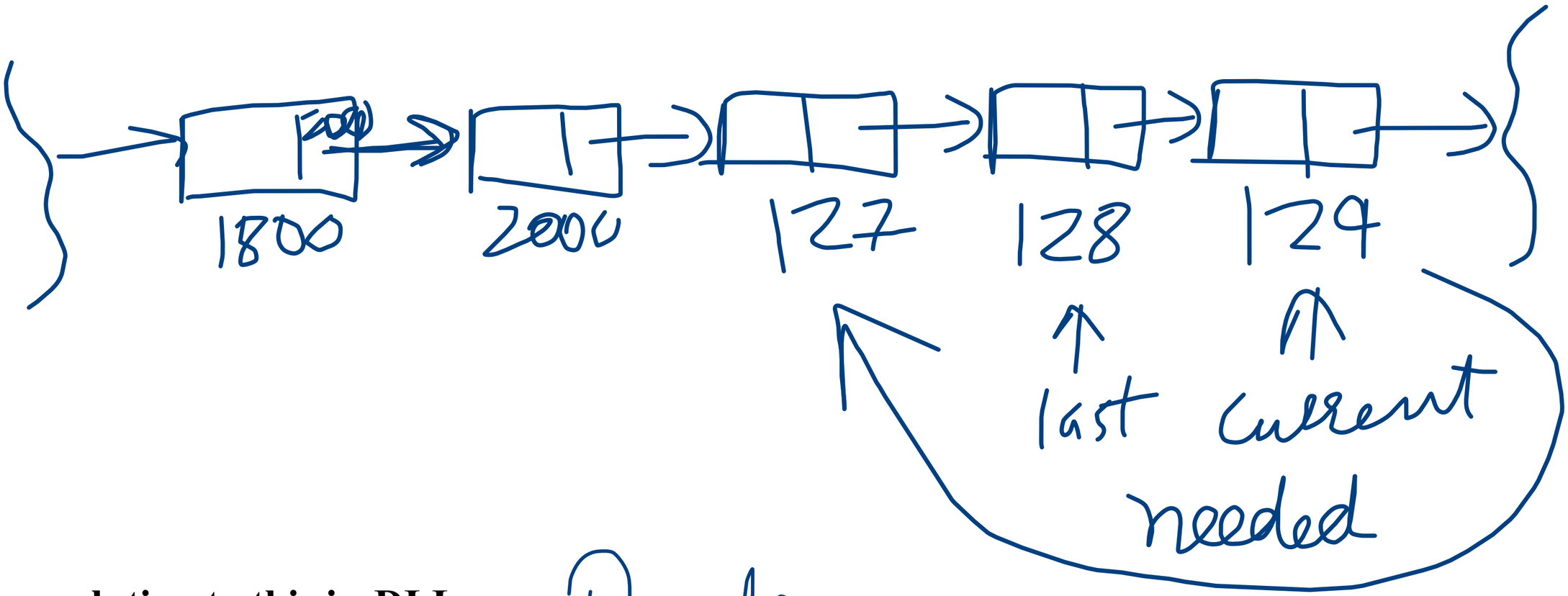
```
void delete_right() {
    if (root == null)//no root
        System.out.println("List is empty");
    else {
        Node t, t2;
        t = t2 = root;//1
    if(root==last)
        root=last=null;
    else
    {
        while (t != last)//2
        {
            t2 = t;
            t = t.next;
        }
        last=t2;//3
     last.next=root;//4
      }
        System.out.println("Deleted:" + t.data);//3 response
message of deletion
                                    (t.next).data
    }
}
```



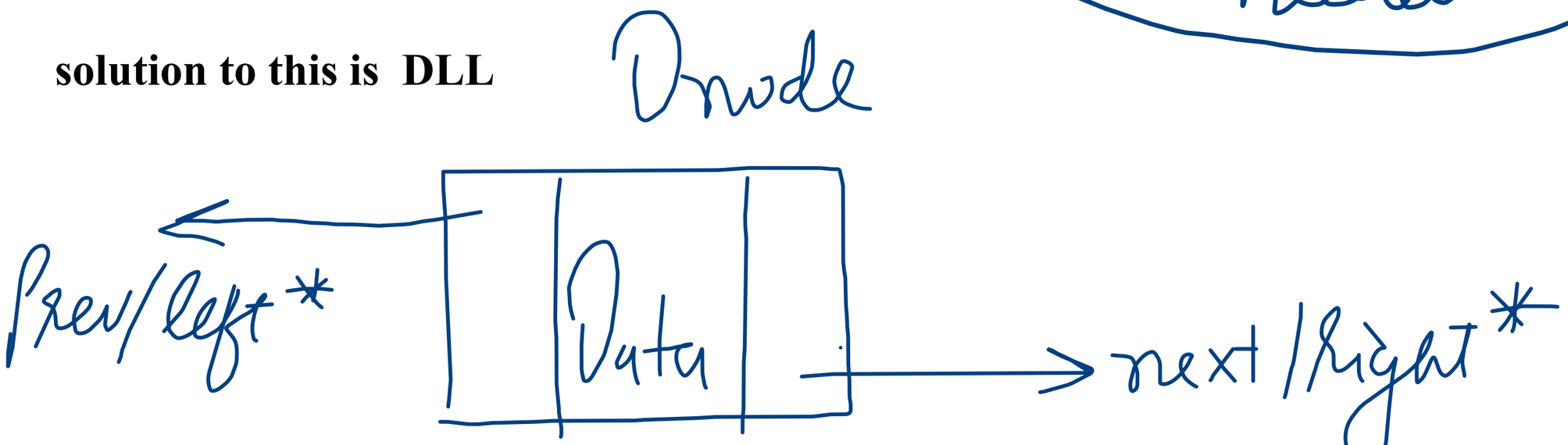
```
void print_list() {
    if (root == null)//no root
        System.out.println("List is
empty");
    else {
        Node t = root;//1
        do{
            System.out.print("| " +
t.data + " |->");
            t = t.next;
        }while (t != root);
    }
}
```

start from root print
till root not reached





solution to this is DLL

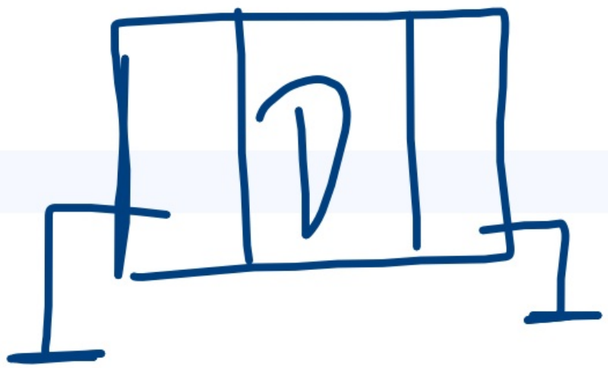


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help

_Structures_VITA Version control

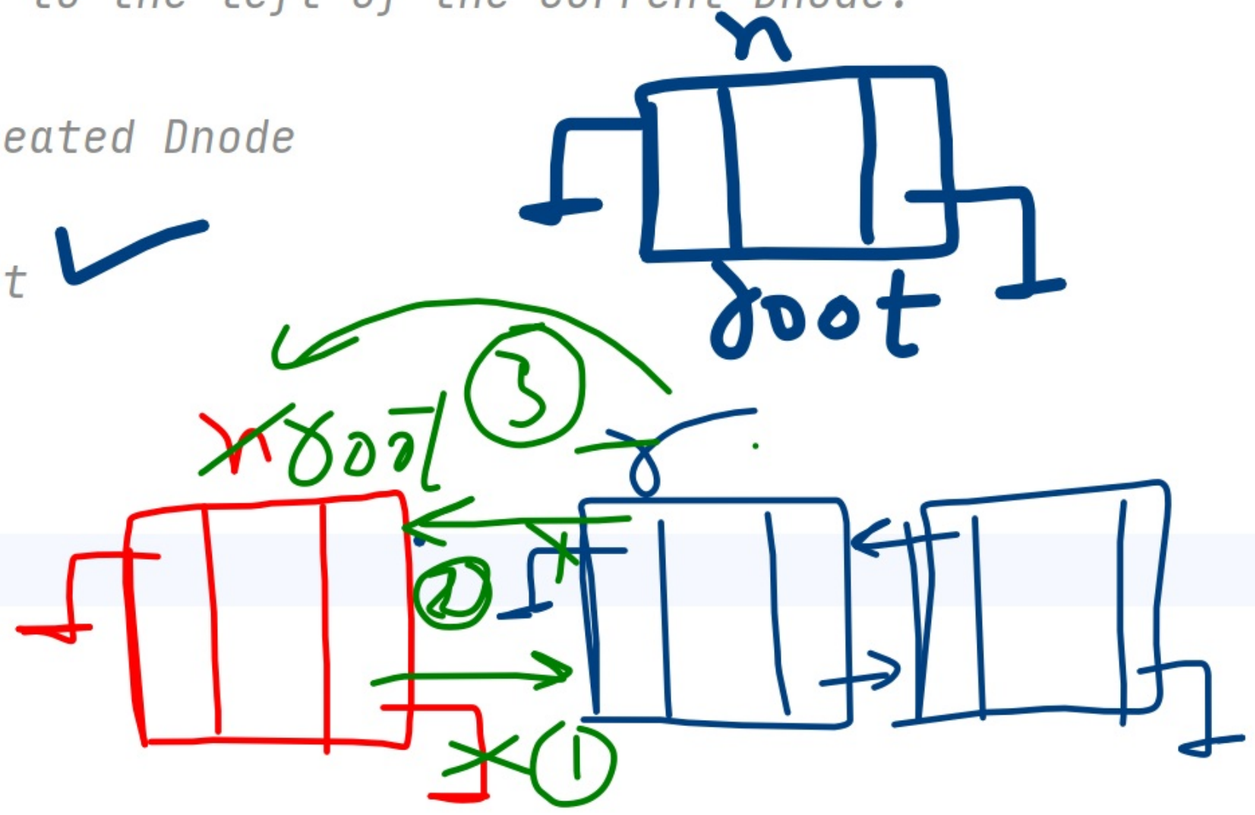
Linear_Linked_List.java | Doubly_Linked_List.java | Node.java | Dnode.java | Circular_Linked_List.java | Linked_List_Class.java | Employee_Manager

```java
package Linked_List_Examples;

public class Dnode
{
    int data;
    Dnode left,right;//self-ref pointer/reference
    Dnode(int data)
    {
        this.data=data;
        this.left=this.right=null;// not ref anyone
    }
}

```

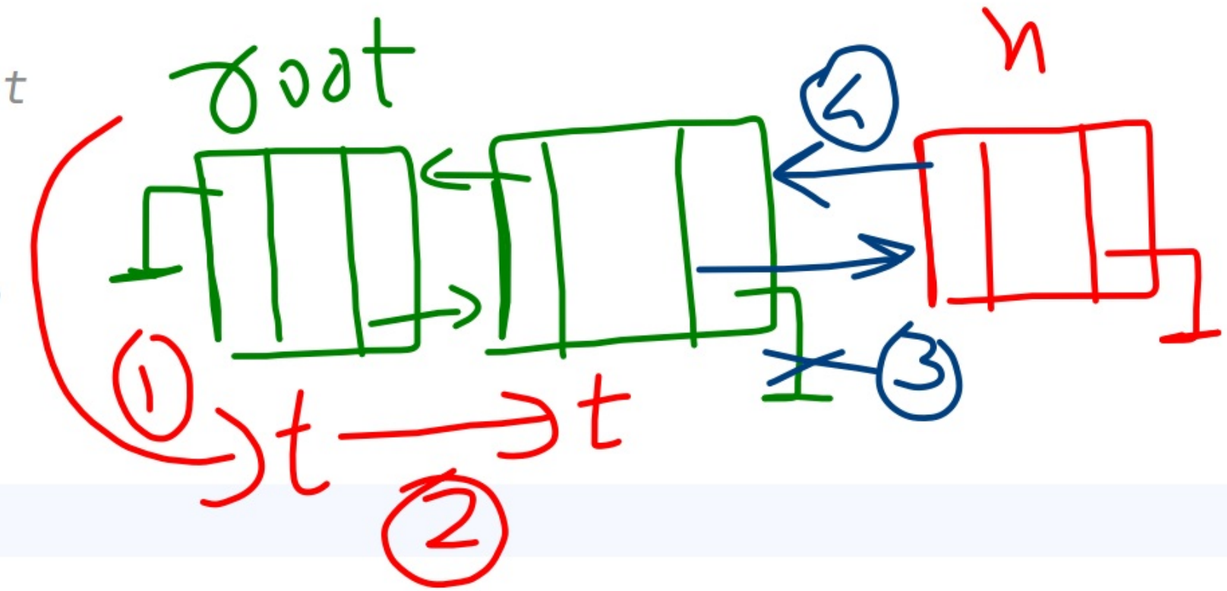


25°C Sunny Search ENG IN 09:37 14-11-2025

```
7       //insert_left:Research new element to the left of the current Dnode.
8       void insert_left(int data) {
9           Dnode n = new Dnode(data);//created Dnode
10          if (root == null)//no root
11              root = n;//1st becomes root
12          else {
13              n.right = root;//1
14              root.left=n;//2
15              root = n;//3
16          }
17      }
18
19      void insert_right(int data) {
20          Dnode n = new Dnode(data);//created Dnode
```



```
    void insert_right(int data) {
        Dnode n = new Dnode(data);//created Dnode
        if (root == null)//no root
            root = n;//1st becomes root
        else {
            Dnode t = root;//1
            while (t.right != null)//2
                t = t.right;
            t.right = n;//3 connected
            n.left=t;//4
        }
    }

    void delete_left() {
```



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help

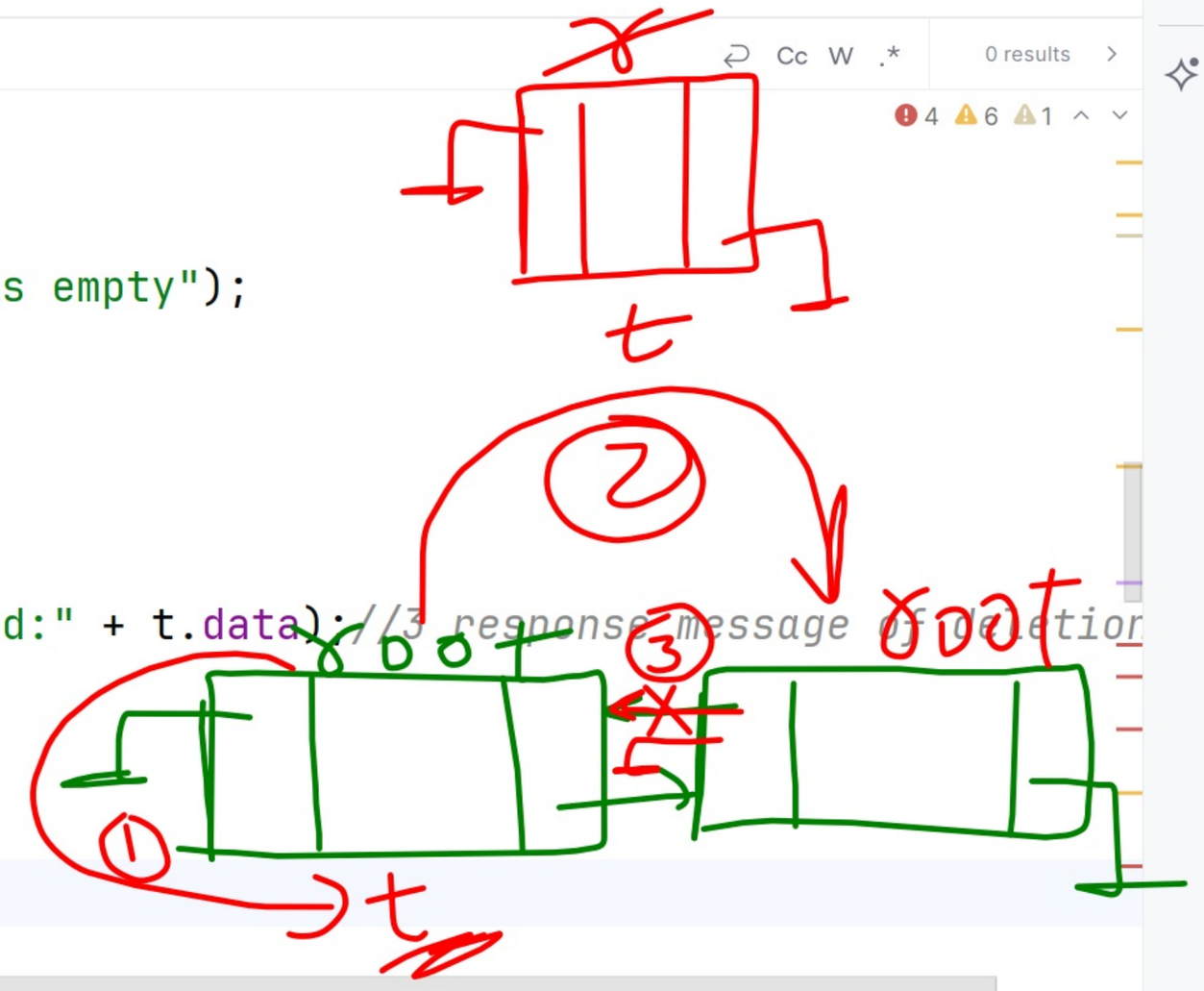
_Structures_VITA Version control Current File

Doubly_Linked_List.java Node.java Dnode.java Circular_Linked_List.java Linked_List_Class.java Employee_Management_Systen

Cc W .* 0 results

4 6 1

```
    void delete_left() {
        if (root == null)//no root
            System.out.println("List is empty");
        else {
            Dnode t = root;//1
            root = root.right;//2
            root.left=null;//3
            System.out.println("Deleted:" + t.data);//3 response message of deletion
        }
    }

    void delete_right() {
        if (root == null)//no root
```



Data_Structures_VITA > src > Linked_List_Examples > Doubly_Linked_List > delete_right 43:14 CRLF UTF-8 4 spaces

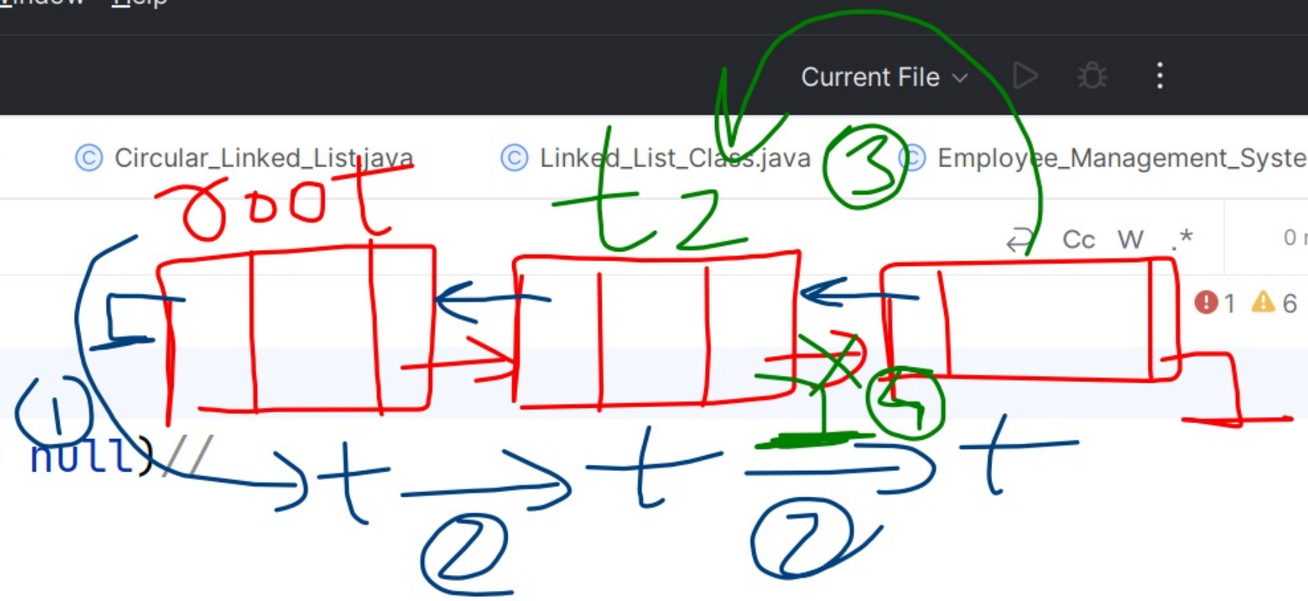
27°C Sunny Search ENG IN 09:52 14-11-2025

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help

_Structures_VITA Version control Current File

Doubly_Linked_List.java Node.java Dnode.java Circular_Linked_List.java Linked_List_Class.java Employee_Management_Syster

0 results

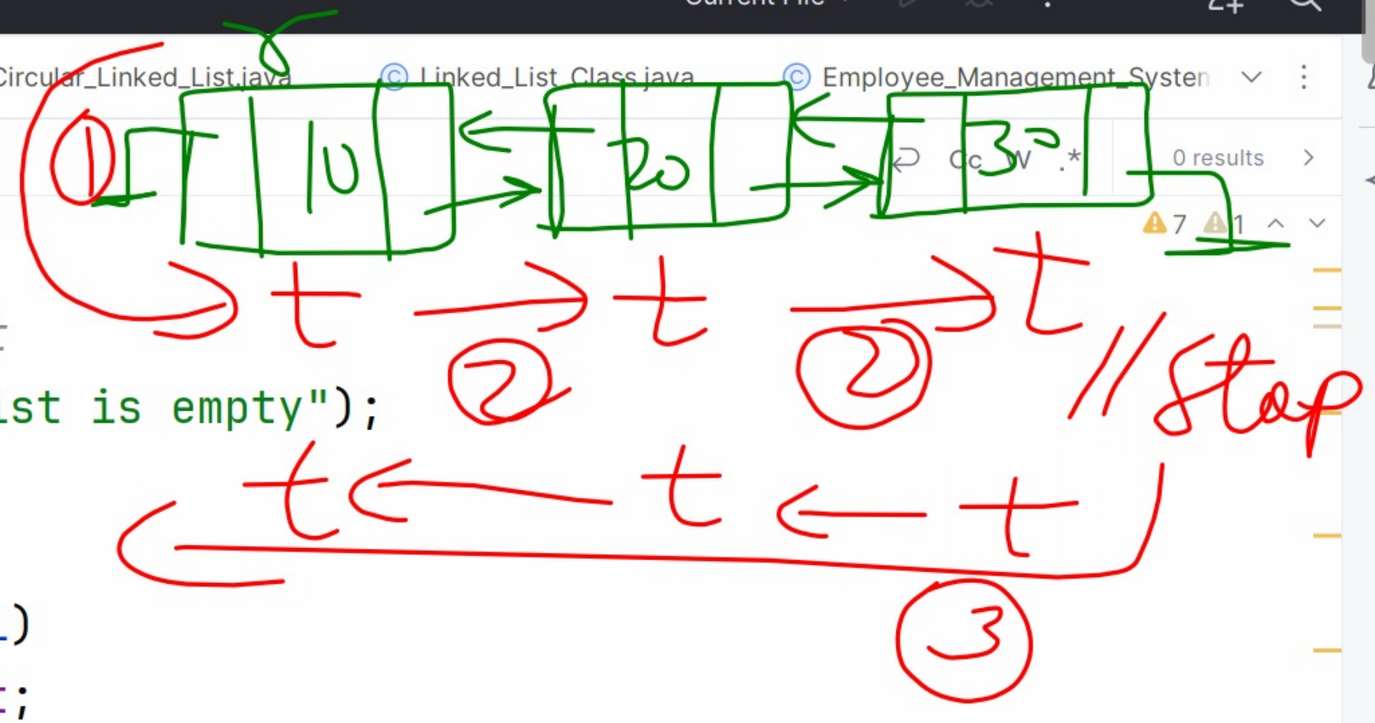



```
48          Dnode t, t2;
49          t = root;//1
50          while (t.right != null)//
51              t = t.right;
52
53          if (t == root)//single Dnode
54              root = null;//manual deletion
55          else
56          {
57              t2=t.left;//3 move back
58              t2.right=null;//4 break link
59          }
60          System.out.println("Deleted:" + t.data);//3 response message of deletior
```

1 6 1

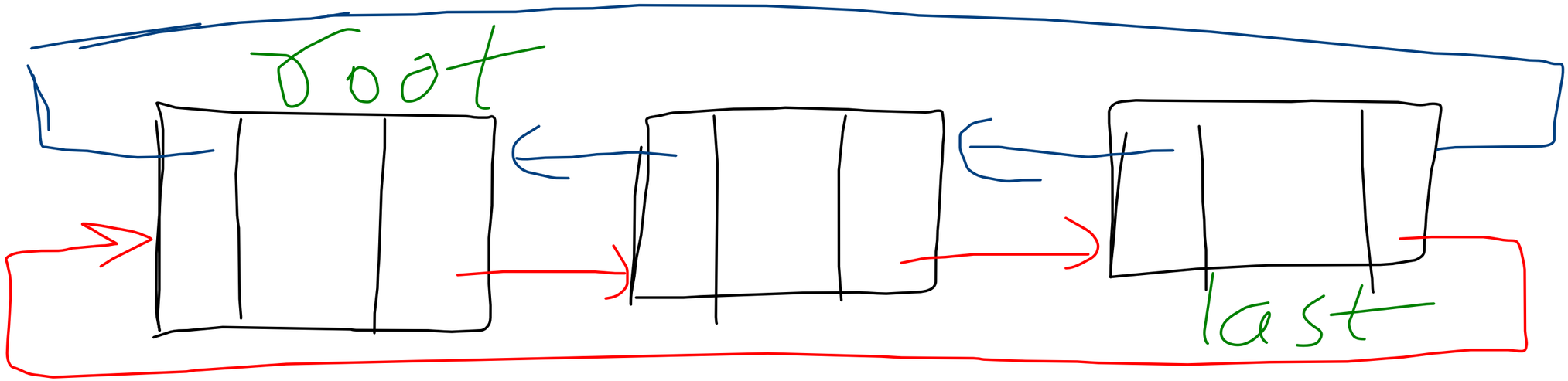
Data_Structures_VITA > src > Linked_List_Examples > Doubly_Linked_List > delete_right 49:25 CRLF UTF-8 4 spaces

27°C Sunny Search ENG IN 09:59 14-11-2025



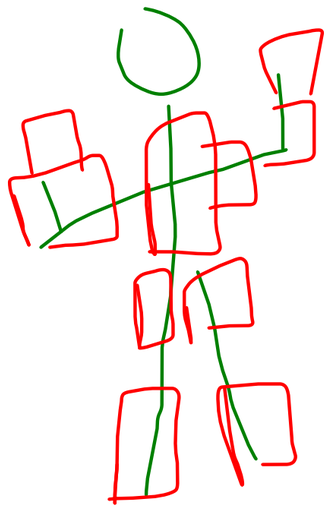

```
76      void print_list_rev() {
77          if (root == null)//no root
78              System.out.println("List is empty");
79          else {
80              Dnode t = root;//1
81              while (t.right != null)
82                      t = t.right;
83              while(t!=null)
84              {
85                  System.out.print("<-| " + t.data +" |->");
86                  t=t.left;
87              }
88          }
```

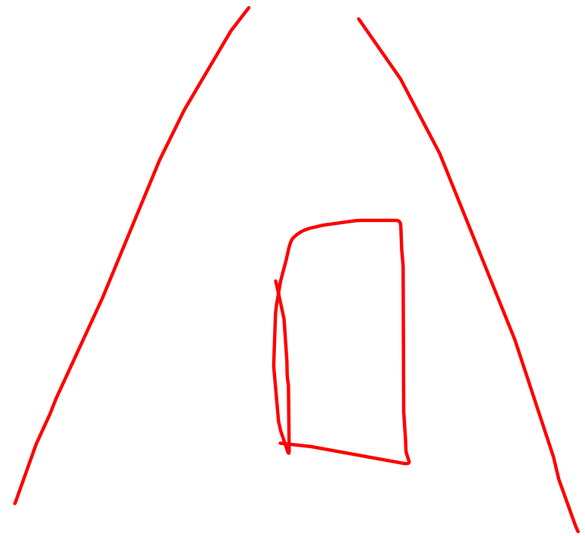

27°C Sunny Search ENG IN 10:09 14-11-2025



DCL

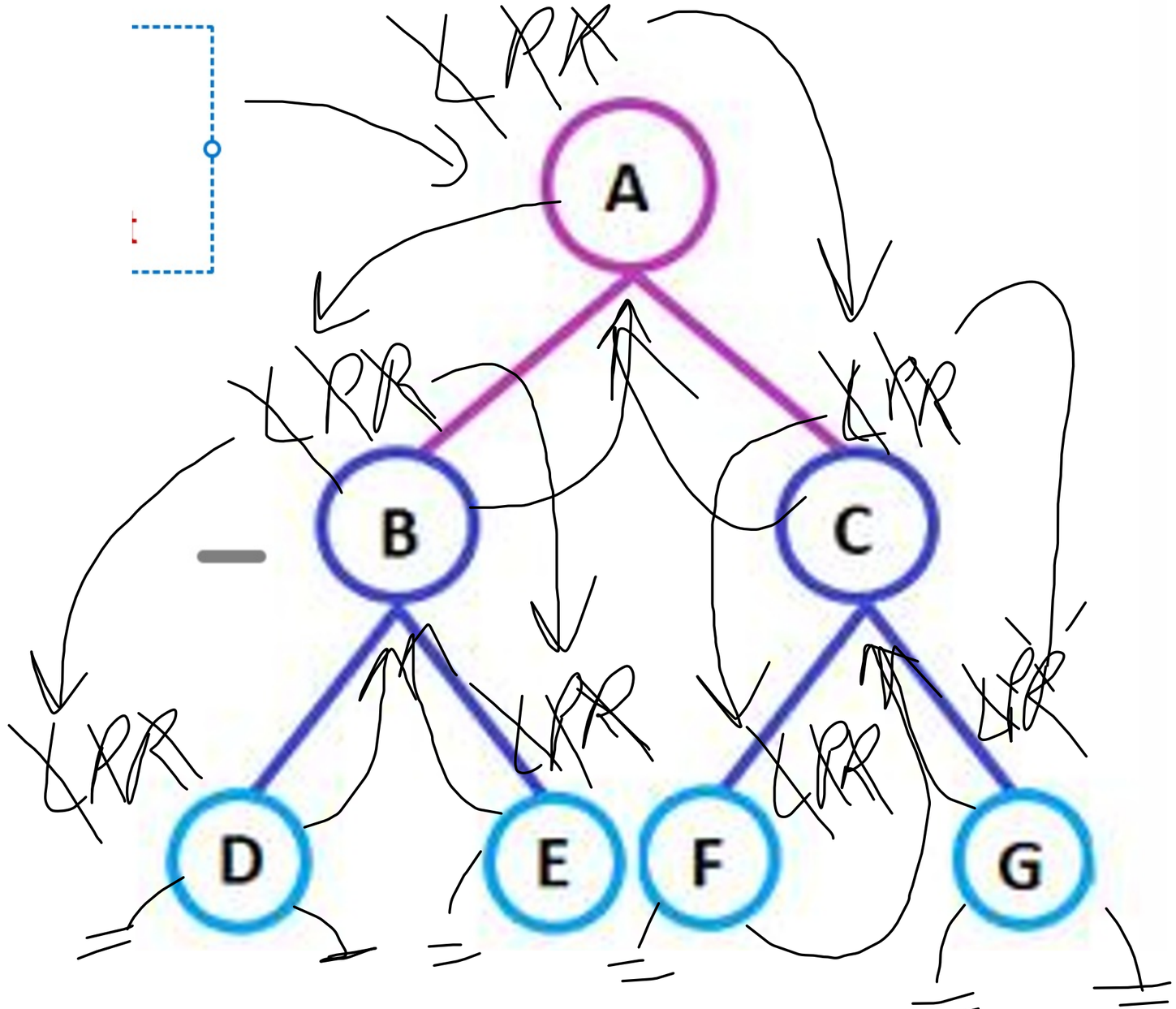
Application





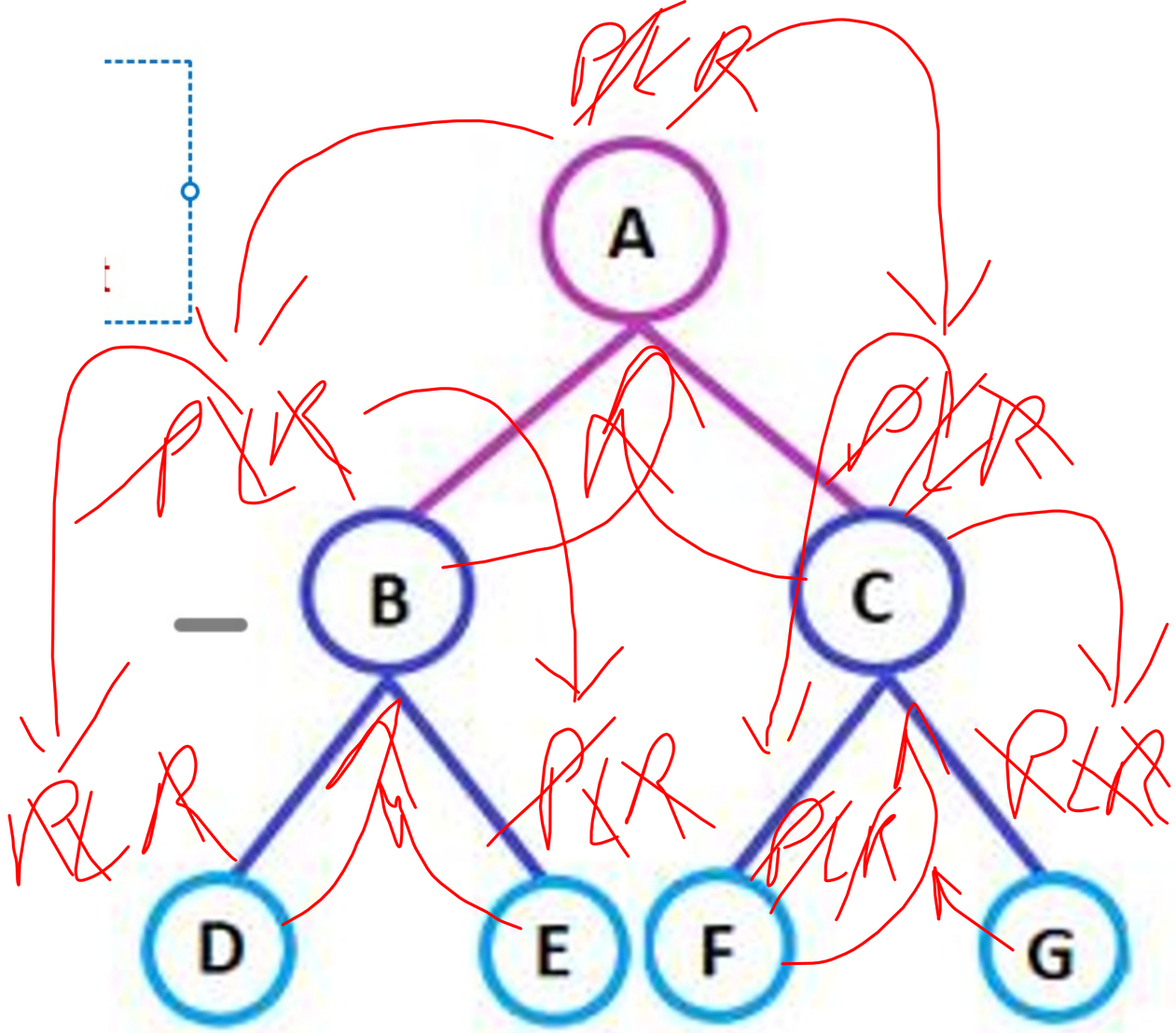
inorder:LPR
when u cut L:go to left
when u cut R:go to right
when u cut P:print
when all done go to parent

D, B, E, A, F, C, G,



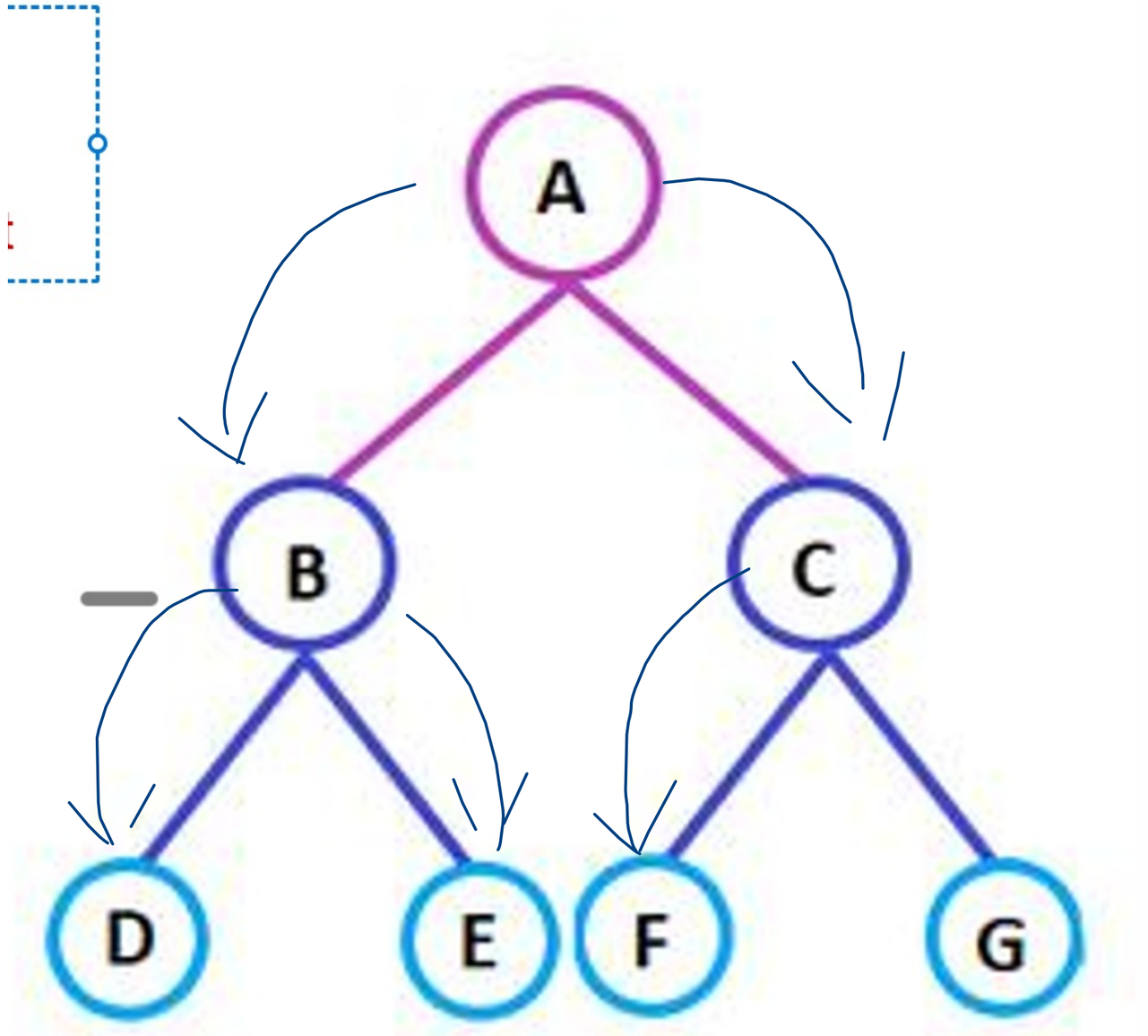
Pre-order:PLR
when u cut L:go to left
when u cut R:go to right
when u cut P:print
when all done go to parent

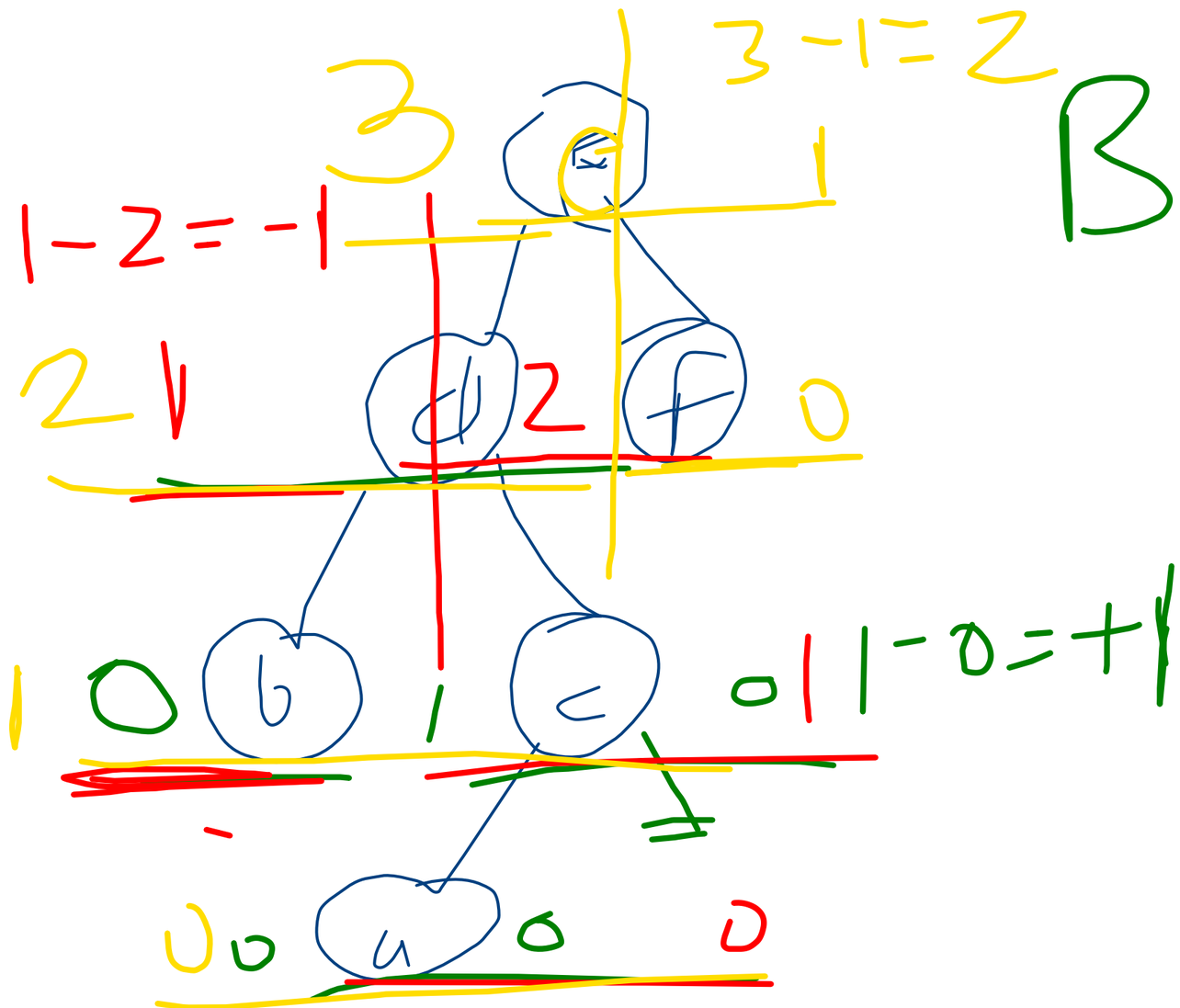
ABDECFG



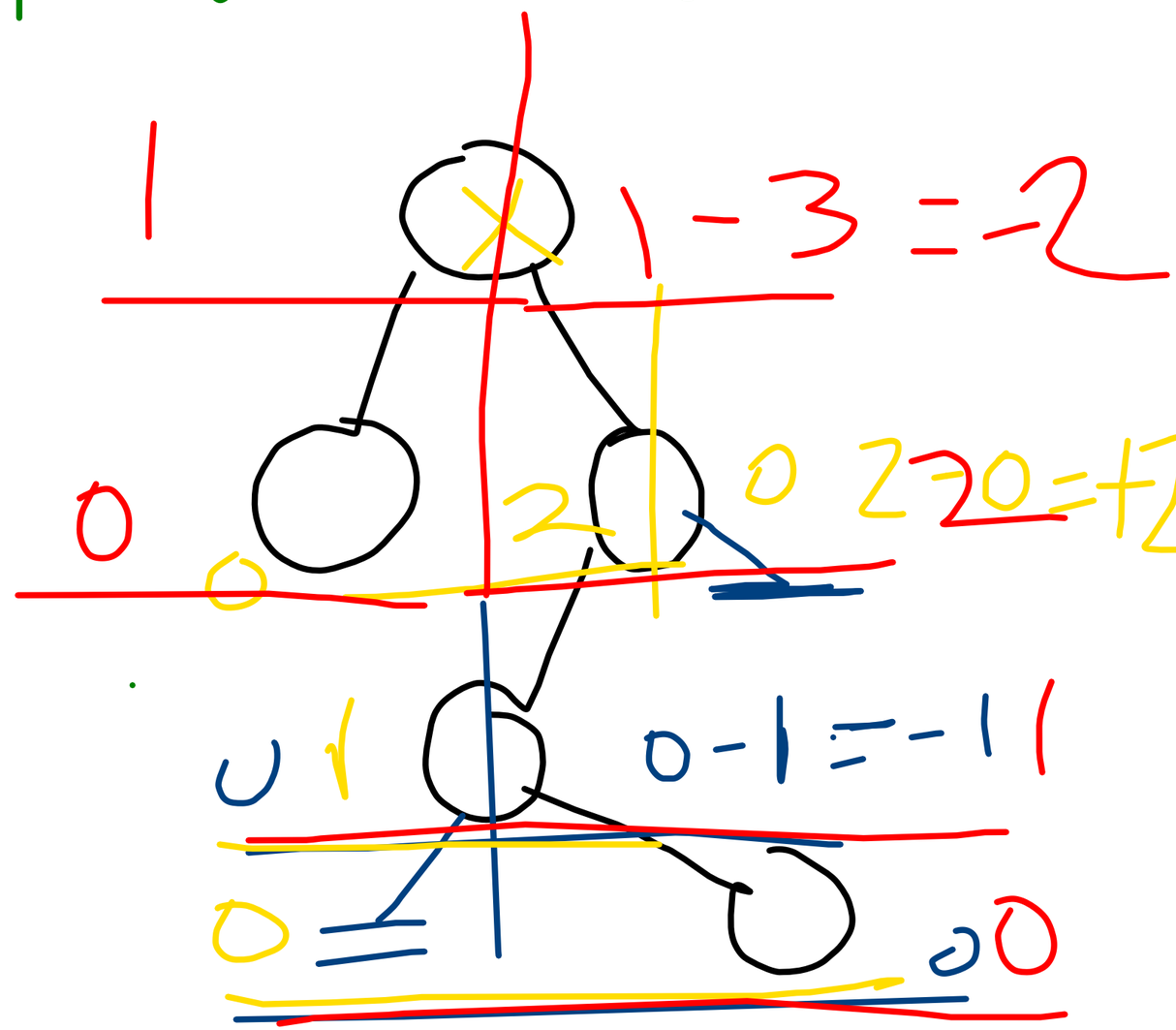
Post-order:LRP
when u cut L:go to left
when u cut R:go to right
when u cut P:print
when all done go to parent

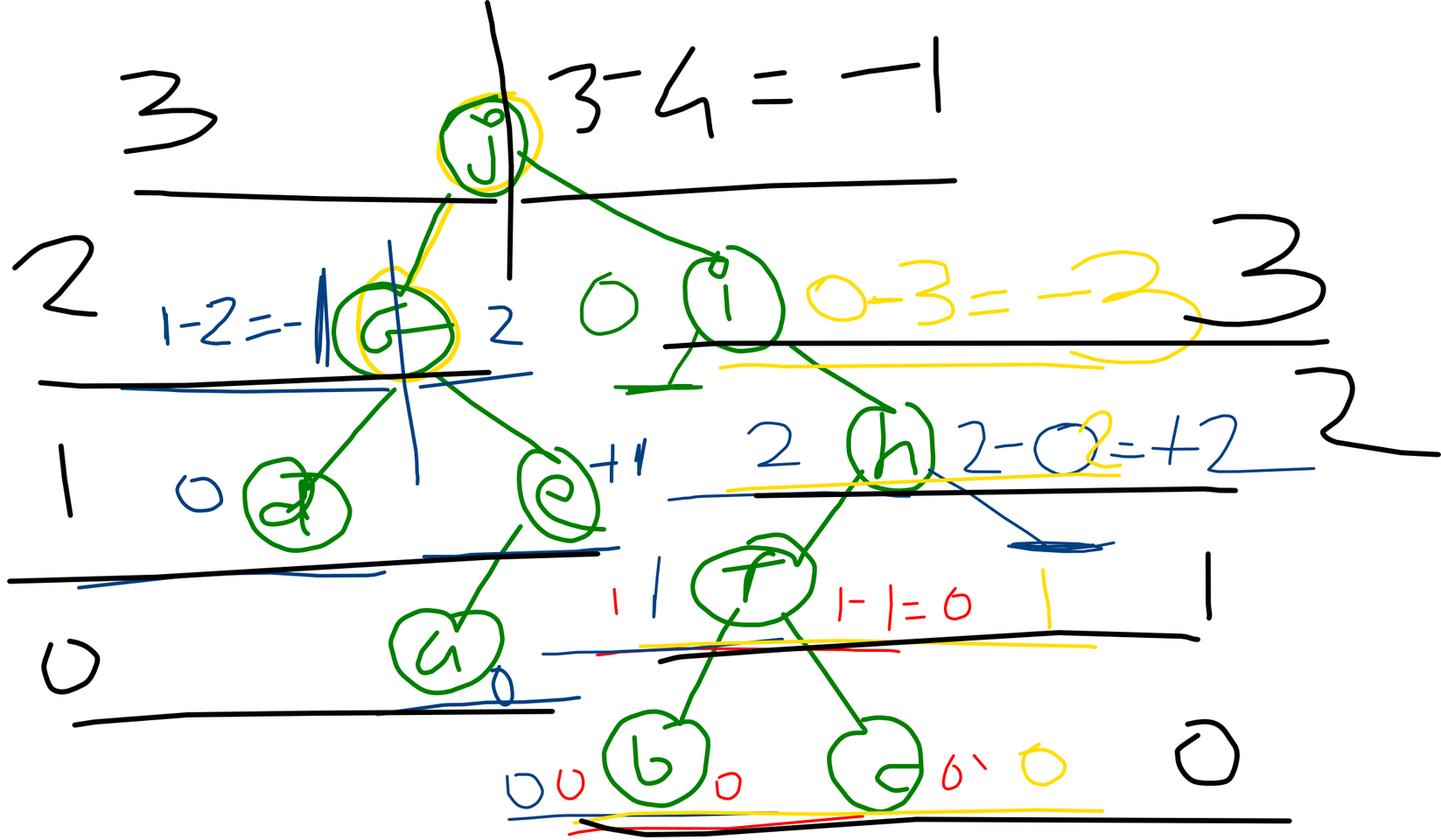
DEBFGCA





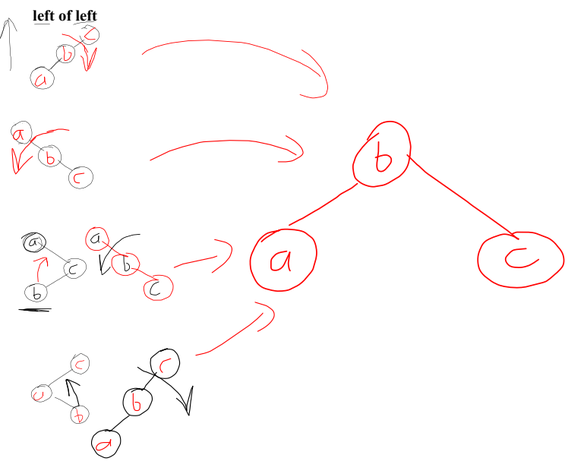
BF = #OLS - #ORS

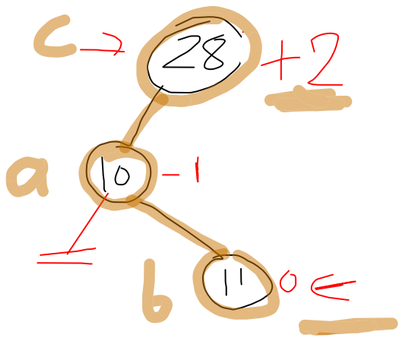


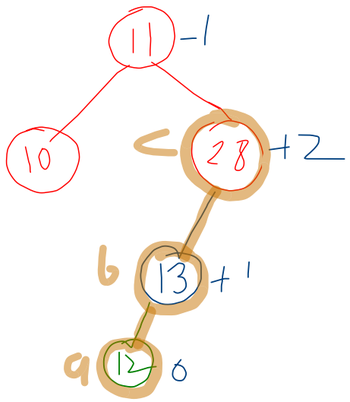

3
3-4=-1
2
1-2=-1
2
0
0-3=-3
3
1
0
-1
2
2-0=+2
2
1
1
1-1=0
1
0
0
0
0
0
0
0
0

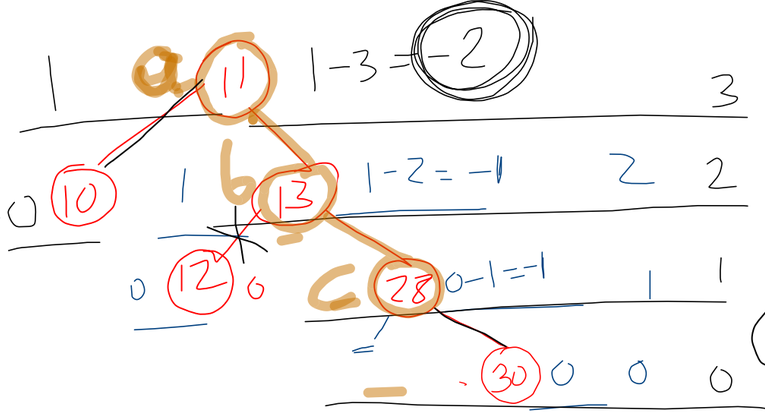
1,In AVL, insertion is done one node at a time in a BST fashion.
2.After every insertion, we calculate balance factor from bottom to top.
3.The first point where the balance factor is out of the ±1 range(stop), see case and follow either of four types of rotation.
(look where insertion is done and where unblanced is)

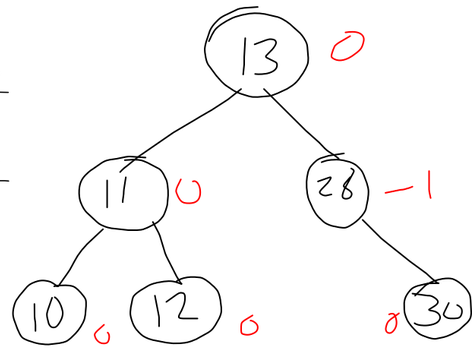
left of left





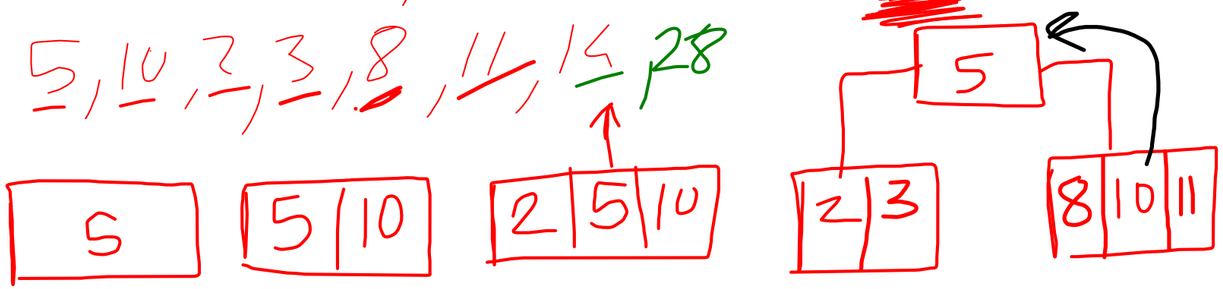


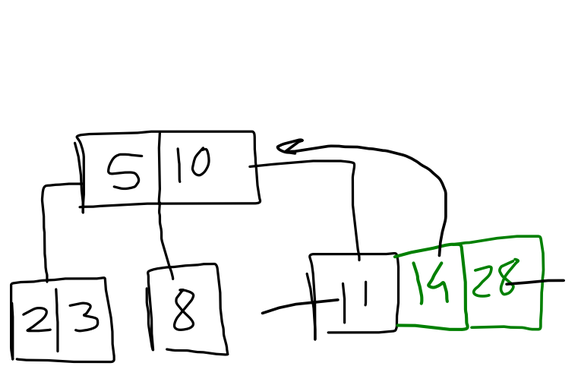


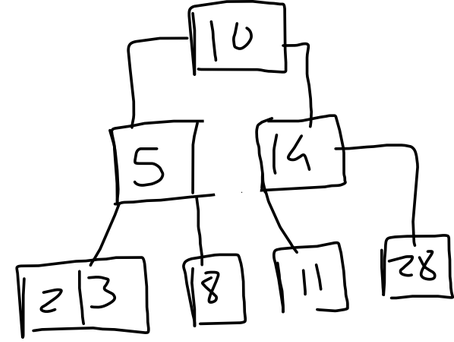


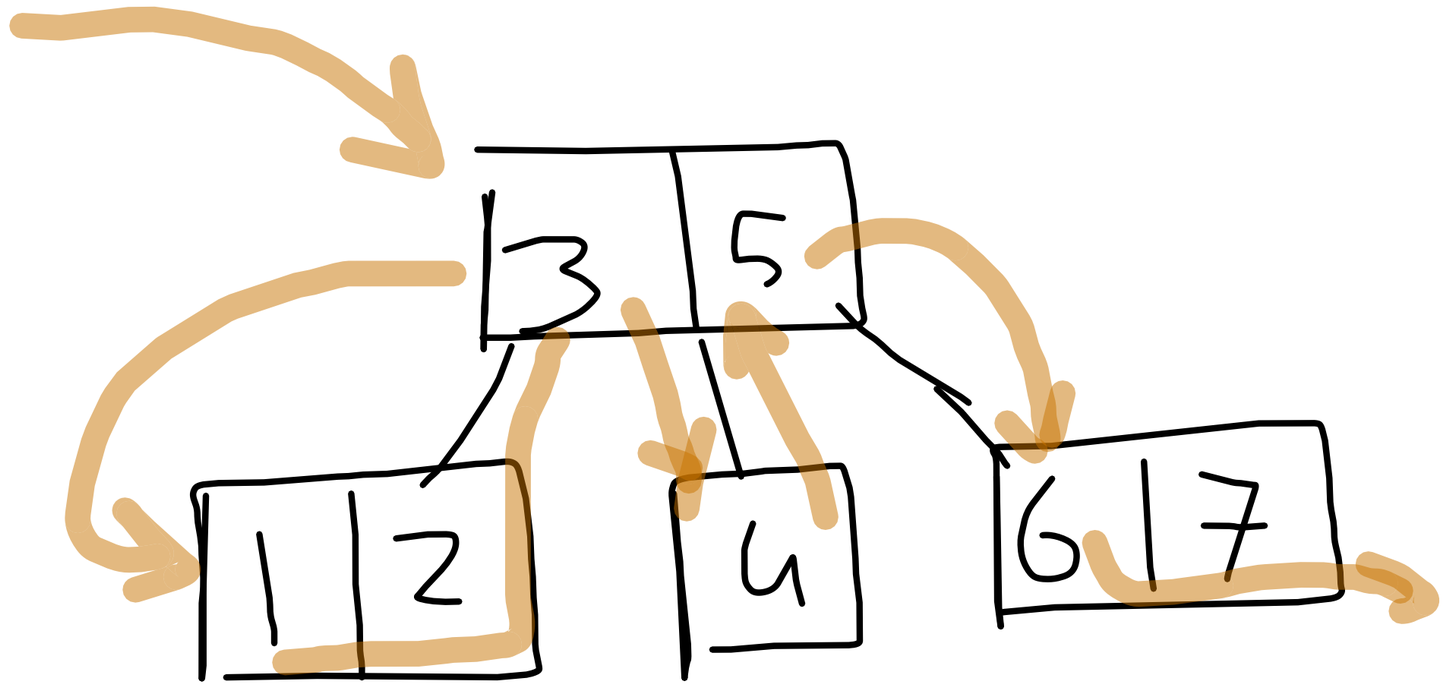
M=3 (min 3/2=1, max 3-1=2, split=3)

5, 10, 2, 3, 8, 11, 14, 28





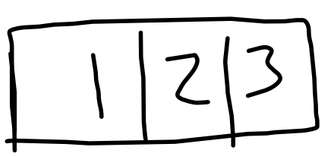


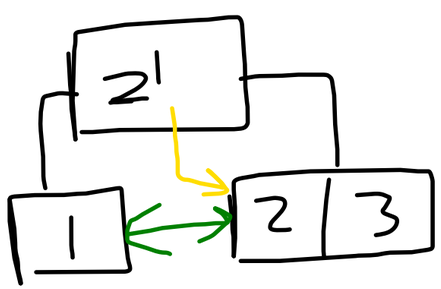


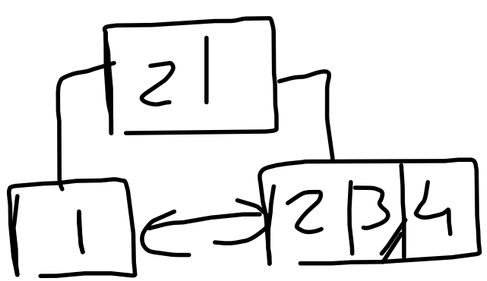
B-tree is excellent for storage but behaves in a very slow manner for sequential access of data because data is scattered among N nodes.

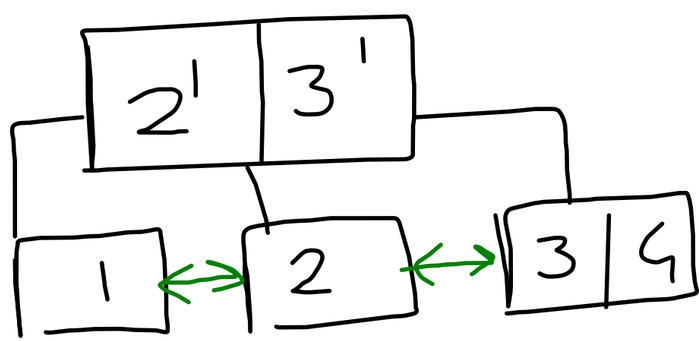
Sol:B+

m=3









1347%3000

1347