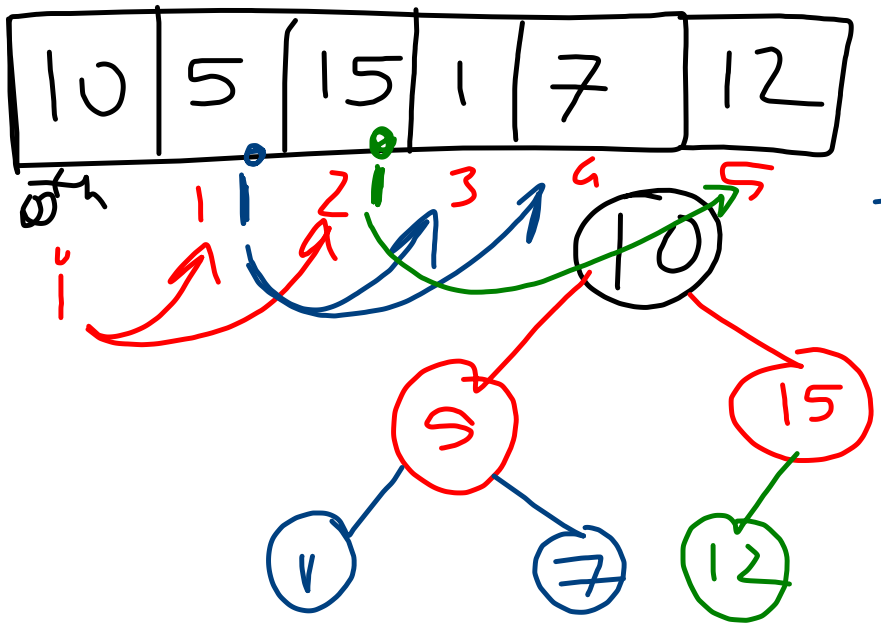


+ve

static

simple

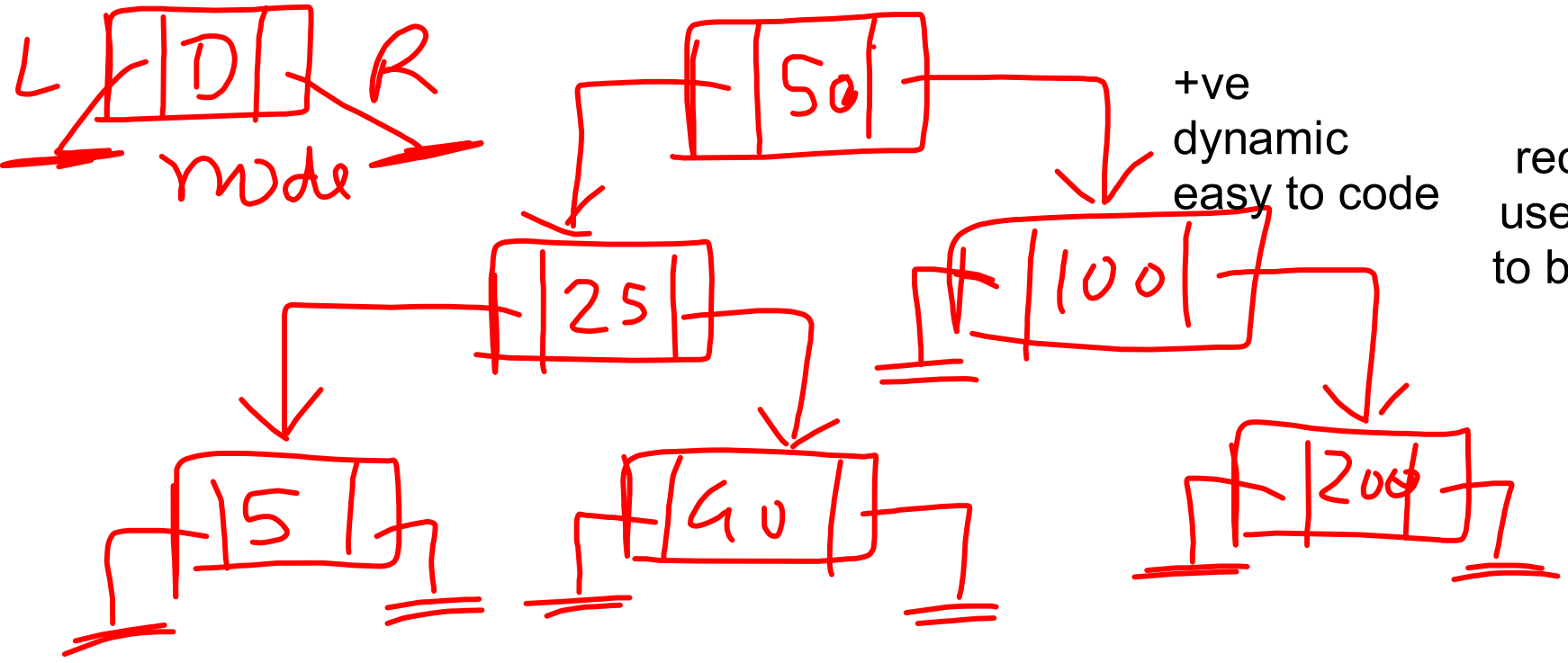
easy to code

-ve

-Static → fixed size

-Flat structure to access

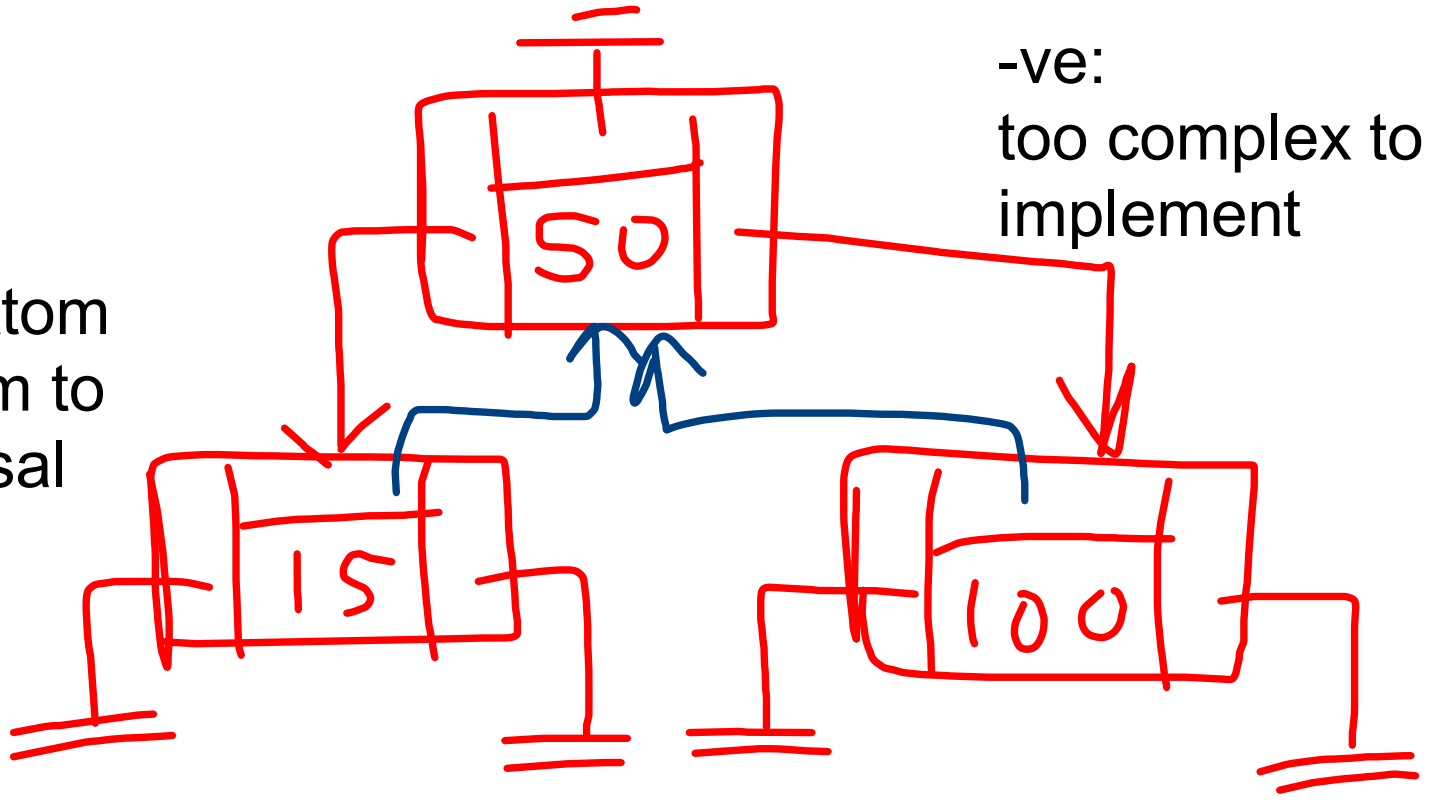
Tree (slow)



+ve
dynamic
easy to code

-ve
recursion to be used as only top to bottom access possible

+ve
-dynamic
-top to bottom and bottom to top traversal possible



-ve:
too complex to implement

DataStructures - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

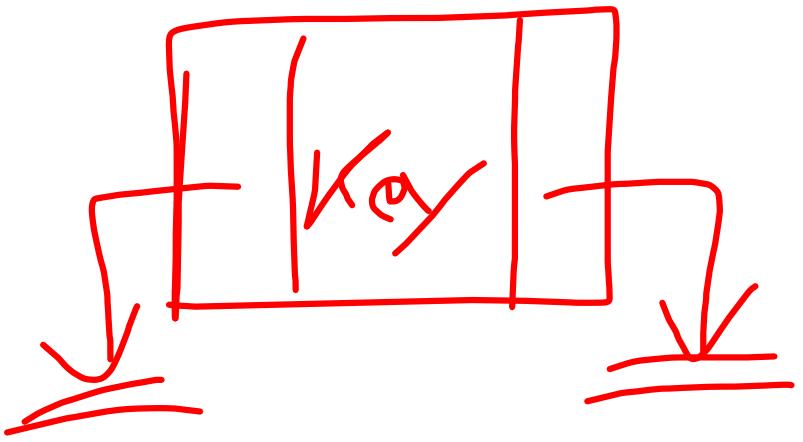
Search (Ctrl+I)

<default confi...

...va | Treemain.java | GraphDemo.java | StackDemo.java | QueueLinearDemo.java | CircularQueueDemo.java | PriorityQueueDemo.java | LinkedListLinear.java | DynamicStackDemo.java | DynamicQueueDemo.ja..

Source | History

```
10      */
11      class Node
12      {
13          int key;
14          Node left, right;
15
16          public Node(int item)
17          {
18              key = item;
19              left = right = null;
20          }
21      }
22      public class Treemain {
23          private Node root;
```



Output

23:4 INS

Search

ENG IN

09:34 29-11-2022

DataStructures - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

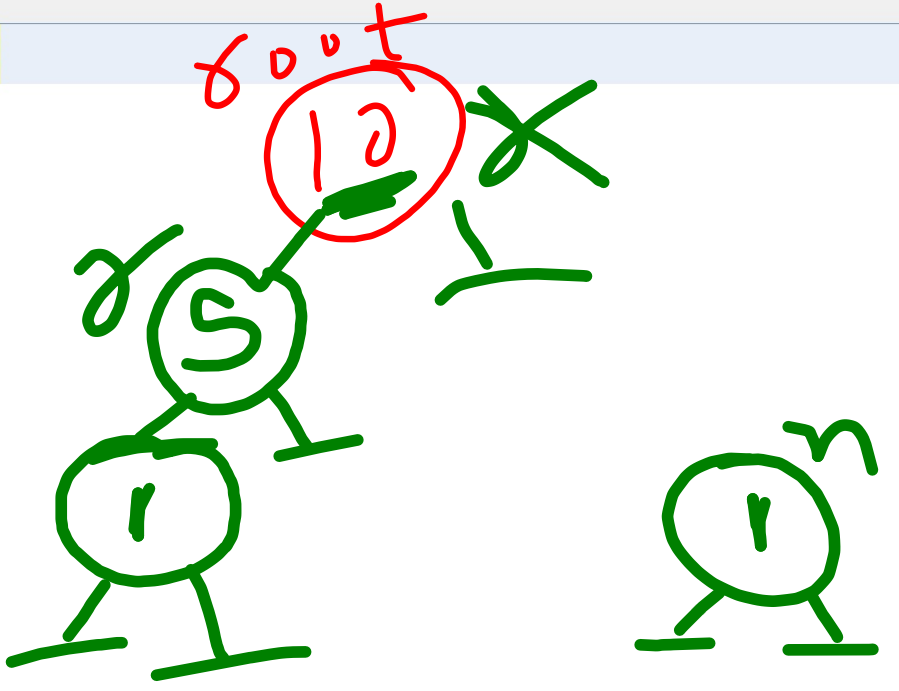
<default confi...

...va Treemain.java | GraphDemo.java | StackDemo.java | QueueLinearDemo.java | CircularQueueDemo.java | PriorityQueueDemo.java | LinkedListLinear.java | DynamicStackDemo.java | DynamicQueueDemo.ja..

Source History

```
29  public void insert(Node r,Node n)
30  {
31     if (root==null)
32     { root=n; }
33     else
34     {
35        if(n.key<r.key)
36        {
37           if(r.left==null)
38              r.left=n;
39           else
40              insert(r.left,n);
41        }
          else
```

insert(10,1);
insert(5,1);



Output

28:38 INS

Search

ENG IN 09:37 29-11-2022

DataStructures - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

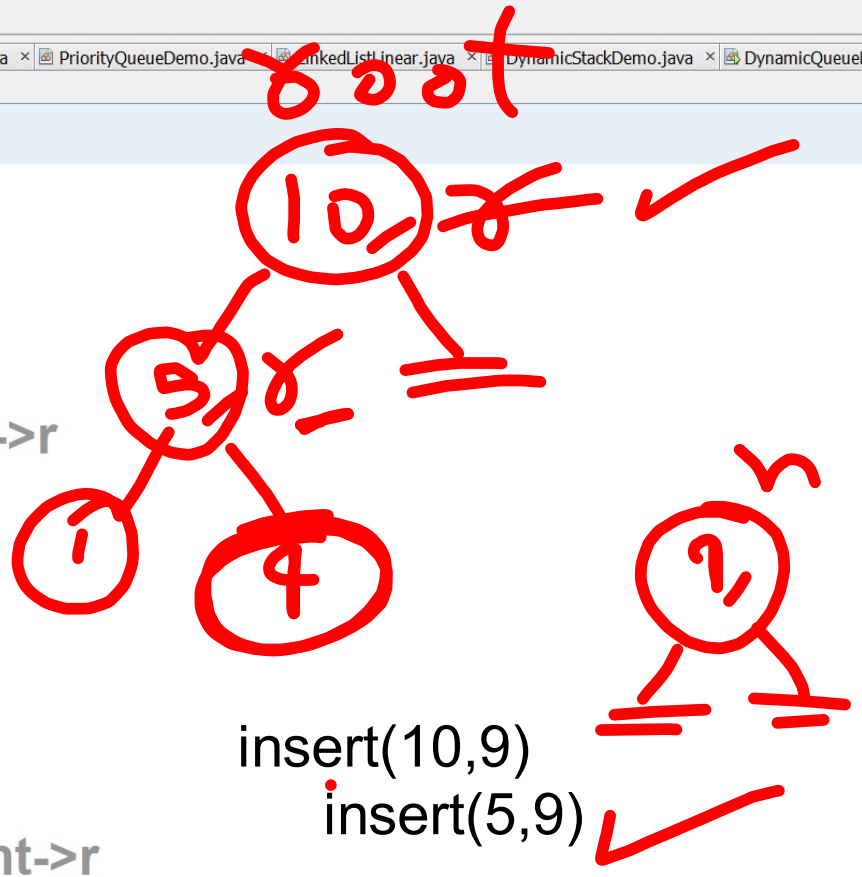
<default confi...

...va | Treemain.java | GraphDemo.java | StackDemo.java | QueueLinearDemo.java | CircularQueueDemo.java | PriorityQueueDemo.java | LinkedListLinear.java | DynamicStackDemo.java | DynamicQueueDemo..

Source | History

Navigator | Projects | Files | Services

```
35          if(n.key<r.key)
36          {
37            if(r.left==null)
38                r.left=n;
39            else
40                insert(r.left,n);//-->r.left->r
41          }
42          else
43          {
44            if(r.right==null)
45                r.right=n;
46            else
47                insert(r.right,n);//r->right->r
48          }
```



Output

34:28 INS

Search

ENG IN 09:45 29-11-2022

DataStructures - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

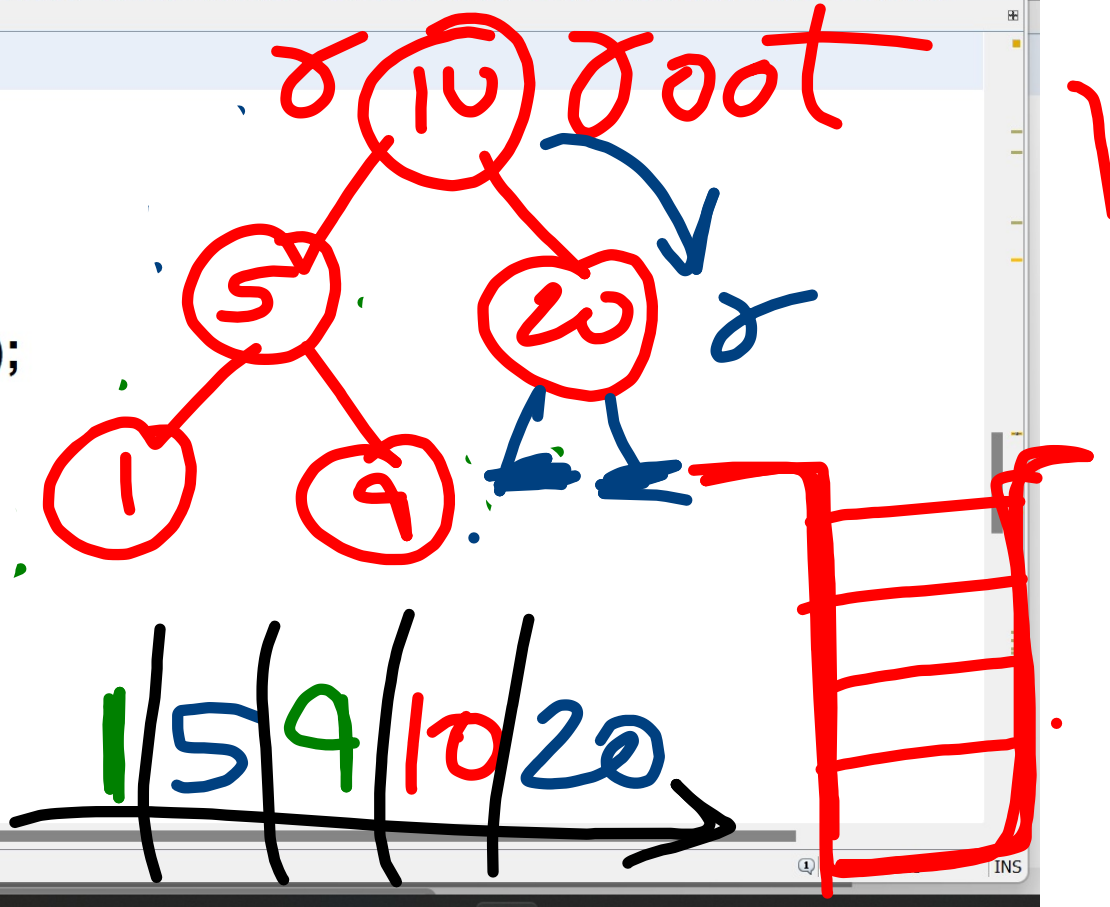
Search (Ctrl+I)

<default confi...

...va Treemain.java | GraphDemo.java | StackDemo.java | QueueLinearDemo.java | CircularQueueDemo.java | PriorityQueueDemo.java | LinkedListLinear.java | DynamicStackDemo.java | DynamicQueueDemo..

Source History

```
52  public void inorder(Node r)
53  {
54      if(r!=null)
55      {
56          inorder(r.left);
57          System.out.println(r.key);
58          inorder(r.right);
59      }
60  }
61
62  public void  counter(Node r)
63  {
64      if(r!=null)
```



Output

INS

Search

ENG IN

09:53 29-11-2022

DataStructures - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

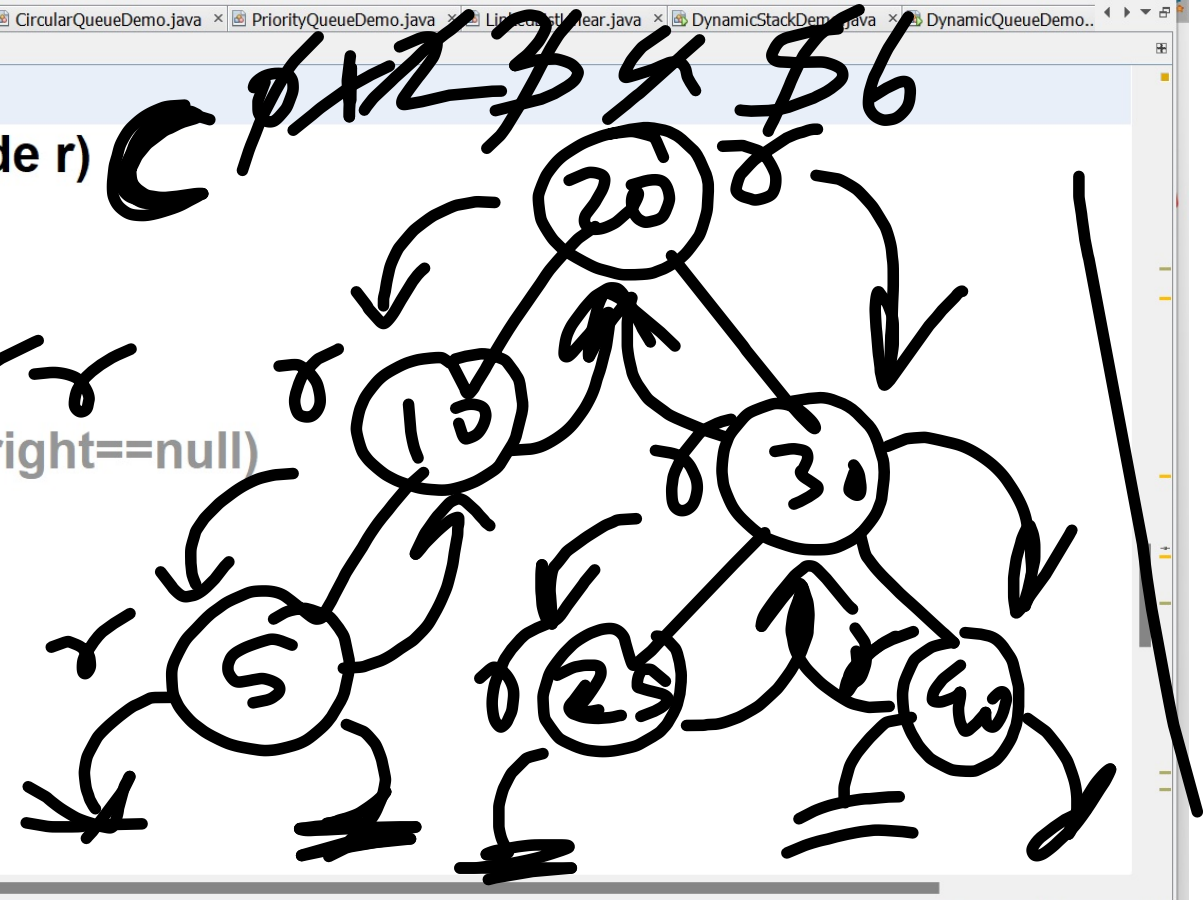
Search (Ctrl+I)

<default confi...

...va | Treemain.java | GraphDemo.java | StackDemo.java | QueueLinearDemo.java | CircularQueueDemo.java | PriorityQueueDemo.java | LinkedListLinear.java | DynamicStackDemo.java | DynamicQueueDemo..

Source History

```
60
62  public void  counter(Node r)
63  {
64     if(r!=null)
65     {
66        counter(r.left);
67      //  if(r.left==null && r.right==null)
68        Treemain.c++;
69        counter(r.right);.
70
71     }
72
73  }
```



Output

60:5 INS

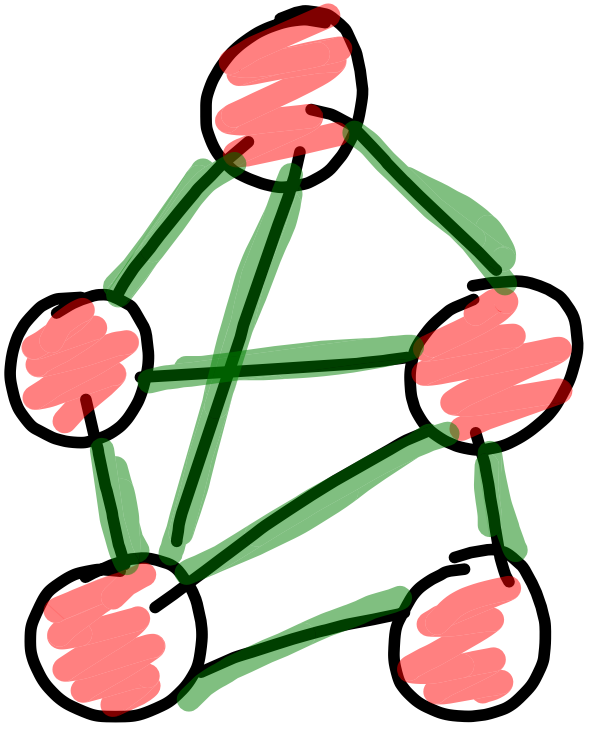
Search

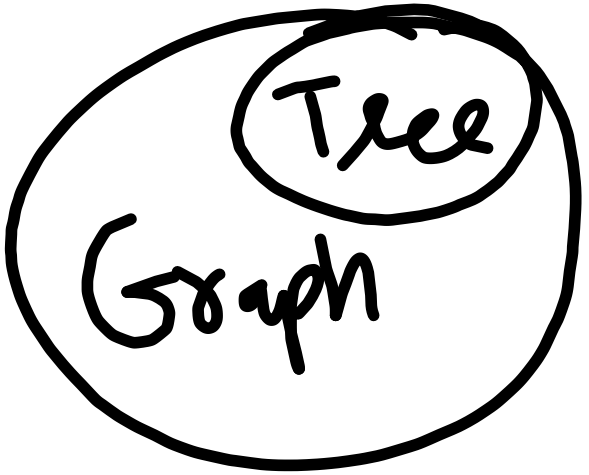
ENG IN

10:13 29-11-2022

Graph is

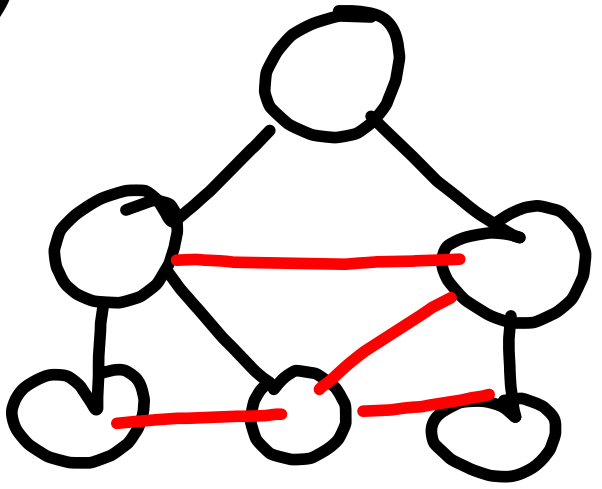
- A data structure that consists of a set of nodes (vertices) and a set of edges that relate the nodes to each other.
- The set of edges describes relationships among the vertices.
- A graph G is defined as follows:
- G=(V,E)
- V(G): a finite, nonempty set of vertices
- E(G): a set of edges (pairs of vertices)

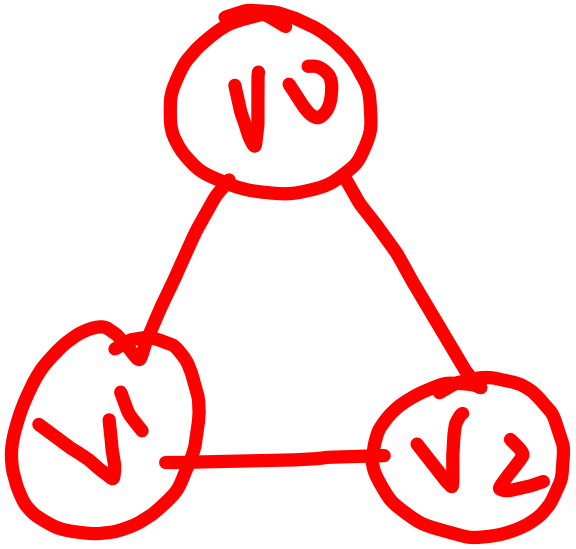




every tree is a graph
but every graph is not tree.

a graph without cycle can be called tree



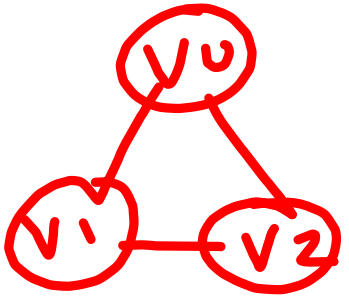
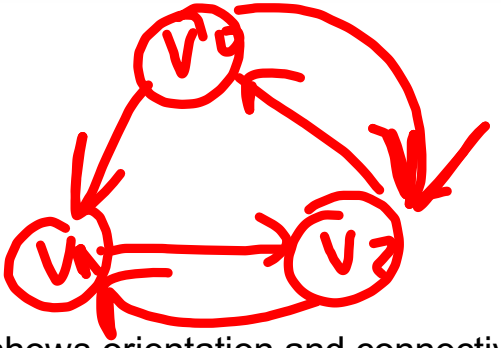
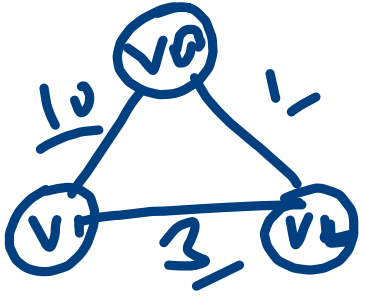
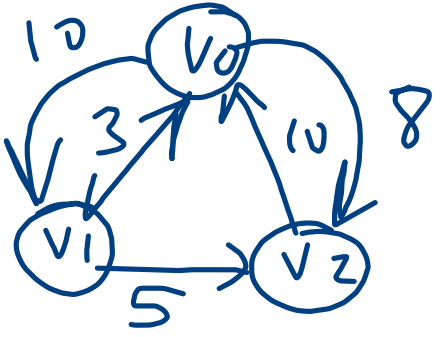


$G(V, E)$

$V = \{v_0, v_1, v_2\}$

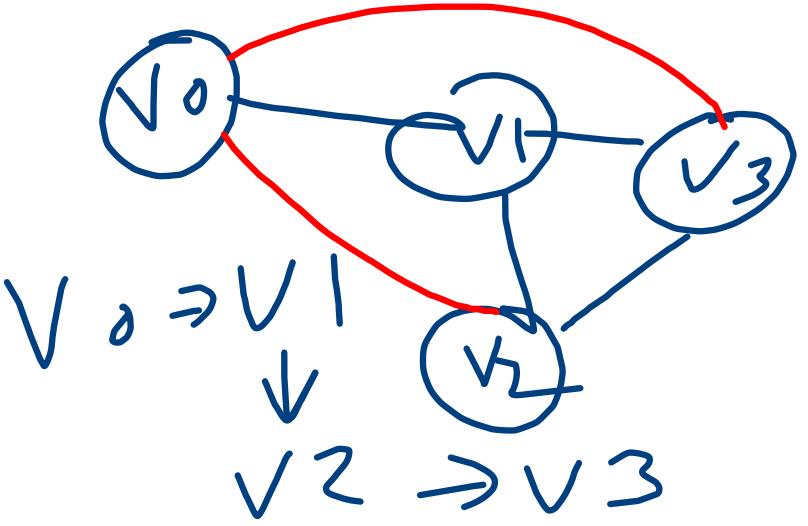
$E = \{(v_0, v_1), (v_1, v_2), (v_2, v_0)\}$

Types

	UNDIRECTED	DIRECTED
UNWEIGHTED	 basic connectivity	 shows orientation and connectivity
WEIGHTED any unit that can be used to take decisions		

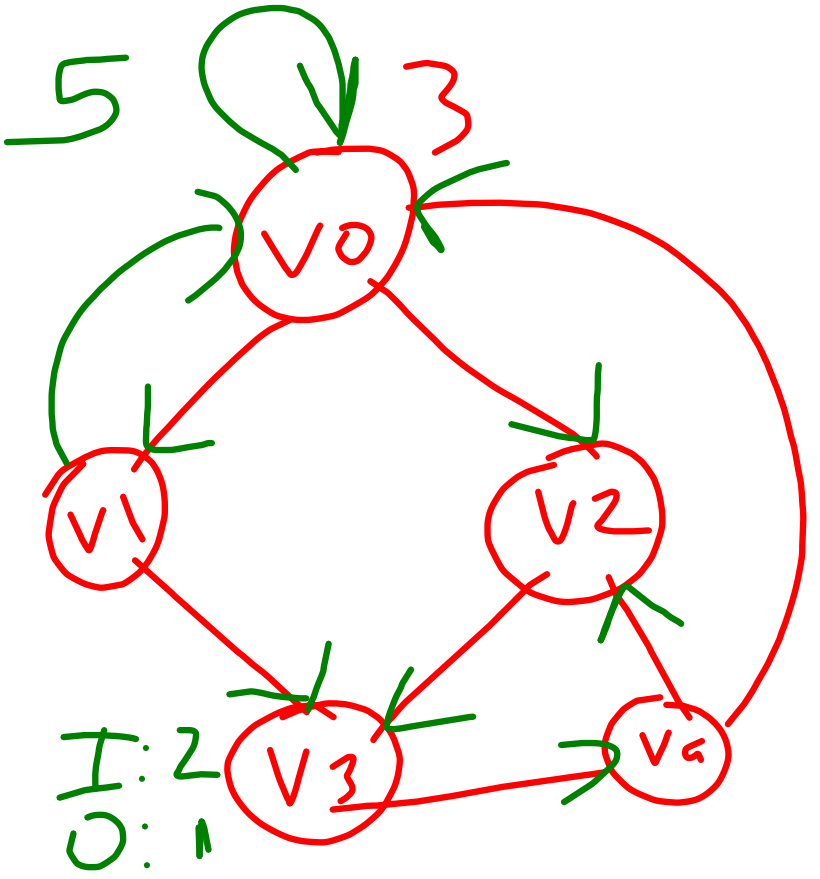
Terminologies

- Adjacent nodes: two nodes are adjacent if they are connected by an edge
- Path: a sequence of vertices that connect two nodes in a graph
- A simple path is a path in which all vertices, except possibly in the first and last, are different.
- Complete graph: a graph in which every vertex is directly connected to every other vertex.
- A cycle is a simple path with the same start and end vertex.

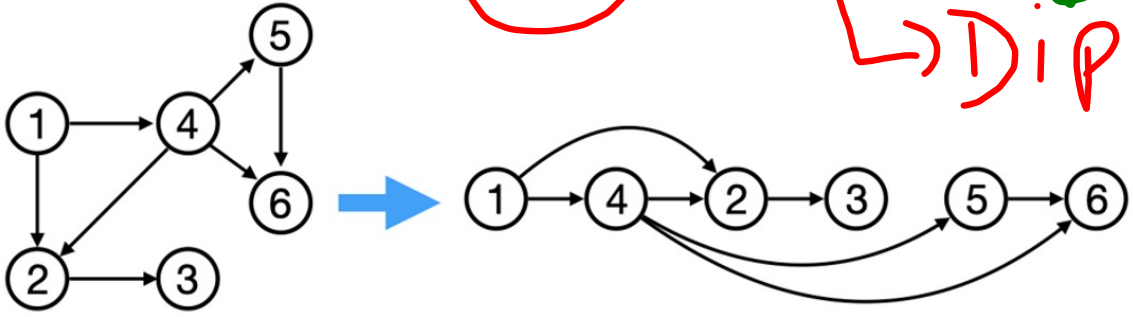


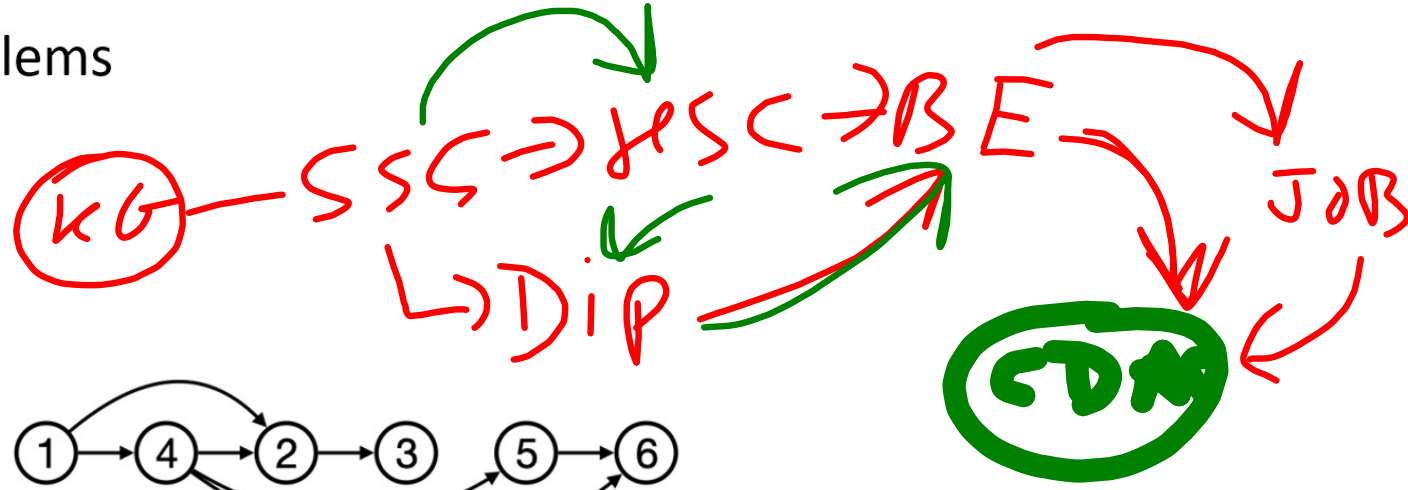
Terminologies

- A cycle is a simple path with the same start and end vertex.
- The degree of vertex / is the no. of edges incident on vertex.
- Directed graphs are strongly connected if there is a path from any one vertex to any other.
- Directed graphs are weakly connected if there is a path between any two vertices, ignoring direction



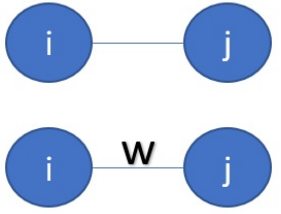
- Handling flow problems
- Topological sort

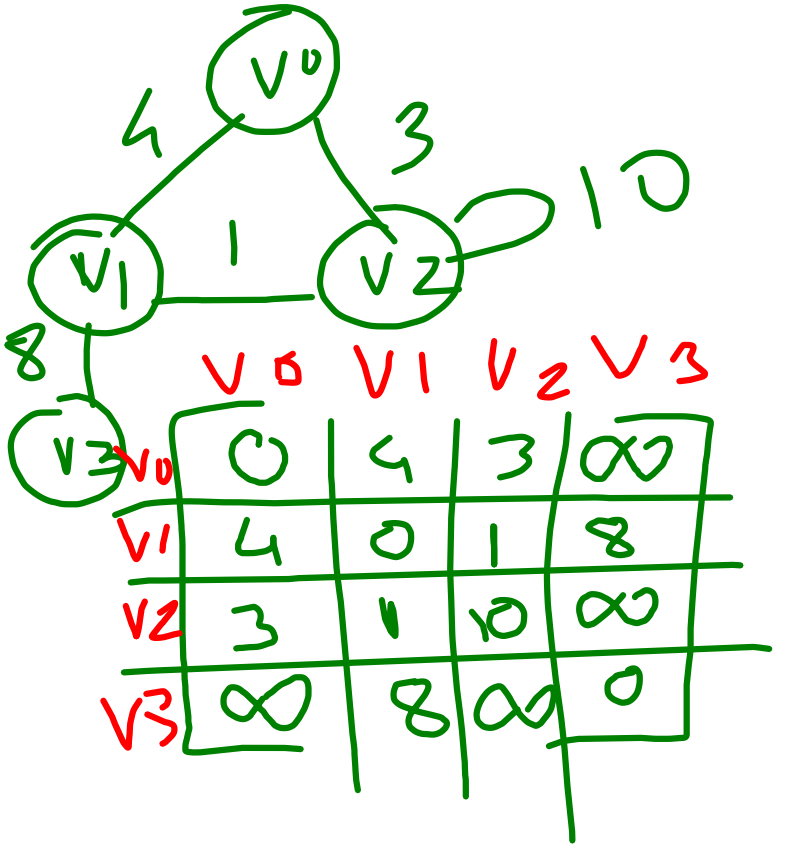




Graph representation

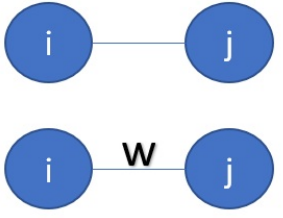
- Adjacency matrix
  - V x V matrix(storage of v x v)
  - Static
  - Easy
  - To fill if unweighted
  - AdjMat[i][j]=1/0
  - To fill if weighted
  - AdjMat[i][j]=w/∞

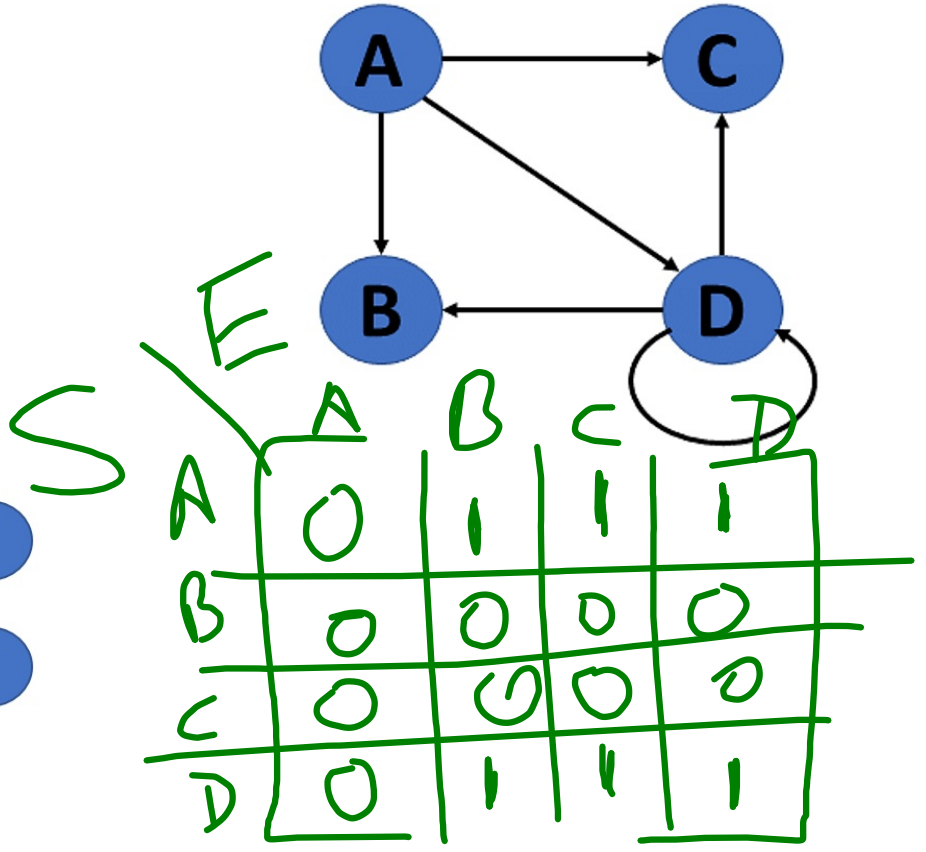




Graph representation

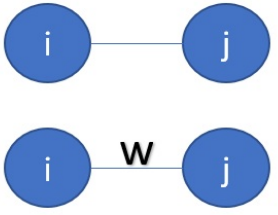
- Adjacency matrix
  - V x V matrix
  - Static
  - Easy
  - To fill if unweighted
  - AdjMat[i][j]=1/0
  - To fill if unweighted
  - AdjMat[i][j]=w/∞

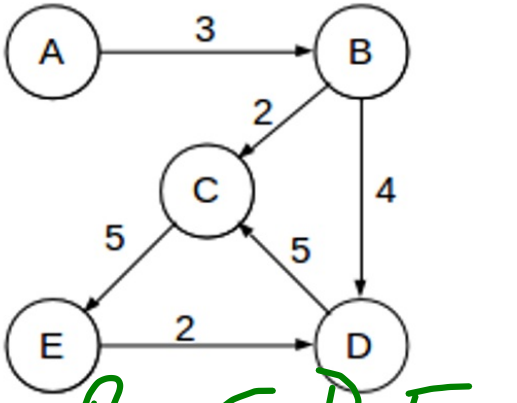


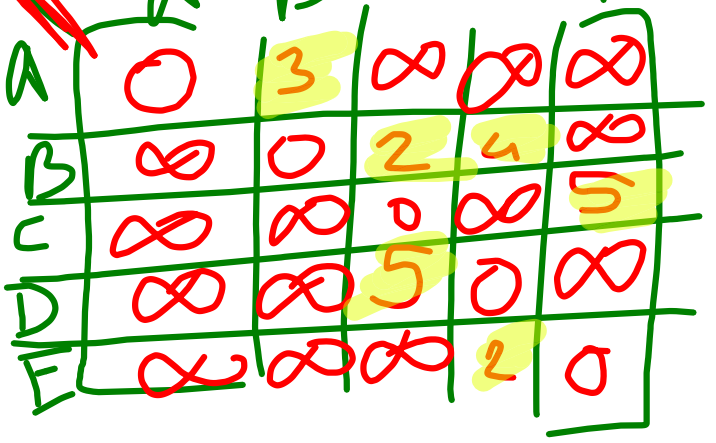


Graph representation

- Adjacency matrix
  - V x V matrix
  - Static
  - Easy
  - To fill if unweighted
  - AdjMat[i][j]=1/0
  - To fill if unweighted
  - AdjMat[i][j]=w/∞



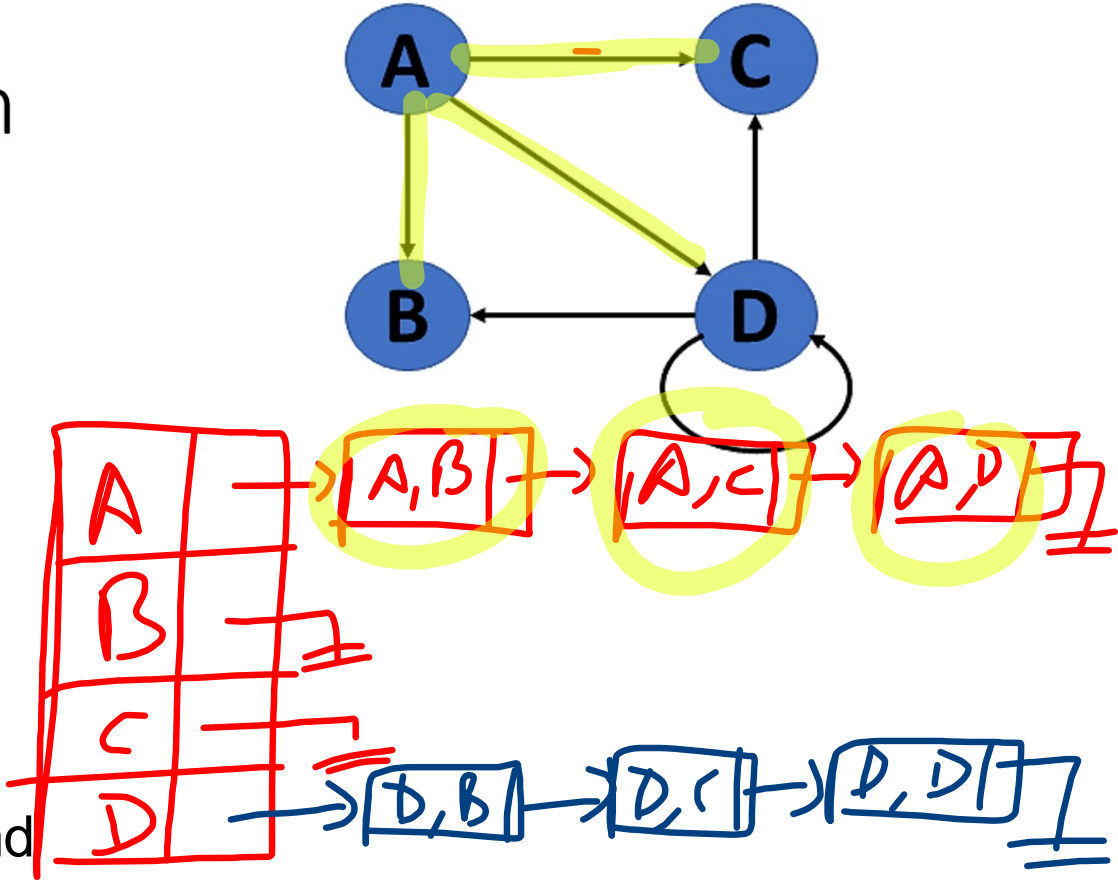




Graph representation

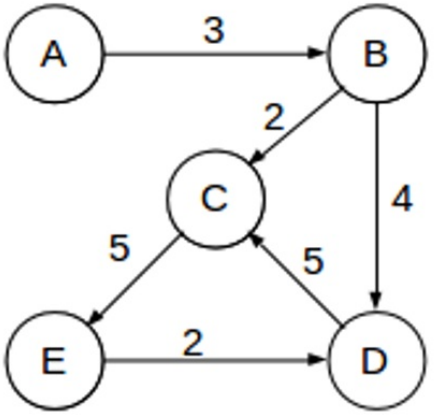
- Adjacency List
  - Array of linked list
  - Each node is an edge
  - Can grow or sharing with need
  - Storage of (v+e)

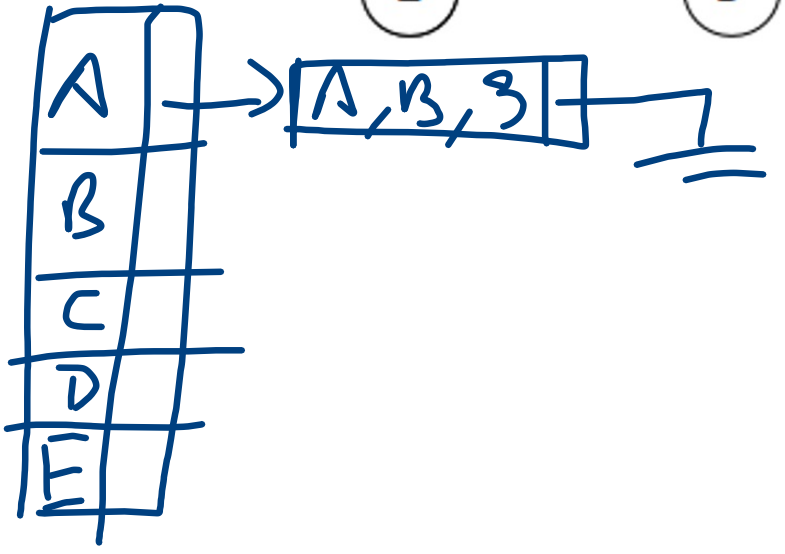
+ve: dynamic and uses space effectively
-ve:too complex to code and use



Graph representation

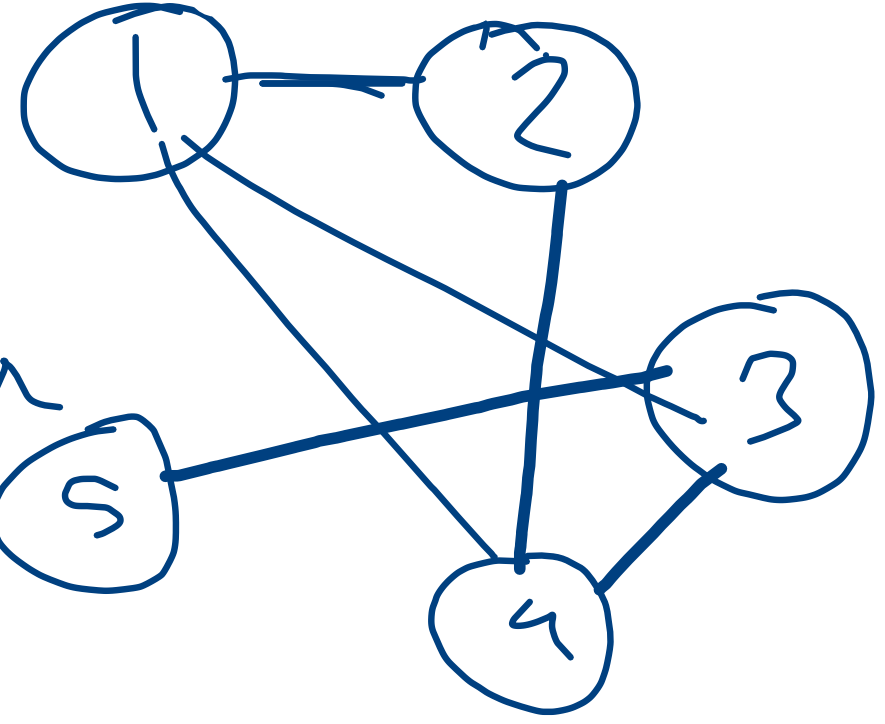
- Adjacency List
  - Array of linked list
  - Each node is an edge
  - Can grow or sharing with need





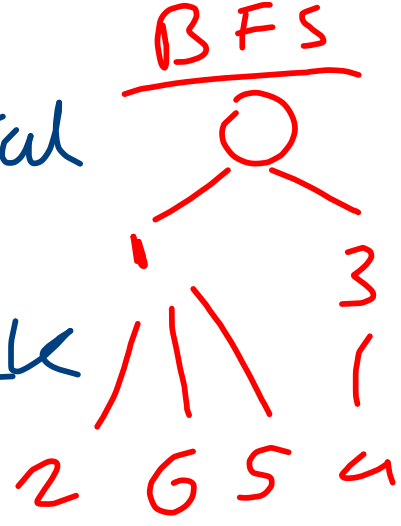
Create a Graph

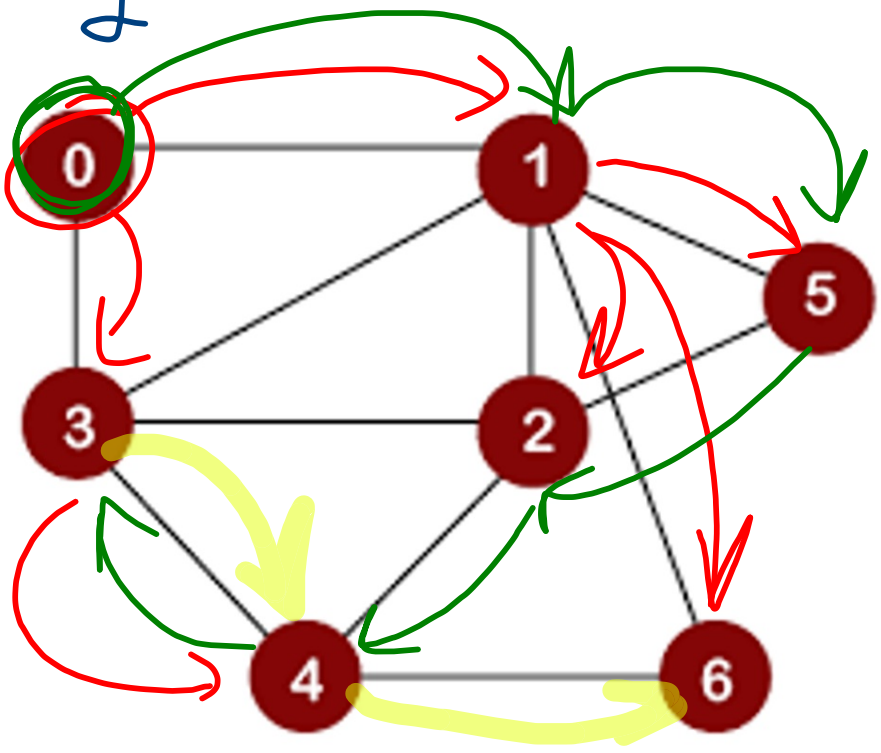
	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	1	0
3	1	0	0	1	1
4	1	1	1	0	1
5	0	0	1	1	0

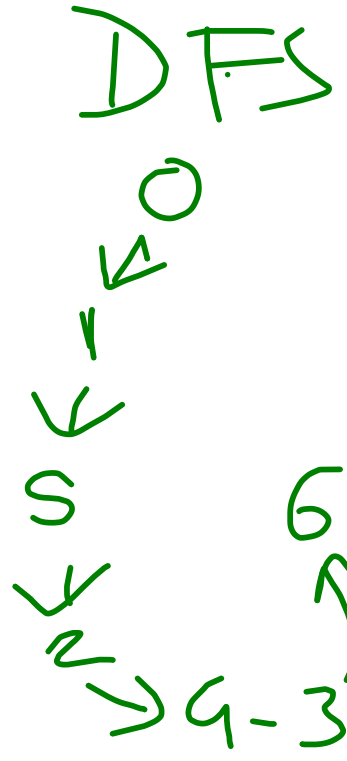


Graph Searching

- BFS → horizontal → queue
- DFS
  - Vertical
  - Stack

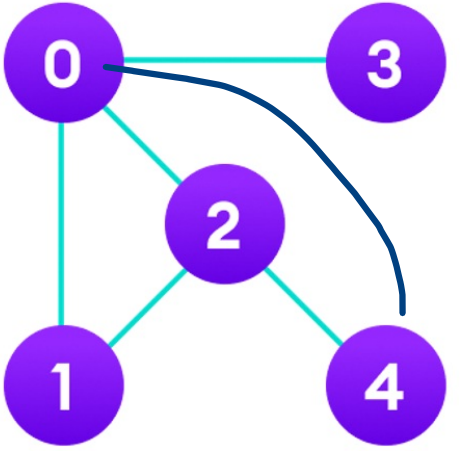






BFS

1. Start by putting any one of the graph's vertices at the back of a queue.
2. Take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4. Keep repeating steps 2 and 3 until the queue is empty.



BFS:(queue)

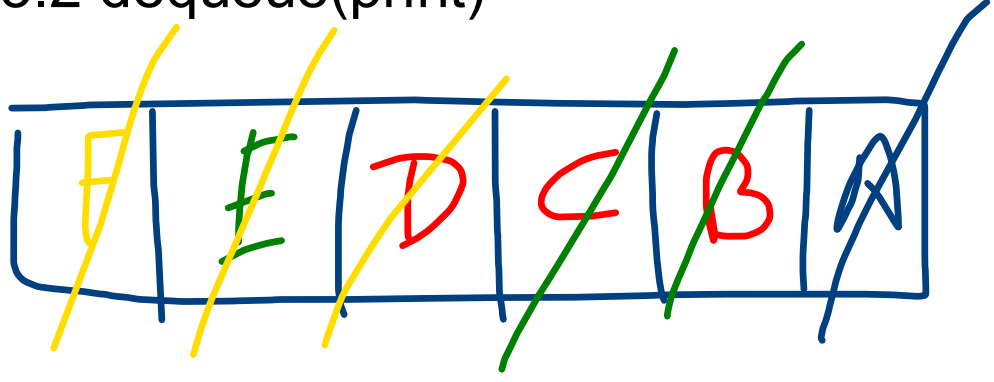
1.accept source

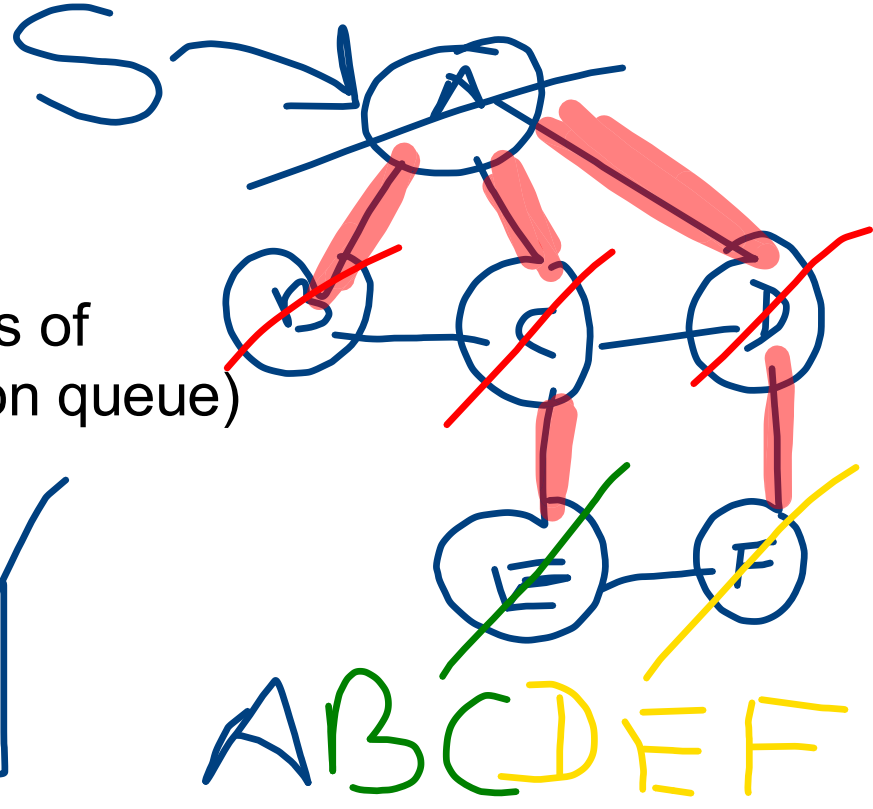
2.mark source visited and enqueue

3.Till queue not empty

3.1 search and insert all neighbours of q[front] in queue(if not already on queue)

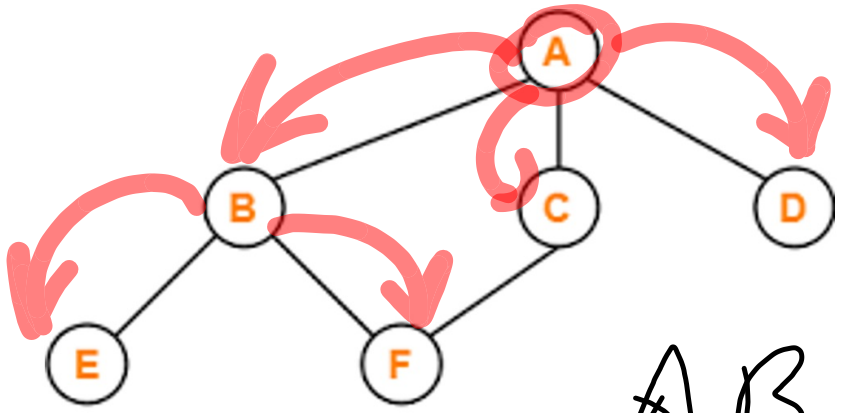
3.2 dequeue(print)





BFS

1. Start by putting any one of the graph's vertices at the back of a queue.
2. Take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4. Keep repeating steps 2 and 3 until the queue is empty.

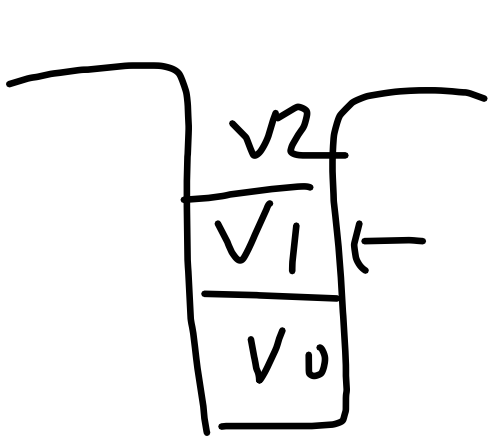


ACDBFE.

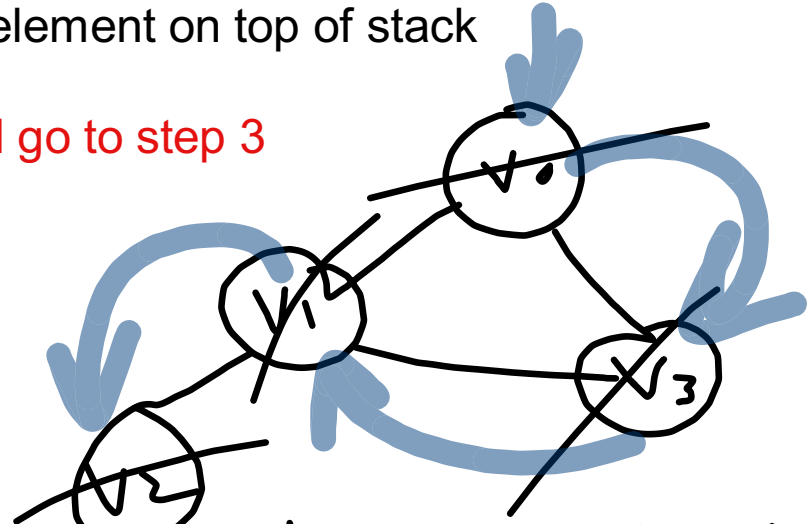
ABCDEF

ADCBFE

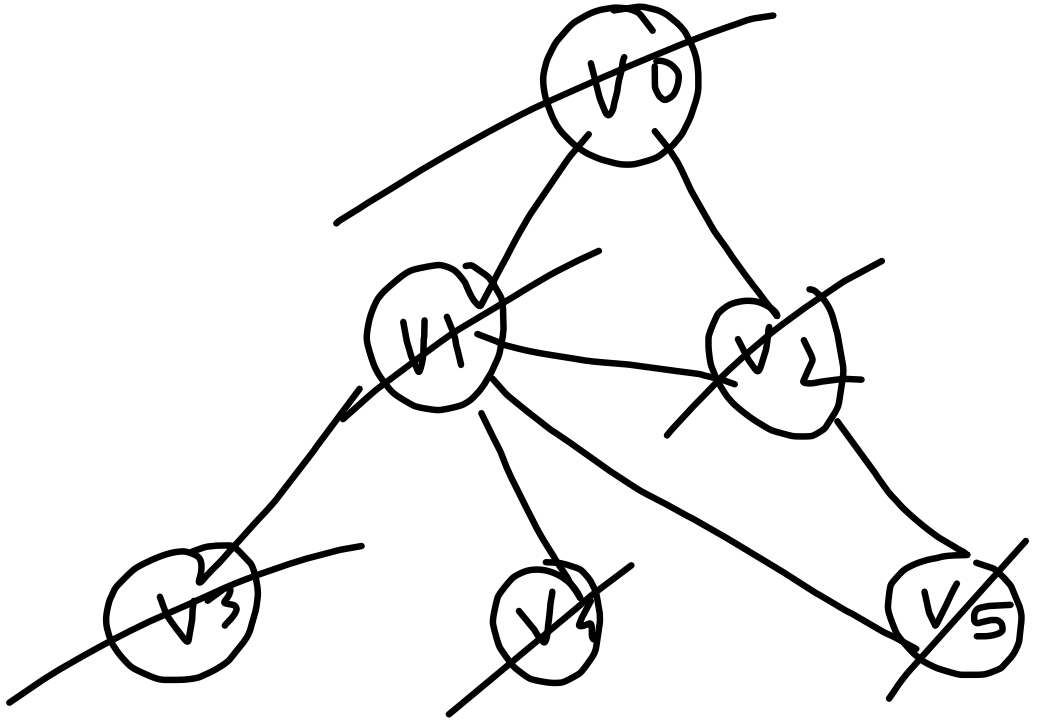
DFS:(stack)
1.accept source
2.mark visited push on stack
3 till stack not over
3.1 search any one unvisited neighbour of element on top of stack
3.2 mark it visited & push on stack
3.3 if no unvisited neighbour found pop and go to step 3



V0
V1
V2
V3



V0 V1 V3 V2



V0V1V2V5
V3V4

DataStructures - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

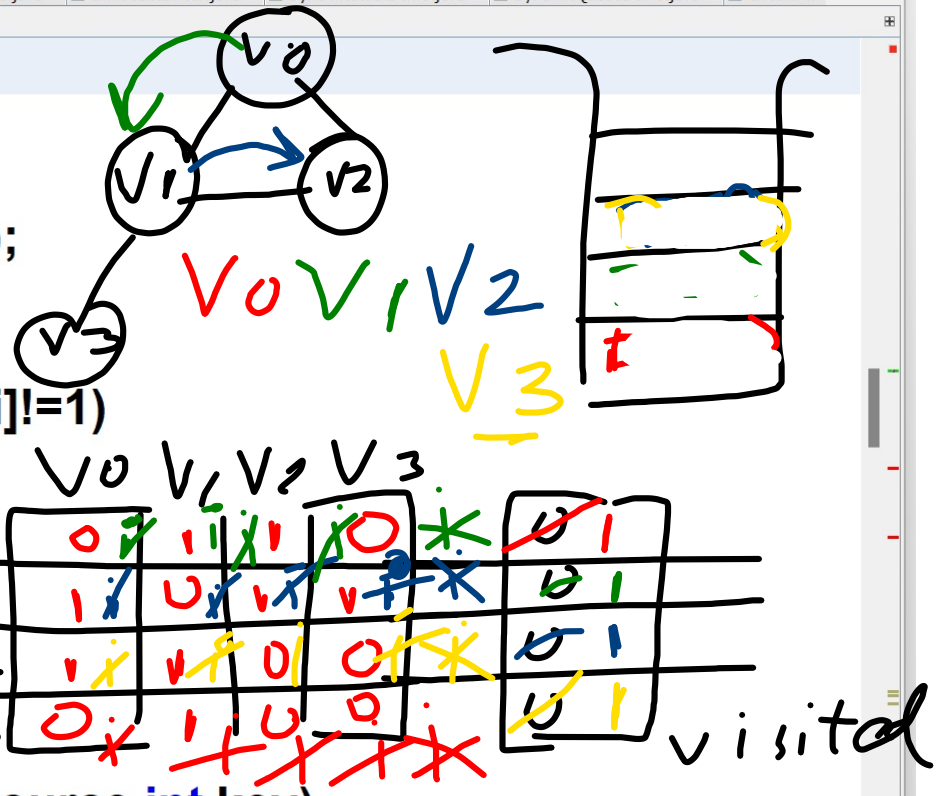
Search (Ctrl+I)

<default confi...

...va GraphDemo.java | StackDemo.java | QueueLinearDemo.java | CircularQueueDemo.java | PriorityQueueDemo.java | LinkedListLinear.java | DynamicStackDemo.java | DynamicQueueDemo.java | Circular...

Source History

```
54  public void DFS(int source)
55  {
56      visited[source]=1;
57      System.out.println("V"+source);
58      for(int i=0;i<v;i++)
59      {
60          if(g[source][i]==1 && visited[i]!=1)
61          //neighbour and unvisited
62          {
63              DFS(i);
64          }
65      }
66  }
    public boolean DFS_search(int source,int key)
```



Output

53:8 INS

DataStructures - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

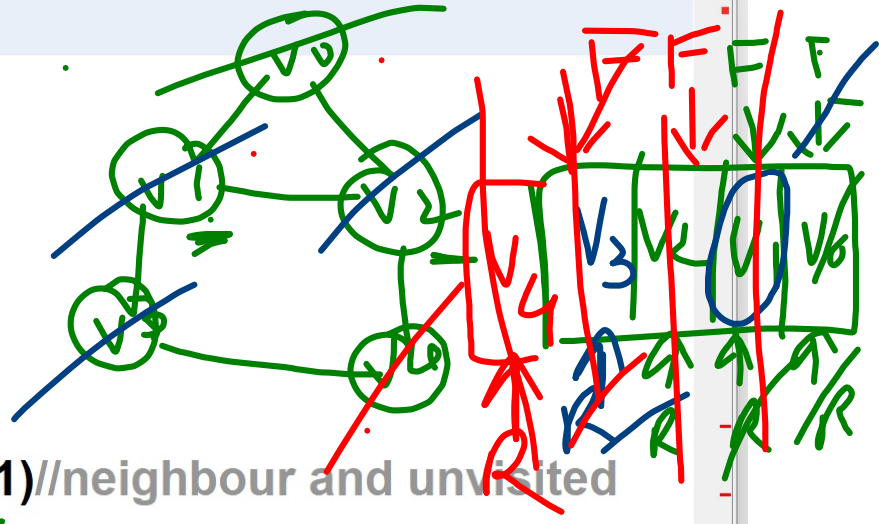
Search (Ctrl+I)

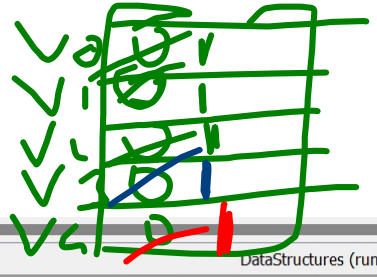
<default confi...

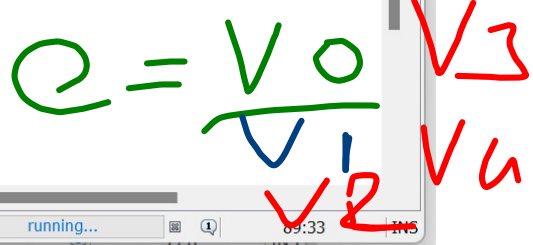
...va GraphDemo.java × StackDemo.java × QueueLinearDemo.java × CircularQueueDemo.java × PriorityQueueDemo.java × LinkedListLinear.java × DynamicStackDemo.java × DynamicQueueDemo.java × CircularLi...

Source History

```
89      q[++rear]=source;//enqueue
90      while(front<=rear)//not empty
91      {
92          int element=q[front++];//dequeue
93          System.out.print("V"+element+"-");
94
95          for(int i=0;i<v;i++)
96           {
97              if(g[element][i]==1 && visited[i]!=1)//neighbour and unvisited
98              {
99                  visited[i]=1;//visited
100                 q[++rear]=i;//enqueue
101             }
102          }
```







Output

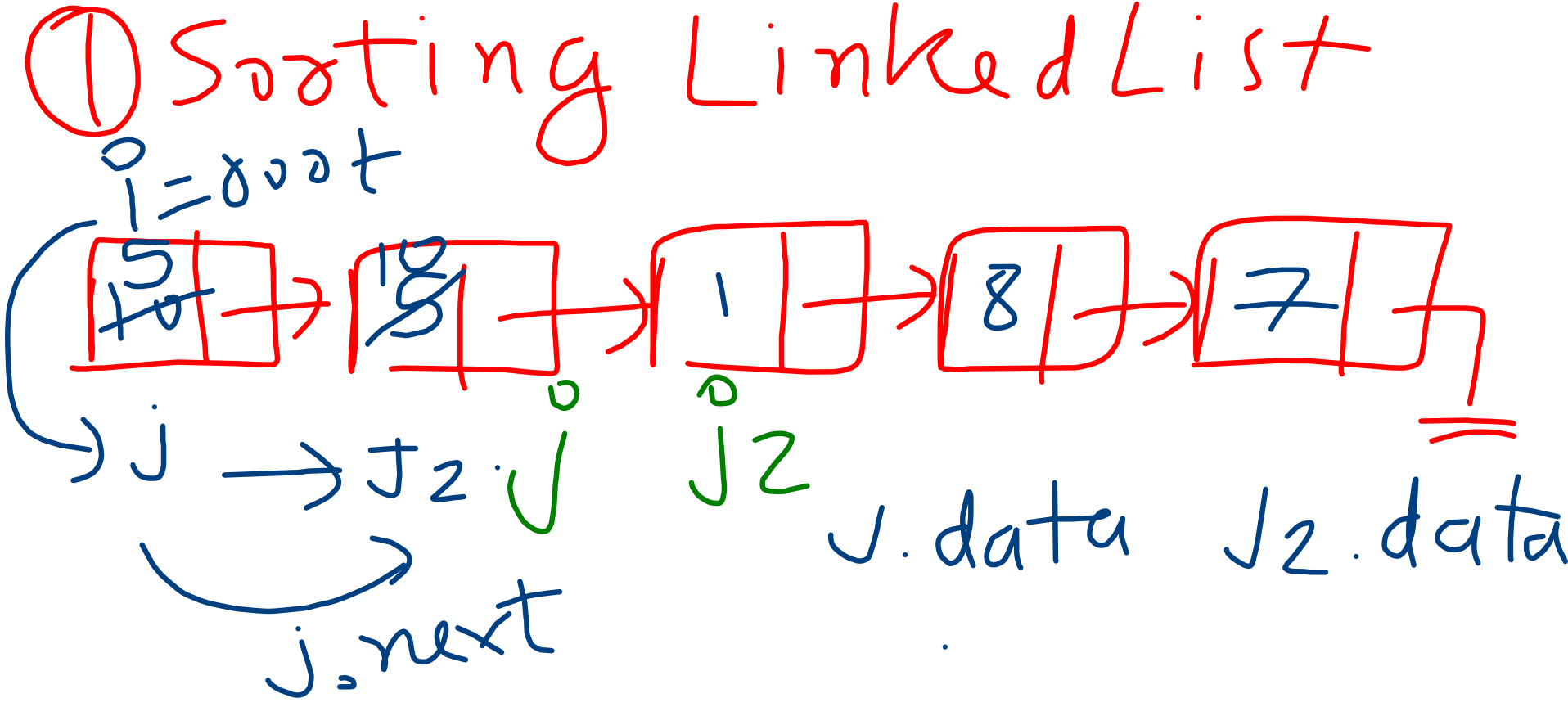
DataStructures (run) running... 89:33 INS

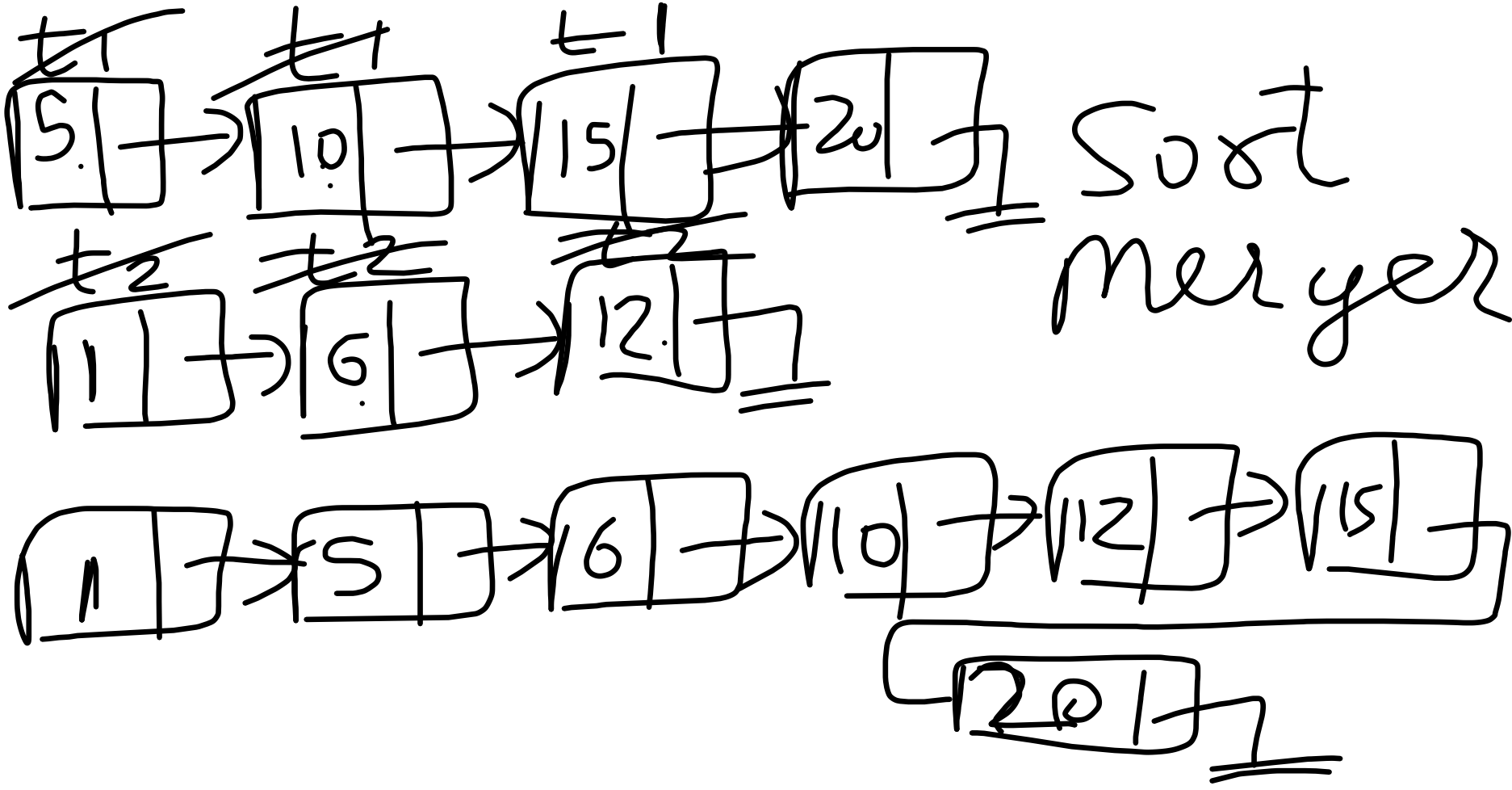
Search

ENG IN

14:28 29-11-2022

① Sorting LinkedList




t1
t1
t1
5
10
15
20
Sort
merger
t2
t2
t2
1
6
12
1
5
6
10
12
15
20