18EC302

Hall Ticket Number:

II/IV B.Tech (Regular) DEGREE EXAMINATION

October, 2019 — **Electronics and Communication Engineering**

Third Semester — **Data Structures Using Python**

Time: Three Hours — **Maximum :** 50 Marks

Answer Question No.1 compulsorily. (1X10 = 10 Marks)

Answer ONE question from each unit. (4X10=40 Marks)

1. Answer all questions (1X10=10 Marks)

a. What are the rules we need to follow for naming the identifiers in python?
b. Illustrate the purpose of invoking append() through the object of list class
c. Define module
d. Define linked list
e. Why we called stack as LIFO system
f. What are the preconditions of a Binary search tree
g. Construct Binary search tree for the elements: 20, 5, 30, 4, 45, 36, 21, 11, and 16
h. Define sub tree
i. What is degree of a node in a tree
j. Define spanning tree

UNIT – I

2.a Explain built-in classes implemented in python 5M

2.b Develop a python program by defining a class complex with Member function to perform : 5M

- real and imaginary values initialization
- complex numbers display
- two complex numbers addition

(OR)

3.a Explain referential arrays and compact arrays 5M

3.b Write a python program to find an element in list using linear search and binary search algorithms 5M

UNIT – II

4.a What is linked list? Explain briefly about singly, circular and doubly linked lists. 5M

4.b Write a python program to implement circular linked list using queue 5M

(OR)

5.a Define queue. Explain how to implement queues using arrays. 5M

5.b Write a python program to implement queue using linked list 5M

UNIT – III

6.a Define tree. Explain basic properties of a tree 5M

6.b Construct BST for the following key sequence 34 12 67 58 22 27 20 45 50. 5M

(OR)

7.a Write the differences between the binary search tree & AVL tree. Construct AVL tree with the followings list of values 3, 1, 4, 6, 9, 2 &5. 5M

7.b Discuss in detail about the rotations in an AVL tree using appropriate examples. 5M

UNIT – IV

8.a Write algorithms for DFS and BFS traversals on a graph. 5M

8.b Write a python program on depth first search in a graph 5M

(OR)

9.a Consider an example graph and show how to construct a breadth first spanning tree and depth first spanning tree. 5M

9.b Illustrate how to determine minimum cost spanning tree of a given graph using Krushkal’s algorithm. 5M

Schema of Evaluation

1. Answer all questions (1X10=10 Marks)

a. What are the rules we need to follow for naming the identifiers in python?

Ans.
i. Alphabets, underscores and digits are permitted for naming the identifiers.
ii. Identifier name should not be start with digit
iii. Uppercase and lower case letters must be distinct
iv. Keywords cannot be used for naming the identifiers

b. Illustrate the purpose of invoking append() through the object of list class

Ans. append ( ) is a member function of list class, which is defined to
- To add a new element to the empty list
- To add a new element at end of the list

c. Define module

Ans. Module is variable or class or function, which is already predefined

d. Define linked list

Ans.
- Linked list is very common data structure, in which data elements are stored in different memory locations.
- Link should be formed between the elements, which are stored in different memory locations, by storing reference of next element in the memory part of previous element.

e. Why we called stack as LIFO system

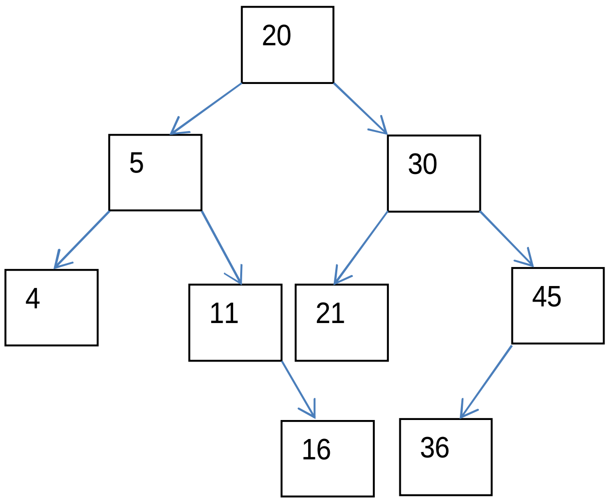
Ans. LIFO means Last in First out, stack is LIFO, because in stack last inserted element should be deleted first.

f. What are the preconditions of a Binary search tree

Ans.
- Root of left-sub tree always be less than Root ------------- L.S<Root
- Root of right-sub tree always be greater than Root -------- R.S>Root

g. Construct Binary search tree for the elements: 20, 5, 30, 4, 45, 36, 21, 11, and 16

Ans.



h. Define sub tree

Ans. Sub tree is a connected tree below the Root

i. What is degree of a node in a tree

Ans. The number of edges connected to a node is called degree of a node

j. Define spanning tree

Ans. Spanning tree is a Solely connection of all edges in a graph.

UNIT – I

2.a Explain built-in classes implemented in python 5M

Ans.

Class	Description	Immutable?
bool	Boolean value	Immutable
int	integer	Immutable
float	Floating point number	Immutable
list	Mutable sequence of objects	
tuple	imutable sequence of objects	Immutable
str	Character strings	Immutable
set	Unordered set of distinct objets	
frozenset	Immutable form of set class	Immutable
dict	Associative mapping (aka dictionary)	

bool class: bool is used to manipulate logical (Boolean)values, and the only two instances of that class are expressed as the literals True and False.

int class: int and float classes are the primary numeric types in python. The int class is designed to represent integer values with arbitrary magnitude. Python automatically chooses the internal representation for an integer based upon the magnitude of its value. The integer constructor int() returns value 0 by default. But this constructor can be used to construct an integer value based upon an existing value of another type. Eg: int(3.7) produce a value 3.

float class: the float class is the sole floating-point type in python, using a fixed precision representation. The constructor form of float () returns 0.0. When given a parameter, the constructor attempts to return the equivalent floating point value. For example the call float (2) returns the floating point value 2.0.

str class:- Python's str class is specifically designed to efficiently represent an immutable sequence of characters, based upon the Unicode international character set. Python string, which is an indexed sequence of characters. String literals can be enclosed in single quotes, as in hello, or double quotes, as in "hello".

List class:- A list is mutable ordered set of values, where each value is identified by an index. The values that make up a list are called its elements. Lists are similar to strings, which are ordered sets of characters, except that the elements of a list can have any type.
There are several ways to create a new list; the simplest is to enclose the elements

in square brackets ([and]):

```
[10, 20, 30, 40]
["spam", "bungee", "swallow"]
```

Tuple:- So far, you have seen two compound types: strings, which are made up of characters; and lists, which are made up of elements of any type. One of the differences we noted is that the elements of a list can be modified, but the characters in a string cannot. In other words, strings are immutable and lists are mutable.

There is another type in Python called a tuple that is similar to a list except that it is immutable. Syntactically, a tuple is a comma-separated list of values:

```
>>> tuple = ’a’, ’b’, ’c’, ’d’, ’e’
```

Although it is not necessary, it is conventional to enclose tuples in parentheses:

```
>>> tuple = (’a’, ’b’, ’c’, ’d’, ’e’)
```

Set and frozenset:- Python’s set class represents the mathematical notion of a set, namely a collection of elements, without duplicates, and without an inherent order to those elements. The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set.

The frozen set class is an immutable form of the set type, so it is legal to have a set of frozensets. Python uses curly braces { and } as delimiters for a set, for example, as {17} or { red , green , blue }. The exception to this rule is that { } does not represent an empty set; for historical reasons, it represents an empty dictionary. Instead, the constructor syntax set( ) produces an empty set. If an iterable parameter is sent to the constructor, then the set of distinct elements is produced. For example, set(hello) produces { h , e , l , o }.

The dict Class:- Python’s dict class represents a dictionary, or mapping, from a set of distinct keys to associated values. For example, a dictionary might map from unique student ID numbers, to larger student records (such as the student’s name, address, and course grades). Python implements a dict using an almost identical approach to that of a set, but with storage of the associated values.

A dictionary literal also uses curly braces, and because dictionaries were introduced in Python prior to sets, the literal form { } produces an empty dictionary. A nonempty dictionary is expressed using a comma-separated series of key:value pairs. For example, the dictionary { ga : Irish , de : German } maps ga to Irish and de to German .

2.b Develop a python program by defining a class complex with Member function to perform : 5M

- real and imaginary values initialization
- complex numbers display
- two complex numbers addition

Ans.

```
class complex:

 def read(self):

   print "enter real and img values"

   self.real=input()

   self.img=input()

 def display(self):

   print self.real,"+i",self.img

 def add(self,c1,c2):

   self.real=c1.real+c2.real

   self.img=c1.img+c2.img


obj1=complex()

obj1.read()

obj1.display()
```

```
obj2=complex()

obj2.read()

obj2.display()


obj3=complex()

obj3.add(obj1,obj2)

obj3.display()
```

(OR)

3.a Explain referential arrays and compact arrays 5M

Ans. Referential Arrays:

In Python, each cell of the array must use the same number of bytes. Given a list of names:

['Rene ', 'Joseph ', 'Janet ', 'Jonas ', 'Helen ', 'Virginia ']

How to represent such a list within an array?

Way 1: Reserve enough space for each cell to hold the maximum length string, but that would be wasteful

Way 2: Python represents a list or tuple instance using an internal storage mechanism of an array of object references. At the lowest level, what is stored is a consecutive sequence of memory addresses at which the elements of the sequence reside.

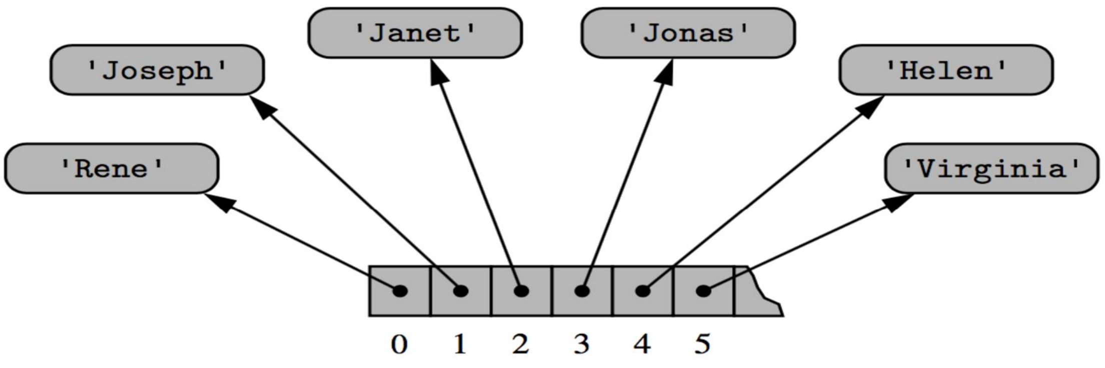
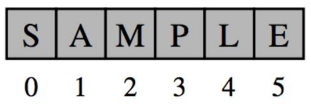


Figure 5.4: An array storing references to strings.

Compact Arrays in Python:

Strings are represented using an array of characters (NOT an array of references). We will refer to this more direct representation as a compact array because the array is storing the bits that represent the primary data (characters, in the case of strings).



Compact arrays have several advantages over referential structures in terms of computing performance.

• overall memory usage will be much lower

• the primary data are stored consecutively in memory

Primary support for compact arrays is in a module named array. That module defines a class, also named array, providing compact storage for arrays of primitive data types.

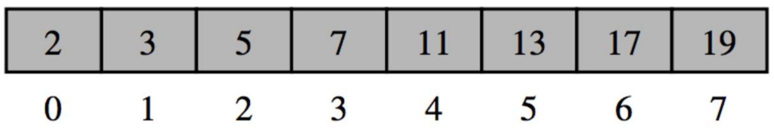


Figure 5.10: Integers stored compactly as elements of a Python array.

3.b Write a python program to find an element in list using linear search and binary search algorithms 5M

Ans Linear search:

```
print(“enter list elements”)
a=input( )
print ("the list  elements are: ",a)

print(“enter a number to search”)
num = input( )
i=0
while i<=9:
        if a[i] = = num:
              break
        i+=1
if  i==10:
         print ("number is not in the list")
else:
         print ("the element is found in ", i ,"th position"
```

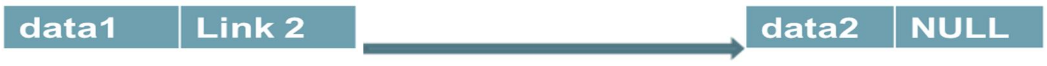
Binary search:

```
Print(“enter list elements”)
a=input( )

print ("the array elements are: ",a)

l=0
u=len(a)-1
flag=1
for i in range(len(a)):
        for j in range(len(a)-i):
                  if(a[j]>a[j+1]):
                      a[j],a[j+1]=a[j+1],a[j]
                  j+=1
          i+=1

print ("the array elements after sorting are: ",a)

print ("enter element to find")
num=input()
mid=(l+u)/2

while l<=u:
      if a[mid]= =num:
                  print ("number is found in ", mid , "th position")
                  flag=0
                  break
      if a[mid]>num:
                  u=mid-1
        else:
                l=mid+1
        mid=(l+u)/2


    if flag= =1:
        print( "the element is not in the list")
```

UNIT – II

4.a What is linked list? Explain briefly about singly, circular and doubly linked lists. 5M

Ans Linked list: linked list is a very common data structure, in which the data elements are stored in different memory locations. The link should be formed between the elements, which are stored in different memory locations, by storing reference of next element in memory part of previous element.
So, in linked list we have two parts in each element memory, data and link part, that's why in linked list every element memory represented as a node.
We can easily understand this concept by using below diagram.

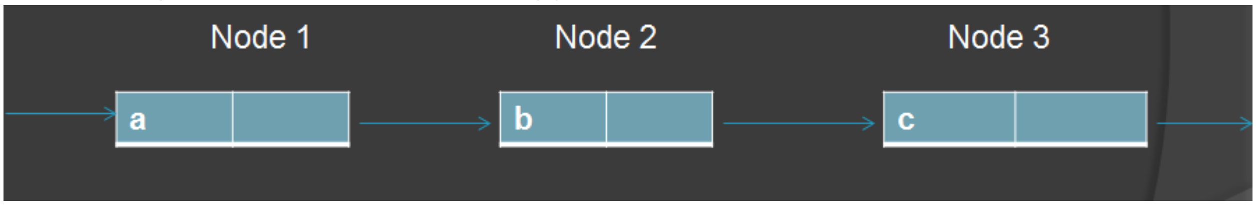


Node-1 Node-2

According to direction of traversing the linked lists are classified in to three types. They are

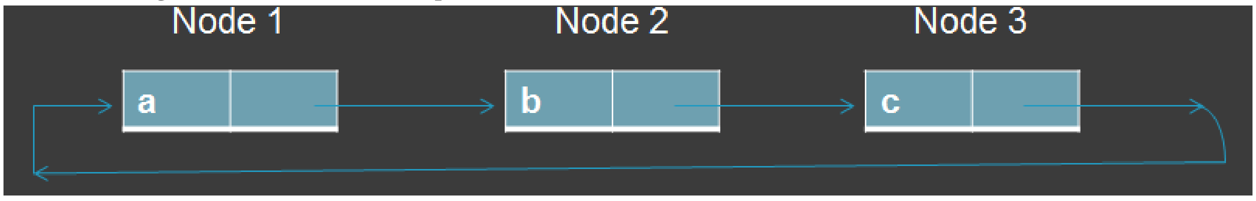
- Singly linked list
- Circular linked list
- Doubly linked list

Singly linked list:- Singly linked list is the linked list, that uses single set of reference variable that is used to refer next node. By using that reference variable, we can traverse in single direction, it not possible come back to previous nodes and in singly linked list the link part of last node consists of None.
The below figure illustrate the concept of singly linked list



In above figure, the link part of last node is None. We cannot come back to previous nodes once we traverse in a single linked list. To overcome this problem we go for circular linked lists

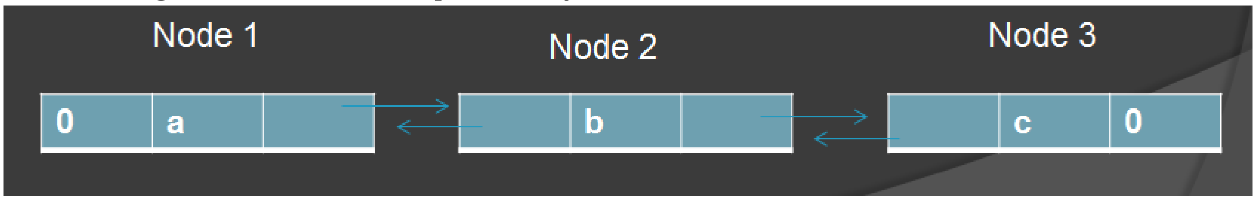
Circular linked list:- Circular linked list is the linked list, that uses single set of reference variable, used to refer next node like a singly linked list. So in a circular linked list, we can traverse in single direction. Here the difference between these two linked lists is we can come back from last node to first node in circular linked list, because the link part of last node contains the reference of first node. But it is not possible in singly linked list.
The below figure illustrate the concept of circular linked list



In circular linked list, although we can come back from last node to first node, it is time taken process to reach last node to last but one node. To overcome this problem we go for doubly linked list.

Doubly linked list: - Doubly linked list is the linked list, that uses two sets of reference variables, one used to refer next node and another used to refer previous node. That's why every node in doubly linked list consists of three parts, one contains reference of previous node, second part contains data and third part contains reference of next node.

The below figure illustrate the concept of doubly linked list:



4.b Write a python program to implement circular linked list using queue 5M

Ans

```
class Node:
        data=0
        link=0


def append(self,num):
        if self.data==0:
           self.data=num
           self.link=self
        else:
           temp=self
           while temp.link!=self:
              temp=temp.link
           r=Node()
           r.data=num
           r.link=self
           temp.link=r



def delet(self):
        temp=Node()
        temp=self
        if temp==None:
           return None

        while temp.link!=self:
           temp=temp.link

        self.data=None
        self=self.link
        temp.link=self
        return self



def disp(self):
        temp=Node()
        temp=self
        while temp!=0:

           print temp.data
           temp=temp.link
           if (self==temp):
```

```
                    break



obj=Node()
append(obj,10)
append(obj,8)
append(obj,12)
append(obj,15)
append(obj,9)
print "the list before deleting "
disp(obj)
obj=delet(obj)
obj=delet(obj)
print "the list after deleting "
disp(obj)
```

(OR)

5.a Define queue. Explain how to implement queues using arrays. 5M

Ans Queue: - Queue is linear data structures in which insertion of new elements takes place at one end called rear, and deletion of existing elements takes place from another end called front.

The operations we are going to perform on queues are

- Enqueue: - allows adding a new element at rear end of queue
- Dequeue: - allows removing an existing element from front end of queue.

Implementation of queue using arrays: -

An array is used to store the ordered list of elements. Hence it is very easy to manage a queue if we represent it using arrays. However the problem with an array is that we are required to declare the size of the array before using it in a program.

Though an array and queues are totally different data structures, an array can be used to store the elements in a queue. We can declare the array with maximum size large enough to manage queue.

Suppose if the user wants to insert on more item to the queue, do the following:

- Increment the rear position by one.
- Check whether it is with in the limit.
- If it is within the limit, insert a new element at this position.

Suppose if the user wants to delete an existing element from the queue, do the following:

- The element at the position pointed out by front will be removed
- The front position will be incremented by one.

The element in Front position				The element in rear position

To implement the queue using array, the following are required:

- list
- A variable to indicate the front position
- Another variable to indicate the rear position.

5.b Write a python program to implement queue using linked list 5M

Ans
```
class Node:
        data=0
        link=0
```

```
def append(self,num):
        if self.data==0:
            self.data=num
            self.link=None
        else:
            temp=self
            while temp.link!=None:
                temp=temp.link
            r=Node()
            r.data=num
            r.link=None
            temp.link=r



def delet(self):
        temp=Node()
        temp=self
        if temp==None:
            return None
        self.data=None
        self=self.link
        return self
def disp(self):
        while self!=None:
                print self.data
                self=self.link




obj=Node()
append(obj,10)
append(obj,8)
append(obj,12)
append(obj,15)
append(obj,9)
print "the list before deleting "
disp(obj)
obj=delet(obj)
obj=delet(obj)
print "the list after deleting "
disp(obj)
```

UNIT – III

6.a Define tree. Explain basic properties of a tree 5M

Ans Tree:-A tree is a nonlinear data structure which is having hierarchal relationship between the finite set of structure elements. Trees consists of finite set of elements, called nodes, and a finite set of directed lines called branches, that connect the nodes.

- Degree: The number of branches associated with a node is the degree of the node.
- When the branch is directed toward the node, it is in degree branch (or) predecessor.
- When the branch is directed away from the node, it is an out degree branch (or) successor.

- The sum of the in degree and out degree branches is the degree of the node.
- If the tree is not empty, the first node is called the root.
- The in degree of the root is, by definition, zero.
- With the exception of the root, all of the nodes in a tree must have an in degree of exactly one; that is, they may have only one predecessor.
- All nodes in the tree can have zero, one, or more branches leaving them; that is, they may have out degree of zero, one, or more.
- A leaf is any node with an out degree of zero, that is, a node with no successors.
- A node that is not a root or a leaf is known as an internal node.
- A node is a parent if it has successor nodes; that is, if it has out degree greater than zero.
- A node with a predecessor is called a child.
- Two or more nodes with the same parents are called siblings.
- An ancestor is any node in the path from the root to the node.
- A descendant is any node in the path below the parent node; that is, all nodes in the paths from a given node to a leaf are descendants of that node.
- A path is a sequence of nodes in which each node is adjacent to the next node.
- The level of a node is its distance from the root. The root is at level 0, its children are at level 1, etc. …
-
- The height of the tree is the level of the leaf in the longest path from the root plus 1. By definition the height of any empty tree is -1.
- A sub tree is any connected structure below the root. The first node in the subtree is known is the root of the sub tree.

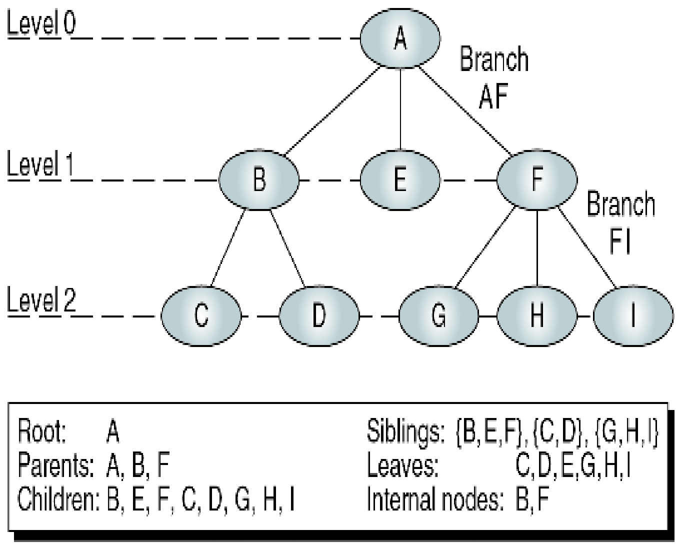


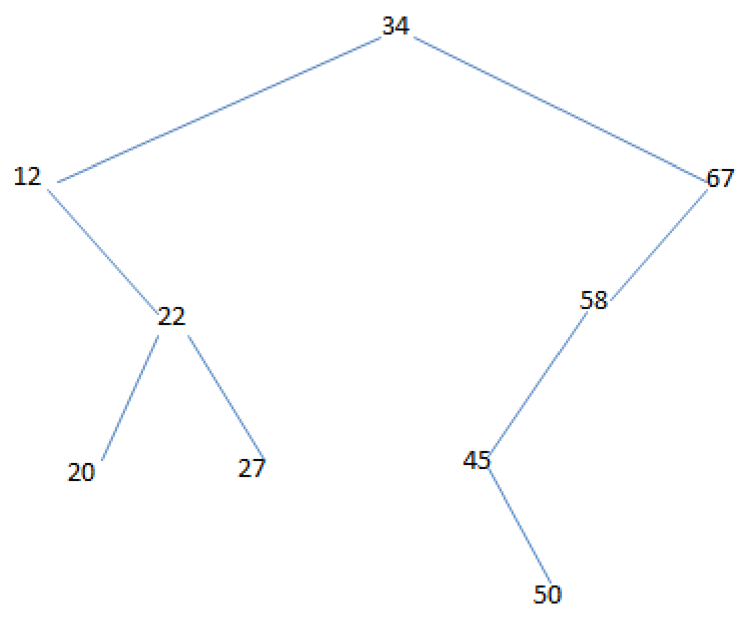
FIGURE 6-2 Tree Nomenclature

6.b Construct BST for the following key sequence 34 12 67 58 22 27 20 45 50. 5M

Ans Consider X is the root of a tree. For every node X, all the keys in its left subtree are smaller than the key value in X, and all the keys in its right subtree are larger than the key value in X

i.e

- Nodes in leftsubtree < root
- Nodes in right subtree > root



(OR)

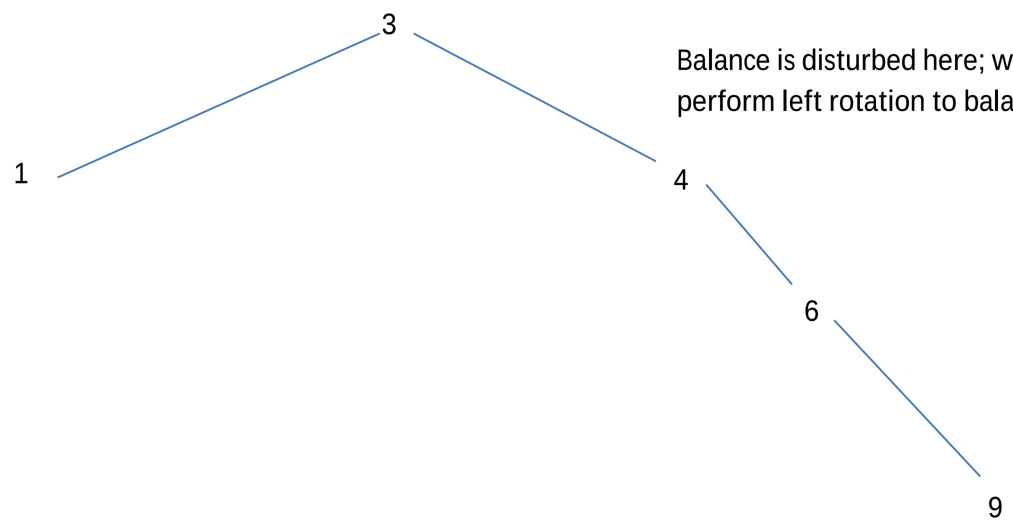
7.a Write the differences between the binary search tree & AVL tree. Construct AVL tree with the followings list of values 3, 1, 4, 6, 9, 2 &5.

Ans Binary search tree: Binary search tree is a binary tree, in which nodes of left sub tree are smaller than root and nodes of right sub tree are greater than root.

AVL tree: AVL tree is a binary search tree, in which the balance factor of each node is at most 1(-1,0,1)

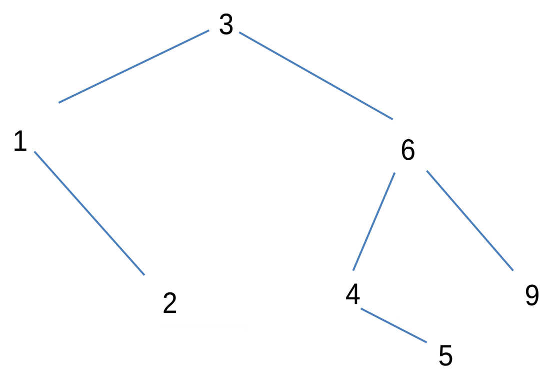
While constructing AVL tree, we have to check balance factor at each and every node after inserting new element.

Step-1:



Balance is disturbed here; we have to perform left rotation to balance

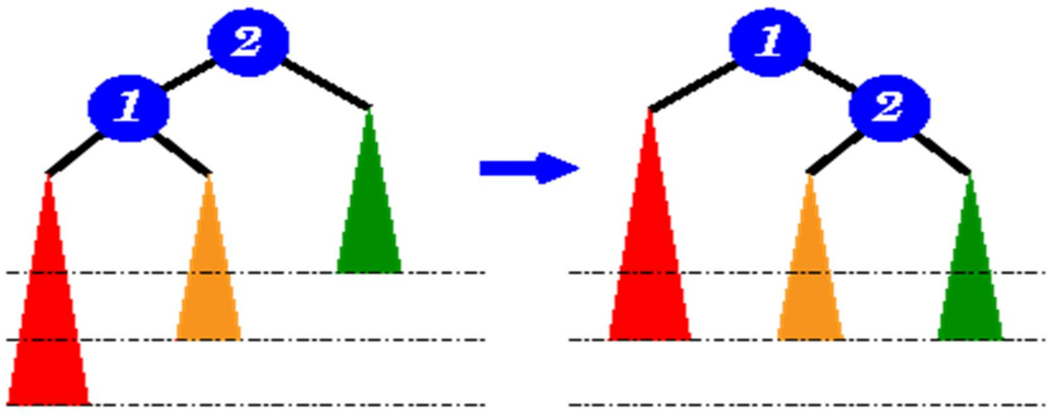
Step-2 :



7.b Discuss in detail about the rotations in an AVL tree using appropriate examples. 5 M

Ans AVL tree is a balanced binary search tree i.e. it is a binary search tree in which each node consists the balance factor at most 1(-1,0,1). The operations performed on AVL trees are insertion, deletion. While constructing AVL trees if any imbalance occurs at any node, we correct that imbalance by performing rotations.

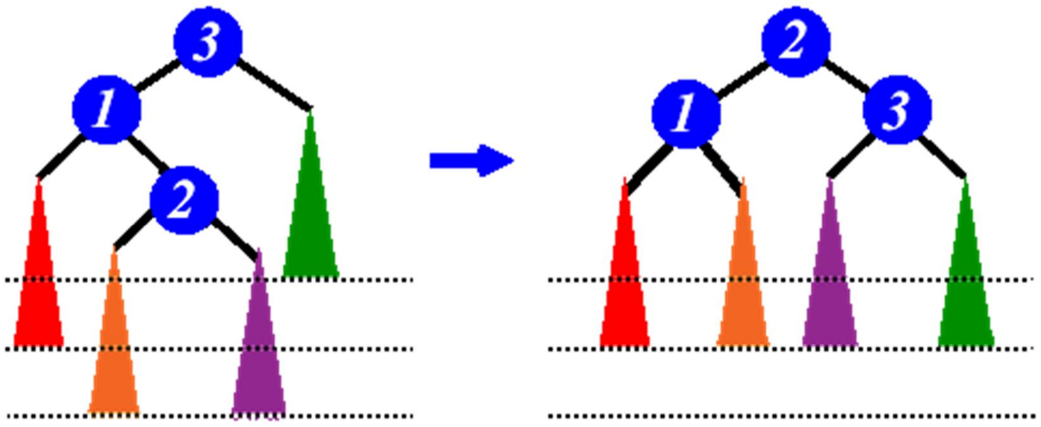
<u>Single Rotation:-</u>



Left sub-tree is two level deeper than the right sub-tree

move ① up a level and ② down a level

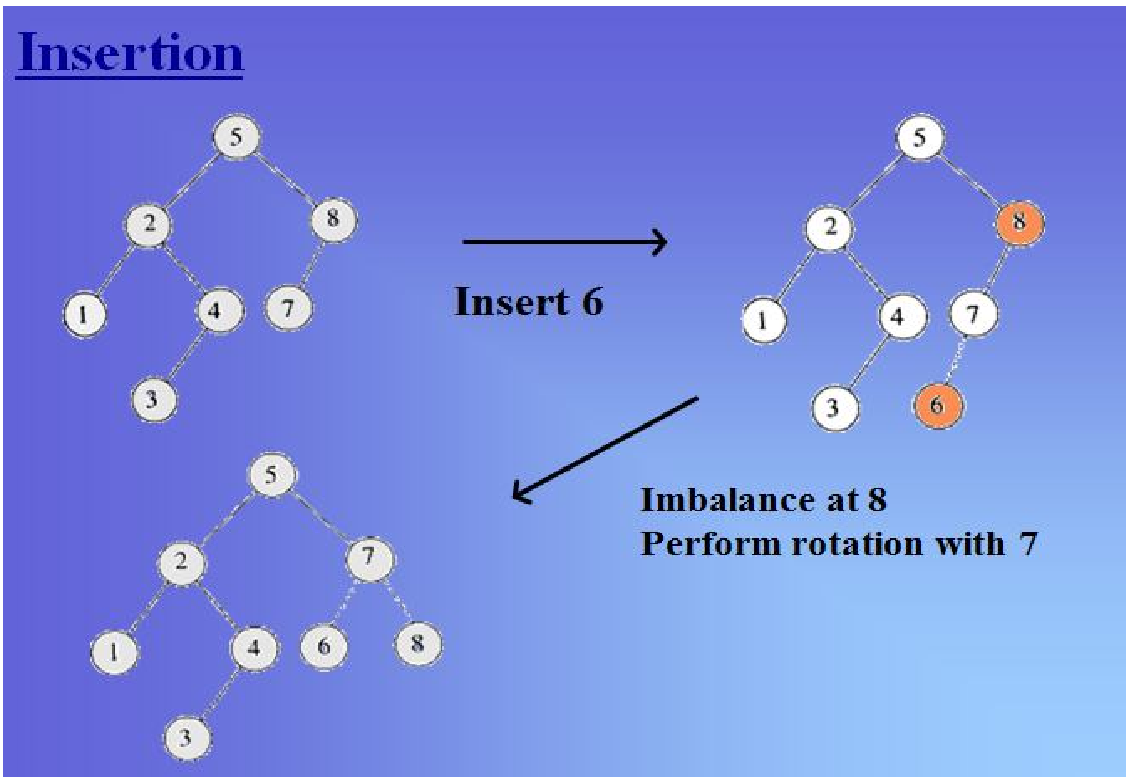
Double rotation:-



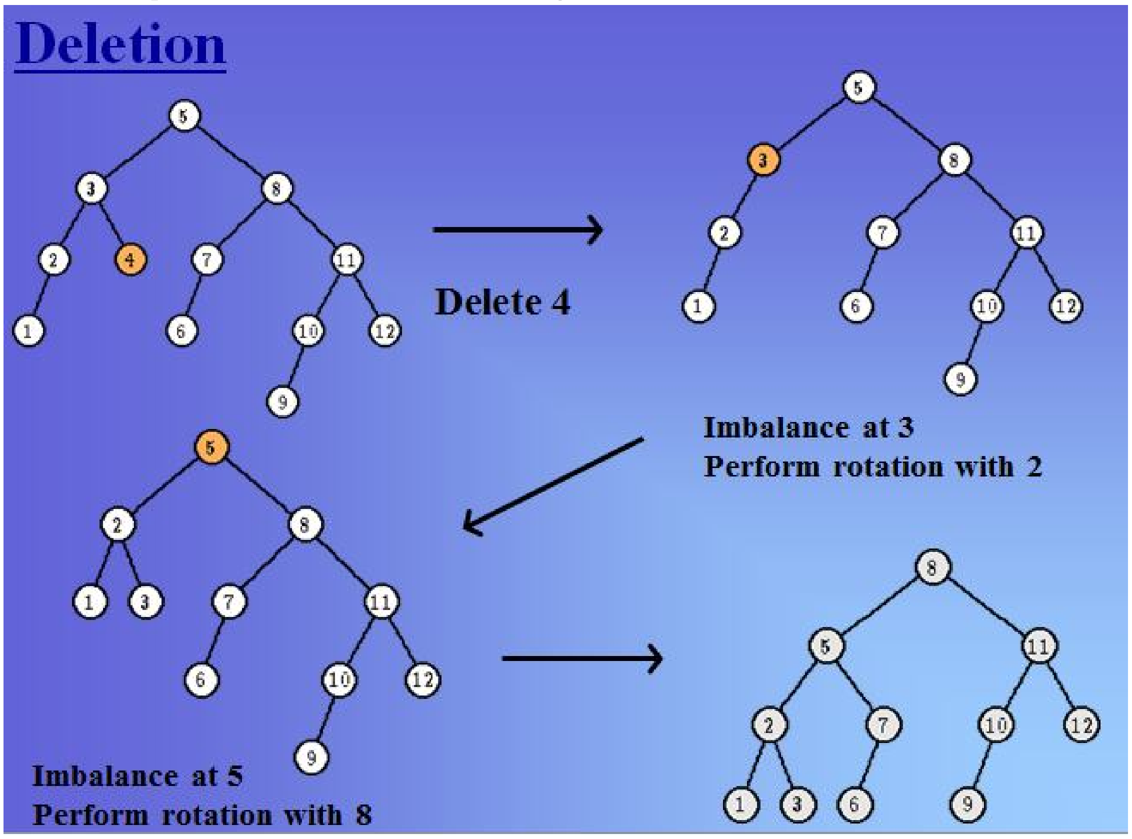
Left sub-tree is two level deeper than the right sub-tree

Move ② up two levels and ③ down a level

Below example shows how to insert a new element in AVL trees



Below example shows how to delete an existing element from AVL trees



UNIT – IV

8.a Write algorithms for DFS and BFS traversals on a graph. 5M

Ans Depth first search (DFS): -

Depth first search algorithm of a graph is as follows:

a) Visit a vertex; say v. Mark this vertex as visited.

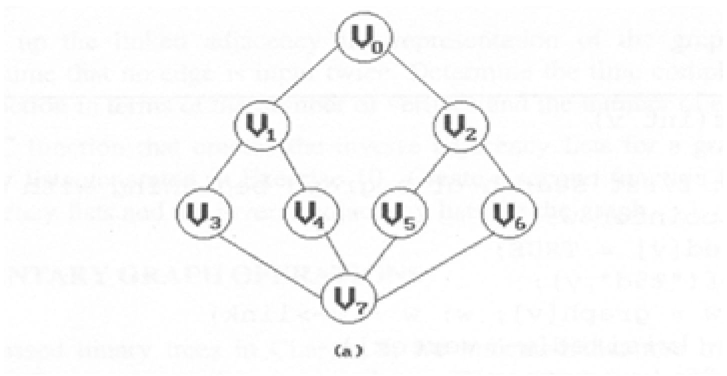
b) Select an unvisited vertex W adjacent to v.

c) Repeat steps (a) and (b) till all adjacent vertices of w are visited.

d) On reaching a vertex whose all adjacent vertices have been visited go back to the last vertex visited which has an unvisited vertex adjacent to it and go back to step (a).

e) Terminate the search when no unvisited vertex can be reached from any of the visited ones.

Breadth first search (BFS) : -
Let us now understand the Breadth first search algorithm. In this algorithm we need to start at some vertex v and marks it as visited. Then visit all unvisited vertices adjacent to these vertices. This process is continued until all the vertices are visited in a graph.
Example:



Depth first Traversal: v0, v1, v3, v7, v4, v5, v2, v6
Breadth first Traversal: v0, v1, v2, v3, v4, v5, v6, v7

8.b Write a python program on depth first search in a graph 5M

Ans

Source code:-

```
 class node:
   data=None
   nex=None




def dfs(v,arr,visited):
   q=node()
   visited[v-1]=True;

   print(v)

   q=arr[v-1];

   while(q!=None):
      if(visited[q.data-1]==False):
         dfs(q.data,arr,visited)
      else:
         q=q.nex;

def getnode(val):
   newnode=node()
   newnode.data=val
   newnode.nex=None
   return newnode


arr=[];
v1=node()
v2=node()
v3=node()
v4=node()
```

```
v1=getnode(2)
arr.append(v1)
v1.nex=v2=getnode(3)
v2.nex=None

v1=getnode(1)
arr.append(v1)
v1.nex=v2=getnode(4)
v2.nex=v3=getnode(5)
v3.nex=None

v1=getnode(1)
arr.append(v1)
v1.nex=v2=getnode(6)
v2.nex=v3=getnode(7)
v3.nex=None

v1=getnode(2)
arr.append(v1)
v1.nex=v2=getnode(8)
v2.nex=None

v1=getnode(2)
arr.append(v1)
v1.nex=v2=getnode(8)
v2.nex=None

v1=getnode(3)
arr.append(v1)
v1.nex=v2=getnode(8)
v2.nex=None

v1=getnode(3)
arr.append(v1)
v1.nex=v2=getnode(8)
v2.nex=None;

v1=getnode(4)
arr.append(v1)
v1.nex=v2=getnode(5);
v2.nex=v3=getnode(6);
v3.nex=v4=getnode(7);
v4.nex=None;

visited=[]
for i in range(len(arr)):
    visited.append(False)

dfs(1,arr,visited)
```

(OR)

9.a Consider an example graph and show how to construct a breadth first spanning tree and depth first spanning tree. 5M

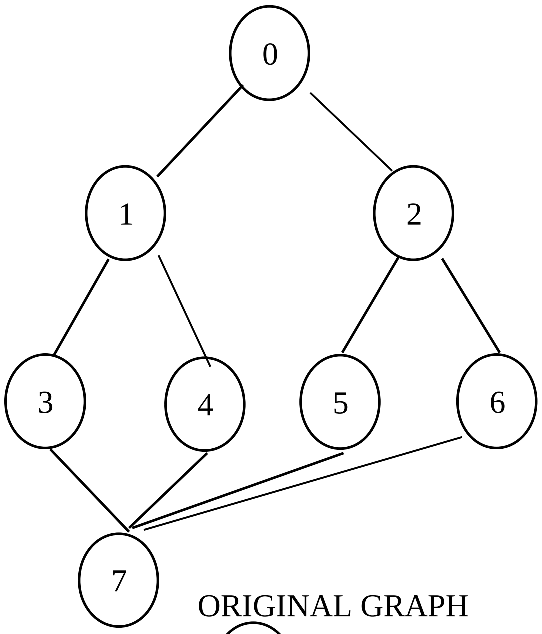
Ans <u>Spanning tree</u>: - A spanning tree is any tree that consists solely (single connection) of edges in G and that includes all the vertices

<u>Creation of spanning tree</u>: -

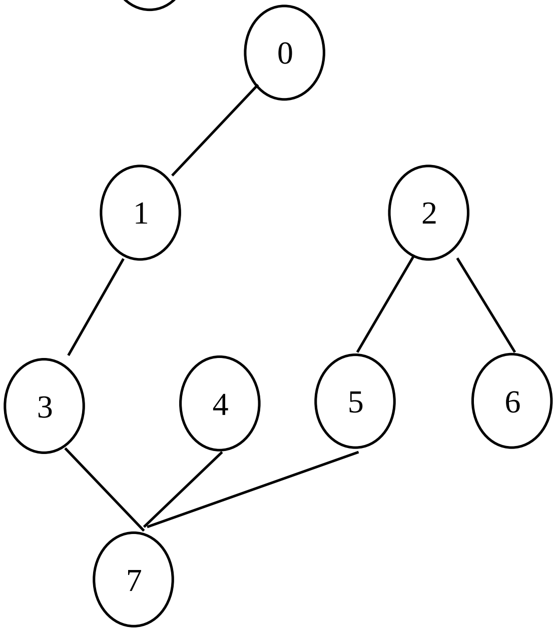
Either DFS or BFS can be used to create a spanning tree

– Depth first spanning tree: -The spanning tree resulting from a call to depth first is known as a depth first spanning tree

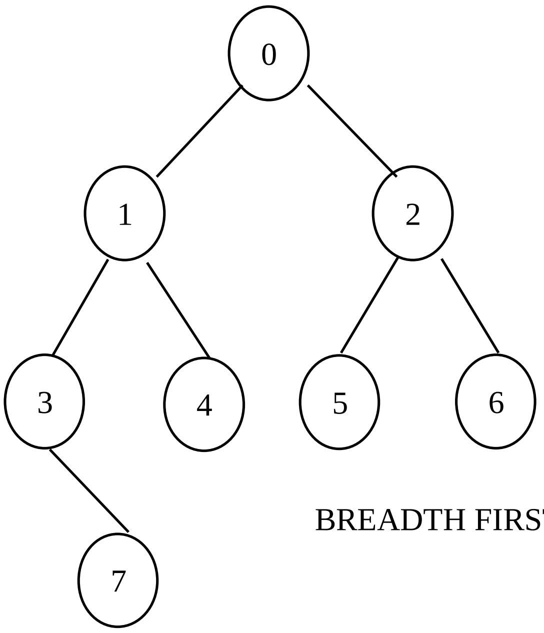
– Breadth first spanning tree: -The spanning tree resulting from a call to breadth first is known as a breadth first spanning tree



ORIGINAL GRAPH



DEPTH FIRST SPANNING TREE



BREADTH FIRST SPANNING TREE

9.b Illustrate how to determine minimum cost spanning tree of a given graph using Krushkal's algorithm. 5M

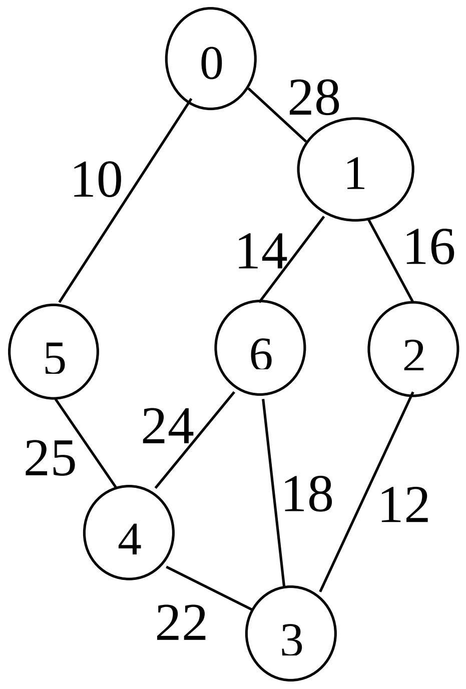
Ans The cost of a spanning tree of a weighted undirected graph is the sum of the costs of the edges in the spanning tree. A minimum cost spanning tree is a spanning tree of least cost.

<u>Kruskals idea</u>: -

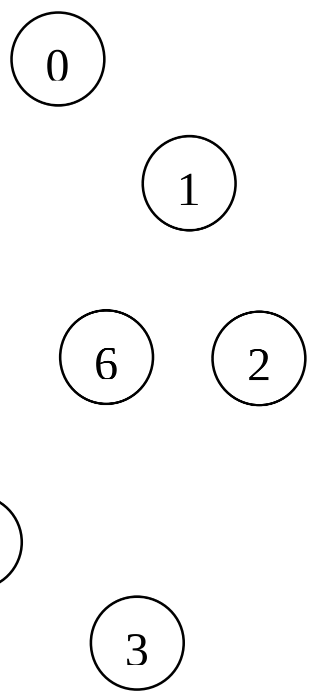
- Build a minimum cost spanning tree T by adding edges to T one at a time.
- Select the edges for inclusion in T in non-decreasing order of the cost
- An edge is added to T if it does not form a cycle
- Since G is connected and has n > 0 vertices, exactly n-1 edges will be selected.

Cost

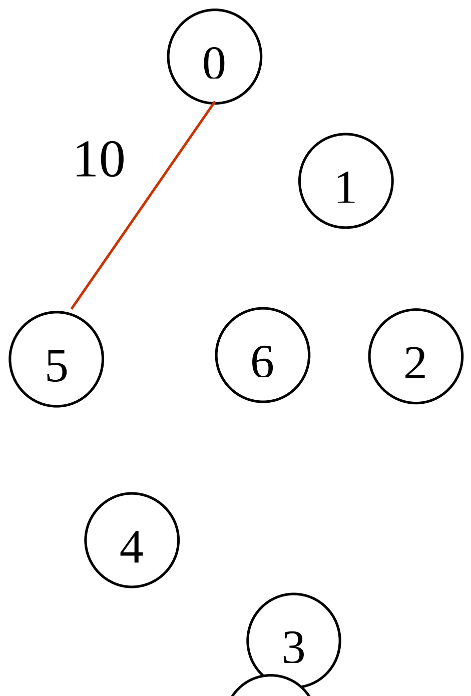
0	10	5
2	12	3
1	14	6
1	16	2
3	18	6
3	22	4
4	24	6
4	25	5
0	28	1



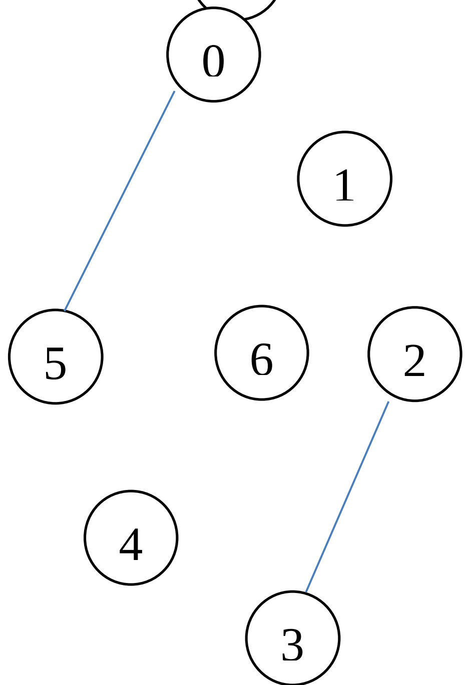
step 1



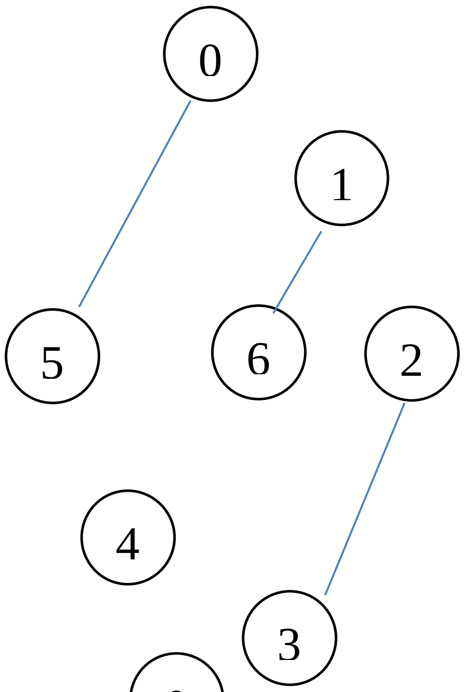
Step 2



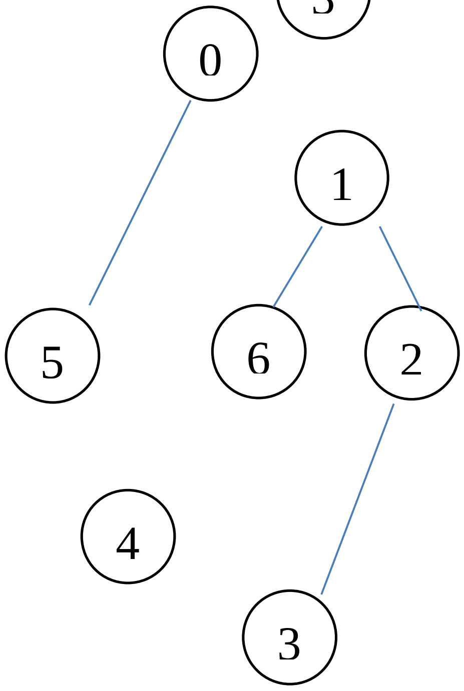
Step 3



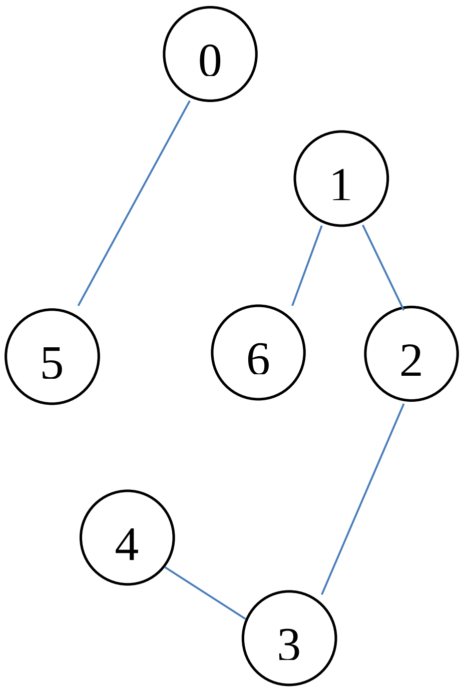
step 4



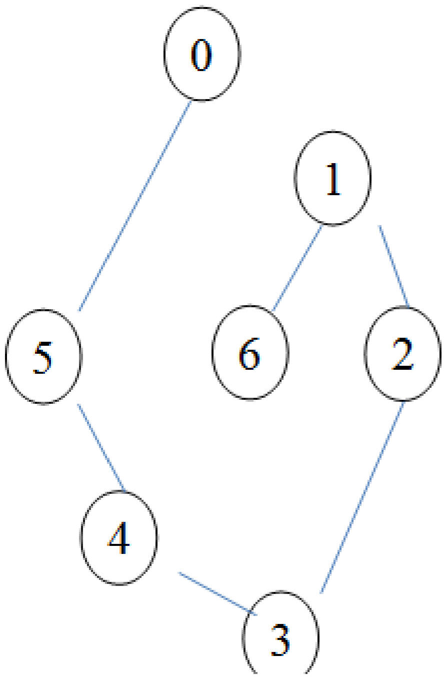
Step 5



Step 6



step 7



[illegible]