Graph Theory

In [94]:

```
from collections import defaultdict
g = defaultdict(list)
edges = []

def addEdge(g,u,v):
    g[u].append(v)
    g[v].append(u)

def createGraph(g):
    for node in g:
        for neighbour in g[node]:
            edges.append((node,neighbour))

def printGraph():
    for j in g:
        print(j," -> ",end = "")
        for i in g[j]:
            print(i,", ",end = '')
        print()

addEdge(g,'a','c')
addEdge(g,'b','d')
addEdge(g,'b','e')
addEdge(g,'a','d')
printGraph()
```

```
a  -> c , d ,
c  -> a ,
b  -> d , e ,
d  -> b , a ,
e  -> b ,
```

In [95]:

```
from collections import defaultdict
g = defaultdict(list)
edges = []

def addEdge(g,u,v,w):
    g[u].append(tuple({v,w}))

def createGraph(g):
    for node in g:
        for neighbour in g[node]:
            edges.append((node,neighbour))
    return g

def printGraph():
    for j in g:
        print(j," -> ",end = "")
        for i in g[j]:
            print(i,", ",end = '')
        print()
addEdge(g,'a','c',4)
addEdge(g,'b','c',3)
addEdge(g,'b','e',2)
addEdge(g,'c','d',7)
addEdge(g,'c','e',1)
addEdge(g,'c','a',11)
printGraph()
#createGraph(g)
```

```
a  -> ('c', 4) ,
b  -> ('c', 3) , ('e', 2) ,
c  -> ('d', 7) , ('e', 1) , ('a', 11) ,
```

Class and Data Structure

In [97]:

```
class Student:
    def __init__(self, name, rollno, cgpa):
        self.name = name
        self.rollno = rollno
        self.cgpa = cgpa
    def readdata(self, Name, Rollno, cgpa ):
        ob = Student(Name, Rollno, cgpa )
        ls.append(ob)
    def printdata(self, ob):
            print("Name   : ", ob.name)
            print("RollNo : ", ob.rollno)
            print("CGPA : ", ob.cgpa)
            print("\n")
ls =[]
stud = Student('', 0, 0)
n = int(input())
for i in range(0,n):
    name = input("enter name")
    roll = int(input("enter roll"))
    cgpa = int(input("enter cgpa"))
    stud.readdata(name, roll, cgpa)
print("\n")
print("\nList of Students\n")
for i in range(ls.__len__()):
    stud.printdata(ls[i])
```

```
List of Students

Name   :  ganesh
RollNo :  20
CGPA :  9


Name   :  raju
RollNo :  22
CGPA :  10
```

In [98]:

```
class Person():
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender
#     def readdata(self, name, age, gender ):
#         ob = Person( name, age, gender )
#         person.append(ob)
    def tell(self):
        print('Name : ', self.name,"\nAge : ",self.age,
'\nGender : ',self.gender)
        print('Empid :',self.empid,"\nDesignation
:",self.desi,"\nSalary :",self.salary)
#     def printdata(self, ob):
#             print(ob.name)
#             print(ob.age)
#             print(ob.gender)
#             print("\n")
class Employee(Person):
    def __init__(self,name,age,gender, empid, desi,
salary):
        Person.__init__(self, name, age, gender)
        self.empid = empid
        self.desi = desi
        self.salary = salary
        Person.tell(self)
#     def readdata(self, name, age, gender ):
#         ob = Employee(name, age, gender , empid, desi,
salary)
#         employ.append(ob)
Employee('Rahul', 20, 'male',345,"intern", 200000)
```

```
Name :  Rahul
```

```
Age :  20
Gender :  male
Empid : 345
Designation : intern
Salary : 200000
```

Out[98]: <__main__.Employee at 0x241f054ea30>

In [99]:

```
class Stack():
    def __init__(self):
        self.stack = []
    def isEmpty(self):
        return self.stack == []
    def PUSH(self,x):
        self.stack.append(x)
    def POP(self):
        return self.stack.pop()
    def printStack(self):
        return self.stack
s = Stack()

while True:
    print('push ')
    print('pop')
    print('quit')
    do = input('What would you like to do? ').split()
    operation = do[0].strip().lower()
    if operation == 'push':
        s.PUSH(int(do[1]))
    elif operation == 'pop':
        if s.isEmpty():
            print('Stack is empty.')
        else:
            print('Popped value: ', s.POP())
    elif operation == 'quit':
        break
    print(s.printStack(),end = "")
    print()
```

```
push
pop
quit
```

```
[10]
push
pop
quit

[10, 20]
push
pop
quit

[10, 20, 30]
push
pop
quit

[10, 20, 30, 40]
push
pop
quit

Popped value:  40
[10, 20, 30]
push
pop
quit

Popped value:  30
[10, 20]
push
pop
quit
```

In [100...

```
class Queue():
    def __init__(self):
        self.queue = []
    def isEmpty(self):
        return self.queue == []
    def INSERT(self,x):
        self.queue.append(x)
    def REMOVE(self):
        return self.queue.pop(0)
    def printQueue(self):
        return self.queue
s = Queue()

while True:
    print('push enter value')
    print('pop')
    print('quit')
    do = input('What would you like to do? ').split()
    operation = do[0].strip().lower()
    if operation == 'push':
        s.INSERT(int(do[1]))
    elif operation == 'pop':
        if s.isEmpty():
            print('Stack is empty.')
        else:
            print('Popped value: ', s.REMOVE())
    elif operation == 'quit':
        break
    print(s.printQueue(),end = "")
    print()
```

```
push enter value
pop
quit
```

```
[10]
push enter value
pop
quit

[10, 30]
push enter value
pop
quit

[10, 30, 40]
push enter value
pop
quit

[10, 30, 40, 100]
push enter value
pop
quit

Popped value:  10
[30, 40, 100]
push enter value
pop
quit

Popped value:  30
[40, 100]
push enter value
pop
quit
```

In [ ]: