In [16]:

```
def Initialise (Gin):
    global c,phi,l,cp,lp,Nsink
    Nsink=1
    for arc in Gin:
        i,j=arc
        Nsink=max(Nsink,i,j)
    # for convenience c[i][j] is capacity of arc [i,j]
    c=[[0 for i in range(Nsink+1)] for j in range(Nsink+1)]
    phi=[[0 for i in range(Nsink+1)] for j in range(Nsink+1)]
    l=[[0 for i in range(Nsink+1)] for j in range(Nsink+1)]
    cp=[[0 for i in range(Nsink+1)] for j in range(Nsink+1)]
    lp=[[0 for i in range(Nsink+1)] for j in range(Nsink+1)]
    print("All values of c,phi and l initialised to zero")
```

In [17]:

```
def Flows (Gin):
    global Nsink,l,phi
    Flowin=[0 for i in range(Nsink+1)]
    Flowout=[0 for i in range(Nsink+1)]
    for arc in Gin:
        i,j=arc
        Flowin[j] = Flowin[j] + phi[i][j]
        Flowout[i] = Flowout[i] + phi[i][j]
    for k in range(2,Nsink):
        if Flowin[k] != Flowout[k]:
            print("*** ERROR *** Flow not conserved at node", k)
    if Flowout[1] != Flowin[Nsink]:
        print("*** ERROR *** Flow not conserved at source or sink")
    Totalcost = 0
    for arc in G:
        i,j=arc
        phi_ij = phi[i][j]
        Totalcost = Totalcost + l[i][j]*phi_ij
        print(arc," has flow ",phi_ij)
    print("Total Cost is ", Totalcost)
```

In [18]:

```
def Links (Gin):
    global Nsink,Out
    Out=[set() for k in range(Nsink)] # labelled 0..Nsink-1
    for arc in Gin:
        i,j = arc
        Out[i - 1] = Out[i - 1] | set([j])
```

In [19]:

```
def SourceSink(Gin):
# finds all paths SinkPaths from source 1 to sink Nsink of network G
    global Nsink,SinkPaths
    Links(Gin)
    Paths = set() # current paths from source stored as set of tuples
    SinkPaths = set() # paths from source to sink Nsink stored as set of tuples
    path = 1 # source node label
    for node in Out[0]:# need out edge from node 1
        pathn = (path,node)
        if node == Nsink:
            SinkPaths = SinkPaths | set([pathn])
        else:
            Paths = Paths | set([pathn])
    Npaths = len(Paths)
    while (0 < Npaths):
        NewPaths = set()
        for oldpath in Paths:
            nold = len(oldpath)
            m = oldpath[-1] # last node in tuple oldpath
            for mout in Out[m-1]:
                if not mout in oldpath:
                    if mout == Nsink:
                        SinkPaths = SinkPaths | set([oldpath+tuple([Nsink])])
                    else:
                        NewPaths = NewPaths | set([oldpath+tuple([mout])])
        Paths = NewPaths
        Npaths = len(Paths)
    print("Paths from source to sink: ",SinkPaths)
```

In [20]:

```
def Newflows(Gin):
# A procedure to modify original flows on Gin along SinkPaths of Gp with minimal cost
    global Gp,phi,c,l,cp,lp,ArcSign,Out
    SourceSink(Gp)
    if SinkPaths == set():
        Links(Gin)
        Flow = 0
        for node in Out[0]:
            Flow = Flow + phi[1][node]
        Cost=0
        for arc in Gin:
            i,j=arc
            Cost=Cost+l[i][j]*phi[i][j]
        print("Maximal flow found:", Flow, " with minimal cost ", Cost)
    else:
        for k in range(len(SinkPaths)):
            cost = 0
            epset = set()
            path=list(SinkPaths)[k]
            for n in range(0, len(path)-1):
                i = path[n]; j = path[n+1];  epset = epset | set([cp[i][j]]); cost = lp
[i][j] + cost
            eps = min(tuple(epset))
            if k == 0: # first path
                mincost = cost; bestpath = path; besteps = eps
            elif cost < mincost:
                mincost = cost; bestpath = path; besteps = eps
        print("A best path in Gp is ", bestpath, " of minimum cost ", mincost)
        print("The min capacity on this path is epsilon ", eps)
        print("The min cost is ", mincost)
        for k in range(0, len(bestpath) - 1):
            i = bestpath[k]; j = bestpath[k+1]
            if ArcSign[i][j] == 1:
                phinewij = phi[i][j] + besteps; phi[i][j]=phinewij
            else:
                phinewji=phi[j][i] = phi[j][i] - besteps; phi[j][i]=phinewji
    #print("Flow=",Flow)
```

In [21]:

```
def Iterate(Gin):
    IncremNet(Gin)
    Newflows(Gin)
    for arc in Gin:
        i,j=arc
        print((i,j)," flow = ", phi[i][j])
```

In [22]:

```
def IncremNet(Gin):
# procedure to create incremental network Gp from given flow network G
    global Gp,Nsink,phi,c,l,cp,lp,ArcSign
# define lists for ArcSign, cp and lp  (indexed by 0..Nsink-1)
    cp=[[0 for i in range(Nsink+1)] for j in range(Nsink+1)]
    lp=[[0 for i in range(Nsink+1)] for j in range(Nsink+1)]
    ArcSign=[[0 for i in range(Nsink+1)] for j in range(Nsink+1)]
    Gp=set([])
    for arc in Gin:
        i,j=arc
        pij = phi[i][j]; pji = phi[j][i]; cij = c[i][j]; lij = l[i][j]
        if (pij < cij and (pji == 0 or not (j,i) in Gin)): # ij arc
            #Gp edges, capacitites and costs added
            cpij = cij - pij; cp[i][j] = cpij; lpij = lij; lp[i][j] =lpij
            ArcSign[i][j] = 1
            Gp=Gp | {(i,j)}
        if pij>0: # ji arc
            cpji = pij; cp[j][i] = cpji; lpji=-lij; lp[j][i] = lpji
            ArcSign[j][i] = -1
            Gp=Gp | {(j,i)}
    print("Incremental Network:",Gp)
```

In [23]:

```
G = {(1, 2), (1, 3), (2, 4), (2, 6), (3, 5), (4, 5), (5, 7), (6, 7), (7, 10)}
```

In [24]:

```
Initialise(G)
```

```
All values of c,phi and l initialised to zero
```

In [25]:

```
Flows(G)
```

```
(1, 2)  has flow  0
(2, 6)  has flow  0
(1, 3)  has flow  0
(6, 7)  has flow  0
(4, 5)  has flow  0
(5, 7)  has flow  0
(7, 10)  has flow  0
(2, 4)  has flow  0
(3, 5)  has flow  0
Total Cost is  0
```

In [26]:

```
#Capacities
c[1][2] = 2000; c[1][3] = 2000; c[2][4] = 1428; c[2][6] =  1428; c[3][5] = 1528; c[4][5
] = 2000
c[5][7] = 2000; c[6][7] = 2400; c[7][10] = 2400
```

In [27]:

```
#Costs
l[1][2] = 0.024; l[1][3] =0.02; l[2][4] =0.015; l[2][6] = 0.02; l[3][5] = 0.05; l[4][5]
= 0.03;
l[5][7] = 0.04; l[6][7] = 0.06; l[7][10] = 0.03;
```

In [28]:

```
Iterate(G)
```

```
Incremental Network: {(1, 2), (2, 6), (6, 7), (1, 3), (4, 5), (5, 7), (7,
10), (2, 4), (3, 5)}
Paths from source to sink:  {(1, 2, 6, 7, 10), (1, 3, 5, 7, 10), (1, 2, 4,
5, 7, 10)}
A best path in Gp is  (1, 2, 6, 7, 10)  of minimum cost  0.134
The min capacity on this path is epsilon  1428
The min cost is  0.134
(1, 2)  flow =  1428
(2, 6)  flow =  1428
(1, 3)  flow =  0
(6, 7)  flow =  1428
(4, 5)  flow =  0
(5, 7)  flow =  0
(7, 10)  flow =  1428
(2, 4)  flow =  0
(3, 5)  flow =  0
```

In [29]:

```
Iterate(G)
```

```
Incremental Network: {(1, 2), (1, 3), (6, 7), (4, 5), (10, 7), (7, 6), (5,
7), (2, 1), (6, 2), (7, 10), (2, 4), (3, 5)}
Paths from source to sink:  {(1, 3, 5, 7, 10), (1, 2, 4, 5, 7, 10)}
A best path in Gp is  (1, 2, 4, 5, 7, 10)  of minimum cost  0.139
The min capacity on this path is epsilon  572
The min cost is  0.139
(1, 2)  flow =  2000
(2, 6)  flow =  1428
(1, 3)  flow =  0
(6, 7)  flow =  1428
(4, 5)  flow =  572
(5, 7)  flow =  572
(7, 10)  flow =  2000
(2, 4)  flow =  572
(3, 5)  flow =  0
```

In [30]:

```
Iterate(G)
```

```
Incremental Network: {(5, 4), (1, 3), (6, 7), (4, 5), (10, 7), (7, 6), (5,
7), (2, 1), (6, 2), (7, 5), (7, 10), (4, 2), (2, 4), (3, 5)}
Paths from source to sink:  {(1, 3, 5, 7, 10)}
A best path in Gp is  (1, 3, 5, 7, 10)  of minimum cost  0.14
The min capacity on this path is epsilon  400
The min cost is  0.14
(1, 2)  flow =  2000
(2, 6)  flow =  1428
(1, 3)  flow =  400
(6, 7)  flow =  1428
(4, 5)  flow =  572
(5, 7)  flow =  972
(7, 10)  flow =  2400
(2, 4)  flow =  572
(3, 5)  flow =  400
```

In [31]:

```
Iterate(G)
```

```
Incremental Network: {(5, 4), (1, 3), (6, 7), (4, 5), (10, 7), (7, 6), (3,
1), (5, 7), (2, 1), (6, 2), (7, 5), (5, 3), (4, 2), (2, 4), (3, 5)}
Paths from source to sink:  set()
Maximal flow found: 2400  with minimal cost  326.85999999999996
(1, 2)  flow =  2000
(2, 6)  flow =  1428
(1, 3)  flow =  400
(6, 7)  flow =  1428
(4, 5)  flow =  572
(5, 7)  flow =  972
(7, 10)  flow =  2400
(2, 4)  flow =  572
(3, 5)  flow =  400
```

In [ ]:

```
Iterate()
```