Problem Statement:

Let G = (V, E) be a connected directed graph with non-negative edge weights,let s and t be vertices of G, and let H be a subgraph of G obtained by deleting some edges. Suppose we want to reinsert exactly one edge from G back into H, so that the shortest path from s to t in the resulting graph is as short as possible.

Idea:

Here we will use BellmanFord algorithm to find the shortest distance from source node to all other nodes. Which will return a array of shortest weights form the source. Also here graph presented of the form [a,b,w], means a-->b with weight w.

Step 1: Using BellmanFord find all source shortest path in H. Return array with D_H. Step 2: Reverse the graph H and apply BellmanFord form t. Return array with D_RH. Step 3: For an edge in (G-H) of the form [a,b,w] add the a'th position of the array in D_H and b'th position of D_RH + w. And take the minimum of those.

Here s=6'th node and t is 0'th node.

BellmanFord

In [1]:
```
import numpy as np
from sys import maxsize
def BellmanFord(graph, V, E, src):
    dis=np.ones(V,dtype=int)*np.infty
    dis[src] = 0
    for i in range(V - 1):
        for j in range(E):
            if dis[graph[j][0]] + \
                graph[j][2] < dis[graph[j][1]]:
                dis[graph[j][1]] = dis[graph[j][0]] + \
                                    graph[j][2]
    for i in range(E):
        x = graph[i][0]
        y = graph[i][1]
        weight = graph[i][2]
        if dis[x] != maxsize and dis[x] + \
                        weight < dis[y]:
            print("Graph contains negative weight cycle")
    return dis
```

Graph G and subgraph H

In [2]:
```
G = [[6, 5, 11], [6, 4, 9], [5, 4, 8],
             [5, 3, 6], [4, 3, 15], [4, 1, 7],
             [4, 2, 5], [3, 0, 5],[3,1,9],[2,6,2],[2,1,8],[1,0,7]]
H= [[6, 5, 11], [6, 4, 9],
             [5, 3, 6],  [4, 1, 7],
             [4, 2, 5], [3, 0, 5],[2,6,2],[2,1,8],[1,0,7]]
```

Step 1

In [3]:
```
# All shortest path in H form source node 6
D_H=BellmanFord(H,7,9,6)
```

Step 2

Reverse graph RH

In [4]:
```
l=len(H)
RH=np.zeros((l,3),dtype=int)
for i in range (0,l):
    RH[i][1]=H[i][0]
    RH[i][0]=H[i][1]
    RH[i][2]=H[i][2]
```

In [5]:
```
# All shortest path in RH form source node 0
D_RH=BellmanFord(RH,7,9,6)
```

Step 3

In [6]:
```
M=np.infty
for i in range(0,len(G)):
    if G[i] not in H:
        T=D_H[G[i][0]]+G[i][2]+D_H[G[i][1]]
        if T<M:
            M=T
            edge=G[i]
```

In [8]:
```
print("The edge can be added from (G-H) such that the path from s to t is minimum with this edge is "
      +str(edge)+" then weight will be "+str(M))
```

```
The edge can be added from (G-H) such that the path from s to t is minimum with this edge is [5, 4, 8] then wei
ght will be 28.0
```