Task 3

For efficiency's sake

Because G2 is ordered by the source of arcs, we no longer need to go over all arcs in G to find out only those sourcing from a particular node tn: these arcs are just listed in G2[tn]. It saves unnecessary enumeration.

Coupling invariant

"The set of arcs in G is the same as in G2":

- For all arcs (nFrom, dist, nTo) in G, a pair (dist, nTo) exists in G2[nFrom].
- For all nodes nFrom, for all pairs (dist, nTo) in G2[nFrom], the arc (nFrom, dist, nTo) exists in G.

Code

```
import sys; INFTY= 1000

print "How many nodes? ",
N= int(sys.stdin.readline()) # N= number of nodes.
print "Nodes are 0..%d, with initial node %d." % (N-1, 0)

# Three white-space separated integers representing (from,dist,to) per
line.
print "Now enter the edges:"
#>> G= [map(int,line.split()) for line in sys.stdin.readlines()]
G2 = [[] for i in range(N)]
for line in sys.stdin.readlines():
    nFrom, dist, nTo = map(int, line.split())
    G2[nFrom].append((dist, nTo))  # put every arc in the respective list
#<<
un1 = [1]*N; nun1 = N   # un1[i] == 1 for all nodes: all nodes are
unmarked
m= [INFTY]*N; m[0]= 0   # m:=  initially INFTY except for initial node
tn= 0                   # tn:= initial node

# Initialisation has established the invariants:
#   I1--- For all marked nodes, m has the least distance from 0. // A.
#   I2--- For all unmarked nodes, m has least all-marked distance from 0.
// B.
#   I3--- Node tn is unmarked, and is m-least among the unmarked nodes. //
C.

while 1:
    un1[tn] = 0; nun1 -= 1

    if nun1 == 0: break

    # Re-establish I2. // G.
#>> for (nFrom,dist,nTo) in G:
```

```
#>>     if nFrom==tn and un1[nTo]:
    for (dist, nTo) in G2[tn]:
        if un1[nTo]:
#<<
            newD= m[tn]+dist
            if newD<m[nTo]: m[nTo]= newD

    # Re-establish I3. // H.
    minD= INFTY
    for n in range(N):
      if un1[n]:
        if m[n]<=minD: tn= n; minD= m[n]

    if minD==INFTY: break # All remaining nodes unreachable. // I.
###
#   I1,I2 and m is INFTY for all nodes in un1. // J,K,L,M.

print "Least distances from Node 0 are:"
for n in range(N):
    if m[n]!=INFTY: print "Distance to Node %d is %d." % (n,m[n])
    else:           print "Node %d is unreachable." % n
```