Task 2

Coupling invariant

For every node i, `un1[i] == 1` iff `i in un`, `un1[i] == 0` iff `i not in un`, and the number of ones `nun1 == len(un)`.

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()]

#>> un= set(range(N))       # un:= all nodes are unmarked
un1 = [1]*N; nun1 = N       # un1[i] == 1 for all nodes: all N 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:
#>> un.remove(tn) # Maintains I1, but breaks I2,I3. // D,E,F.
    un1[tn] = 0; nun1 -= 1  # It's ensured by I3 that un1[tn] == 1
beforehand, so no need to test.
#<<

#>> if len(un)==0: break # No unmarked nodes left.
    if nun1 == 0: break
#<<

    # Re-establish I2. // G.
    for (nFrom,dist,nTo) in G:
#>>     if nFrom==tn and nTo in un:
        if nFrom==tn and un1[nTo]:
#<<
            newD= m[tn]+dist
```

```
            if newD<m[nTo]: m[nTo]= newD

    # Re-establish I3. // H.
    minD= INFTY
#>> for n in un:
    for n in range(N):
      if un1[n]:  # go over un1 on unmarked nodes only
#<<
        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
```