

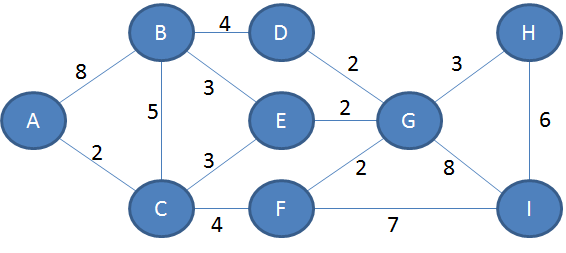
Faculty of Science

**Course**: CSCI 3070U: Design and Analysis of Algorithms

**Tutorial:** #11

**Topic:** All pairs shortest path

For this tutorial, you will implement the Floyd-Warshall algorithm in C, C++, Java, or Python. Test your algorithm using the graph below, with its indicated edge weights. How you implement the graph is up to you, but an adjacency matrix is recommended. Include print statements in your program that display the Dk matrix after each step (i.e. for k=0...|V|).



Solution

def getAdjacency(g):

vertices = g.keys()

d = {}

for i in vertices:

d[i] = {}

for j in vertices:

try:

d[i][j] = g[i][j]

except KeyError:

if i == j:

# distance between node and itself is 0

d[i][j] = 0

else:

# distance to other nodes is infinite

d[i][j] = float('inf')

return d

def floydWarshall(g):

vertices = g.keys()

d = getAdjacency(g)

printD(d,0)

for k in vertices:

for i in vertices:

for j in vertices:

d[i][j] = min(d[i][j],d[i][k]+d[k][j])

printD(d,k)

return d

def printD(d,k):

print "k = ",

print k

for row in d:

print "[",

for col in d[row]:

print repr(d[row][col]).rjust(4),

print "]"

g = {

1: {2: 8, 3: 2},

2: {1: 8, 3: 5, 4: 4, 5: 3},

3: {1: 2, 2: 5, 5: 3, 6: 4},

4: {2: 4, 7: 2},

5: {2: 3, 3: 3, 7: 2},

6: {3: 4, 7: 2, 9: 7},

7: {4: 2, 5: 2, 6: 2, 8: 3, 9: 8},

8: {7: 3},

9: {6: 7, 7: 8},

}

dn = floydWarshall(g)