Owen McEvoy
CIS 315
Spring 2019

Assignment 4

1.)

$$W^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}\begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & \infty & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & 3 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

int k=0, k<6, k++
{ int i=0, i<6, i++
{ int k=0, k<6, i++
{ if dist[i][j] > (i][k]+[k][j]
[i][j] =
}}} ....

$$W_1 = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{bmatrix}$$

$D^0[2,3]$ $D^0[2,1]+D^0[1,3]$
$\infty >$ $1 + \infty$ ✗
$D^0[2,4]$ $D^0[2,1]$ $D^0[1,4]$
$2 >$ $0 + \infty$ ✗
$D[2,5]$ $D[2,1] + D[1,5]$
$\infty >$ $1 - 1 = 0$ ✓
$D[2,6]$
$\infty >$ $1+\infty$ ✗

$\infty > -4 + \infty$ ✗
$\infty > -4 + \infty$ ✗
$3 > -4 - 1 = -5$ ✓

$$W_2 = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{bmatrix}$$

$D[3,1] > D[3,2] + D[2,1]$
$\infty > 2 + 1$ ✓ $D[3,1] = 3$
$D[3,4] > D[3,2] + D[2,4]$
$\infty > 2 + 2$ ✓ $D[3,4] = 4$
$D[3,5] > D[3,2] + D[2,5]$
$\infty > 2 + 0$ ✓ $D[3,5] = 2$

$$W = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{bmatrix}$$

$D[5,1] > D[5,2] + D[2,1]$
$\infty > 7 + 1 = 8$ ✓ $D[5,1] = 8$
$D[5,3] > D[5,4] + D[4,3]$
$\infty > \infty$ ✗
$D[5,4] > D[5,2] + D[2,4]$
$\infty > 7 + 2$ ✓ $D[5,4] = 9$
$D[5,6] > D[5,2] + D[2,6]$
∞ $0 + \infty$ ✗

$$W = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{bmatrix}$$

$D[6,1] > D[6,2] + D[2,1]$
$\infty > 5 + 1$ ✓ $D[6,1] = 6$
$D[6,4] > D[6,2] + D[2,4]$
$\infty > 5 + 2 = 7$ ✓ $D[6,4] = 7$
$D[6,5] > D[6,2] + D[2,5]$
$\infty > 5 + 0 = 5$ ✓ $D[6,5] = 5$

★ not doing step by step to save space & sanity

$$W = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{bmatrix}$$

$D[2,1] \geq D[2,4] + D(4,1]$

$2 > -4+2$ ✓ $D[2,1] = -2$

$0 > 2+\infty$ ×

$\infty > 2+\infty$ ×

$2 > 2+0$ ×

$0 > 2-5$ ✓ $D[2,5] = -3$

$\infty > 2+\infty$ ×

$D[3,1] > D[3,4] + D[4,1]$

$3 > 4-4 = 0$ $D[3,1] = 0$

$2 > 4+\infty$ × $4 > 4+0$ ×

$0 > 4+\infty$ × $2 > 4+5$ $D[3,5] = -1$

$D(5,1] > D(5,4) + D(4,1]$

$8 > 9-4$ ✓ $D[5,1] = 5$

$D[6,1] > D[6,4] + D[4,1]$

$6 > 7-4 = 3 = D[6,1]$ ✓

$5 > 7-5 = 2 = D[6,5]$ ✓

$D[1,2] > D[1,5] + D[5,2]$

$\infty > -1+7 = 6 = D[1,2]$ ✓

$D[1,4] > D[1,5] + D[5,4]$

$\infty > -1+9 = 8 = D[1,4]$ ✓

$D[4,2] > D[4,5] + D[5,2]$

$\infty > -5+7 = 2 = D[4,2]$ ✓

$\vdots$

$D[3,1] > D[3,6] + D[6,1]$

$0 > -8+3 = -5 = D[3,1]$ ✓

$2 > -8+5 = -3 = D[3,2]$ ✓

$0 > -8+10$ ×

$4 > -8+7 = -1$ ✓ $-1 = D[3,4]$

$-1 > -8+2 = -6 = D[3,5]$ ✓

$$W = \begin{bmatrix} 0 & -6 & \infty & 8 & -1 & \infty \\ -2 & 2 & \infty & 2 & -3 & \infty \\ -5 & -3 & 0 & -1 & -6 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix}$$

final matrix is

$$\begin{bmatrix} 0 & -6 & \infty & 8 & -1 & \infty \\ -2 & 2 & \infty & 2 & -3 & \infty \\ -5 & -3 & 0 & -1 & -6 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix}$$

Owen McEvoy
CIS 315
Spring 2019

Assignment 4

2.) given a graph $G=(V,E)$ w/ weights W. V is number of cities & E is all paths between the cities. Determine the capacity needed to travel between any 2 cities w/ refueling only once.

1- Using the Floyd-Warshall algo., find shortest distance between each city
2- loop through each pair in G & keep track of max weight globally & locally
3- return max capacity

So...

```
One_stop(G):
        Floyd_Warshall(G)
        one_stop = 0
        local_max = 0
        for each (U,V) in G:      // let U,V, be any 2 cities
            for each path from U to V:
                if (max(W[U,P], W[P,V]) > local_max):
                    local_max = max(W[U,P], W[P,V])
            if (local_max > one_stop):
                one_stop = local_max
        return one_stop
```

3.) must place a sign at m_1 & m_n. Signs are placed every 100 miles. The penalty for placing a sign x miles apart is $(100-x)^2$. We want to place them to minimize the penalty total.

The subproblem for this problem is calculating the best possible location for each sign while minimizing the cost. To do so, you must look at all the sign locations & determine which one will result in the minimum

penalty & then choosing that one. This process is done until m_n is reached.
the recurrence for placing sign i where $i<j<n$
$(100-(j-i))^2 + \min(\text{penalty}(j))$
in this case, j is the sign that was placed in the initial step

4) given M, fixed cost of moving between 2 cities
city $C=\{c_1,\dots,c_n\}$
city $D=\{d_1,\dots,d_n\}$
let i = month of operation
c_i = operating in C in month d_i = operating in D in mont

The subproblem is determining which city has the cheaper costs of operation in month i. The other thing to consider, is lets say $D_i < C_i$. The owner should only move if $D_i < C_i - M$ to account for the cost of moving.

A method to solve this problem would be maintaining a minimum array. For each month, i, put the least expensive city in this array.

```
min_array = []
Cheap_City(M, C, D):
        if len(min-array) = len(C) + len(D):
                return min_array
        else:
            Cheap = min(Ci, Di)
            if Cheap + M > (Di/Ci): # whichever one wasnt cheap
                Cheap = Di/Ci #
        min_array.append(cheap)
        Cheap_City(M, C[1:], D[1:])
```