FALL SEM – (2020 – 21)

MAT2003

SUBMITTED BY : SRIHARSHITHA DEEPALA

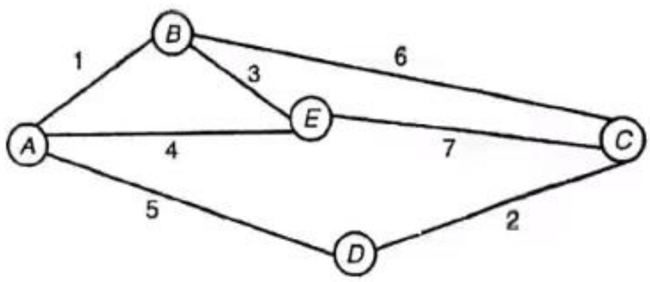
REG NO : 19BCD7246

LAB NO : 11

SLOT : L6

QUESTION – 1:

2412. *Use Dijkstra's algorithm to determine a Shortest Path from A to C for the following network*



CODE:

```
function [sp, spcost] = lab11(cost_matrix, source,
destination)
    % Displaying the given graph
    W = input('Enter the weights of the edges :');
    dg = sparse(input('Enter the vector-1'), input('Enter
the vector-2'),W)
    ug = tril(dg + dg')
names = input('Enter the names of nodes')
```

```
h=
view(biograph(ug,names,'ShowArrows','off','ShowWeights','o
n'))
    % Finding the shortest path and shortest distance
using dijkstra Algorithm
    n = input('Enter the number of nodes :');
    Alphabets  = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    source = input('Enter the source node index :');
    letter1 = Alphabets(source);
    fprintf('The equivalent alphabet to source node index
is %c\n',letter1);
    destination = input('Enter the destination node index
:');
    letter2 = Alphabets(destination);
    fprintf('The equivalent alphabet to destination node
index is %c\n',letter2)
    cost_matrix = input('Enter the cost matrix :');
    n = size(cost_matrix,1);
    S(1:n) = 0;
    dist(1:n) = inf;
    prev(1:n) = n+1;
    dist(source) = 0;

    while sum(S)~=n
        c = [];
        for i = 1:n
            if S(i) == 0
                c = [c dist(i)];
            else
                c = [c inf];
            end
        end
        [u_index u ] = min(c);
        S(u) = 1;
        for i = 1:n
            if(dist(u)+cost_matrix(u, i)) < dist(i)
                dist(i) = dist(u) + cost_matrix(u, i);
                prev(i) = u;
            end
        end
    end
    sp = [destination];
```

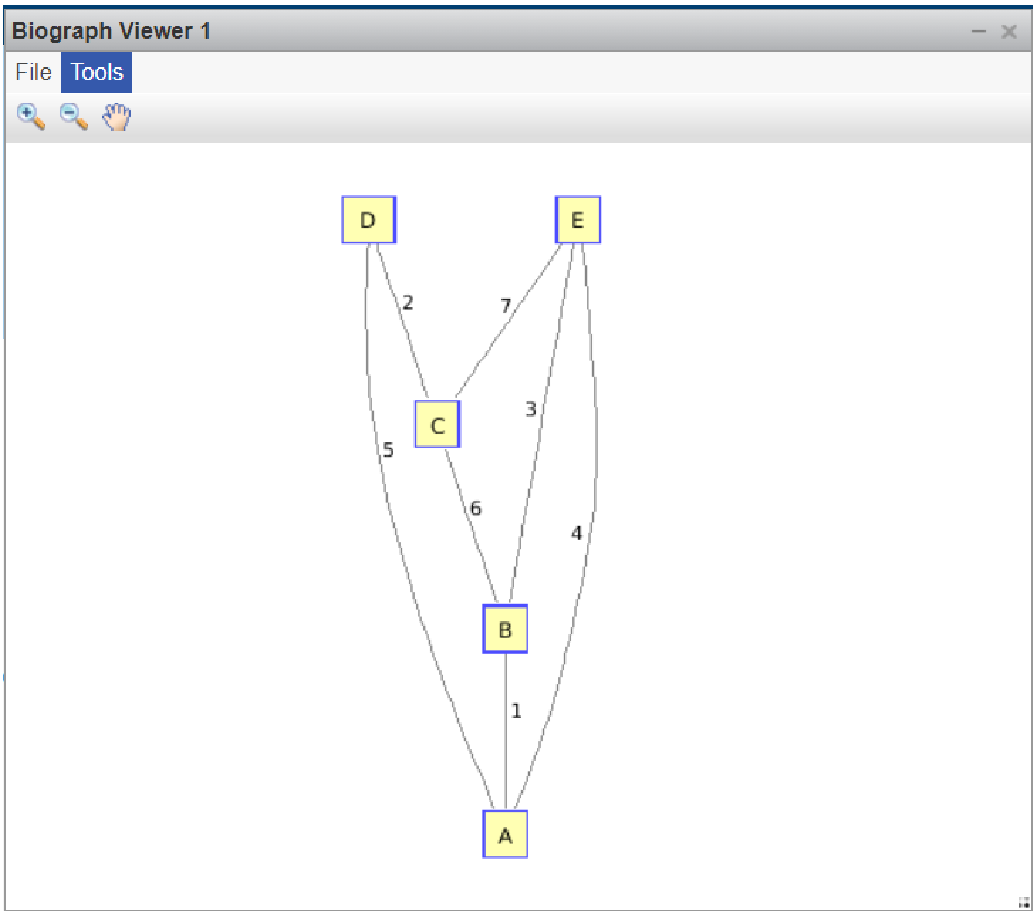
```
    while sp(1)~=source
        if prev(sp(1))<= n
            sp = [prev(sp(1)) sp];
        else
           error;
        end
    end
    spcost = dist(destination);
fprintf('The shortest path is %d -> %d -> %d\n', sp);
fprintf('The shortest length is %d\n', spcost);


end
```

OUTPUT:

```
lab11
Enter the weights of the edges :
[1 3 6 4 7 5 2]
Enter the vector-1
[1 2 2 1 5 1 4]
Enter the vector-2
[2 5 3 5 3 4 3]

dg =

   (1,2)        1
   (2,3)        6
   (4,3)        2
   (5,3)        7
   (1,4)        5
   (1,5)        4
   (2,5)        3


ug =

   (2,1)        1
   (4,1)        5
   (5,1)        4
```

```
   (3,2)        6
   (5,2)        3
   (4,3)        2
   (5,3)        7

Enter the names of nodes
{'A', 'B', 'C', 'D', 'E'}


names =


  1×5 cell array


    {'A'}    {'B'}    {'C'}    {'D'}    {'E'}


Biograph object with 5 nodes and 7 edges.
```

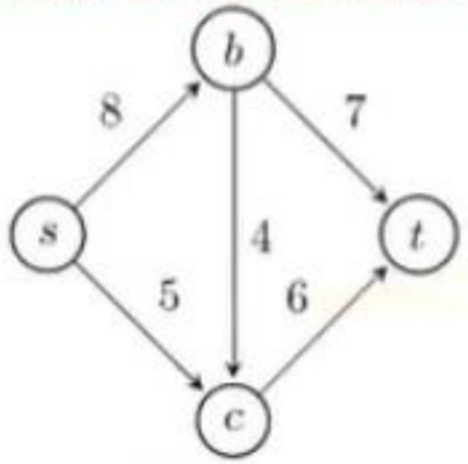
```
Biograph object with 5 nodes and 7 edges.
```

```
Enter the number of nodes :
5
Enter the source node index :
1
The equivalent alphabet to source node index is A
Enter the destination node index :
3
The equivalent alphabet to destination node index is C
Enter the cost matrix :
[0 1 Inf 5 4; 1 0 6 Inf 3; Inf 6 0 2 7; 5 0 2 0 Inf; 4 3 7
Inf 0]
The shortest path is 1 -> 2 -> 3
The shortest length is 7

ans =

     1     2     3
```

QUESTION – 2:

Find the maximum flow for given a network



CODE:

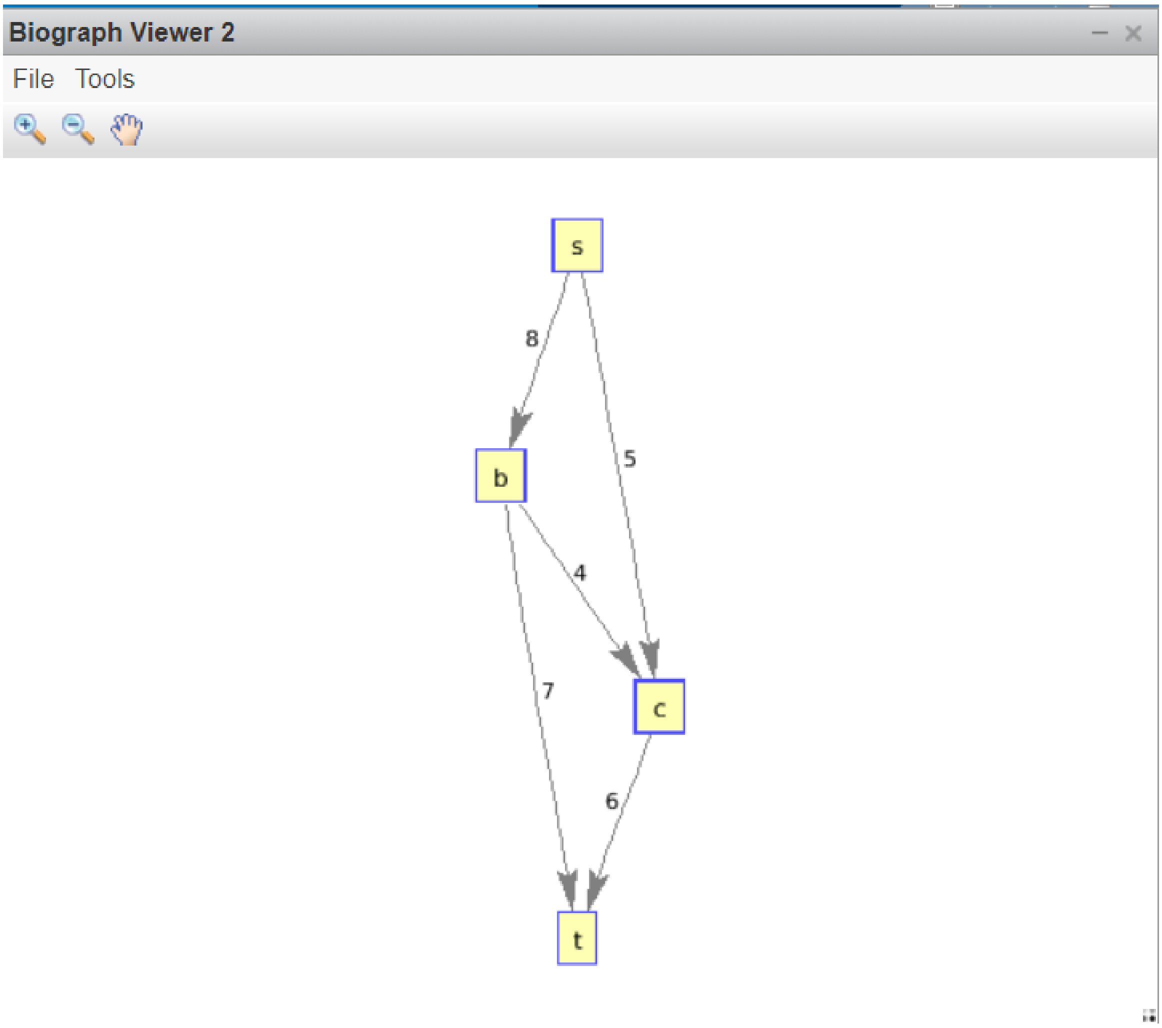
```
function lab11_1
clc;
clear;
% Displaying the graph
W = input('Enter the weights of the edges :');
dg = sparse(input('Enter the vector-1'), input('Enter the
vector-2'),W)
names = input('Enter the names of the nodes :')
h =
view(biograph(dg,names,'ShowArrows','on','ShowWeights','on
'))
% Solving max flow problem using ford fulkerson algori
s = input('Enter the value of source :');
t = input('Enter the value of destination :');
f = 0;
cap = input('Input the matrix :');
len = length(cap);
while true
p = findPath(cap);
if p(1) == 0, break; end
flow = max(max(cap));
for j = 2:length(p)
flow = min(flow,cap(p(j),p(j-1)));
end
for j = 2:length(p)
a = p(j); b = p(j-1);
cap(a,b) = cap(a,b) - flow
cap(b,a) = cap(b,a) + flow
end
f = f + flow
end
disp(['Max flow is ' num2str(f)]);
disp('Residual graph:');
disp(cap);
% Find an Augmenting Path
function F = findPath(A) % BFS (Breadth-first Search)
q = zeros(1,len); % queue
        pred = zeros(1,len);            % predecessor
array
        front = 1; back = 2;
```

```
        pred(s) = s; q(front) = s;
        while front ~= back
            v = q(front);
            front = front + 1;
            for i = 1:len
                if pred(i) == 0 && A(v,i) > 0
                    q(back) = i;
                    back = back + 1;
                    pred(i) = v;
                end
            end
        end
        path = zeros(1,len);
        if pred(t) ~= 0
            i = t; c = 1;
            while pred(i) ~= i
                path(c) = i;
                c = c + 1;
                i = pred(i);
            end
            path(c) = s;
            path(c+1:len) = [];
        end
        F = path;
    end
end
```

OUTPUT:

```
Enter the weights of the edges :
[8 5 7 6 4]
Enter the vector-1
[1 1 2 4 2]
Enter the vector-2
[2 4 3 3 4]

dg =

   (1,2)        8
   (2,3)        7
   (4,3)        6
   (1,4)        5
```

```
   (2,4)        4

Enter the names of the nodes :
{'s', 'b', 't', 'c'}


names =


  1×4 cell array

    {'s'}    {'b'}    {'t'}    {'c'}


Biograph object with 4 nodes and 5 edges.
```



```
Enter the value of source :
1
Enter the value of destination :
4
Input the matrix :
[0 8 5 0; 0 0 4 7; 0 0 0 6; 0 0 0 0]

f =

     7


f =

    12


f =

    13

Max flow is 13
Residual graph:
     0     0     0     0
     8     0     3     0
     5     1     0     0
     0     7     6     0
```