William Wijaya
3035992465

Lab 3: The Travelling Salesman Problem

Problem 3 only

Problem 3

There are 7 functions that are needed to do problem 3. Functions are listed below:

visGraph.m

```
function s=visGraph(edges) %Fill the input with graph.edges
e_12=edges(:,1:2);
e3=edges(:,3);
C=unique(edges(:,1:2));
Z=zeros(1,length(C));
Com=[];
for i=1:length(C)
    Com(i,:)=Z;
    e1=edges(:,1);
    e1_1=e_12(find(e1==i),:);
    e2=e1_1(:,2);
    Com(i,e2)=e3(find(e1==i),:);
end
bg=biograph(Com,[],'ShowWeights','on');
view(bg)
```

formLP.m

```
function prob=formLP(graph) %Fill the input graph=graph.edges
g=graph;
f=graph(:,3); %f Matrix
Aeq=[];
g1=graph(:,1);
g2=graph(:,2);
J=unique(g(:,1:2));
Z=zeros(1,length(g1));

% Building Aeq matrix
for i=1:(length(J)*2)
    if i<=length(J)
        Aeq(i,:)=Z;
        g1_1=find(g1==i);
        Aeq(i,g1_1)=1;
    else
        Aeq(i,:)=Z;
        g2_1=find(g2==i-length(J));
        Aeq(i,g2_1)=1;
```

```
end
end

%Building beq matrix
beq=ones(size(Aeq(:,1)));

%Building lb and ub
lb=zeros(size(Aeq(1,:)))';
ub=ones(size(Aeq(1,:)))';

%Building Aineq and Bineq matrix

Aineq=[];
bineq=[];
if Aineq~=[]
bineq=zeros(size(A(1,:)))';
end

solver='linprog';
options=optimoptions('linprog','Algorithm','dual-simplex');
prob=struct('edges',g,'f',f,'Aineq',Aineq,'bineq',bineq,'Aeq',(Aeq),'beq',(b
eq),'lb',(lb),'ub',(ub),...
    'options',options,'solver',solver)
end
```

branch.m

```
function prob=branch(prob,graph) %fill prob=prob and graph=graph.edges
prob=prob;
graph=graph;

%Keep track what nodes are passed starting from node 1

    edges=prob.sol_edges;
    prob=[];
    Z=zeros(length(edges(:,1)),1);
    edges=[edges,Z];
    leftS=edges(:,1);
    allS=edges(find(leftS==1),:); % Start subtour progress from node 1
    rightS=allS(1,2);
    findS=find(edges(:,2)==rightS)
    edges(findS,3)=1 % Assign the zero equal to one if the edges is passed


% The while loop to keep track of the edges that are passed
% until this reach node 1 again
    while rightS~=1
        leftS=edges(:,1);
        allS=edges(find(leftS==rightS),:);
        rightS=allS(1,2);
```

```
        findS=find(edges(:,2)==rightS);
        edges(findS,3)=1;
    end

 % Now we can use the information from the third column matrix

 %Finding edges_solution with third column of 1
  edges11=edges(find(edges(:,3)==1),:);
  edges12=edges11(:,1:2); % Delete the third column
  g1=graph(:,1:2); % Matrix graph.edges column 3 deleted
  member=ismember(g1,edges12,'rows'); % Comparing g1 with edges12
  g2=[graph,member]; %Adding the fourth column with member

fmember=find(member==1); %Finding what edge need to be branched

for i=1:length(fmember)
    myField=strcat('p',num2str(i));
    prob.(myField)=formLP(graph);
    Aineq=zeros(1,length(graph(:,1)));
    Aineq(fmember(i))=1;
    prob.(myField).Aineq=Aineq;
    prob.(myField).bineq=zeros(1,1);
    prob.(myField)=solveLP(prob.(myField));
end
end
```

solveLP.m

```
function prob=solveLP(prob) %Fill prob=prob
[sol,fval,exitflag,output,lambda]=linprog(prob);
g=prob.edges;
g1=g(:,1);
g1_2=g(:,1:2);
g3=g(:,3);
sol_edges=g1_2(find(sol==1),:);
cost=fval;
cost_edges=g3(find(sol==1));
isFeasible=exitflag==1;
hasSubtours=[];

if isFeasible==1
hasSubtours=subtourDetect(sol_edges);
end

prob=struct('f',prob.f,'Aeq',prob.Aeq,'beq',prob.beq,...
    'lb',prob.lb,'ub',prob.ub,'options',prob.options,...
    'solver',prob.solver,'sol',sol,'sol_edges',sol_edges,...
    'cost',cost,'cost_edges',cost_edges,...
    'isFeasible',isFeasible,'hasSubtours',hasSubtours);

end
```

subtourDetect.m

```
function hasSubtours=subtourDetect(sol_edges)
edges=sol_edges;
Z=zeros(length(edges(:,1)),1);
edges=[edges,Z];
leftS=edges(:,1);
allS=edges(find(leftS==1),:); % Start subtour progress from node 1
rightS=allS(1,2);
findS=find(edges(:,2)==rightS);
edges(findS,3)=1; % Assign the zero equal to one if the edges is passed

% The while loop to keep track of the edges that are passed
% until this reach node 1 again
while rightS~=1
    leftS=edges(:,1);
    allS=edges(find(leftS==rightS),:);
    rightS=allS(1,2);
    findS=find(edges(:,2)==rightS);
    edges(findS,3)=1;
end

% All of the above program will generate 3 column matrix.
% The first two column matrix will be the sol_edges
% The third column matrix is a matrix of 0 or 1

% If all third column matrix is equal to one,
% then all edges starting from one are passed, meaning there is no
% subtours.
if all(edges(:,3))==1
    hasSubtours=0;
else
    hasSubtours=1;
end
```

tourFun.m

```
function tour=tourFun(sol_edges) %fill sol_edges=prob.sol_edges
trackTour=[];
edges=sol_edges;
leftS=edges(:,1);
allS=edges(find(leftS==1),:); % Start subtour progress from node 1
tracktour=[1];
rightS=allS(1,2);
tracktour(1,2)=rightS;
```

```
i=3;
while rightS~=1
    leftS=edges(:,1);
    allS=edges(find(leftS==rightS),:);
    rightS=allS(1,2);
    tracktour(1,i)=rightS;
    i=i+1;
end

tour=tracktour;
```

Finally, the function that is asked for the assignment problem 3:

solveTSP.m

```
function [tour,cost]=solveTSP(graph) %fill graph=graph.edges
prob=formLP(graph);
prob=solveLP(prob);
bestSolM=[inf,0];

    if prob.hasSubtours==1
  %% Begin branching

 prob=branch(prob,graph);
 bestSolution=inf;
 i=1;
 while isempty(struct2cell(prob))==0
     myField=strcat('p',num2str(i));
     if isempty(struct2cell(prob))==1
         cost=bestSolution;
     else
         extractProb=prob.(myField);
         prob=rmfield(prob,myField);
         if extractProb.isFeasible==0
             extractProb=[];
         elseif extractProb.hasSubtours==0
             if bestSolution>extractProb.cost
                 bestSolution=extractProb.cost;
                 bestProb=extractProb;
             end
             extractProb=[];
         elseif bestSolM(end)==bestSolM(end-1)
                 break
         elseif extractProb.hasSubtours==1
            extractProb=branch(extractProb,graph);
            pname=fieldnames(prob);
            pend=pname(end,1);
            pendString=string(pend);
            pNumber=str2double(regexp(pendString,'[\d.]+','match'));
            j=pNumber+1;
             while isempty(struct2cell(extractProb))==0
```

```
                myField1=strcat('p',num2str(j));
                myField2=strcat('p',num2str(j-pNumber));
                prob.(myField1)=extractProb.(myField2);
                extractProb=rmfield(extractProb,myField2);
                j=j+1;
            end
        end
    end
    bestSolM(1,i)=bestSolution;
    i=i+1;
 end
    elseif prob.hasSubtours==0 % No Subtour problem
        bestSolution=prob.cost;
        bestProb=prob;
    else
        bestSolution=[];
        bestProb=[];
    end

 %Cost
 if bestSolution~=inf
 cost=bestSolution;
 else
     cost=[];
 end

 %Tour
if bestSolution~=inf
 tour=tourFun(bestProb.sol_edges);
else
    tour=[];
end
 fprintf("\nThe final answer is:\n")
 fprintf("\ncost= %4.2f \n",cost)
 display(tour)
 if isempty(cost)==1
 fprintf("\nEmpty answer means the problem is infeasible\n")
 elseif isempty(cost)==0
     visGraph([bestProb.sol_edges,bestProb.cost_edges])
 end
end
```

After defining all the functions needed for problem 3, we can start problem 3 by using:

William_Wijaya_lab3problem3.m

```
clear all;clc;close all
%This is for problem 3 only!
for i=1:6
    if i==1
        load('graph1.mat')
    elseif i==2
        load('graph2.mat')
    elseif i==3
```

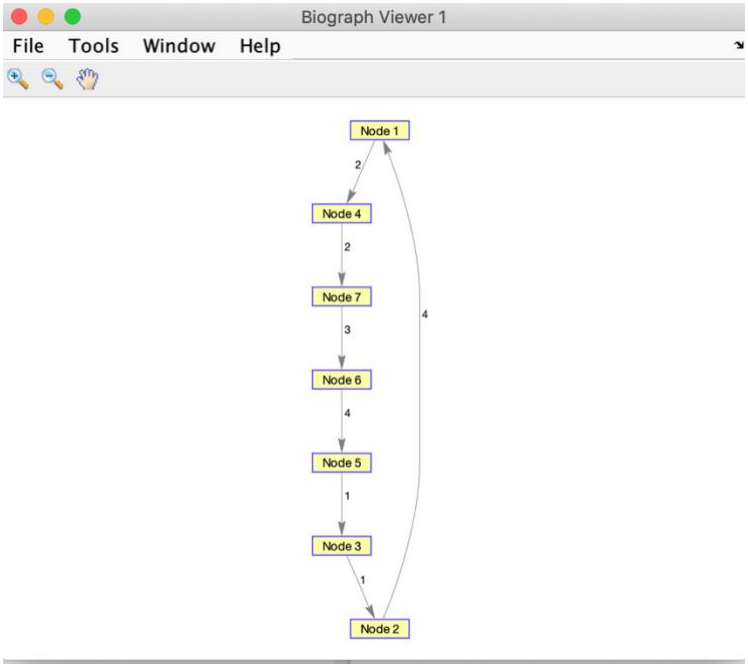
```
        load('graph3.mat')
    elseif i==4
        load('graph4.mat')
    elseif i==5
        load('graph5.mat')
    else
        load('graph6.mat')
    end

[tour,cost]=solveTSP(graph.edges);
fprintf('\nPress enter to continue next problem:\n');pause;
clc;clear all;close all
end
fprintf('lab 3 problem 3 Done!\n')
```

Then, the following output is generated:

Note that only some part of the output is shown that is important for the assignment.

graph1.mat:

```
The final answer is:

cost= 17.00

tour =

  Columns 1 through 6

     1     4     7     6     5     3

  Columns 7 through 8

     2     1
```



graph2.mat:

```
The final answer is:

cost=

tour =

    []


Empty answer means the problem is infeasible
```

graph3.mat:

```
The final answer is:

cost=

tour =

    []


Empty answer means the problem is infeasible
```

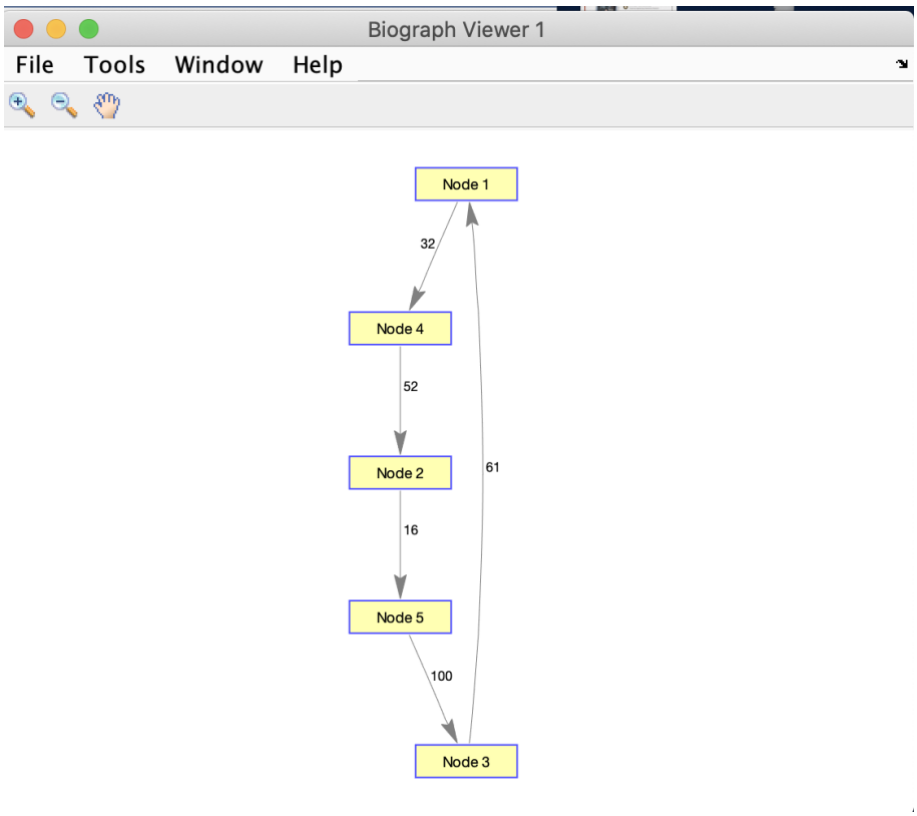
graph4.mat:

The final answer is:

cost= 261.00

tour =

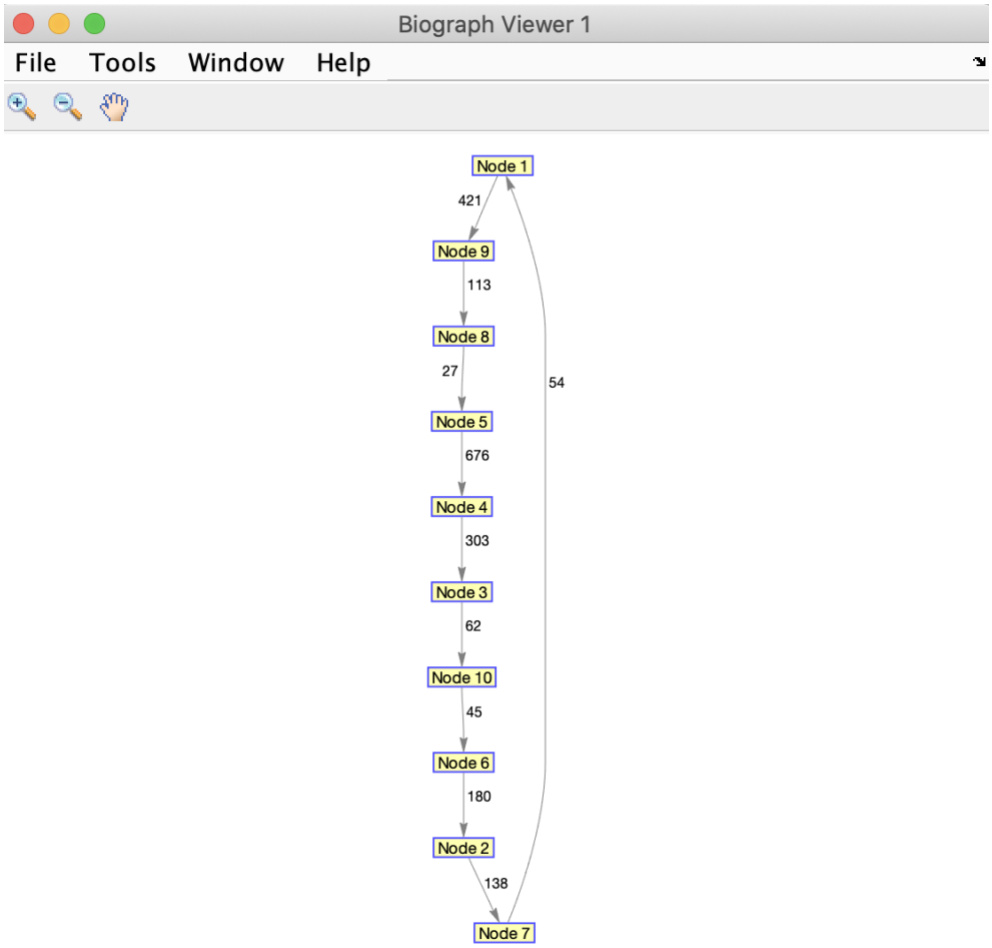
1 4 2 5 3 1



graph5.mat:

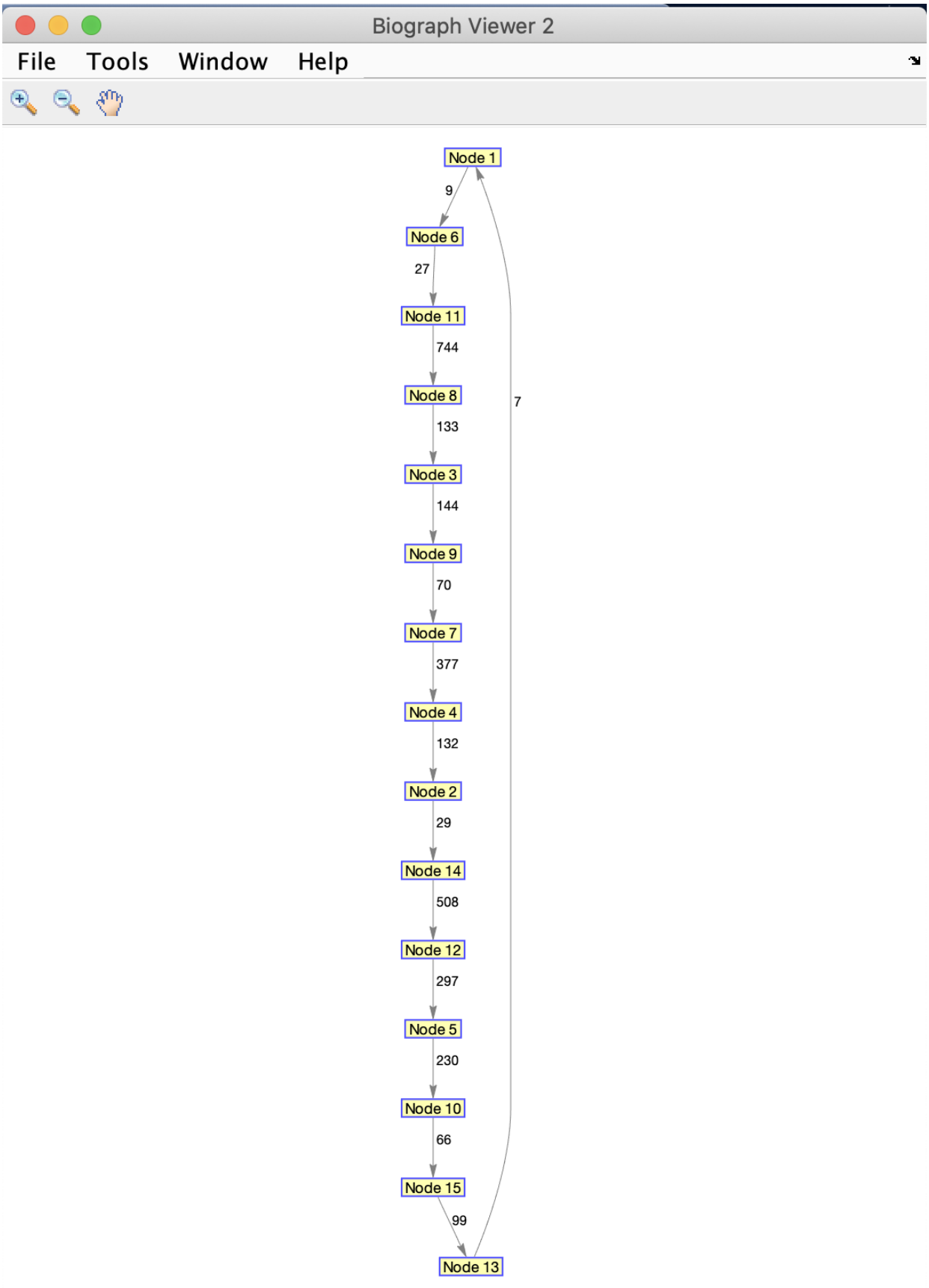
The final answer is:

```
cost= 2019.00

tour =

  Columns 1 through 6

     1     9     8     5     4     3

  Columns 7 through 11

    10     6     2     7     1
```



graph6.mat:

```
The final answer is:

cost= 2872.00

tour =

  Columns 1 through 6

     1     6    11     8     3     9

  Columns 7 through 12

     7     4     2    14    12     5

  Columns 13 through 16

    10    15    13     1
```


Biograph Viewer 2
File
Tools
Window
Help
Node 1
9
Node 6
27
Node 11
744
Node 8
133
Node 3
144
Node 9
70
Node 7
377
Node 4
132
Node 2
29
Node 14
508
Node 12
297
Node 5
230
Node 10
66
Node 15
99
Node 13
7