CS325 Project 3

Rikki Gibson, James Sandoval, Johanes Sunarto

1 Transshipment Model

A. I. Objective function and constraints formulated in LINDO:

```
MIN
        10CP11 + 15CP12 +
        11CP21 + 8CP22 +
        13CP31 + 8CP32 + 9CP33 +
        14CP42 + 8CP43 +
        5CW11 + 6CW12 + 7CW13 + 10CW14 +
        12CW23 + 8CW24 + 10CW25 + 14CW26 +
        14CW34 + 12CW35 + 12CW36 + 6CW37
ST
        CP11 + CP12 = 150
        CP21 + CP22 = 450
        CP31 + CP32 + CP33 = 250
        CP42 + CP43 = 150
        CW11 = 100
        CW12 = 150
        CW13 + CW23 = 100
        CW14 + CW24 + CW34 = 200
        CW25 + CW35 = 200
        CW26 + CW36 = 150
        CW37 = 100
```

II. Optimal solution from LINDO program:

LP OPTIMUM FOUND AT STEP 5

OBJECTIVE FUNCTION VALUE

1) 16400.00

VARIABLE	VALUE	REDUCED COST
CP11	150.000000	0.000000
CP12	0.000000	5.000000
CP21	0.000000	3.000000
CP22	450.000000	0.000000
CP31	0.000000	5.000000
CP32	250.000000	0.000000
CP33	0.000000	1.000000
CP42	0.000000	6.000000
CP43	150.000000	0.000000
CW11	100.000000	0.000000
CW12	150.000000	0.000000
CW13	100.000000	0.000000
CW14	0.000000	2.000000
CW23	0.000000	5.000000
CW24	200.000000	0.000000
CW25	200.000000	0.000000
CW26	0.000000	2.000000
CW34	0.000000	6.000000
CW35	0.000000	2.000000
CW36	150.000000	0.000000
CW37	100.000000	0.000000

III. The minimum cost to ship from plants, to warehouses, to retailers is \$16400. The optimal routes to use are $p_1 \to w_1$, $p_2 \to w_2$, $p_3 \to w_2$, $p_4 \to w_3$, $w_1 \to r_1$, $w_1 \to r_2$, $w_1 \to r_3$, $w_2 \to r_4$, $w_2 \to r_5$, $w_3 \to r_6$, and $w_3 \to r_7$.

B. This can be determined by eliminating variables involving warehouse 2 from the original LINDO program. The minimum cost in this version of the problem is $17200. It is feasible to do this because every plant and retailer that might have depended on warehouse 2 has an alternative it can choose, and warehouses do not have an indicated maximum capacity. The modified LINDO program is below:

```
MIN
        10CP11 + 15CP12 +
        11CP21 + 8CP22 +
        13CP31 + 8CP32 + 9CP33 +
        14CP42 + 8CP43 +
        5CW11 + 6CW12 + 7CW13 + 10CW14 +
        14CW34 + 12CW35 + 12CW36 + 6CW37
ST
        CP11 + CP12 = 150
        CP21 + CP22 = 450
        CP31 + CP32 + CP33 = 250
        CP42 + CP43 = 150
        CW11 = 100
        CW12 = 150
        CW13 = 100
        CW14 + CW34 = 200
        CW35 = 200
        CW36 = 150
        CW37 = 100
```

C. This can be expressed by adding a constraint to version A that describes the maximum amount of product that can go through warehouse 2. The system does not have a minimum load that warehouse 2 needs to bear in order to have a feasible solution. In this new version of the problem, the minimum cost is $17000.

MIN

10CP11 + 15CP12 +
11CP21 + 8CP22 +
13CP31 + 8CP32 + 9CP33 +
14CP42 + 8CP43 +
5CW11 + 6CW12 + 7CW13 + 10CW14 +
12CW23 + 8CW24 + 10CW25 + 14CW26 +
14CW34 + 12CW35 + 12CW36 + 6CW37

ST

CP11 + CP12 = 150
CP21 + CP22 = 450
CP31 + CP32 + CP33 = 250
CP42 + CP43 = 150
CW11 = 100
CW12 = 150
CW13 + CW23 = 100
CW14 + CW24 + CW34 = 200
CW25 + CW35 = 200
CW26 + CW36 = 150
CW37 = 100
CW23 + CW24 + CW25 + CW26 < 100

2 Mixture Problem

A Minimizing calories and meeting nutritional requirements

I) Decision variables:
x_t = amount of tomatoes
x_l = amount of lettuce
x_s = amount of spinach
x_c = amount of carrots
x_{ss} = amount of sunflower seeds
x_{st} = amount of smoked tofu
x_{cp} = amount of chickpeas
x_o = amount of oil

Objective Function: minimize $Cal = 21x_t + 16x_l + 40x_s + 41x_c + 585x_{ss} + 120x_{st} + 164x_{cp} + 884x_o$

Resource Constraints:

Protein: $0.85x_t + 1.62x_l + 2.86x_s + 0.93x_c + 23.4x_{ss} + 16x_{st} + 9x_{cp} + 0x_o \geq 15$

Fat: $2 \leq 0.33x_t + 0.2x_l + 0.39x_s + 0.24x_c + 48.7x_{ss} + 5x_{st} + 2.6x_{cp} + 100x_o \leq 8$

Carbs: $4.64x_t + 2.37x_l + 3.63x_s + 9.58x_c + 15x_{ss} + 3x_{st} + 27x_{cp} + 0x_o \geq 4$

Sodium: $9x_t + 28x_l + 65x_s + 69x_c + 3.8x_{ss} + 120x_{st} + 78x_{cp} + 0x_o \leq 200$

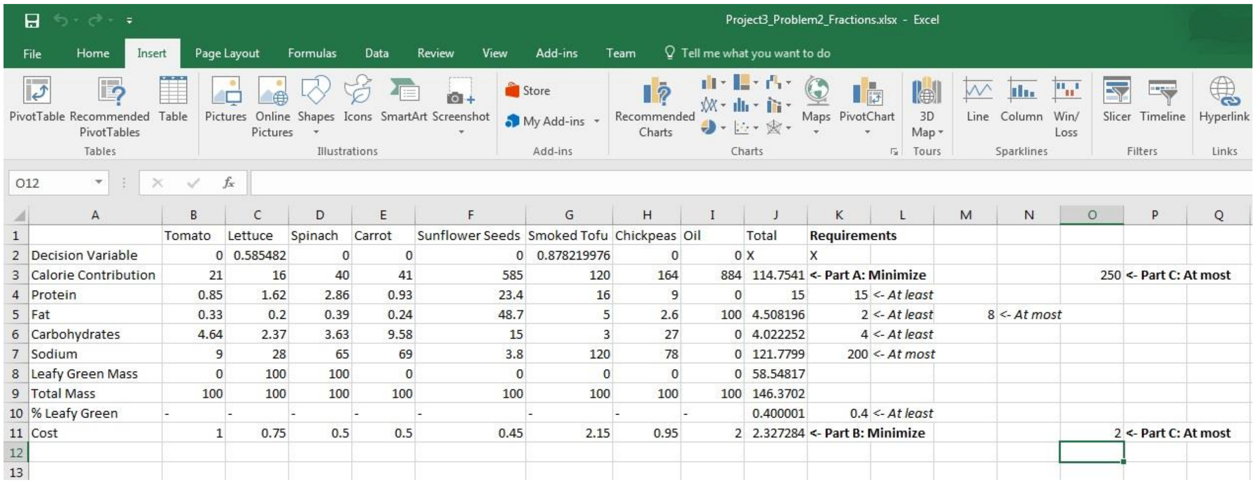
LeafyGreen: $(x_l + x_s) / (x_t + x_l + x_s + x_c + x_{ss} + x_{st} + x_{cp} + x_o) \geq 0.40$

Non-negativity Constraints: $x_t \geq 0, x_l \geq 0, x_s \geq 0, x_c \geq 0, x_{ss} \geq 0, x_{st} \geq 0, x_{cp} \geq 0, x_o \geq 0$

Project3_Problem2_Integer.xlsx - Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Tomato	Lettuce	Spinach	Carrot	Sunflower Seeds	Smoked Tofu	Chickpeas	Oil	Total	**Requirements**						
2	Decision Variable	0	1	0	0	0	1	0	0	X	X						
3	Calorie Contribution	21	16	40	41	585	120	164	884	136	**<- Part A: Minimize**				250	**<- Part C: At most**	
4	Protein	0.85	1.62	2.86	0.93	23.4	16	9	0	17.62	15	<- At least					
5	Fat	0.33	0.2	0.39	0.24	48.7	5	2.6	100	5.2	2	<- At least	8	<- At most			
6	Carbohydrates	4.64	2.37	3.63	9.58	15	3	27	0	5.37	4	<- At least					
7	Sodium	9	28	65	69	3.8	120	78	0	148	200	<- At most					
8	Leafy Green Mass	0	100	100	0	0	0	0	0	100							
9	Total Mass	100	100	100	100	100	100	100	100	200							
10	% Leafy Green	-	-	-	-	-	-	-	-	0.5	0.4	<- At least					
11	Cost	1	0.75	0.5	0.5	0.45	2.15	0.95	2	2.9	**<- Part B: Minimize**				2	**<- Part C: At most**	
12																	
13																	

II)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Tomato	Lettuce	Spinach	Carrot	Sunflower Seeds	Smoked Tofu	Chickpeas	Oil	Total	**Requirements**						
2	Decision Variable	0	0.585482	0	0	0	0.878219976	0	0	X	X						
3	Calorie Contribution	21	16	40	41	585	120	164	884	114.7541	**<- Part A: Minimize**				250	**<- Part C: At most**	
4	Protein	0.85	1.62	2.86	0.93	23.4	16	9	0	15	15	*<- At least*					
5	Fat	0.33	0.2	0.39	0.24	48.7	5	2.6	100	4.508196	2	*<- At least*		8	*<- At most*		
6	Carbohydrates	4.64	2.37	3.63	9.58	15	3	27	0	4.022252	4	*<- At least*					
7	Sodium	9	28	65	69	3.8	120	78	0	121.7799	200	*<- At most*					
8	Leafy Green Mass	0	100	100	0	0	0	0	0	58.54817							
9	Total Mass	100	100	100	100	100	100	100	100	146.3702							
10	% Leafy Green	-	-	-	-	-	-	-	-	0.400001	0.4	*<- At least*					
11	Cost	1	0.75	0.5	0.5	0.45	2.15	0.95	2	2.327284	**<- Part B: Minimize**				2	**<- Part C: At most**	
12																	
13																	

III) Assuming that each item has to have integer quantity, calories was minimized to *136*, and the cost was *$2.90*. However, if each item could have fractional quantity, calories was minimized to *114.75* and the cost was *$2.32*.

B Minimizing cost

I) Decision variables: x_t = amount of tomatoes

x_l = amount of lettuce

x_s = amount of spinach

x_c = amount of carrots

x_{ss} = amount of sunflower seeds

x_{st} = amount of smoked tofu

x_{cp} = amount of chickpeas

x_o = amount of oil

Objective Function: minimize Cost = $1x_t + 0.75x_l + 0.50x_s + 0.50x_c + 0.45x_{ss} + 2.15x_{st} + 0.95x_{cp} + 2x_o$

Resource Constraints:

Protein: $0.85x_t + 1.62x_l + 2.86x_s + 0.93x_c + 23.4x_{ss} + 16x_{st} + 9x_{cp} + 0x_o \geq 15$

Fat: $2 \leq 0.33x_t + 0.2x_l + 0.39x_s + 0.24x_c + 48.7x_{ss} + 5x_{st} + 2.6x_{cp} + 100x_o \leq 8$

Carbs: $4.64x_t + 2.37x_l + 3.63x_s + 9.58x_c + 15x_{ss} + 3x_{st} + 27x_{cp} + 0x_o \geq 4$

Sodium: $9x_t + 28x_l + 65x_s + 69x_c + 3.8x_{ss} + 120x_{st} + 78x_{cp} + 0x_o \leq 200$

LeafyGreen%: $(x_l + x_s) / (x_t + x_l + x_s + x_c + x_{ss} + x_{st} + x_{cp} + x_o) \geq 0.40$

Non-negativity Constraints: $x_t \geq 0$, $x_l \geq 0$, $x_s \geq 0$, $x_c \geq 0$, $x_{ss} \geq 0$, $x_{st} \geq 0$, $x_{cp} \geq 0$, $x_o \geq 0$

Project3_Problem2_Integer.xlsx - Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Tomato	Lettuce	Spinach	Carrot	Sunflower Seeds	Smoked Tofu	Chickpeas	Oil	Total	**Requirements**						
2	Decision Variable	0	0	1	0	0	1	0	0	X	X						
3	Calorie Contribution	21	16	40	41	585	120	164	884	160	**<- Part A: Minimize**					250	**<- Part C: At most**
4	Protein	0.85	1.62	2.86	0.93	23.4	16	9	0	18.86	15	<- At least					
5	Fat	0.33	0.2	0.39	0.24	48.7	5	2.6	100	5.39	2	<- At least		8	<- At most		
6	Carbohydrates	4.64	2.37	3.63	9.58	15	3	27	0	6.63	4	<- At least					
7	Sodium	9	28	65	69	3.8	120	78	0	185	200	<- At most					
8	Leafy Green Mass	0	100	100	0	0	0	0	0	100							
9	Total Mass	100	100	100	100	100	100	100	100	200							
10	% Leafy Green	-	-	-	-	-	-	-	-	0.5	0.4	<- At least					
11	Cost	1	0.75	0.5	0.5	0.45	2.15	0.95	2	2.65	**<- Part B: Minimize**					2	**<- Part C: At most**
12																	

II)

Project3_Problem2_Fractions.xlsx - Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Tomato	Lettuce	Spinach	Carrot	Sunflower Seeds	Smoked Tofu	Chickpeas	Oil	Total	**Requirements**						
2	Decision Variable	0	0	0.8323	0	0.096083325	0	1.1523635	0	X	X						
3	Calorie Contribution	21	16	40	41	585	120	164	884	278.4884	**<- Part A: Minimize**					250	**<- Part C: At most**
4	Protein	0.85	1.62	2.86	0.93	23.4	16	9	0	15	15	*<- At least*					
5	Fat	0.33	0.2	0.39	0.24	48.7	5	2.6	100	8	2	*<- At least*		8	*<- At most*		
6	Carbohydrates	4.64	2.37	3.63	9.58	15	3	27	0	35.57631	4	*<- At least*					
7	Sodium	9	28	65	69	3.8	120	78	0	144.349	200	*<- At most*					
8	Leafy Green Mass	0	100	100	0	0	0	0	0	83.23003							
9	Total Mass	100	100	100	100	100	100	100	100	208.0747							
10	% Leafy Green	-	-	-	-	-	-	-	-	0.400001	0.4	*<- At least*					
11	Cost	1	0.75	0.5	0.5	0.45	2.15	0.95	2	1.554133	**<- Part B: Minimize**					2	**<- Part C: At most**
12																	

III) Assuming that each item has to have integer quantity, cost was minimized to *$2.60*, and the calorie count was *160*. However, if each item could have fractional quantity, cost was minimized to *$1.55* and the calorie count was *278.48*.

C Comparing A to B

I) In her ideal world Veronica would like to sell the salad for $5 and make a profit of at least $3, so the cost of the salad has to be at most $2. She would also like to advertise that the salad has less than 250 calories.

In order to do this, Veronica could use hold one of the value as a constrained rather than a minimize objective. For example, treat 250 calorie as another maximum constraint, while minimizing cost; or treat $2 as a maximum constraint and minimize calories.

II) Adding these 2 constraints to the integer constrained formula yields no feasible results. For one it will not lower the cost under $2.60 while still maintaining nutritional value. This leads us to believe that the ingredients should not be a choice of 0 or 1 (0 or 100mg).

Using the non integer constrained formula with cost minimized actually found a solution: 0.761996 Spinach, 0.09383033 Sunflower Seeds, 0.16894129 Smoked Tofu, and 0.8802223 Chickpeas. This yields a total calorie of *250* and a total cost of *$1.62*.

Another option was to minimize calories while keeping cost under $2: 0.550343 Spinach, 0.029429026 Sunflower Seeds, 0.79608621 Smoked Tofu. This yields a total calorie of *134.76* and a cost of exactly *$2*.

3 Solving Shortest Paths

A. I. The shortest path problem may be solved using the following linear program formulation:

minimize - d_t
subject to $d_s = 0$
$d_v - d_u \leq \ell(u,v)\ \forall\ edges\ uv$
$d_u \leq\ 0\ \forall\ vertices\ u\ \in\ V$

Constraints (derived from Project3Problem3.txt):
$d_a - d_b \leq 2$
$d_a - d_c \leq 3$
$d_a - d_d \leq 8$
$d_a - d_h \leq 9$
$d_b - d_4 \leq 4$
$d_b - d_c \leq 5$
$d_b - d_e \leq 7$
$d_b - d_f \leq 4$
$d_c - d_d \leq 10$
$d_c - d_b \leq 5$
$d_c - d_g \leq 9$
$d_c - d_i \leq 11$
$d_c - d_f \leq 4$
$d_d - d_a \leq 8$
$d_d - d_g \leq 2$
$d_d - d_j \leq 5$
$d_d - d_f \leq 1$
$d_e - d_h \leq 5$
$d_e - d_c \leq 4$
$d_e - d_i \leq 10$
$d_f - d_i \leq 2$
$d_f - d_g \leq 2$
$d_g - d_d \leq 2$
$d_g - d_j \leq 8$
$d_g - d_k \leq 12$
$d_h - d_i \leq 5$
$d_h - d_k \leq 10$
$d_1 - d_a \leq 20$

$d_i - d_k \leq 6$
$d_i - d_j \leq 2$
$d_i - d_m \leq 12$
$d_j - d_i \leq 2$
$d_j - d_k \leq 4$
$d_j - d_l \leq 5$
$d_k - d_h \leq 10$
$d_k - d_m \leq 10$
$d_l - d_m \leq 2$

II. The approach used to find the solution was to use the linprog function contained within Matlab.The resulting distances are as follows:

```
Shortest Distance from a to a =  0
Shortest Distance from a to b =  2
Shortest Distance from a to c =  3
Shortest Distance from a to d =  8
Shortest Distance from a to e =  9
Shortest Distance from a to f =  6
Shortest Distance from a to g =  8
Shortest Distance from a to h =  9
Shortest Distance from a to i =  8
Shortest Distance from a to j = 10
Shortest Distance from a to k = 14
Shortest Distance from a to l = 15
Shortest Distance from a to m = 17
```

III. The following code was used to find the solution:

```
% Problem3a.m Solution Script
% 23Feb 2017
% *** Remove all headers and white space from
    input text***

clear variables
close all
clc

%% GET DATA FROM FILE
%open file
fid = fopen('Project3Problem3-1.txt');
```

```
%read lines while data
rline = fgets(fid);
rowidx = 0;

while ischar(rline)
    % inc count
    rowidx = rowidx + 1;

    % splits the string at the specified
       delimiter
    C = strsplit(rline, ' ');

    % convert nodes to indexes using ascii codes
        (a = 1, b = 2, etc.)
    edgeStart(rowidx) = double(C{1}) - double('a
       ') + 1;
    edgeEnd(rowidx) = double(C{2}) - double('a')
        + 1;
    edgeWeight(rowidx) = str2num(C{3});
    %disp(double(C{1}) - double('a') + 1);

    % go for the next line
    rline = fgetl(fid);
end

fclose(fid);

%% PROCESS DATA - FIND SHORTEST PATHS
% Shortest paths from a to all
numberOfNodes = max([edgeStart, edgeEnd]);

% Build A and B matrices from end to start
% Size A is num of inequal by num of nodes -
   numel is num of elements
A = zeros(numel(edgeWeight), numberOfNodes);

for j = 1:numel(edgeWeight)
    A(j, edgeStart(j)) = -1;
    A(j, edgeEnd(j)) = 1;
```

```
end

b = edgeWeight';

% Add constraints < 0
% identity matrix
A = [A; -eye(numberOfNodes)];
% set zeros
b = [b; zeros(numberOfNodes, 1)];

% single equality constraint - distance to A =
    0;
Aeq = zeros(1, numberOfNodes);
Aeq(1, 1) = 1;
beq = 0;

% Minimize Constraint to Max negative sum of
    distances
f = -ones(numberOfNodes, 1);

% This is where the magic happens ... call the
    linprog function to utilize
% the simplex method
 [x, fval, exitflag] = linprog(f, A, b, Aeq, beq
    );

 % Open the output file
 fid = fopen('Problem3A_Solution.txt', 'w');

 fprintf(fid, 'Problem 3 A solution:\n\n');

 % for all numbers in x, print the results
 for j = 1:numel(x)
     fprintf(fid, 'Distance from a to %c = %2.0f
         \n', char('a' + j - 1), x(j));
 end

fclose(fid);
```

B. I. Using the same code as problem 3A; Vertex z is added and is made unreachable to vertex a. This results in the following error message generated via Matlab:

Exiting: One or more of the residuals, duality gap, or total relative error has stalled: the dual appears to be infeasible and the primal unbounded since the primal objective $< -1e + 10$ and the dual objective $< 1e + 6$

The error message basically alludes to the notion that the optimization problem cannot be optimized due to the fact that no path exists between vertex a and vertex z.

C. I. In order to find the shortest path from each vertex to vertex m, we must flip the directionality of each edge. After flipping the directionality of each edge we must then find the shortest path from m to each vertex. This results in the correct answer using one linear program.

II. The resulting distances are as follows:

```
Problem 3 C solution:

Distance from a to m = 17
Distance from b to m = 15
Distance from c to m = 15
Distance from d to m = 12
Distance from e to m = 19
Distance from f to m = 11
Distance from g to m = 14
Distance from h to m = 14
Distance from i to m =  9
Distance from j to m =  7
Distance from k to m = 10
Distance from l to m =  2
Distance from m to m =  0
```

III. The following code was used to the find the solution:

```
% Problem3c.m Solution Script
% 23Feb 2017
% *** Remove all headers and white space from
   input text***
% lengths of Shortest paths from all vertexes to
    vertex m

clear variables
close all
clc

%% GET DATA FROM FILE
%open file
fid = fopen('Project3Problem3-1.txt');

%read lines while data
rline = fgets(fid);
rowidx = 0;
```

```
while ischar(rline)
    % inc count
    rowidx = rowidx + 1;

    % splits the string at the specified
       delimiter
    C = strsplit(rline, ' ');

    % convert nodes to indexes using ascii codes
        (a = 1, b = 2, etc.)
    edgeStart(rowidx) = double(C{1}) - double('a
       ') + 1;
    edgeEnd(rowidx) = double(C{2}) - double('a')
        + 1;
    edgeWeight(rowidx) = str2num(C{3});
    %disp(double(C{1}) - double('a') + 1);

    % go for the next line
    rline = fgetl(fid);
end

fclose(fid);

%% PROCESS DATA - FIND SHORTEST PATHS
% Shortest paths from a to all
numberOfNodes = max([edgeStart, edgeEnd]);

% Build A and B matrices from end to start
% Size A is num of inequal by num of nodes -
   numel is num of elements
A = zeros(numel(edgeWeight), numberOfNodes);

% reverse the direction of the edges by swapping
    1's:
% previously: edgeStart(j) = -1) and edgeEnd(j)=
    1
for j = 1:numel(edgeWeight)
    A(j, edgeStart(j)) = 1;
    A(j, edgeEnd(j)) = -1;
end
```

```
b = edgeWeight';

% Add constraints < 0
% identity matrix
A = [A; -eye(numberOfNodes)];
% set zeros
b = [b; zeros(numberOfNodes, 1)];

% single equality constraint - distance to m =
    0;
Aeq = zeros(1, numberOfNodes);
startNode = double('m') - double('a') + 1
Aeq(1, startNode) = 1;
beq = 0;

% Minimize Constraint to Max negative sum of
    distances
f = -ones(numberOfNodes, 1);

% This is where the magic happens ... call the
    linprog function to utilize
% the simplex method
 [x, fval, exitflag] = linprog(f, A, b, Aeq, beq
    );

 % Open the output file
 fid = fopen('Problem3C_Solution.txt', 'w');

 fprintf(fid, 'Problem 3 C solution:\n\n');

 % for all numbers in x, print the results
 for j = 1:numel(x)
     fprintf(fid, 'Distance from m to %c = %2.0f
         \n', char('a' + j - 1), x(j));
 end

fclose(fid);
```

D. I. The first term of the problem is equivalent to problem 3c in which we must find the shortest path from all vertices to a given vertex.

The second term of the problem is equivalent to problem 3a in which we must find the shortest path from a singular vertex to all other vertices. If we combine the lengths of these two sub-problems we will arrive at a correct solution.

As noted in the question, there are two vertices that are unreachable to i (l and m). This prevented finding shortest paths for those vertices for the sub-problem in part I; However, in the second sub-problem (part II) all vertices are reachable from i. All distances equaling NaN denote unreachable vertices.

II. The resulting Distances are as follows:

```
Shortest Distance from a to a passing  i = 28
Shortest Distance from a to b passing  i = 30
Shortest Distance from a to c passing  i = 31
Shortest Distance from a to d passing  i = 36
Shortest Distance from a to e passing  i = 37
Shortest Distance from a to f passing  i = 34
Shortest Distance from a to g passing  i = 36
Shortest Distance from a to h passing  i = 24
Shortest Distance from a to i passing  i =  8
Shortest Distance from a to j passing  i = 10
Shortest Distance from a to k passing  i = 14
Shortest Distance from a to l passing  i = 15
Shortest Distance from a to m passing  i = 17
Shortest Distance from b to b passing  i = 28
Shortest Distance from b to c passing  i = 29
Shortest Distance from b to d passing  i = 34
Shortest Distance from b to e passing  i = 35
Shortest Distance from b to f passing  i = 32
Shortest Distance from b to g passing  i = 34
Shortest Distance from b to h passing  i = 22
Shortest Distance from b to i passing  i =  6
Shortest Distance from b to j passing  i =  8
Shortest Distance from b to k passing  i = 12
Shortest Distance from b to l passing  i = 13
Shortest Distance from b to m passing  i = 15
Shortest Distance from c to c passing  i = 29
Shortest Distance from c to d passing  i = 34
Shortest Distance from c to e passing  i = 35
```

```
Shortest Distance from c to f passing  i = 32
Shortest Distance from c to g passing  i = 34
Shortest Distance from c to h passing  i = 22
Shortest Distance from c to i passing  i =  6
Shortest Distance from c to j passing  i =  8
Shortest Distance from c to k passing  i = 12
Shortest Distance from c to l passing  i = 13
Shortest Distance from c to m passing  i = 15
Shortest Distance from d to d passing  i = 31
Shortest Distance from d to e passing  i = 32
Shortest Distance from d to f passing  i = 29
Shortest Distance from d to g passing  i = 31
Shortest Distance from d to h passing  i = 19
Shortest Distance from d to i passing  i =  3
Shortest Distance from d to j passing  i =  5
Shortest Distance from d to k passing  i =  9
Shortest Distance from d to l passing  i = 10
Shortest Distance from d to m passing  i = 12
Shortest Distance from e to e passing  i = 39
Shortest Distance from e to f passing  i = 36
Shortest Distance from e to g passing  i = 38
Shortest Distance from e to h passing  i = 26
Shortest Distance from e to i passing  i = 10
Shortest Distance from e to j passing  i = 12
Shortest Distance from e to k passing  i = 16
Shortest Distance from e to l passing  i = 17
Shortest Distance from e to m passing  i = 19
Shortest Distance from f to f passing  i = 28
Shortest Distance from f to g passing  i = 30
Shortest Distance from f to h passing  i = 18
Shortest Distance from f to i passing  i =  2
Shortest Distance from f to j passing  i =  4
Shortest Distance from f to k passing  i =  8
Shortest Distance from f to l passing  i =  9
Shortest Distance from f to m passing  i = 11
Shortest Distance from g to g passing  i = 33
Shortest Distance from g to h passing  i = 21
Shortest Distance from g to i passing  i =  5
Shortest Distance from g to j passing  i =  7
Shortest Distance from g to k passing  i = 11
```

```
Shortest Distance from g to l passing  i = 12
Shortest Distance from g to m passing  i = 14
Shortest Distance from h to h passing  i = 21
Shortest Distance from h to i passing  i =  5
Shortest Distance from h to j passing  i =  7
Shortest Distance from h to k passing  i = 11
Shortest Distance from h to l passing  i = 12
Shortest Distance from h to m passing  i = 14
Shortest Distance from i to i passing  i =  0
Shortest Distance from i to j passing  i =  2
Shortest Distance from i to k passing  i =  6
Shortest Distance from i to l passing  i =  7
Shortest Distance from i to m passing  i =  9
Shortest Distance from j to j passing  i =  4
Shortest Distance from j to k passing  i =  8
Shortest Distance from j to l passing  i =  9
Shortest Distance from j to m passing  i = 11
Shortest Distance from k to k passing  i = 21
Shortest Distance from k to l passing  i = 22
Shortest Distance from k to m passing  i = 24
Shortest Distance from l to l passing  i = NaN
Shortest Distance from l to m passing  i = NaN
Shortest Distance from m to m passing  i = NaN
```

III. The Following code was used to find the solution:

```
% Problem3d.m Solution Script
% 23Feb 2017
% *** Remove all headers and white space from
   input text***
% lengths of Shortest paths from all vertices to
    vertex i, and
% lengths of shortest paths from i to all
   vertices

clear variables
close all
clc

%% GET DATA FROM FILE
%open file
```

```
fid = fopen('Project3Problem3-1.txt');

%read lines while data
rline = fgets(fid);
rowidx = 0;

while ischar(rline)
    % inc count
    rowidx = rowidx + 1;

    % splits the string at the specified
        delimiter
    C = strsplit(rline, ' ');

    % convert nodes to indexes using ascii codes
         (a = 1, b = 2, etc.)
    edgeStart(rowidx) = double(C{1}) - double('a
        ') + 1;
    edgeEnd(rowidx) = double(C{2}) - double('a')
         + 1;
    edgeWeight(rowidx) = str2num(C{3});
    %disp(double(C{1}) - double('a') + 1);

    % go for the next line
    rline = fgetl(fid);
end

fclose(fid);

%% PROCESS DATA - PART I - FIND SHORTEST PATHS
   FROM VERTICES TO i
% Shortest paths from a to all
numberOfNodes = max([edgeStart, edgeEnd]);

% Build A and B matrices from end to start
% Size A is num of inequal by num of nodes -
   numel is num of elements
A = zeros(numel(edgeWeight), numberOfNodes);

% reverse the direction of the edges by swapping
```

```
    1's:
% previously: edgeStart(j) = -1) and edgeEnd(j)=
    1
for j = 1:numel(edgeWeight)
    A(j, edgeStart(j)) = 1;
    A(j, edgeEnd(j)) = -1;
end
b = edgeWeight';

% Add constraints < 0
% identity matrix
A = [A; -eye(numberOfNodes)];
% set zeros
b = [b; zeros(numberOfNodes, 1)];

% single equality constraint - distance to i =
   0;
% Account for unreachable nodes l & m

Aeq = zeros(3, numberOfNodes);
startNode = double('i') - double('a') + 1;
nodeL = double('l') - double('a') + 1;
nodeM = double('m') - double('a') + 1;

Aeq(1, startNode) = 1;
Aeq(2, nodeL) = 1;
Aeq(3, nodeM) = 1;
beq = [0; 99999;99999];

% Minimize Constraint to Max negative sum of
   distances
f = -ones(numberOfNodes, 1);

% This is where the magic happens ... call the
   linprog function to utilize
% the simplex method
 [x, fval, exitflag] = linprog(f, A, b, Aeq, beq
    );

 %% PROCESS DATA - PART II - FIND SHORTEST PATHS
```

```
    FROM I TO ALL OTHER VERTICES

 distanceToNodei = x;

 % numberOfNodes -> highest numbered node
 numberOfNodes = max([edgeStart, edgeEnd]);

 % Build a and b matrices from edgeStart ->
    edgeEnd
 % A -> number of inequalities by num of nodes
 A = zeros(numel(edgeWeight), numberOfNodes);

 for j = 1:numel(edgeWeight)
    A(j, edgeStart(j)) = -1;
    A(j, edgeEnd(j)) = 1;
 end

b = edgeWeight';

% Add constraints < 0
% identity matrix
A = [A; -eye(numberOfNodes)];
% set zeros
b = [b; zeros(numberOfNodes, 1)];

% single equality constraint
Aeq = zeros(1, numberOfNodes);
startNode = double('i') - double('a') + 1;
Aeq(1, startNode) = 1;
beq = 0;

% Minimize Constraint to Max negative sum of
    distances
f = -ones(numberOfNodes, 1);

% call linprog
[x, fval, exitflag] = linprog(f, A, b, Aeq, beq)
    ;

distanceFromNodei = x;
```

```
%% COMBINE RESULTS FROM PART I AND PART II
for i = 1:numberOfNodes
    for j = 1:numberOfNodes
        distFromTo(i,j) = distanceToNodei(i) +
            distanceFromNodei(j);
        if distFromTo(i,j) > 999
            distFromTo(i,j) = NaN;
        end
    end
end

% Open the output file and print results
fid = fopen('Problem3D_Solution.txt', 'w');
fprintf(fid, 'Problem 3.D Solution:\n\n');
for i = 1:numberOfNodes
    for j = i:numberOfNodes
        fprintf(fid, 'Shortest Distance from %c
            to %c passing through i = %2.0f \n',
            char('a'+i-1), char('a'+j-1),
            distFromTo(i,j));
    end
end

fclose(fid);
```