CS 325 Project 3: Linear Programming

Kyle Guthrie
Michael C. Stramel
Alex Miranda

November 13, 2016

Problem 1

Problem 1 Part A

Determine the number of refrigerators to be shipped from plants to warehouses, and then warehouses to retailers to minimize the cost.

Solution

- In all 1000 units will travel through the network at a minimum cost of \$17100.

- Ship 150 units from plant #1 to warehouse #1 at a cost of \$1500.
- Ship 200 units from plant #2 to warehouse #1 at a cost of \$2200.
- Ship 250 units from plant #2 to warehouse #2 at a cost of \$2000.
- Ship 150 units from plant #3 to warehouse #2 at a cost of \$1200.
- Ship 100 units from plant #3 to warehouse #3 at a cost of \$ 900.
- Ship 150 units from plant #4 to warehouse #3 at a cost of \$1200.

- Ship 100 units from warehouse #1 to retailer #1 at a cost of \$ 500.
- Ship 150 units from warehouse #1 to retailer #2 at a cost of \$ 900.
- Ship 100 units from warehouse #1 to retailer #3 at a cost of \$ 700.
- Ship 200 units from warehouse #2 to retailer #4 at a cost of \$1600.
- Ship 200 units from warehouse #2 to retailer #5 at a cost of \$2000.
- Ship 150 units from warehouse #3 to retailer #6 at a cost of \$1800.
- Ship 100 units from warehouse #3 to retailer #7 at a cost of \$ 600.

- 150 total units will leave plant #1 (capacity is 150).
- 450 total units will leave plant #2 (capacity is 450).
- 250 total units will leave plant #3 (capacity is 250).
- 150 total units will leave plant #4 (capacity is 150).

- 350 total units will enter warehouse #1, 350 units will leave.

- 400 total units will enter warehouse #2, 400 units will leave.
- 250 total units will enter warehouse #3, 250 units will leave.

- 100 total units will enter retailer #1 (demand is 100).
- 150 total units will enter retailer #2 (demand is 150).
- 100 total units will enter retailer #3 (demand is 100).
- 200 total units will enter retailer #4 (demand is 200).
- 200 total units will enter retailer #5 (demand is 200).
- 150 total units will enter retailer #6 (demand is 150).
- 100 total units will enter retailer #7 (demand is 100).

Linear Program Formulation

1. Overall idea of problem
    - Refrigerators moving from $n = 4$ plants to $q = 3$ warehouses to $m = 7$ retailers.
    - Not all plants deliver to all warehouses.
    - Not all warehouses deliver to all retailers.
    - Costs of shipping from plants to warehouses vary by pair.
    - Costs of shipping from warehouses to retailers vary by pair.
    - Each plant has a capacity in terms of number of refrigerators it can supply.
    - Each retailer has a capacity in terms of number of refrigerators it demands.
2. What is the goal? What are you trying to achieve?
    - Determine optimal shipping routes (n to q and q to m).
    - Determine number of refrigerators moving along each route (n to q and q to m).
    - Satisfy the demand of the retailers.
    - Minimize the cost.
3. Identify variables
    - cp_{ij} = cost of moving a refrigerator between plant i and warehouse j
        - ex. $cp_{32} = 8$ = cost of moving from plant 3 to warehouse 2
        - 9 variables
    - cw_{jk} = cost of moving a refrigerator between warehouse j and retailer k
        - ex. $cp_{14} = 10$ = cost of moving from warehouse 1 to retailer 4

  - 12 variables
- s_i = capacity (supply) of each plant
  - ex. $s_2 = 450$ = number of refrigerators that plant 2 can supply
  - 4 variables
- d_k = capacity (demand) of each retailer
  - ex. $d_6 = 150$ = number of refrigerators that plant 6 demands
  - 7 variables
- np_{ij} = number of refrigerators shipped from plant i to warehouse j
  - 9 variables
- nw_{jk} = number of refrigerators shipped from warehouse j to retailer k
  - 12 variables

4. Identify constraints

- $s_1 \leq 150$
- $s_2 \leq 450$
- $s_3 \leq 250$
- $s_4 \leq 150$
- $d_1 \geq 100$
- $d_2 \geq 150$
- $d_3 \geq 100$
- $d_4 \geq 200$
- $d_5 \geq 200$
- $d_6 \geq 150$
- $d_7 \geq 100$
- $np_{11} + np_{21} + np_{31} = nw_{11} + nw_{12} + nw_{13} + nw_{14}$
- $np_{12} + np_{22} + np_{32} + np_{42} = nw_{23} + nw_{24} + nw_{25} + nw_{26}$
- $np_{33} + np_{43} = nw_{34} + nw_{35} + nw_{36} + nw_{37}$
- $s_1 = np_{11} + np_{12}$
- $s_2 = np_{21} + np_{22}$
- $s_3 = np_{31} + np_{32} + np_{33}$
- $s_4 = np_{42} + np_{43}$
- $d_1 = nw_{11}$
- $d_2 = nw_{12}$
- $d_3 = nw_{13} + nw_{23}$
- $d_4 = nw_{14} + nw_{24} + nw_{34}$

- $d_5 = nw_{25} + nw_{35}$
- $d_6 = nw_{26} + nw_{36}$
- $d_7 = nw_{37}$
- $np_{11} \geq 0$
- $np_{12} \geq 0$
- $np_{21} \geq 0$
- $np_{22} \geq 0$
- $np_{31} \geq 0$
- $np_{32} \geq 0$
- $np_{33} \geq 0$
- $np_{42} \geq 0$
- $np_{43} \geq 0$
- $nw_{11} \geq 0$
- $nw_{12} \geq 0$
- $nw_{13} \geq 0$
- $nw_{14} \geq 0$
- $nw_{23} \geq 0$
- $nw_{24} \geq 0$
- $nw_{25} \geq 0$
- $nw_{26} \geq 0$
- $nw_{34} \geq 0$
- $nw_{35} \geq 0$
- $nw_{36} \geq 0$
- $nw_{37} \geq 0$

5. Identify inputs and outputs that you can control
   - np_{ij}
   - nw_{jk}
   - $cost$

6. Specify all quantities mathematically
   - Many have been defined above already. A few more will be added here.
   - $cost = \sum_{i=1}^{n} \sum_{j=1}^{q} np_{ij} * cp_{ij} + \sum_{j=1}^{q} \sum_{k=1}^{m} nw_{jk} * cw_{jk}$
   - $cp_{11} = 10$
   - $cp_{12} = 15$

- $cp_{21} = 11$
- $cp_{22} = 8$
- $cp_{31} = 13$
- $cp_{32} = 8$
- $cp_{33} = 9$
- $cp_{42} = 14$
- $cp_{43} = 8$
- $cw_{11} = 5$
- $cw_{12} = 6$
- $cw_{13} = 7$
- $cw_{14} = 10$
- $cw_{23} = 12$
- $cw_{24} = 8$
- $cw_{25} = 10$
- $cw_{26} = 14$
- $cw_{34} = 14$
- $cw_{35} = 12$
- $cw_{36} = 12$
- $cw_{37} = 6$

7. Check the model for completeness and correctness
   - All variables are positive.

Matlab Code

```
% --------------------------------------------------------------
% reference: array index to variable mapping
% --------------------------------------------------------------
% (1) np11
% (2) np12
% (3) np21
% (4) np22
% (5) np31
% (6) np32
% (7) np33
% (8) np42
% (9) np43
% (10) nw11
% (11) nw12
% (12) nw13
```

```
% (13) nw14
% (14) nw23
% (15) nw24
% (16) nw25
% (17) nw26
% (18) nw34
% (19) nw35
% (20) nw36
% (21) nw37
% (22) s1
% (23) s2
% (24) s3
% (25) s4
% (26) d1
% (27) d2
% (28) d3
% (29) d4
% (30) d5
% (31) d6
% (32) d7

% --------------------------------------------------------------
% lower bounds vector
%   note matlab arrays/vectors start at index 1 (not 0)
% --------------------------------------------------------------
lb = zeros(32,1);
lb(26) = 100;   % d1
lb(27) = 150;   % d2
lb(28) = 100;   % d3
lb(29) = 200;   % d4
lb(30) = 200;   % d5
lb(31) = 150;   % d6
lb(32) = 100;   % d7

% --------------------------------------------------------------
% upper bounds vector
%   note matlab arrays/vectors start at index 1 (not 0)
% --------------------------------------------------------------
ub = Inf(32,1);
ub(1)  = 150;   % np11
ub(2)  = 150;   % np12
ub(3)  = 450;   % np21
ub(4)  = 450;   % np22
ub(5)  = 250;   % np31
ub(6)  = 250;   % np32
ub(7)  = 250;   % np33
ub(8)  = 150;   % np42
ub(9)  = 150;   % np43
ub(22) = 150;   % s1
ub(23) = 450;   % s2
ub(24) = 250;   % s3
ub(25) = 150;   % s4

% --------------------------------------------------------------
```

```
% linear inequality matrix and vector
%   note matlab arrays/vectors start at index 1 (not 0)
% ------------------------------------------------------------
A = [];
b = [];

% ------------------------------------------------------------
% linear equality matrix and vector
%   note matlab arrays/vectors start at index 1 (not 0)
%   14 equations in 32 variables
% ------------------------------------------------------------
Aeq = zeros(14, 32);
beq = zeros(14, 1);
%np11 + np21 + np31        = nw11 + nw12 + nw13 + nw14
%np11 + np21 + np31 - nw11 - nw12 - nw13 - nw14 = 0
Aeq(1,[1,3,5,10,11,12,13]) = [1,1,1,-1,-1,-1,-1];
%np12 + np22 + np32 + np42 =               nw23 + nw24 + nw25 + nw26
%np12 + np22 + np32 + np42 - nw23 - nw24 - nw25 - nw26 = 0
Aeq(2,[2,4,6,8,14,15,16,17]) = [1,1,1,1,-1,-1,-1,-1];
%               np33 + np43 =                     nw34 + nw35 + nw36 + nw37
%np33 + np43 - nw34 - nw35 - nw36 - nw37 = 0
Aeq(3,[7,9,18,19,20,21]) = [1,1,-1,-1,-1,-1];
%s1 = np11 + np12
%s1 - np11 - np12 = 0
Aeq(4,[22,1,2]) = [1,-1,-1];
%s2 = np21 + np22
%s2 - np21 - np22 = 0
Aeq(5,[23,3,4]) = [1,-1,-1];
%s3 = np31 + np32 + np33
%s3 - np31 - np32 - np33 = 0
Aeq(6,[24,5,6,7]) = [1,-1,-1,-1];
%s4 =        np42 + np43
%s4 - np42 - np43 = 0
Aeq(7,[25,8,9]) = [1,-1,-1];
%d1 = nw11
%d1 - nw11 = 0
Aeq(8,[26,10]) = [1,-1];
%d2 = nw12
%d2 - nw12 = 0
Aeq(9,[27,11]) = [1,-1];
%d3 = nw13 + nw23
%d3 - nw13 - nw23 = 0
Aeq(10,[28,12,14]) = [1,-1,-1];
%d4 = nw14 + nw24 + nw34
%d4 - nw14 - nw24 - nw34 = 0
Aeq(11,[29,13,15,18]) = [1,-1,-1,-1];
%d5 =        nw25 + nw35
%d5 - nw25 - nw35 = 0
Aeq(12,[30,16,19]) = [1,-1,-1];
%d6 =        nw26 + nw36
%d6 - nw26 - nw36 = 0
Aeq(13,[31,17,20]) = [1,-1,-1];
%d7 =                      nw37
%d7 - nw37 = 0
```

```
Aeq(14,[32,21]) = [1,-1];

% ------------------------------------------------------------
% objective function vector
%   note matlab arrays/vectors start at index 1 (not 0)
% ------------------------------------------------------------
f = zeros(32,1);
f(1)  = 10;    % np11(value in f is cp11)
f(2)  = 15;    % np12(value in f is cp12)
f(3)  = 11;    % np21(value in f is cp21)
f(4)  = 8;     % np22(value in f is cp22)
f(5)  = 13;    % np31(value in f is cp31)
f(6)  = 8;     % np32(value in f is cp32)
f(7)  = 9;     % np33(value in f is cp33)
f(8)  = 14;    % np42(value in f is cp42)
f(9)  = 8;     % np43(value in f is cp43)
f(10) = 5;     % nw11(value in f is cw11)
f(11) = 6;     % nw12(value in f is cw12)
f(12) = 7;     % nw13(value in f is cw13)
f(13) = 10;    % nw14(value in f is cw14)
f(14) = 12;    % nw23(value in f is cw23)
f(15) = 8;     % nw24(value in f is cw24)
f(16) = 10;    % nw25(value in f is cw25)
f(17) = 14;    % nw26(value in f is cw26)
f(18) = 14;    % nw34(value in f is cw34)
f(19) = 12;    % nw35(value in f is cw35)
f(20) = 12;    % nw36(value in f is cw36)
f(21) = 6;     % nw37(value in f is cw37)

% ------------------------------------------------------------
% call solver and obtain solution
% ------------------------------------------------------------
[x fval] = linprog(f,A,b,Aeq,beq,lb,ub);

% ------------------------------------------------------------
% print the optimum shipping routes and min cost
% ------------------------------------------------------------
fileID = fopen('partA.out','w');
fprintf(fileID, '-----------------------------------------------------\n');
fprintf(fileID, 'Project 3 Problem 1 Part A Solution\n');
fprintf(fileID, '-----------------------------------------------------\n');
fprintf(fileID, '\n');
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(1), 1, 1, x(1) * f(1));
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(2), 1, 2, x(2) * f(2));
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(3), 2, 1, x(3) * f(3));
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(4), 2, 2, x(4) * f(4));
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(5), 3, 1, x(5) * f(5));
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(6), 3, 2, x(6) * f(6));
```

```
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(7), 3, 3, x(7) * f(7));
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(8), 4, 2, x(8) * f(8));
fprintf(fileID, ['Ship %3.0f units from plant #%d to warehouse #%d ', ...
    'at a cost of $%4.0f.\n'], x(9), 4, 3, x(9) * f(9));
fprintf(fileID, '\n');
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(10), 1, 1, x(10) * f(10));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(11), 1, 2, x(11) * f(11));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(12), 1, 3, x(12) * f(12));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(13), 1, 4, x(13) * f(13));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(14), 2, 3, x(14) * f(14));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(15), 2, 4, x(15) * f(15));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(16), 2, 5, x(16) * f(16));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(17), 2, 6, x(17) * f(17));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(18), 3, 4, x(18) * f(18));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(19), 3, 5, x(19) * f(19));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(20), 3, 6, x(20) * f(20));
fprintf(fileID, ['Ship %3.0f units from warehouse #%d to retailer ', ...
    '#%d at a cost of $%4.0f.\n'], x(21), 3, 7, x(21) * f(21));
fprintf(fileID, '\n');
fprintf(fileID, ['%3.0f total units will leave plant #%d (capacity ', ...
    'is %3.0f).\n'], x(1) + x(2), 1, x(22));
fprintf(fileID, ['%3.0f total units will leave plant #%d (capacity ', ...
    'is %3.0f).\n'], x(3) + x(4), 2, x(23));
fprintf(fileID, ['%3.0f total units will leave plant #%d (capacity ', ...
    'is %3.0f).\n'], x(5) + x(6) + x(7), 3, x(24));
fprintf(fileID, ['%3.0f total units will leave plant #%d (capacity ', ...
    'is %3.0f).\n'], x(8) + x(9), 4, x(25));
fprintf(fileID, '\n');
temp = x(10) + x(11) + x(12) + x(13);
fprintf(fileID, ['%3.0f total units will enter warehouse #%d, %3.0f ', ...
    'units will leave.\n'], x(1) + x(3) + x(5), 1, temp);
temp = x(14) + x(15) + x(16) + x(17);
fprintf(fileID, ['%3.0f total units will enter warehouse #%d, %3.0f ', ...
    'units will leave.\n'], x(2) + x(4) + x(6) + x(8), 2, temp);
temp = x(18) + x(19) + x(20) + x(21);
fprintf(fileID, ['%3.0f total units will enter warehouse #%d, %3.0f ', ...
    'units will leave.\n'], x(7) + x(9), 3, temp);
fprintf(fileID, '\n');
fprintf(fileID, ['%3.0f total units will enter retailer #%d (demand ', ...
    'is %3.0f).\n'], x(10), 1, x(26));
fprintf(fileID, ['%3.0f total units will enter retailer #%d (demand ', ...
```

```
    'is %3.0f).\n'], x(11), 2, x(27));
fprintf(fileID, ['%3.0f total units will enter retailer #%d (demand ', ...
    'is %3.0f).\n'], x(12) + x(14), 3, x(28));
fprintf(fileID, ['%3.0f total units will enter retailer #%d (demand ', ...
    'is %3.0f).\n'], x(13) + x(15) + x(18), 4, x(29));
fprintf(fileID, ['%3.0f total units will enter retailer #%d (demand ', ...
    'is %3.0f).\n'], x(16) + x(19), 5, x(30));
fprintf(fileID, ['%3.0f total units will enter retailer #%d (demand ', ...
    'is %3.0f).\n'], x(17) + x(20), 6, x(31));
fprintf(fileID, ['%3.0f total units will enter retailer #%d (demand ', ...
    'is %3.0f).\n'], x(21), 7, x(32));
fprintf(fileID, '\n');
total = x(22) + x(23) + x(24) + x(25);
fprintf(fileID, ['In all %3.0f units will travel through the network ', ...
    'at a minimum cost of $%5.0f.\n'], total, fval);
fclose(fileID);
```

Problem 1 Part B

Determine the number of refrigerators to be shipped from plants to warehouses, and then warehouses to retailers to minimize the cost. For part B warehouse 2 is closed along with all associated routes. Changes to the problem statement or solution relative to Problem 1 Part A are highlighted in red.

Solution

There is no solution when warehouse 2 is closed. The following is the error message that is returned by Matlab function linprog():

Exiting: One or more of the residuals, duality gap, or total relative error has grown 100000 times greater than its minimum value so far:
    the primal appears to be infeasible (and the dual unbounded).
    (The dual residual < TolFun=1.00e-08.)

So Matlab tells us that there is no feasible solution. Why is that? If you look back at the network diagram and the supply and demand tables, you'll note that with warehouse 2 out of commission, retailers 5, 6, and 7 can only receive shipments from warehouse 3 and their total demand is 450 units. At the same time, warehouse 3 can only receive shipments from plant 3 and 4. The total supply capacity of those plants is only 400 units. Therefore, warehouse 3 gets 50 less units from plants 3 and 4 than are demanded from retailers 5, 6, and 7.

Linear Program Formulation

1. Overall idea of problem
    - Refrigerators moving from $n = 4$ plants to $q = 2$ warehouses to $m = 7$ retailers.
    - Not all plants deliver to all warehouses.
    - Not all warehouses deliver to all retailers.
    - Costs of shipping from plants to warehouses vary by pair.
    - Costs of shipping from warehouses to retailers vary by pair.
    - Each plant has a capacity in terms of number of refrigerators it can supply.
    - Each retailer has a capacity in terms of number of refrigerators it demands.
    - Warehouse 2 has closed and all associate routes have been eliminated.
2. What is the goal? What are you trying to achieve?
    - Unchanged from part A.
3. Identify variables
    - Unchanged from part A.

4. Identify constraints

    - All constraints from part A remain in effect with the addition of two new constraints:
    - $np_{12} + np_{22} + np_{32} + np_{42} = 0$
    - $nw_{23} + nw_{24} + nw_{25} + nw_{26} = 0$

5. Identify inputs and outputs that you can control

    - Unchanged from part A.

6. Specify all quantities mathematically

    - Unchanged from part A.

7. Check the model for completeness and correctness

    - All variables are positive.

Matlab Code

Code minimally changed from part A. Only changes are 2 additional constraints (16 total equations) in the linear equality matrix and vector. Identical code from part A is not shown below (to save space).

```
% -------------------------------------------------------------
% linear equality matrix and vector
%   note matlab arrays/vectors start at index 1 (not 0)
%   16 equations in 32 variables
% -------------------------------------------------------------
Aeq = zeros(16, 32);
beq = zeros(16, 1);
%np11 + np21 + np31        = nw11 + nw12 + nw13 + nw14
%np11 + np21 + np31 - nw11 - nw12 - nw13 - nw14 = 0
Aeq(1,[1,3,5,10,11,12,13]) = [1,1,1,-1,-1,-1,-1];
%np12 + np22 + np32 + np42 =               nw23 + nw24 + nw25 + nw26
%np12 + np22 + np32 + np42 - nw23 - nw24 - nw25 - nw26 = 0
Aeq(2,[2,4,6,8,14,15,16,17]) = [1,1,1,1,-1,-1,-1,-1];
%              np33 + np43 =                      nw34 + nw35 + nw36 + nw37
%np33 + np43 - nw34 - nw35 - nw36 - nw37 = 0
Aeq(3,[7,9,18,19,20,21]) = [1,1,-1,-1,-1,-1];
%s1 = np11 + np12
%s1 - np11 - np12 = 0
Aeq(4,[22,1,2]) = [1,-1,-1];
%s2 = np21 + np22
%s2 - np21 - np22 = 0
Aeq(5,[23,3,4]) = [1,-1,-1];
%s3 = np31 + np32 + np33
%s3 - np31 - np32 - np33 = 0
Aeq(6,[24,5,6,7]) = [1,-1,-1,-1];
%s4 =        np42 + np43
%s4 - np42 - np43 = 0
```

```
Aeq(7,[25,8,9]) = [1,-1,-1];
%d1 = nw11
%d1 - nw11 = 0
Aeq(8,[26,10]) = [1,-1];
%d2 = nw12
%d2 - nw12 = 0
Aeq(9,[27,11]) = [1,-1];
%d3 = nw13 + nw23
%d3 - nw13 - nw23 = 0
Aeq(10,[28,12,14]) = [1,-1,-1];
%d4 = nw14 + nw24 + nw34
%d4 - nw14 - nw24 - nw34 = 0
Aeq(11,[29,13,15,18]) = [1,-1,-1,-1];
%d5 =        nw25 + nw35
%d5 - nw25 - nw35 = 0
Aeq(12,[30,16,19]) = [1,-1,-1];
%d6 =        nw26 + nw36
%d6 - nw26 - nw36 = 0
Aeq(13,[31,17,20]) = [1,-1,-1];
%d7 =                      nw37
%d7 - nw37 = 0
Aeq(14,[32,21]) = [1,-1];
%np12 + np22 + np32 + np42 = 0
Aeq(15,[2,4,6,8]) = [1,1,1,1];
%nw23 + nw24 + nw25 + nw26 = 0
Aeq(16,[14,15,16,17]) = [1,1,1,1];
```

Problem 1 Part C

Determine the number of refrigerators to be shipped from plants to warehouses, and then warehouses to retailers to minimize the cost. For part C warehouse 2 is limited to just 100 units per week. Changes to the problem statement or solution relative to Problem 1 Part A are highlighted in blue.

Solution

- In all 1000 units will travel through the network at a minimum cost of $18300.

- Ship 150 units from plant #1 to warehouse #1 at a cost of $1500.
- Ship 350 units from plant #2 to warehouse #1 at a cost of $3850.
- Ship 100 units from plant #2 to warehouse #2 at a cost of $800.
- Ship 0 units from plant #3 to warehouse #2 at a cost of $ 0.
- Ship 250 units from plant #3 to warehouse #3 at a cost of $2250.
- Ship 150 units from plant #4 to warehouse #3 at a cost of $1200.

- Ship 100 units from warehouse #1 to retailer #1 at a cost of $ 500.
- Ship 150 units from warehouse #1 to retailer #2 at a cost of $ 900.
- Ship 100 units from warehouse #1 to retailer #3 at a cost of $ 700.
- Ship 150 units from warehouse #1 to retailer #4 at a cost of $1500.
- Ship 50 units from warehouse #2 to retailer #4 at a cost of $ 400.
- Ship 50 units from warehouse #2 to retailer #5 at a cost of $ 500.
- Ship 150 units from warehouse #3 to retailer #5 at a cost of $1800.
- Ship 150 units from warehouse #3 to retailer #6 at a cost of $1800.
- Ship 100 units from warehouse #3 to retailer #7 at a cost of $ 600.

- 150 total units will leave plant #1 (capacity is 150).
- 450 total units will leave plant #2 (capacity is 450).
- 250 total units will leave plant #3 (capacity is 250).
- 150 total units will leave plant #4 (capacity is 150).

- 500 total units will enter warehouse #1, 500 units will leave.
- 100 total units will enter warehouse #2, 100 units will leave.
- 400 total units will enter warehouse #3, 400 units will leave.

- 100 total units will enter retailer #1 (demand is 100).
- 150 total units will enter retailer #2 (demand is 150).
- 100 total units will enter retailer #3 (demand is 100).
- 200 total units will enter retailer #4 (demand is 200).
- 200 total units will enter retailer #5 (demand is 200).
- 150 total units will enter retailer #6 (demand is 150).
- 100 total units will enter retailer #7 (demand is 100).

Linear Program Formulation

1. Overall idea of problem
    - Refrigerators moving from $n = 4$ plants to $q = 3$ warehouses to $m = 7$ retailers.
    - Not all plants deliver to all warehouses.
    - Not all warehouses deliver to all retailers.
    - Costs of shipping from plants to warehouses vary by pair.
    - Costs of shipping from warehouses to retailers vary by pair.
    - Each plant has a capacity in terms of number of refrigerators it can supply.
    - Each retailer has a capacity in terms of number of refrigerators it demands.
    - Warehouse 2 is limited to just 100 units in and out per week.
2. What is the goal? What are you trying to achieve?
    - Unchanged from part A.
3. Identify variables
    - Unchanged from part A.
4. Identify constraints
    - All constraints from part A remain in effect with the addition of two new constraints:
    - $np_{12} + np_{22} + np_{32} + np_{42} = 100$
    - $nw_{23} + nw_{24} + nw_{25} + nw_{26} = 100$

5. Identify inputs and outputs that you can control
    - Unchanged from part A.
6. Specify all quantities mathematically
    - Unchanged from part A.
7. Check the model for completeness and correctness
    - All variables are positive.

Matlab Code

Code minimally changed from part A. Only changes are 2 additional constraints (16 total equations) in the linear equality matrix and vector. Identical code from part A is not shown below (to save space).

```
% -------------------------------------------------------------
% linear equality matrix and vector
%   note matlab arrays/vectors start at index 1 (not 0)
%   16 equations in 32 variables
% -------------------------------------------------------------
Aeq = zeros(16, 32);
beq = zeros(16, 1);
%np11 + np21 + np31        = nw11 + nw12 + nw13 + nw14
%np11 + np21 + np31 - nw11 - nw12 - nw13 - nw14 = 0
Aeq(1,[1,3,5,10,11,12,13]) = [1,1,1,-1,-1,-1,-1];
%np12 + np22 + np32 + np42 =               nw23 + nw24 + nw25 + nw26
%np12 + np22 + np32 + np42 - nw23 - nw24 - nw25 - nw26 = 0
Aeq(2,[2,4,6,8,14,15,16,17]) = [1,1,1,1,-1,-1,-1,-1];
%              np33 + np43 =                      nw34 + nw35 + nw36 + nw37
%np33 + np43 - nw34 - nw35 - nw36 - nw37 = 0
Aeq(3,[7,9,18,19,20,21]) = [1,1,-1,-1,-1,-1];
%s1 = np11 + np12
%s1 - np11 - np12 = 0
Aeq(4,[22,1,2]) = [1,-1,-1];
%s2 = np21 + np22
%s2 - np21 - np22 = 0
Aeq(5,[23,3,4]) = [1,-1,-1];
%s3 = np31 + np32 + np33
%s3 - np31 - np32 - np33 = 0
Aeq(6,[24,5,6,7]) = [1,-1,-1,-1];
%s4 =        np42 + np43
%s4 - np42 - np43 = 0
Aeq(7,[25,8,9]) = [1,-1,-1];
%d1 = nw11
%d1 - nw11 = 0
Aeq(8,[26,10]) = [1,-1];
%d2 = nw12
%d2 - nw12 = 0
Aeq(9,[27,11]) = [1,-1];
```

```
%d3 = nw13 + nw23
%d3 - nw13 - nw23 = 0
Aeq(10,[28,12,14]) = [1,-1,-1];
%d4 = nw14 + nw24 + nw34
%d4 - nw14 - nw24 - nw34 = 0
Aeq(11,[29,13,15,18]) = [1,-1,-1,-1];
%d5 =        nw25 + nw35
%d5 - nw25 - nw35 = 0
Aeq(12,[30,16,19]) = [1,-1,-1];
%d6 =        nw26 + nw36
%d6 - nw26 - nw36 = 0
Aeq(13,[31,17,20]) = [1,-1,-1];
%d7 =                      nw37
%d7 - nw37 = 0
Aeq(14,[32,21]) = [1,-1];
%np12 + np22 + np32 + np42 = 100
Aeq(15,[2,4,6,8]) = [1,1,1,1];
beq(15) = 100;
%nw23 + nw24 + nw25 + nw26 = 100
Aeq(16,[14,15,16,17]) = [1,1,1,1];
beq(16) = 100;
```

Problem 1 Part D

Formulate a generalized linear programming model for the transshipment problem. Give the objective function and constraints as mathematical formulas.

Note

I can't help but feel that I've already done this in Parts A through C but I'll repeat (and try to expand) here in a more general form.

Objective Function

Whereas...

n = number of plants ≥ 1
q = number of warehouses ≥ 1
m = number of retailers ≥ 1
np_{ij} = number of refrigerators shipped from plant p_i to warehouse $w_j \geq 0$
cp_{ij} = cost of moving a refrigerator between plant p_i and warehouse $w_j > 0$
nw_{jk} = number of refrigerators shipped from warehouse w_j to retailer $r_k \geq 0$
cw_{jk} = cost of moving a refrigerator between warehouse w_j and retailer $r_k > 0$

The **objective function** is to...

minimize $cost = \sum_{i=1}^{n} \sum_{j=1}^{q} np_{ij} * cp_{ij} + \sum_{j=1}^{q} \sum_{k=1}^{m} nw_{jk} * cw_{jk}$

Constraints

The objective function (designed to minimize *cost*) is **subject to** the following contraints...

$0 \leq$ capacity of each plant $= s_i \leq$ some maximum
$0 \leq$ throughput capability of each warehouse $= q_j \leq$ some maximum
$0 \leq$ demand of each retailer $= m_k \leq$ some maximum

and also...

$\sum_{i=1}^{n} np_{ij}$ = capacity of a given plant p_i

$\sum_{i=1}^{n} np_{ij} = \sum_{k=1}^{m} nw_{jk}$ = throughput capability for a given warehouse w_j

$\sum_{k=1}^{m} nw_{jk}$ = demand of a given retailer r_k

Problem 2

Problem 2 Part A

Determine the combination of ingredients that minimizes calories but meets all nutritional requirements.

i. Formulate the problem as a linear program with an objective function and all constraints.

   We can formulate this problem as the following linear program:

$$
\begin{aligned}
\text{minimize} \quad & 21 \cdot w_{\text{tomato}} + 16 \cdot w_{\text{lettuce}} + 40 \cdot w_{\text{spinach}} + 41 \cdot w_{\text{carrot}} \\
& + 585 \cdot w_{\text{sunflower seed}} + 120 \cdot w_{\text{smoked tofu}} + 164 \cdot w_{\text{chickpea}} + 884 \cdot w_{\text{oil}} \\
\\
\text{subject to} \quad & - 0.85 \cdot w_{\text{tomato}} - 1.62 \cdot w_{\text{lettuce}} - 2.86 \cdot w_{\text{spinach}} - 0.93 \cdot w_{\text{carrot}} \\
& - 23.4 \cdot w_{\text{sunflower seed}} - 16 \cdot w_{\text{smoked tofu}} - 9 \cdot w_{\text{chickpea}} - 0 \cdot w_{\text{oil}} \leq -15, \\
\\
& - 0.33 \cdot w_{\text{tomato}} - 0.2 \cdot w_{\text{lettuce}} - 0.39 \cdot w_{\text{spinach}} - 0.24 \cdot w_{\text{carrot}} \\
& - 48.7 \cdot w_{\text{sunflower seed}} - 5 \cdot w_{\text{smoked tofu}} - 2.6 \cdot w_{\text{chickpea}} - 100 \cdot w_{\text{oil}} \leq -2, \\
\\
& 0.33 \cdot w_{\text{tomato}} + 0.2 \cdot w_{\text{lettuce}} + 0.39 \cdot w_{\text{spinach}} + 0.24 \cdot w_{\text{carrot}} \\
& + 48.7 \cdot w_{\text{sunflower seed}} + 5 \cdot w_{\text{smoked tofu}} + 2.6 \cdot w_{\text{chickpea}} + 100 \cdot w_{\text{oil}} \leq 8, \\
\\
& - 4.64 \cdot w_{\text{tomato}} - 2.37 \cdot w_{\text{lettuce}} - 3.63 \cdot w_{\text{spinach}} - 9.58 \cdot w_{\text{carrot}} \\
& - 15 \cdot w_{\text{sunflower seed}} - 3 \cdot w_{\text{smoked tofu}} - 27 \cdot w_{\text{chickpea}} - 0 \cdot w_{\text{oil}} \leq -4, \\
\\
& 9 \cdot w_{\text{tomato}} + 28 \cdot w_{\text{lettuce}} + 65 \cdot w_{\text{spinach}} + 69 \cdot w_{\text{carrot}} \\
& + 3.8 \cdot w_{\text{sunflower seed}} + 120 \cdot w_{\text{smoked tofu}} + 78 \cdot w_{\text{chickpea}} + 0 \cdot w_{\text{oil}} \leq 200, \\
\\
& 0.4 \cdot w_{\text{tomato}} - 0.6 \cdot w_{\text{lettuce}} - 0.6 \cdot w_{\text{spinach}} + 0.4 \cdot w_{\text{carrot}} \\
& + 0.4 \cdot w_{\text{sunflower seed}} + 0.4 \cdot w_{\text{smoked tofu}} + 0.4 \cdot w_{\text{chickpea}} + 0.4 \cdot w_{\text{oil}} \leq 0, \\
\\
& - w_{\text{tomato}} \leq 0 \\
& - w_{\text{lettuce}} \leq 0 \\
& - w_{\text{spinach}} \leq 0 \\
& - w_{\text{carrot}} \leq 0 \\
& - w_{\text{sunflower seed}} \leq 0 \\
& - w_{\text{smoked tofu}} \leq 0 \\
& - w_{\text{chickpea}} \leq 0 \\
& - w_{\text{oil}} \leq 0
\end{aligned}
$$

where the w parameters are the weight of each ingredient, in 100's of grams. The equations, from top to bottom, are:

1) Minimize total calories
2) Subject to total protein ≥ 15 grams
3) Subject to total fat ≥ 2 grams
4) Subject to total fat ≤ 8 grams
5) Subject to total carbohydrates ≥ 4 grams
6) Subject to total sodium ≤ 200 milligrams
7) Subject to total leafy green mass $\geq 40\%$ of total mass
8) Subject to individual ingredient weights ≥ 0

ii. Determine the optimal solution for the linear program using any software you want. Include a copy of the code/file in the report.

The following MATLAB code was used to generate the solution:

```
f = [21, 16, 40, 41, 585, 120, 164, 884];

A = [-0.85, -1.62, -2.86, -0.93, -23.4, -16, -9, 0;
     -0.33, -0.2, -0.39, -0.24, -48.7, -5, -2.6, -100;
     0.33, 0.2, 0.39, 0.24, 48.7, 5, 2.6, 100;
     -4.64, -2.37, -3.63, -9.58, -15, -3, -27, 0;
     9, 28, 65, 69, 3.8, 120, 78, 0;
     0.4, -0.6, -0.6, 0.4, 0.4, 0.4, 0.4, 0.4;
     -eye(8)];

b = [-15; -2; 8; -4; 200;  0; zeros(8,1)];

[X,FVAL,EXITFLAG] = linprog(f,A,b);
```

where `X` stores the resulting weights of ingredients (in 100's of grams), `FVAL` stores the minimized number of calories, and `EXITFLAG` stores the status of the `linprog` optimization.

iii. What is the cost of the low calorie salad?

The optimal low calorie salad contains the following weights of ingredients (in 100's of grams):

$$
\begin{aligned}
w_{\text{tomato}} &\approx 0\\
w_{\text{lettuce}} &\approx 0.5855\\
w_{\text{spinach}} &\approx 0\\
w_{\text{carrot}} &\approx 0\\
w_{\text{sunflower seed}} &\approx 0\\
w_{\text{smoked tofu}} &\approx 0.8782\\
w_{\text{chickpea}} &\approx 0\\
w_{\text{oil}} &\approx 0
\end{aligned}
$$

The optimal low calorie salad costs approximately \$2.33. It contains approximately 114.75 kcal

Problem 2 Part B

Veronica realizes that it is also important to minimize the cost associated with the new salad. Unfortunately some of the ingredients can be expensive. Determine the combination of ingredients that minimizes cost.

i. Formulate the problem as a linear program with an objective function and all constraints.

   We can formulate this problem as the following linear program:

$$
\begin{aligned}
\text{minimize} \quad & 1 \cdot w_{\text{tomato}} + 0.75 \cdot w_{\text{lettuce}} + 0.5 \cdot w_{\text{spinach}} + 0.5 \cdot w_{\text{carrot}} \\
& + 0.45 \cdot w_{\text{sunflower seed}} + 2.15 \cdot w_{\text{smoked tofu}} + 0.95 \cdot w_{\text{chickpea}} + 2.00 \cdot w_{\text{oil}} \\
\text{subject to} \quad & - 0.85 \cdot w_{\text{tomato}} - 1.62 \cdot w_{\text{lettuce}} - 2.86 \cdot w_{\text{spinach}} - 0.93 \cdot w_{\text{carrot}} \\
& - 23.4 \cdot w_{\text{sunflower seed}} - 16 \cdot w_{\text{smoked tofu}} - 9 \cdot w_{\text{chickpea}} - 0 \cdot w_{\text{oil}} \leq -15, \\
& - 0.33 \cdot w_{\text{tomato}} - 0.2 \cdot w_{\text{lettuce}} - 0.39 \cdot w_{\text{spinach}} - 0.24 \cdot w_{\text{carrot}} \\
& - 48.7 \cdot w_{\text{sunflower seed}} - 5 \cdot w_{\text{smoked tofu}} - 2.6 \cdot w_{\text{chickpea}} - 100 \cdot w_{\text{oil}} \leq -2, \\
& 0.33 \cdot w_{\text{tomato}} + 0.2 \cdot w_{\text{lettuce}} + 0.39 \cdot w_{\text{spinach}} + 0.24 \cdot w_{\text{carrot}} \\
& + 48.7 \cdot w_{\text{sunflower seed}} + 5 \cdot w_{\text{smoked tofu}} + 2.6 \cdot w_{\text{chickpea}} + 100 \cdot w_{\text{oil}} \leq 8, \\
& - 4.64 \cdot w_{\text{tomato}} - 2.37 \cdot w_{\text{lettuce}} - 3.63 \cdot w_{\text{spinach}} - 9.58 \cdot w_{\text{carrot}} \\
& - 15 \cdot w_{\text{sunflower seed}} - 3 \cdot w_{\text{smoked tofu}} - 27 \cdot w_{\text{chickpea}} - 0 \cdot w_{\text{oil}} \leq -4, \\
& 9 \cdot w_{\text{tomato}} + 28 \cdot w_{\text{lettuce}} + 65 \cdot w_{\text{spinach}} + 69 \cdot w_{\text{carrot}} \\
& + 3.8 \cdot w_{\text{sunflower seed}} + 120 \cdot w_{\text{smoked tofu}} + 78 \cdot w_{\text{chickpea}} + 0 \cdot w_{\text{oil}} \leq 200, \\
& 0.4 \cdot w_{\text{tomato}} - 0.6 \cdot w_{\text{lettuce}} - 0.6 \cdot w_{\text{spinach}} + 0.4 \cdot w_{\text{carrot}} \\
& + 0.4 \cdot w_{\text{sunflower seed}} + 0.4 \cdot w_{\text{smoked tofu}} + 0.4 \cdot w_{\text{chickpea}} + 0.4 \cdot w_{\text{oil}} \leq 0, \\
& - w_{\text{tomato}} \leq 0 \\
& - w_{\text{lettuce}} \leq 0 \\
& - w_{\text{spinach}} \leq 0 \\
& - w_{\text{carrot}} \leq 0 \\
& - w_{\text{sunflower seed}} \leq 0 \\
& - w_{\text{smoked tofu}} \leq 0 \\
& - w_{\text{chickpea}} \leq 0 \\
& - w_{\text{oil}} \leq 0
\end{aligned}
$$

where the w parameters are the weight of each ingredient, in 100's of grams. The equations, from top to bottom, are:

1) Minimize total cost
2) Subject to total protein $\geq$ 15 grams
3) Subject to total fat $\geq$ 2 grams
4) Subject to total fat $\leq$ 8 grams
5) Subject to total carbohydrates $\geq$ 4 grams
6) Subject to total sodium $\leq$ 200 milligrams
7) Subject to total leafy green mass $\geq$ 40% of total mass
8) Subject to individual ingredient weights $\geq$ 0

ii. Determine the optimal solution for the linear program using any software you want. Include a copy of the code/file in the report.

The following MATLAB code was used to generate the solution:

```
f = [1, 0.75, 0.5, 0.5, 0.45, 2.15, 0.95, 2.0];

A = [-0.85, -1.62, -2.86, -0.93, -23.4, -16, -9, 0;
    -0.33, -0.2, -0.39, -0.24, -48.7, -5, -2.6, -100;
    0.33, 0.2, 0.39, 0.24, 48.7, 5, 2.6, 100;
    -4.64, -2.37, -3.63, -9.58, -15, -3, -27, 0;
    9, 28, 65, 69, 3.8, 120, 78, 0;
    0.4, -0.6, -0.6, 0.4, 0.4, 0.4, 0.4, 0.4;
    -eye(8)];

b = [-15; -2; 8; -4; 200;  0; zeros(8,1)];

[X,FVAL,EXITFLAG] = linprog(f,A,b);
```

where `X` stores the resulting weights of ingredients (in 100's of grams), `FVAL` stores the minimized number of calories, and `EXITFLAG` stores the status of the `linprog` optimization.

iii. How many calories are in the low cost salad?

The optimal low cost salad contains the following weights of ingredients (in 100's of grams):

$$
\begin{aligned}
w_{\text{tomato}} &\approx 0 \\
w_{\text{lettuce}} &\approx 0 \\
w_{\text{spinach}} &\approx 0.8323 \\
w_{\text{carrot}} &\approx 0 \\
w_{\text{sunflower seed}} &\approx 0.0961 \\
w_{\text{smoked tofu}} &\approx 0 \\
w_{\text{chickpea}} &\approx 1.1524 \\
w_{\text{oil}} &\approx 0
\end{aligned}
$$

The optimal low cost salad costs approximately \$1.55. It contains approximately 278.49 kcal

Problem 2 Part C

Compare the results from part A and B. Veronicas goal is to create a Very Veggie Salad that is both low calorie and low cost. She would like to sell the salad for \$5.00 and still have a profit of at least \$3.00. However if she can advertise that the salad has under 250 calories then she may be able to sell more.

i. Suggest some possible ways that she select a combination of ingredients that is near optimal for both objectives. This is a type of multi-objective optimization.

   What Veronica should do is attempt to find a "pareto optimal" solution that accounts for both calories and cost. In order to do this, she should introduce a new parameter λ to her optimization formulation. In essence, she would like to:

$$\begin{array}{ll} \text{minimize} & (1-\lambda)\cdot \text{CALORIES} + \lambda \cdot \text{PRICE} \\ \text{subject to} & \text{CONSTRAINTS} \end{array}$$

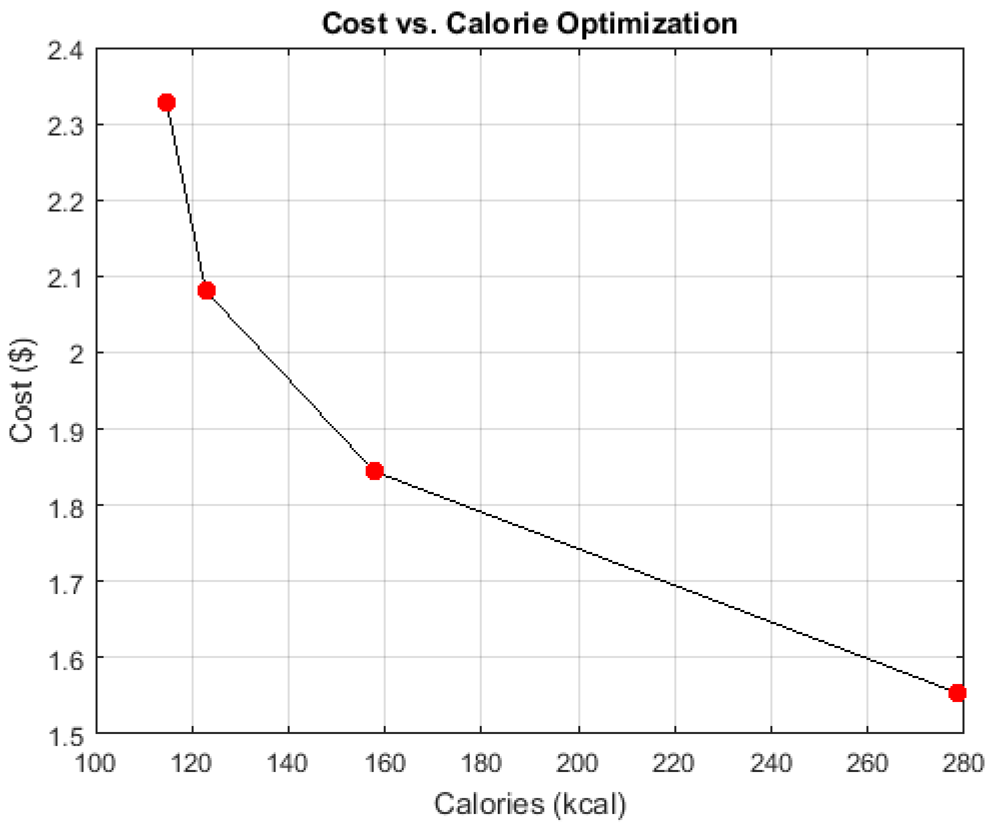
   for values of λ between 0 and 1. When $\lambda = 0$, she will be finding the minimum calorie combination of ingredients. When $\lambda = 1$, she will be finding the minimum price combination of ingredients. For all values of λ inbetween, she will be finding the "pareto optimal" combination of ingredients. So, she should solve the optimization problem for values of λ between 0 and 1. She can then examine these pareto optimal combinations of ingredients to determine which would best meet her goals of \$3.00 profit and under 250 calories.

   We can express this as the following linear program:

$$
\begin{aligned}
\text{minimize} \quad & (1-\lambda)\cdot(21\cdot w_{\text{tomato}} + 16\cdot w_{\text{lettuce}} + 40\cdot w_{\text{spinach}} + 41\cdot w_{\text{carrot}} \\
& + 585\cdot w_{\text{sunflower seed}} + 120\cdot w_{\text{smoked tofu}} + 164\cdot w_{\text{chickpea}} + 884\cdot w_{\text{oil}}) \\
& + \lambda\cdot(1\cdot w_{\text{tomato}} + 0.75\cdot w_{\text{lettuce}} + 0.5\cdot w_{\text{spinach}} + 0.5\cdot w_{\text{carrot}} \\
& + 0.45\cdot w_{\text{sunflower seed}} + 2.15\cdot w_{\text{smoked tofu}} + 0.95\cdot w_{\text{chickpea}} + 2.00\cdot w_{\text{oil}}) \\
\\
\text{subject to} \quad & -0.85\cdot w_{\text{tomato}} - 1.62\cdot w_{\text{lettuce}} - 2.86\cdot w_{\text{spinach}} - 0.93\cdot w_{\text{carrot}} \\
& - 23.4\cdot w_{\text{sunflower seed}} - 16\cdot w_{\text{smoked tofu}} - 9\cdot w_{\text{chickpea}} - 0\cdot w_{\text{oil}} \leq -15, \\
\\
& -0.33\cdot w_{\text{tomato}} - 0.2\cdot w_{\text{lettuce}} - 0.39\cdot w_{\text{spinach}} - 0.24\cdot w_{\text{carrot}} \\
& - 48.7\cdot w_{\text{sunflower seed}} - 5\cdot w_{\text{smoked tofu}} - 2.6\cdot w_{\text{chickpea}} - 100\cdot w_{\text{oil}} \leq -2, \\
\\
& 0.33\cdot w_{\text{tomato}} + 0.2\cdot w_{\text{lettuce}} + 0.39\cdot w_{\text{spinach}} + 0.24\cdot w_{\text{carrot}} \\
& + 48.7\cdot w_{\text{sunflower seed}} + 5\cdot w_{\text{smoked tofu}} + 2.6\cdot w_{\text{chickpea}} + 100\cdot w_{\text{oil}} \leq 8, \\
\\
& -4.64\cdot w_{\text{tomato}} - 2.37\cdot w_{\text{lettuce}} - 3.63\cdot w_{\text{spinach}} - 9.58\cdot w_{\text{carrot}} \\
& - 15\cdot w_{\text{sunflower seed}} - 3\cdot w_{\text{smoked tofu}} - 27\cdot w_{\text{chickpea}} - 0\cdot w_{\text{oil}} \leq -4, \\
\\
& 9\cdot w_{\text{tomato}} + 28\cdot w_{\text{lettuce}} + 65\cdot w_{\text{spinach}} + 69\cdot w_{\text{carrot}} \\
& + 3.8\cdot w_{\text{sunflower seed}} + 120\cdot w_{\text{smoked tofu}} + 78\cdot w_{\text{chickpea}} + 0\cdot w_{\text{oil}} \leq 200, \\
\\
& 0.4\cdot w_{\text{tomato}} - 0.6\cdot w_{\text{lettuce}} - 0.6\cdot w_{\text{spinach}} + 0.4\cdot w_{\text{carrot}} \\
& + 0.4\cdot w_{\text{sunflower seed}} + 0.4\cdot w_{\text{smoked tofu}} + 0.4\cdot w_{\text{chickpea}} + 0.4\cdot w_{\text{oil}} \leq 0, \\
\\
& -w_{\text{tomato}} \leq 0 \\
& -w_{\text{lettuce}} \leq 0 \\
& -w_{\text{spinach}} \leq 0 \\
& -w_{\text{carrot}} \leq 0 \\
& -w_{\text{sunflower seed}} \leq 0 \\
& -w_{\text{smoked tofu}} \leq 0 \\
& -w_{\text{chickpea}} \leq 0 \\
& -w_{\text{oil}} \leq 0
\end{aligned}
$$

ii. What combination of ingredient would you suggest and what is the associated cost and calorie.

Varying λ between 0 and 1, and solving the resulting optimization problems, we obtain the following pareto optimal combinations of total calories and total price:



Based on this result, I would suggest the following combination of ingredients:

$$\begin{aligned}
w_{\text{tomato}} &\approx 0\\
w_{\text{lettuce}} &\approx 0\\
w_{\text{spinach}} &\approx 0.5346\\
w_{\text{carrot}} &\approx 0\\
w_{\text{sunflower seed}} &\approx 0.0865\\
w_{\text{smoked tofu}} &\approx 0.7154\\
w_{\text{chickpea}} &\approx 0\\
w_{\text{oil}} &\approx 0
\end{aligned}$$

This pareto optimal salad costs approximately $1.84. It contains approximately 157.86 kcal. This meets both of Veronicas goals.

iii. Note: There is not one right answer. Discuss how you derived your solution.

The following MATLAB code was used to derive the answer, based on the discussion above.

```
lambda = 0:0.0005:1;

for j = 1:numel(lambda)
    L = lambda(j);
    f1 = [21, 16, 40, 41, 585, 120, 164, 884];
    f2 = [1, 0.75, 0.5, 0.5, 0.45, 2.15, 0.95, 2.0];

    f = (1 - L)*f1 + L*f2;

    A = [-0.85, -1.62, -2.86, -0.93, -23.4, -16, -9, 0;
        -0.33, -0.2, -0.39, -0.24, -48.7, -5, -2.6, -100;
        0.33, 0.2, 0.39, 0.24, 48.7, 5, 2.6, 100;
        -4.64, -2.37, -3.63, -9.58, -15, -3, -27, 0;
        9, 28, 65, 69, 3.8, 120, 78, 0;
        0.4, -0.6, -0.6, 0.4, 0.4, 0.4, 0.4, 0.4;
        -eye(8)];

    b = [-15; -2; 8; -4; 200;  0; zeros(8,1)];

    [X,FVAL,EXITFLAG] = linprog(f,A,b);

    calories_per_ingredient = [21, 16, 40, 41, 585, 120, 164, 884];
    cost_per_ingredient = [1, 0.75, 0.5, 0.5, 0.45, 2.15, 0.95, 2];

    if (EXITFLAG == 1)
        cost(j) = cost_per_ingredient * X;
        calories(j) = calories_per_ingredient * X;
    else
        cost(j) = NaN;
        calories(j) = NaN;
    end
end

figure(1)
plot(calories, cost, 'k-')
hold on
plot(calories, cost, 'r.', 'markersize', 25)
grid on
xlabel('Calories (kcal)')
ylabel('Cost ($)')
title('Cost vs. Calorie Optimization')
```

Problem 3

a) What are the lengths of the shortest paths from vertex a to all other vertices?

The shortest path problem can be solved using the following linear programming formulation:

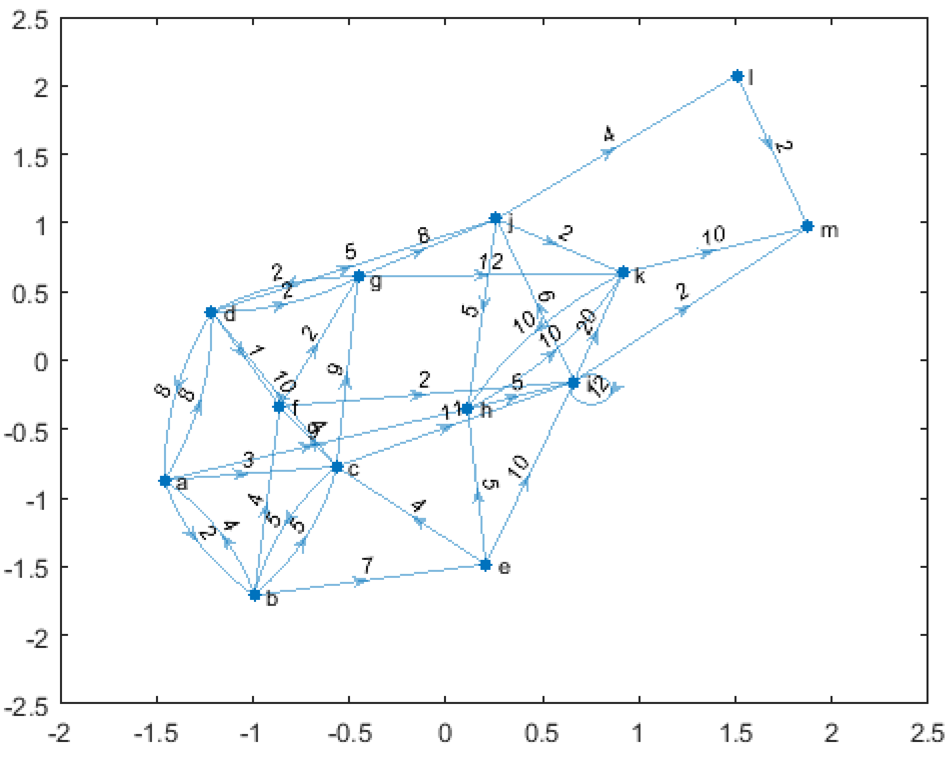
$$\begin{aligned} \text{minimize} \quad & -d_t \\ \text{subject to} \quad & d_s = 0 \\ & d_v - d_u \leq \ell(uv) \quad \forall \text{ edges } uv \\ & -d_u \leq 0 \quad \forall \text{ vertices } u \in V \end{aligned}$$

Based on the vertices and weights provided in the Project3Problem3-1.txt file, the constraints for the linear programming formulation to find shortest paths will be:

- $d_b - d_a \leq 2$
- $d_c - d_a \leq 3$
- $d_d - d_a \leq 8$
- $d_h - d_a \leq 9$
- $d_a - d_b \leq 4$
- $d_c - d_b \leq 5$
- $d_e - d_b \leq 7$
- $d_f - d_b \leq 4$
- $d_d - d_c \leq 10$
- $d_b - d_c \leq 5$
- $d_g - d_c \leq 9$
- $d_i - d_c \leq 11$
- $d_f - d_c \leq 4$
- $d_a - d_d \leq 8$
- $d_g - d_d \leq 2$
- $d_j - d_d \leq 5$
- $d_f - d_d \leq 1$
- $d_h - d_e \leq 5$
- $d_c - d_e \leq 4$
- $d_i - d_e \leq 10$
- $d_i - d_f \leq 2$
- $d_g - d_f \leq 2$
- $d_d - d_g \leq 2$

- $d_j - d_g \leq 8$
- $d_k - d_g \leq 12$
- $d_i - d_h \leq 5$
- $d_k - d_h \leq 10$
- $d_a - d_i \leq 20$
- $d_k - d_i \leq 6$
- $d_j - d_i \leq 2$
- $d_m - d_i \leq 12$
- $d_i - d_j \leq 2$
- $d_k - d_j \leq 4$
- $d_l - d_j \leq 5$
- $d_h - d_k \leq 10$
- $d_m - d_k \leq 10$
- $d_m - d_l \leq 2$

An image of the digraph is below:



The MATLAB function `linprog` was utilized to find the solution. The following are the lengths of the shortest paths from a to each of the other vertices in the graph.

```
----- P3.A SOLUTION -----
Distance from a to a = -0
Distance from a to b =  2
Distance from a to c =  3
Distance from a to d =  8
Distance from a to e =  9
Distance from a to f =  6
Distance from a to g =  8
Distance from a to h =  9
Distance from a to i =  8
Distance from a to j = 10
Distance from a to k = 14
Distance from a to l = 15
Distance from a to m = 17
```

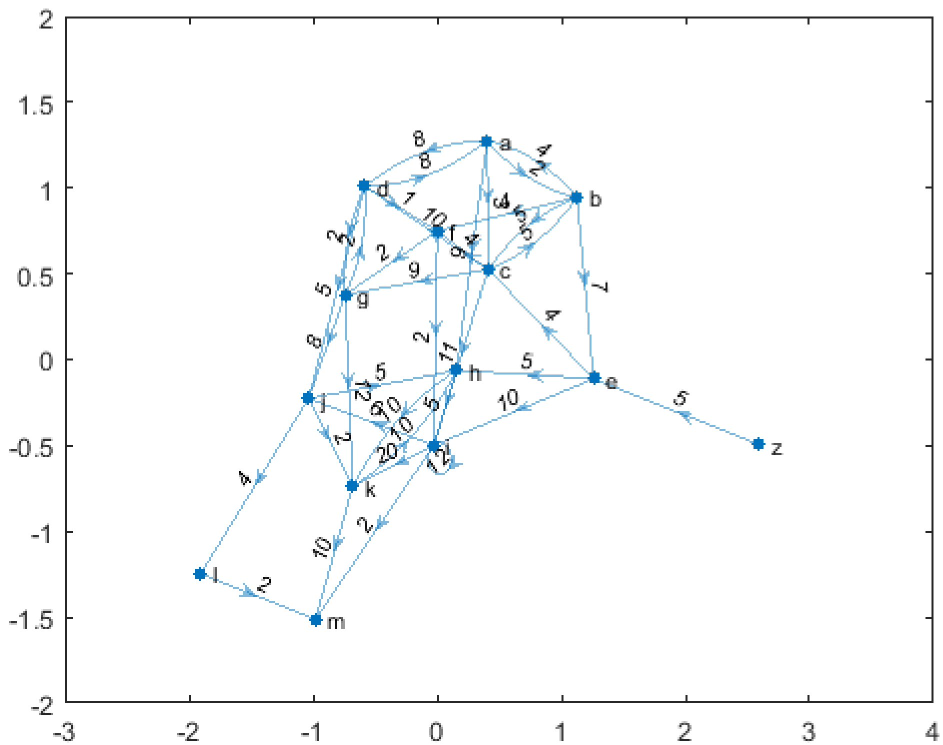
The following MATLAB code was used to generate the solution.

```
clear variables
close all
clc

%% PARSE INPUT DATA
% Open file
fid = fopen('p3_input.txt');

% Read line while there is data remaining
tline = fgets(fid);
rowIdx = 0;
while ischar(tline)
    % Increment row counter to store data
    rowIdx = rowIdx + 1;

    % fgetl reads line in as char array -- split on spaces
    C = strsplit(tline, ' ');

    % Convert node letter to index using ASCII codes (a = 1, b = 2, ...)
    edgeStart(rowIdx) = double(C{1}) - double('a') + 1;
    edgeEnd(rowIdx) = double(C{2}) - double('a') + 1;
    edgeWeight(rowIdx) = str2num(C{3});
    % Grab the next line
    tline = fgetl(fid);
end

fclose(fid);

%% SHORTEST PATH FROM A TO OTHER VERTICES

% Number of nodes is highest numbered node in our data
numberOfNodes = max([edgeStart, edgeEnd]);

% Build A and b matrices from edgeEnd and edgeStart
% Size of A is number of inequalities by number of nodes
A = zeros(numel(edgeWeight), numberOfNodes);
```

```matlab

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

b = edgeWeight';

% Additional constriant that all distances must be greater than 0
A = [A; -eye(numberOfNodes)];

b = [b; zeros(numberOfNodes, 1)];

% Single equality constraint -- distance to a must be zero, since we start at a
Aeq = zeros(1, numberOfNodes);
Aeq(1, 1) = 1;
beq = 0;

% Minimization constriant is to maximimize negative sum of distances
f = -ones(numberOfNodes, 1);

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

fid = fopen('P3A_solution.txt','w');

fprintf(fid, '----- P3.A SOLUTION -----\n');
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) If a vertex z is added to the graph for which there is no path from vertex a to vertex z, what will be the result when you attempt to find the lengths of shortest paths as in part a).

Consider the following possible digraph with unreachable z:



The solution code is the same as was presented in part (a). When the code is run, the following error message is presented:

```
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 is saying that the optimization function has failed to converge on a solution because the new vertex z is unreachable. It is not feasible to find the shortest path to an unreachable vertex.

There are two possible work-arounds for this problem. The first step in either method is to identify the vertex that the optimization routine is having difficulties with. You can determine this by examining the preliminary values for the distance to each vertex. The unreachable vertices will have very large values.

The first approach is to remove the unreachable vertices from your optimization function. Suppose you had vertices a, b, c, and d, with vertex d being unreachable from a. You could simply change your optimization statement from:

$$\text{minimize} \quad -d_a - d_b - d_c - d_d$$

to the form:

$$\text{minimize} \quad -d_a - d_b - d_c$$

This allows the optimization routine to find a feasible solution.

Another alernative approach is to specify the distance of the unreachable vertices to be a very large value (large being relative to the maximum path length in the graph) using an equality constraint. By fixing the value, you essentially remove it from consideration by the optimization routine. By giving it a large value, you ensure that it will not conflict in any way with the optimal result. In post processing of the results, you would note that the fixed value you provided denoted that the vertex was unreachable.

Following either of these methods would then yield the same result as in part (a).

c) What are the lengths of the shortest paths from each vertex to vertex m? How can you solve this problem with just one linear program?

Finding the shortest path from each vertex to the vertex m is equivalent to flipping the directionality of each edge, and then finding the shortest path from m to each vertex. The resulting path lengths are equivalent to the distance from a given vertex to m with the original edge directions.

The MATLAB function `linprog` was utilized to find the solution. The following are the lengths of the shortest paths from each vertex to m.

```
----- P3.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
```

The following MATLAB code was used to generate the solution.

```
clear variables
close all
clc

%% PARSE INPUT DATA
% Open file
fid = fopen('p3_input.txt');

% Read line while there is data remaining
tline = fgets(fid);
rowIdx = 0;
while ischar(tline)
    % Increment row counter to store data
    rowIdx = rowIdx + 1;

    % fgetl reads line in as char array -- split on spaces
    C = strsplit(tline, ' ');

    % Convert node letter to index using ASCII codes (a = 1, b = 2, ...)
    edgeStart(rowIdx) = double(C{1}) - double('a') + 1;
    edgeEnd(rowIdx) = double(C{2}) - double('a') + 1;
    edgeWeight(rowIdx) = str2num(C{3});
    % Grab the next line
```

```
    tline = fgetl(fid);
end

fclose(fid);

%% SHORTEST PATH FROM OTHER VERTICES TO M
% Going from other vertices to M is equivalent to reversing the edge
% directions and going from M to other vertcies.

% Number of nodes is highest numbered node in our data
numberOfNodes = max([edgeStart, edgeEnd]);

% Build A and b matrices from edgeEnd and edgeStart
% Size of A is number of inequalities by number of nodes
A = zeros(numel(edgeWeight), numberOfNodes);

% Since we're reversing the direction of the edges, we'll swap the +1 and
% -1 values to capture that.
for j = 1:numel(edgeWeight)
    A(j, edgeStart(j)) = 1;
    A(j, edgeEnd(j)) = -1;
end

b = edgeWeight';

% Additional constriant that all distances must be greater than 0
A = [A; -eye(numberOfNodes)];

b = [b; zeros(numberOfNodes, 1)];

% Single equality constraint -- distance to m must be zero, since we start
% at m
Aeq = zeros(1, numberOfNodes);
startNode = double('m') - double('a') + 1;
Aeq(1, startNode) = 1;
beq = 0;

% Minimization constriant is to maximimize negative sum of distances
f = -ones(numberOfNodes, 1);

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

fid = fopen('P3C_solution.txt', 'w');

fprintf(fid, '----- P3.C SOLUTION -----\n');
for j = 1:numel(x)
    fprintf(fid, 'Distance from %c to m = %2.0f \n', char('a'+ j - 1), x(j));
end

fclose(fid);
```

d) Suppose that all paths must pass through vertex i. How can you calculate the length of the shortest path from any vertex x to vertex y that pass through vertex i (for all $x, y \in V$)? Calculate the lengths of these paths for the given graph. (Note: for some vertices x and y, it may be impossible to pass through vertex i).

The length of the shortest path from any vertex x to any vertex y that passes through the vertex i is equivalent to:

$$(\text{Shortest path from } x \text{ to } i) + (\text{Shortest path from } i \text{ to } y)$$

The first term in this equation is equivalent to part (c) of this problem - find the shortest path from each vertex to a given vertex. The second term in this equation is equivalent to part (a) of this problem - find the shortest path from a given vertex to each other vertex.

In the first portion, there are two vertices that are unable to reach vertex i. Looking back at the graphical representation of the digraph in part (a), it's easy to see that these vertices are l and m. l is only able to reach vertex m, and vertex m is unable to reach any other vertex. In the second portion, all vertices are accessible from vertex i.

The MATLAB function `linprog` was utilized to find the solution for each aspect of the problem (distances to i, and distances from i). The following are the lengths of the shortest paths from each vertex to every other vertex, passing through vertex i. A distance of `NaN` denotes that there is not a path between the vertices that travels through vertex i.

```
----- P3.D SOLUTION -----
Distance from a to a through i = 28
Distance from a to b through i = 30
Distance from a to c through i = 31
Distance from a to d through i = 36
Distance from a to e through i = 37
Distance from a to f through i = 34
Distance from a to g through i = 36
Distance from a to h through i = 24
Distance from a to i through i =  8
Distance from a to j through i = 10
Distance from a to k through i = 14
Distance from a to l through i = 15
Distance from a to m through i = 17
Distance from b to a through i = 26
Distance from b to b through i = 28
Distance from b to c through i = 29
Distance from b to d through i = 34
Distance from b to e through i = 35
Distance from b to f through i = 32
Distance from b to g through i = 34
Distance from b to h through i = 22
Distance from b to i through i =  6
Distance from b to j through i =  8
```

```
Distance from b to k through i = 12
Distance from b to l through i = 13
Distance from b to m through i = 15
Distance from c to a through i = 26
Distance from c to b through i = 28
Distance from c to c through i = 29
Distance from c to d through i = 34
Distance from c to e through i = 35
Distance from c to f through i = 32
Distance from c to g through i = 34
Distance from c to h through i = 22
Distance from c to i through i =  6
Distance from c to j through i =  8
Distance from c to k through i = 12
Distance from c to l through i = 13
Distance from c to m through i = 15
Distance from d to a through i = 23
Distance from d to b through i = 25
Distance from d to c through i = 26
Distance from d to d through i = 31
Distance from d to e through i = 32
Distance from d to f through i = 29
Distance from d to g through i = 31
Distance from d to h through i = 19
Distance from d to i through i =  3
Distance from d to j through i =  5
Distance from d to k through i =  9
Distance from d to l through i = 10
Distance from d to m through i = 12
Distance from e to a through i = 30
Distance from e to b through i = 32
Distance from e to c through i = 33
Distance from e to d through i = 38
Distance from e to e through i = 39
Distance from e to f through i = 36
Distance from e to g through i = 38
Distance from e to h through i = 26
Distance from e to i through i = 10
Distance from e to j through i = 12
Distance from e to k through i = 16
Distance from e to l through i = 17
Distance from e to m through i = 19
Distance from f to a through i = 22
Distance from f to b through i = 24
Distance from f to c through i = 25
Distance from f to d through i = 30
Distance from f to e through i = 31
Distance from f to f through i = 28
Distance from f to g through i = 30
Distance from f to h through i = 18
Distance from f to i through i =  2
Distance from f to j through i =  4
Distance from f to k through i =  8
Distance from f to l through i =  9
```

```
Distance from f to m through i = 11
Distance from g to a through i = 25
Distance from g to b through i = 27
Distance from g to c through i = 28
Distance from g to d through i = 33
Distance from g to e through i = 34
Distance from g to f through i = 31
Distance from g to g through i = 33
Distance from g to h through i = 21
Distance from g to i through i =  5
Distance from g to j through i =  7
Distance from g to k through i = 11
Distance from g to l through i = 12
Distance from g to m through i = 14
Distance from h to a through i = 25
Distance from h to b through i = 27
Distance from h to c through i = 28
Distance from h to d through i = 33
Distance from h to e through i = 34
Distance from h to f through i = 31
Distance from h to g through i = 33
Distance from h to h through i = 21
Distance from h to i through i =  5
Distance from h to j through i =  7
Distance from h to k through i = 11
Distance from h to l through i = 12
Distance from h to m through i = 14
Distance from i to a through i = 20
Distance from i to b through i = 22
Distance from i to c through i = 23
Distance from i to d through i = 28
Distance from i to e through i = 29
Distance from i to f through i = 26
Distance from i to g through i = 28
Distance from i to h through i = 16
Distance from i to i through i =  0
Distance from i to j through i =  2
Distance from i to k through i =  6
Distance from i to l through i =  7
Distance from i to m through i =  9
Distance from j to a through i = 22
Distance from j to b through i = 24
Distance from j to c through i = 25
Distance from j to d through i = 30
Distance from j to e through i = 31
Distance from j to f through i = 28
Distance from j to g through i = 30
Distance from j to h through i = 18
Distance from j to i through i =  2
Distance from j to j through i =  4
Distance from j to k through i =  8
Distance from j to l through i =  9
Distance from j to m through i = 11
Distance from k to a through i = 35
```

```
Distance from k to b through i = 37
Distance from k to c through i = 38
Distance from k to d through i = 43
Distance from k to e through i = 44
Distance from k to f through i = 41
Distance from k to g through i = 43
Distance from k to h through i = 31
Distance from k to i through i = 15
Distance from k to j through i = 17
Distance from k to k through i = 21
Distance from k to l through i = 22
Distance from k to m through i = 24
Distance from l to a through i = NaN
Distance from l to b through i = NaN
Distance from l to c through i = NaN
Distance from l to d through i = NaN
Distance from l to e through i = NaN
Distance from l to f through i = NaN
Distance from l to g through i = NaN
Distance from l to h through i = NaN
Distance from l to i through i = NaN
Distance from l to j through i = NaN
Distance from l to k through i = NaN
Distance from l to l through i = NaN
Distance from l to m through i = NaN
Distance from m to a through i = NaN
Distance from m to b through i = NaN
Distance from m to c through i = NaN
Distance from m to d through i = NaN
Distance from m to e through i = NaN
Distance from m to f through i = NaN
Distance from m to g through i = NaN
Distance from m to h through i = NaN
Distance from m to i through i = NaN
Distance from m to j through i = NaN
Distance from m to k through i = NaN
Distance from m to l through i = NaN
Distance from m to m through i = NaN
```

The following MATLAB code was used to generate the solution.

```
clear variables
close all
clc

%% PARSE INPUT DATA
% Open file
fid = fopen('p3_input.txt');

% Read line while there is data remaining
tline = fgets(fid);
rowIdx = 0;
while ischar(tline)
```

```
    % Increment row counter to store data
    rowIdx = rowIdx + 1;

    % fgetl reads line in as char array -- split on spaces
    C = strsplit(tline, ' ');

    % Convert node letter to index using ASCII codes (a = 1, b = 2, ...)
    edgeStart(rowIdx) = double(C{1}) - double('a') + 1;
    edgeEnd(rowIdx) = double(C{2}) - double('a') + 1;
    edgeWeight(rowIdx) = str2num(C{3});
    % Grab the next line
    tline = fgetl(fid);
end

fclose(fid);

%% SHORTEST PATH FROM OTHER VERTICES TO I
% Going from other vertices to I is equivalent to reversing the edge
% directions and going from I to other vertcies.

% Number of nodes is highest numbered node in our data
numberOfNodes = max([edgeStart, edgeEnd]);

% Build A and b matrices from edgeEnd and edgeStart
% Size of A is number of inequalities by number of nodes
A = zeros(numel(edgeWeight), numberOfNodes);

% Since we're reversing the direction of the edges, we'll swap the +1 and
% -1 values to capture that.
for j = 1:numel(edgeWeight)
    A(j, edgeStart(j)) = 1;
    A(j, edgeEnd(j)) = -1;
end

b = edgeWeight';

% Additional constriant that all distances must be greater than 0
A = [A; -eye(numberOfNodes)];

b = [b; zeros(numberOfNodes, 1)];

% Single equality constraint -- distance to i must be zero, since we start
% at i
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];

% Minimization constriant is to maximimize negative sum of distances
f = -ones(numberOfNodes, 1);
```

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

fprintf('----- P3.C SOLUTION -----\n')
for j = 1:numel(x)
    fprintf('Distance from %c to i = %2.0f \n', char('a'+ j - 1), x(j))
end

distanceToNodeI = x;

%% SHORTEST PATH FROM I TO OTHER VERTICES

% Number of nodes is highest numbered node in our data
numberOfNodes = max([edgeStart, edgeEnd]);

% Build A and b matrices from edgeEnd and edgeStart
% Size of A is number of inequalities by number 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';

% Additional constriant that all distances must be greater than 0
A = [A; -eye(numberOfNodes)];

b = [b; zeros(numberOfNodes, 1)];

% Single equality constraint -- distance to i must be zero, since we start
% at i
% Aeq = zeros(3, numberOfNodes);
Aeq = zeros(1, 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];
beq = 0;

% Minimization constriant is to maximimize negative sum of distances
f = -ones(numberOfNodes, 1);

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

fprintf('----- P3.C SOLUTION -----\n')
for j = 1:numel(x)
    fprintf('Distance from i to %c = %2.0f \n', char('a'+ j - 1), x(j))
end
```

```
distanceFromNodeI = x;

%% COMBINE THE RESULTS:
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

fid = fopen('P3D_solution.txt','w');
fprintf(fid, '----- P3.D SOLUTION -----\n');
for i = 1:numberOfNodes
    for j = 1:numberOfNodes
        fprintf(fid, 'Distance from %c to %c through i = %2.0f \n', ...
            char('a' + i - 1), char('a'+ j - 1), distFromTo(i,j));
    end
end

fclose(fid);
```