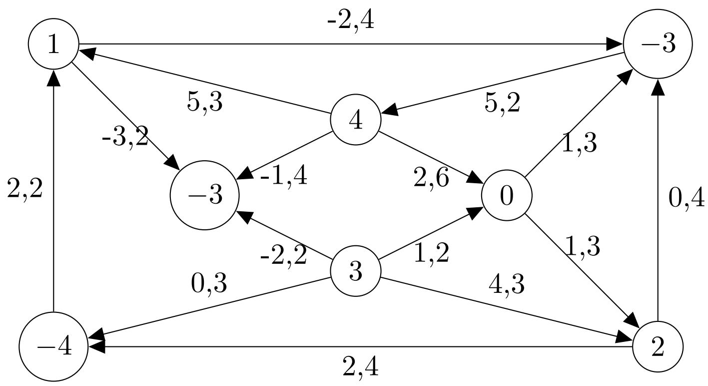
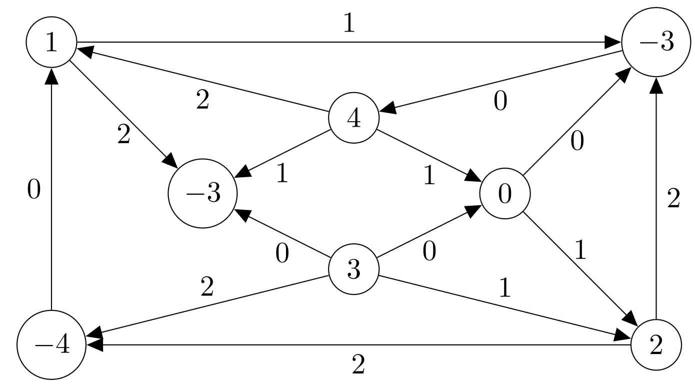
Caleb Logemann
MATH 566 Discrete Optimization
Homework 8

Consider the following network M with costs and capacities depicted on edges and boundary in vertices.



1. Consider the following b-flow f in M.



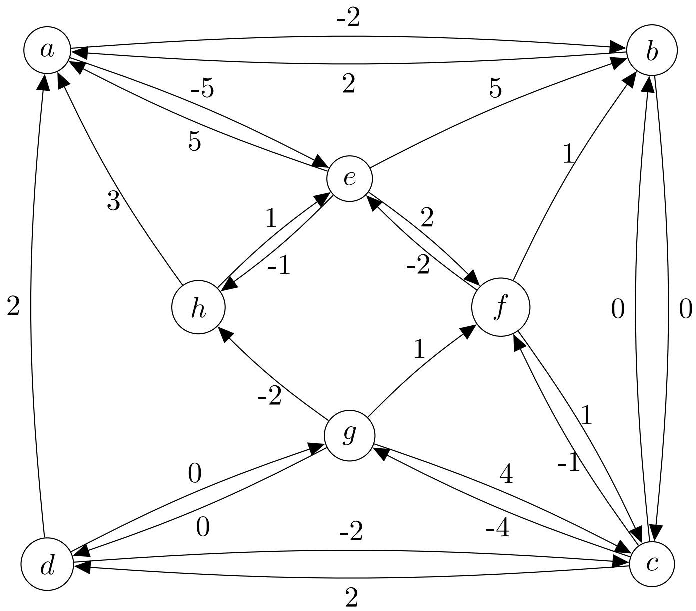
Compute the cost of f. Start computing the minimum cost b-flow by finding a sequence of augmenting cycles starting from f.

First I will compute the cost of the given flow f.

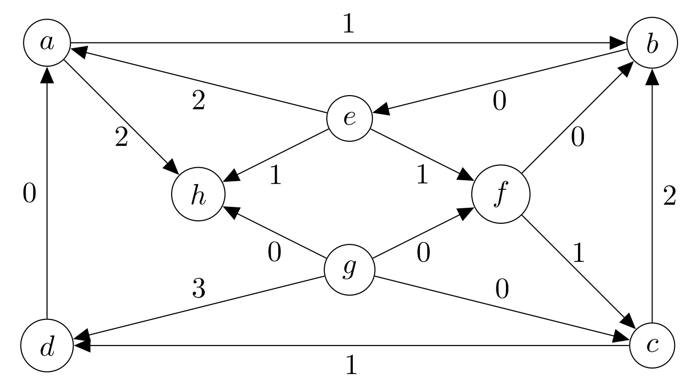
$$1 \times -2 + 2 \times 5 + 2 \times -3 + 1 \times -1 + 1 \times 2 + 1 \times 1 + 1 \times 4 + 2 \times 2 + 2 \times 0 + 2 \times 0 = 12$$

The cost of this flow is 12.

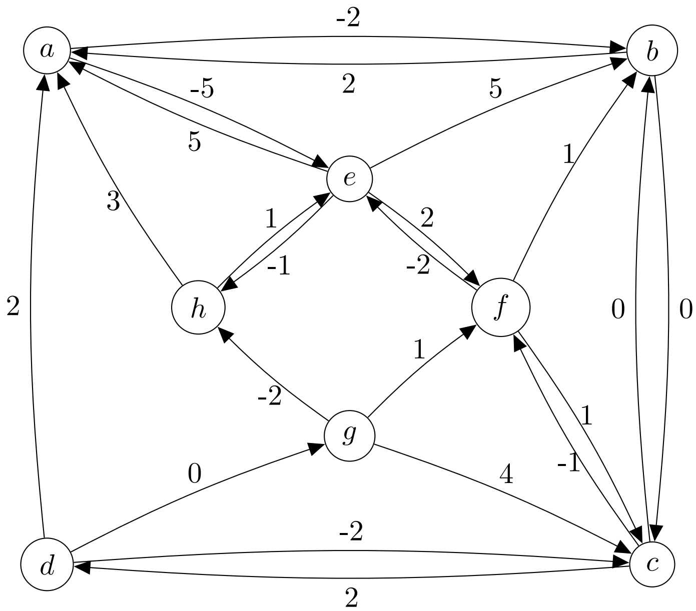
Now I will do 2 augmentations of this flow, first the residual graph must be computed, I relabeled the vertices while doing this. The boundaries at each vertex are not important for this algorithm as the augmentations preserve this property.



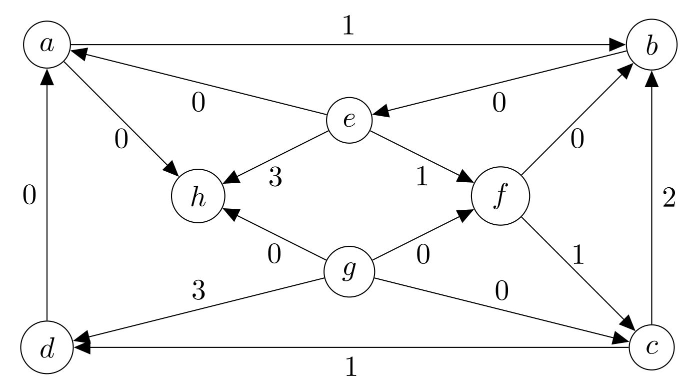
The first negative cycle that I will augment along the cycle $c \to d \to g \to c$, the value of this cycle is -6. The minimum available capacity along this cycle is 1 along the $g \to d$ edge. The new flow is shown below, and the value of this new flow is 6.



The new residual graph is shown below.



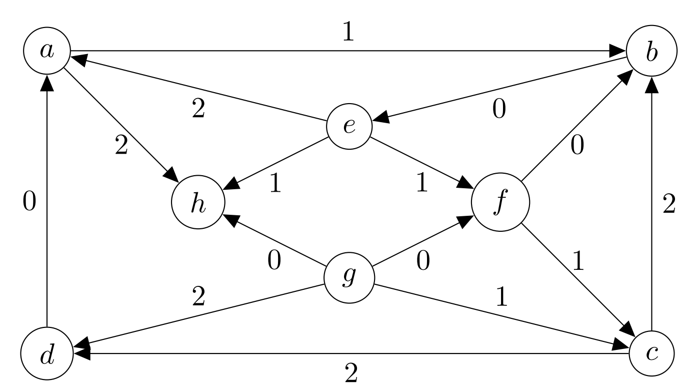
The next cycle I will augment on is $a \to e \to h \to a$, which has value -3. The minimum capacity on this cycle is 2. The new flow will have value 0, and is shown below.



From problem 2 we know that this is the optimal min flow on this network.

2. Solve minimum cost b-flow for M using linear programming. That is, formulate the problem using linear programming and solve it using Sage or APMonitor. Then draw the resulting network.

   First I will relabel the nodes with letters in order to better describe the flow.



In order to solve the min flow problem we can consider the flow on each edge, which I will denote $f(e)$ for each edge in $E(G)$. Let $c(e)$ be the cost of edge e, $u(e)$ be the capacity of each edge, and $b(v)$ be the boundary at vertex v. Any solution needs to satisfy two types of constraints. First the flow must be greater than zero and less than the capacity, symbolically this can be expressed as

$$0 \leq f(e) \leq u(e)$$

for all $e \in E(G)$. Second the difference between the flow out and the flow in of each vertex must be the boundary of each vertex, symbolically this can expressed as

$$\sum_{(u,v)\in E(G)} f((u,v)) - \sum_{(v,u)\in E(G)} f((v,u)) = b(v)$$

for each vertex v in $V(G)$.

Also the objective function for this linear program is minimizing the flow times the cost of each edge. Therefore the entire linear program is

$$(P) \begin{cases} \text{minimize} & \sum_{e\in E(G)} c(e)f(e) \\ \text{subject to} & f(e) \leq u(e) \quad \forall e \in E(G) \\ & \sum_{e=(u,v)\in E(G)} f(e) - \sum_{e=(v,u)\in E(G)} f(e) = b(v) \quad \forall v \in V(G) \\ & f(e) \geq 0 \quad \forall e \in E(G) \end{cases}$$

This linear program can be solved using the following Sage script.

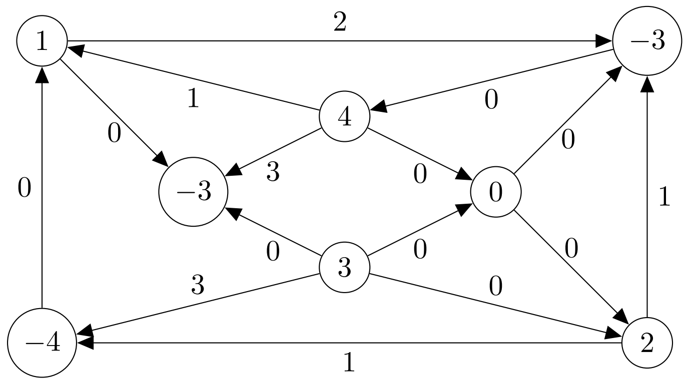
```
import operator
#                       a     b     c     d     e     f     g     h
costs = Matrix([[0,   -2,    0,    0,    0,    0,    0,   -3],  #a
                [0,    0,    0,    0,    5,    0,    0,    0],  #b
                [0,    0,    0,    2,    0,    0,    0,    0],  #c
                [2,    0,    0,    0,    0,    0,    0,    0],  #d
                [5,    0,    0,    0,    0,    2,    0,   -1],  #e
                [0,    1,    1,    0,    0,    0,    0,    0],  #f
                [0,    0,    4,    0,    0,    1,    0,   -2],  #g
                [0,    0,    0,    0,    0,    0,    0,    0]]) #h

#                             a  b  c  d  e  f  g  h
capacities = Matrix([[0,  4, 0, 0, 0, 0, 0, 2],  #a
                     [0,  0, 0, 0, 2, 0, 0, 0],  #b
                     [0,  4, 0, 4, 0, 0, 0, 0],  #c
                     [2,  0, 0, 0, 0, 0, 0, 0],  #d
                     [3,  0, 0, 0, 0, 6, 0, 4],  #e
                     [0,  3, 3, 0, 0, 0, 0, 0],  #f
                     [0,  0, 3, 3, 0, 2, 0, 2],  #g
                     [0,  0, 0, 0, 0, 0, 0, 0]]) #h

# sources/sinks
b = {'a':1, 'b':-3, 'c':2, 'd':-4, 'e':'4', 'f':0, 'g':3, 'h':-3}
G = DiGraph(capacities, weighted=True)

indexToVertex = {0:'a', 1:'b', 2:'c', 3:'d', 4:'e', 5:'f', 6:'g', 7:'h'}
vertexToIndex = {v: k for k, v in indexToVertex.iteritems()}
```

```
G.relabel(indexToVertex)
milp = MixedIntegerLinearProgram(maximization=False)
f = milp.new_variable(nonnegative=True)

milp.set_objective(sum([costs[vertexToIndex[e[0]], vertexToIndex[e[1]]]*
    ↪ f[e[0], e[1]] for e in G.edges()]))

# Flows less than capacities
for edge in G.edges():
    # edge[2] is weight or label of edge
    milp.add_constraint(f[edge[0], edge[1]] <= edge[2])

# Flows into and out of vertices must be equal to source or sink
for vertex in G.vertices():
    flow_in = sum([f[v, vertex] for v in G.neighbors_in(vertex)])
    flow_out = sum([f[vertex, v] for v in G.neighbors_out(vertex)])
    milp.add_constraint(flow_out - flow_in == b[vertex])

print('Objective␣Value:␣{}'.format(milp.solve()))
sol = milp.get_values(f)
sol = sorted(sol.items(), key=operator.itemgetter(0))
for i, v in sol:
    print('f[%s]␣=␣%s' % (i, v))
```

The output of this script is shown below.

```
Objective Value: 0.0
f[('a', 'b')] = 2.0
f[('a', 'h')] = 0.0
f[('b', 'e')] = 0.0
f[('c', 'b')] = 1.0
f[('c', 'd')] = 1.0
f[('d', 'a')] = 0.0
f[('e', 'a')] = 1.0
f[('e', 'f')] = 0.0
f[('e', 'h')] = 3.0
f[('f', 'b')] = 0.0
f[('f', 'c')] = 0.0
f[('g', 'c')] = 0.0
f[('g', 'd')] = 3.0
f[('g', 'f')] = 0.0
f[('g', 'h')] = 0.0
```

This shows that the minimum cost is 0 and the flow is shown below.



3. Show that the Maximum Flow Problem can be regarded as a special case of the Minimum Cost Flow problem. That is, for an instance of Maximum Flow Problem find a reformulation to Minimum Cost Flow problem whose solution can be interpreted as a solution to Maximum Flow Problem. That is, find *simple* algorithm that is solving Maximum Flow Problem and using Minimum Cost Flow as a black box subroutine once.

   Let G be a directed graph with capacity u and source vertex s and target vertex t be a max flow problem. In order to transform this into a minimum cost flow problem, I will let the cost of all edges be 0. I will let the boundaries of all vertices be 0. Also if there is an edge $t \to s$, note that the flow on this edge will not be a part of the maximal flow from s to t because it would only decrease the flow. This edge could be a part of a cycle, but if it is part of a cycle then removing the flow on this edge will increase the overall flow. Therefore this edge doesn't contribute to a maximal flow, so if it exists it can be removed from G. Now that this edge is removed in can be added into the min cost flow problem with cost -1. Therefore in minimizing the cost of the flow, this will maximize the flow over the edge $t \to s$. The value of the flow over this edge in the min flow problem is the value of the maximal flow. The flow on all of the other edges is the maximal flow on G as well.

   In summary in order to use a min cost flow algorithm to maximize the flow on G, set the boundaries of all the vertices to 0, set the costs of all edges to 0, and add an edge from t to s with cost -1. Running the min cost flow algorithm on this graph will give a flow. This flow with the $t \to s$ edge removed is a maximal flow on G.