CSCI 570 – Homework 4
Due: April 4th, 2021

Name: Rohan Mahendra Chaudhari
USC ID: 6675-5653-85
Email ID: rmchaudh@usc.edu

1. You are given the following graph G. Each edge is labelled with the capacity of that edge.

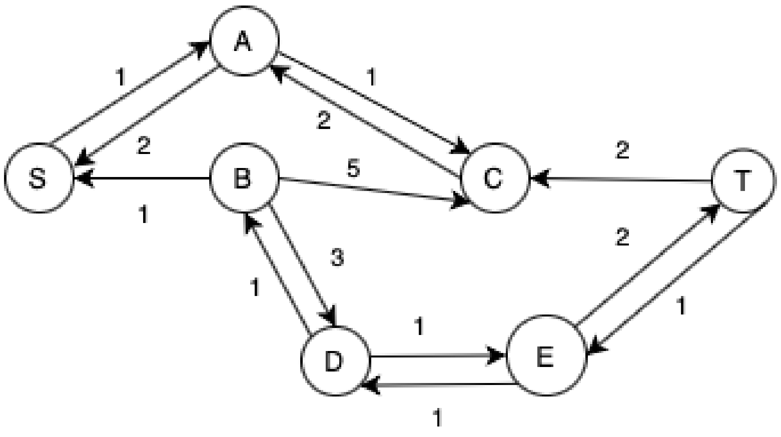
   a. Find a max-flow in G using the Ford-Fulkerson algorithm. Draw the residual graph G_f corresponding to the max flow. You do not need to show all intermediate steps. (10 pts)
   b. Find the max-flow value and a min-cut. (5 pts)

   Solution:

   a. On running Ford-Fulkerson algorithm on the above graph -
   Max flow = 3

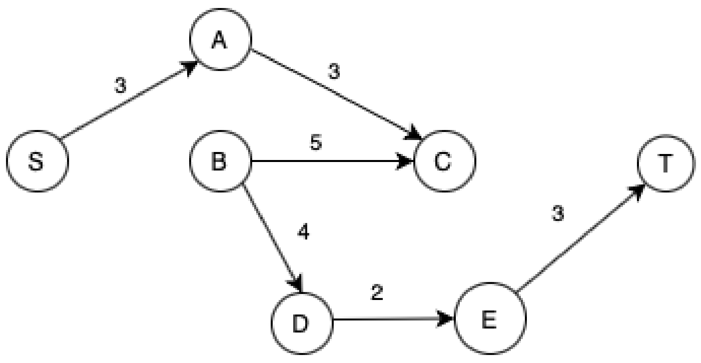
   Final residual graph (G_f):

   Taking path
   1. S → A → C → T (bottleneck - 2 : C → T)
   2. S → B → D → E → T (bottleneck – 1 : S → B)



b. Max flow value = 3
The min cut edges are: S → B and C → T. Since by taking these edges out we can note that there will be no direct path from source(S) to sink(T).

Min cut graph:



2. A group of traders are leaving Switzerland, and need to convert their Francs into various international currencies. There are n traders t_1, t_2, . . . , t_n and m currencies c_1 , c_2 , . . . , c_m . Trader t_k has F_k Francs to convert. For each currency cj, the bank can convert at most B_j Francs to c_j. Trader t_k is willing to trade as much as S_{kj} of his Francs for currency cj .(For example, a trader with 1000 Francs might be willing to convert up to 200 of his Francs for USD, up to 500 of his Francs for Japanese's Yen, and up to 200 of his Francs for Euros). Assuming that all traders give their requests to the bank at the same time, design an algorithm that the bank can use to satisfy the requests (if it is possible).

<u>Solution</u>:

This is set up as a flow network, with Francs flowing from the source 's' to the sink 'x'.
A row of vertices t_1,….,t_n represents the traders, and a row of vertices c_1,….,c_m represents the currencies held by the bank.
The edge (s, t_k) has capacity F_k which gives the amount trader t_k wants to change.
The edge (t_k, c_j) has capacity S_{kj} giving the maximum number of Francs trader t_k wants to trade into currency c_j.
Finally, the edge (c_j, x) with capacity B_j gives the limit on the amount of currency c_j that is available at the bank.

Claim: The above problem has a solution if and only if the constructed network has a max flow value of $\Sigma_k F_k$.

Proof ➔) Assume there is a solution. It means that all the traders will be able to convert their Francs to the required currency and that the bank will have will have enough currency to do so.
So we push a flow of F_k from source to t_k. On the edge between traders and currency we have a flow of S_{kj} which mentions the number of Francs trader t_k wants to trade into currency c_j. Further we have a flow of B_j which shows the amount of currency c_j that is available at the bank.

🡐) If there is a max flow 'f' in the network with $|f| = \Sigma_k F_k$ then all the traders are able to convert their currencies.

3. After getting vaccinated, students will be able to return to in-person classes next semester. There will be n students, s1, s2, ..., sn, return to in-person classes, and there will be k in-person classes. Each in-person class consists of several students and a student can be enrolled in more than one in-person class. We need to select one student from each in-person class and the maximum times a student is selected should be less than m. Design an algorithm that decides if such a selection exists, or not.

<u>Solution</u>:

We start by setting the problem as a bipartite graph problem. In this graph one partition is a set of vertices s_i, representing all n students. Another partition is a set of k classes c_j. Then we connect each student s_i to all classes c_j by directed edges with the capacity 1 since a student can only enrol in a class once.

Next, we extend the bipartite graph to a network flow. We add the source 'src' and connect it to every student vertex s_i by an edge (src,s_i) of capacity 'k' since k is the max number of classes a student can enrol in. We add the target 't' and for every class vertex c_j, we add an edge (c_j,t) of capacity as the maximum number of students that can enrol in the class which here is 'n'(max-flow of capacity).

Claim: The original problem has a solution (a valid selection is indeed possible) if and only if the constructed network has a max-flow of value $e_1+e_2+\ldots+e_n$ where e_i is the number of times a particular student has been picked and where the max value for e_i can be m-1.

Proof ➔) Assume there is a solution. It means that a student is selected from every class and that the maximum times a particular student is selected is less than m.

So we can push a flow of k from source to each student. On the edge between student and classes, we assign a flow of 1. On the edges between classes and target, we assign a flow value equal to the number of times a particular student has been selected which must be less than m. This must be possible, since we have a valid selection.

Conversely 🡸) Assume there is a max-flow of value $e_1+e_2+\ldots+e_n$. This means that each student vertex will get a flow of k. Due to capacity constraint on each edge (s_i,c_j), no student will be selected more than m - 1 times. We also observe that 1 student is selected from every class due to capacity condition n.

Lastly, we get a selection by running a network flow algorithm and pick edges (s_i,c_j) with a unit flow.

4. USC Admissions Centre needs your help in planning paths for Campus tours given to prospective students or interested groups. Let USC campus be modelled as a weighted, directed graph G containing locations V connected by one-way roads E. On a busy day, let k be the number of campus tours that have to be done at the same time. It is required that the paths of campus tours do not use the same roads. Let the tour have k starting locations $A = \{ a1, a2,..., ak \} \subset V$. From the starting locations the groups are taken by a guide on a path through G to some ending location in $B = \{ b1, b2,..., bk \} \subset V$. Your goal is to find a path for each group i from the starting location, ai , to any ending location bj such that no two paths share any edges, and no two groups end in the same location bj.

   a. Design an algorithm to find k paths $ai \rightarrow bj$ that start and end at different vertices and such that they do not share any edges. (15 pts)
   b. Modify your algorithm to find k paths $ai \rightarrow bj$, that start and end in different locations and such that they share neither vertices nor edges. (10 pts)

Solution:

a. The algorithm is as follows:
   1. Create a flow network G' containing all vertices in V, all directed edges in E with capacity 1, and additionally a source vertex 's' and target vertex 't'. Connect the source to each starting location with a directed edge (s, a_i) and each ending location to the target with a directed edge (b_j, t), all with capacity 1.
   2. Run Ford-Fulkerson on this network to get a maximum flow 'f'. If |f| = k, then there is a solution else if |f| < k, then there is no solution and hence we return 'False'. We will later show that it is always the case that |f| <= k.
   3. To extract the paths a_i to b_j as well as which starting location ultimately connects to which ending location, run DFS on the returned max flow f starting from 's', tracing a path to 't'. Remove these edges and repeat 'k' times until we have the 'k' disjoint paths.

   To prove correctness, we must show that there is a flow of size 'k' if and only if there are k disjoint paths.

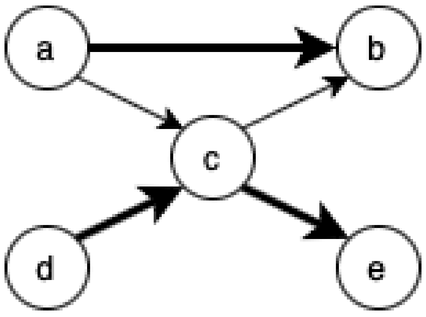
   $\rightarrow$) Claim : An integral flow of size f in a network with unit capacities can be decomposed into a unit flow over a simple path p: s$\rightarrow$t and a flow of size f - 1 which does not use p.

   Proof: Pick any simple path p: s$\rightarrow$t over edges with non-zero flow. One must exists otherwise we would have a cut with no flow. In the residual graph, we send an augmenting flow of -1 over the path that has capacity 1. We then have a resulting flow of size f-1, and the flow on each of the edges of p is 0.

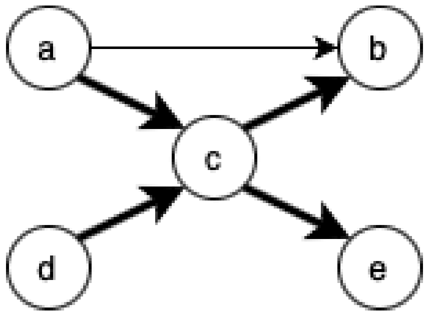
   $\leftarrow$) If there is a set of k disjoint paths from a_i to b_j, then the flow with one unit of flow on each of the edges contained in those paths defines a flow of size k. Since the cut (s, V'- s) has capacity k, this is a maximum flow on the network.

   Time complexity: The run time for this algorithm is O(k|E|). In the modified graph G', we have |E| + 2k = O(|E|) edges, and the maximum possible flow value is k, therefore, running Ford-Fulkerson takes O(k|E|) time. Extracting paths takes at most O(|E|) time, as we traverse each edge at most once. In total, then, the algorithm takes O(k|E|) time.

b. The modifications are as follows:



a→b & c→e share neither vertices nor edges



a→b & c→e share a vertex

Duplicate each vertex ‘v’ into 2 vertices v_{in} and v_{out}, with a directed edge between them. All edges (u, v) now become (u, v_{in}) and all edges (v, w) now become (v_{out}, w). Assign the edge (v_{in}, v_{out}) capacity 1. But while doing so we must make sure we include the starting and ending locations a_i and b_i to the vertices duplicated.

With this transformation, we now have a graph in which there is a single edge corresponding to each vertex, and thus any paths that previously shared vertices would be required to share this edge. Now, we can use the same algorithm mentioned above in (a) on the modified graph to find ‘k’ disjoint paths sharing neither edges nor vertices, if they exist.

Time complexity: The transformation of the graph takes O(|E|) time, as we are adding that many new vertices and edges to the graph. The new graph has O(|E| + |V| + 2k) = O(|E|) edges, and therefore the run time is unchanged.

5. In the network below, the demand values are shown on vertices (supply value if negative). Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. You need to show all your steps.

   a. Turn the circulation with lower bounds problem into a circulation problem without lower bounds. (8 pts)
   b. Turn the circulation with demands problem into the maximum flow problem. (8 pts)

c. Does a feasible circulation exist? Explain your answer. (4 pts)

Solution:

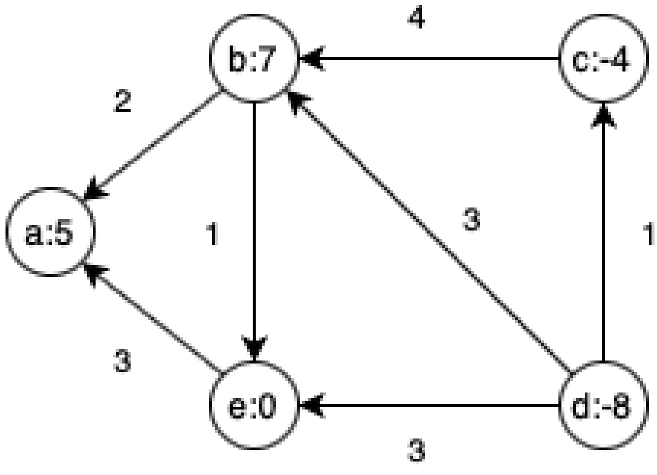
a. Let a:9, b:5, c:-4, d:-13, e:3
First we check the necessary condition for a feasible circulation. The sum of demands must be zero.
Here, 9+5+(-4)+(-13)+3 = 0, thus the necessary condition for a feasible solution is met and we can turn the circulation with lower bounds problem into a circulation problem without lower bounds.
We push a flow with the value of the lower bound l(u,v) on each edge and compute the flow excess $L(v) = f^{in}(v) – f^{out}(v)$ for each vertex v.

L(a) = (2+2) - 0 = 4
L(b) = (2+1) – (3+2) = -2
L(c) = 2 – 2 = 0
L(d) = 0 – (2+1+2) = -5
L(e) = (2+3) – 2 = 3

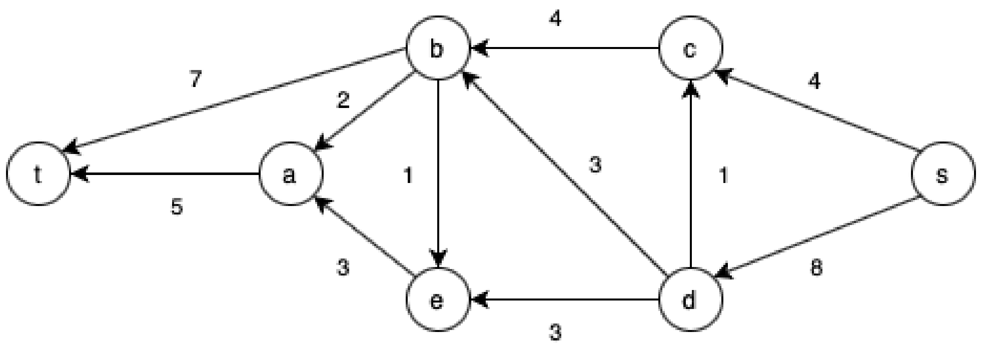
Next, we compute the demands d'(v) = d(v) – L(v) to get,

d'(a) = 9 - 4 = 5
d'(b) = 5 – (-2) = 7
d'(c) = -4 – 0 = -4
d'(d) = -13 – (-5) = -8
d'(e) = 3 – (3) = 0

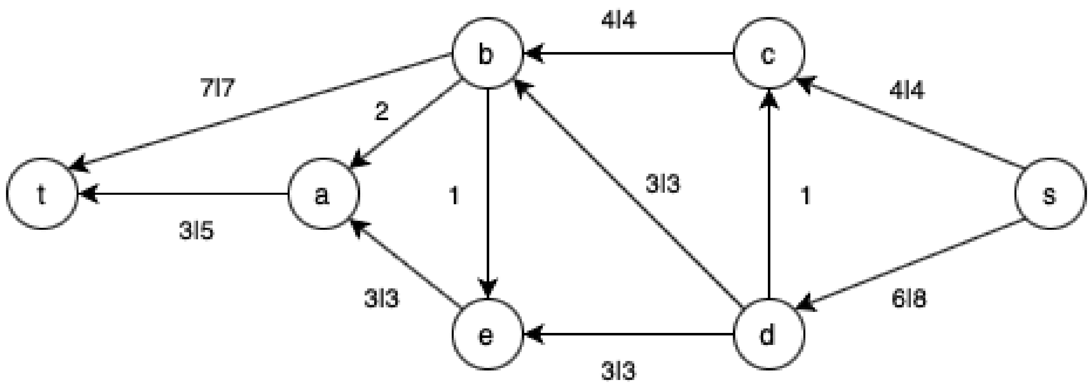
Thus we have reduced the original problem into a circulation problem without lower bounds.



b. In order to reduce the circulation problem from part (a) into the max flow network problem, we construct a new graph by adding 2 extra vertices, s and t. We connect the source ‘s’ with vertices c and d by edges with capacities 4 and 8, respectively. We connect vertices a and b with the target ‘t’ by edges with capacities 5 and 7,respectively.



c. Running the Ford-Fulkerson algorithm, we find that max flow has value 10 and no path exists from source ‘s’ to target ‘t’. Thus we have a feasible circulation to the original problem and the figure below is a feasible circulation to the original problem. Also the sum of demands of the max flow graph is 0 which shows a feasible circulation.



b
c
t
a
s
e
d
4|4
7|7
2
4|4
1
3|3
1
3|5
3|3
6|8
3|3