

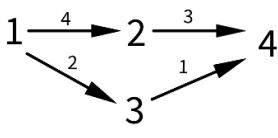
Lab 7

Document 1, Problem 3

a.) $1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 6$, The augmenting path proving the existence of the bottleneck.

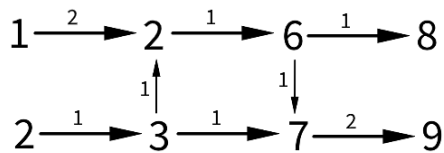
b.) $1 \xrightarrow{1} 2 \xrightarrow{1} 3 \xrightarrow{1} 4 \xrightarrow{1} 5$ Simple example of a graph with no bottlenecks.

c.) $1 \xrightarrow{2} 2 \xrightarrow{1} 3 \xrightarrow{2} 4 \xrightarrow{2} 5$ Simple example of a graph with one bottleneck.

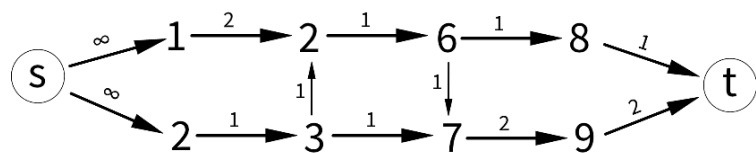
d.)  Simple example of a graph with two bottlenecks.

Document 1, Problem 4

Original:



Transformed:



Transformations are as follows:

Create a singular source node s and connect it to all source nodes i . The connecting edges should have an infinite capacity.

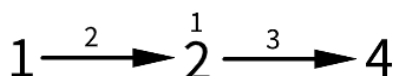
Create a singular sink node t and connect it to all sink nodes j . The capacity of the connecting edge should be $\sum u_{ij}$, or the sum of all edge capacities that connect to sink node j .

This maintains the property that is assumed of sources, having infinite flow, and the property of the sinks, having no more flow than the sum of their edge capacities allows.

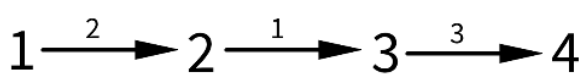
The maximum flow is still deducible from the transformed graph, as the new sink edges also maintain both their conservation and capacity constraints.

Document 1, Problem 5

Original:



Transformed:



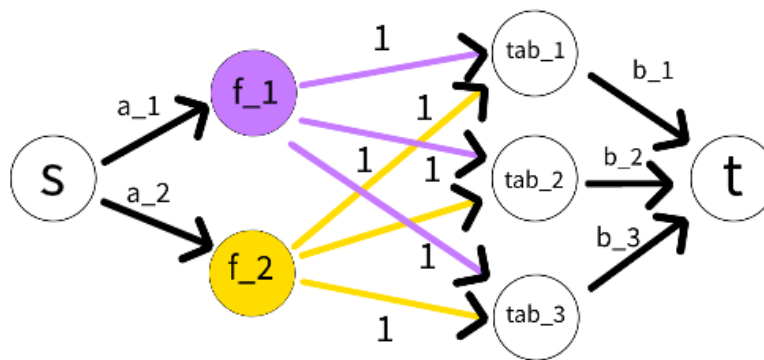
Transformations as follows:

Create an intermediate edge with the same edge capacity as the node with capacity previously had. The edge should connect to new intermediate node, that then inherits all the same outgoing edges incident to the original node. The original node should maintain all incoming edges previously incident to it.

The transform maintains the same capacity constraint previously held by the pre-transformed graph, while now allowing the problem to be considered as a maximum flow problem.

Document 2, Problem 6

Flow network representation:



A.) From left to right, every “layer” of the network represents a constraint on the problem. For simplicity in the example, there is two families and three tables. This can be expanded for any number families p and tables q with their respective attributes.

The edges with capacities a_1 and a_2 are representative of a family f_k with a_i members. By constraining this edge, it ensures no more than the family size of members are distributed to the dining tables.

The nodes f_1 and f_2 are representative of a family f_k .

The edges with capacities one are representative of the maximum number of members from a family f_k that are allowed to be seated at anyone table. Following the constraints of this problem, only one member from each family is to be seated at any one table.

The nodes tab_1 and tab_2 are representative of a table j .

The edges with capacities b_1 , b_2 , and b_3 are representative of the total capacities of any table j . This ensures the capacity constraint of each table is held, i.e. no more diners than the table can allow are seated at said table.

B.) There are two scenarios where there exists no solution. First, if some family f_k has members a_i such that $a_i > q$ (the number of tables). Second, if $\sum_{i=1}^k a_i > \sum_{j=1}^q b_j$, or where the total number of people to be seated exceeds the total capacity of all tables.

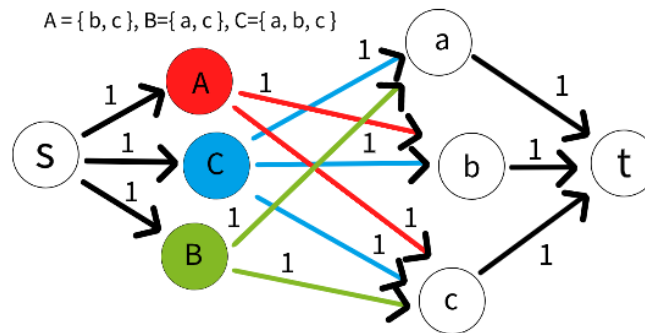
For the first scenario, the algorithm will return a flow x_{s,f_k} on edges $(s, f_k) < u_{s,f_k}$, or in other words, since the algorithm ensures the conservation constraint is met, it will not allow more members to “flow” out of node f_k than there is capacity to flow. This results in the flow on edge (s, f_k) to be less than its capacity, and thus not every member of said family is seated.

For the second scenario, the algorithm upon completion, will return a maximum flow $x < \sum_{i=1}^k a_i$, or in other words, not every person was able to be seated, since the maximum flow is equivalent to the number of diners seated.

C.) To determine where family members are divided amongst tables, after a maximum flow solution is discovered, we can inspect the edges between a family f_k and all tables j to determine where family members from family k are seated amongst all tables. If there is a flow of one on an edge (f_k, j) then a member from the family k must be seated at a table j .

Document 2, Problem 6

Flow network representation:



A.) Again, from left to right, every “layer” of the network represents a constraint on the problem. For simplicity in the example, there is three communities and three representatives. This can be expanded for any number of n subsets consisting of m objects. The examples subsets are detailed above the network.

The edges from the source to each subset S_i (or A, B, C in this example), are of weight 1. This is representative of the constraint that only one object from each subset S_j may be chosen.

Each of the colored nodes represents a one of the sets S_j .

An edge from any one colored node to the next “layer” of nodes is representative of one object contained in the S_j subset, which the colored node represents. The fact that they have a weight of one is to maintain the constraint that only one may be chosen.

The nodes not colored, but labeled a, b, c are representative of A and its elements, the n objects of which each subset S_j is constructed of. The limit of three nodes a, b, c is for this example only.

The sink node and its edges are to again maintain the capacity constraint that only one each object may be represented for each subset.

- B.) For this problem, there are many different combinations of subsets that do not produce a solution. Far too many to list. However, we can tell there is no solution if when after finished the shortest augmenting path algorithm, we receive either a maximum flow $x < |S_j|$, indicating that not every subset has been represented in the final solution, or after inspecting the resulting maximum flow graph, find a subset containing 0 flow, indicating that there were not enough elements in A to represent every subset, S_j .
- C.) Assuming there is a solution, simply inspect the resulting maximum flow graph and the edges between each subset node (the colored nodes in this example) and the object nodes (non-colored, lowercase nodes), and see which of the connecting edges (S_j, a_i) has a non-zero flow. Such a flow implies that object a_i represents for subset S_j .