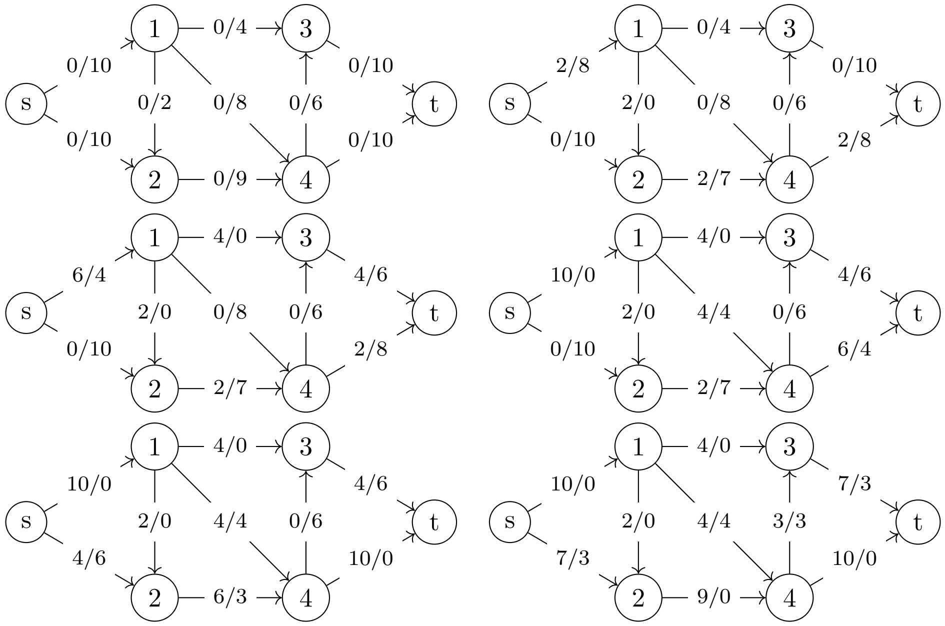
CS240 Homework 3

Shen Linfeng Student ID: 2018E8018461028

April 1, 2019

Problem 1



The path: $s \to 1 \to 2 \to 4 \to t$, $s \to 1 \to 3 \to t$, $s \to 1 \to 4 \to t$, $s \to 2 \to 4 \to t$, $s \to 2 \to 4 \to 3 \to t$.

The edge $a \xrightarrow{\text{c/d}} b$ represent an edge from a to b with capacity of d, and an edge from b to a with capacity of c.

Problem 2

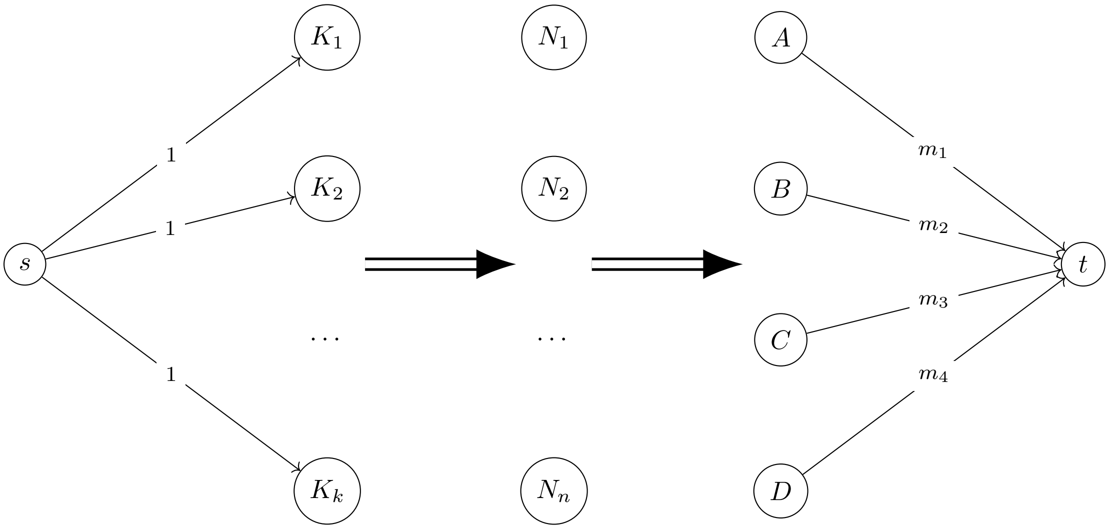
Let N to represent n staff members, K represent k departments, A, B, C, D represent four classes.

Create digraph $G = (\{s, t\} \cup A \cup B \cup C, \cup D \cup K \cup N, E)$:

- Add source s, and unit capacity edges from s to each node in K.
- Add sink t, and m_1 capacity edges from A to t, m_2 capacity edges from B to t, m_3 capacity edges from C to t, m_4 capacity edges from D to t,
- Direct edges from K_i to N_j if N_j belongs to K_i for every j, and assign unit capacity.
- Direct edges from N_j to X if N_j belongs to X(X is one of $\{A, B, C, D\}$) for every j, and assign unit capacity.

Suppose the maximum flow of the network G is f, and assume there is a solution to the problem, H. Apply H on G we can get the flow of it is k, then we should have $k \leqslant f$. And with $c(\{s\} \cup A \cup B \cup C, \cup D \cup K \cup N, \{t\}) = k \geqslant f$, we have $f = k$.

Then use Ford-Fulkerson Algorithm to G, we can get K' and N', where every $K_j \in K'$ and every $N_i \in N'$ is on the path. With $f = k$ and the structure of G, we can get $|K'| = |N'| = k$, so N' is a set satisfy the requirements of this problem, every $N_i \in N'$ is the person we need.



Problem 3

(a) If there exists a min-cut which edge (v, u) don't cross, then the min-cut will not change, the same with max-flow. If edge (v, u) cross a min-cut, the max-flow may increase by 1.

So we can use BFS to search for if there exists an augmenting path on the residual graph. If there exists an augmenting path, then we can add the path to update max-flow, and increase the value of max-flow by 1, otherwise the max-flow will stay the same. It will run in $O(|V| + |E|)$.

(b) If we have $c_f(v, u) \geqslant 1$, then nothing change.

Otherwise, we can use BFS to find a path from t to s on the residual graph. For every edge (a, b)(besides (u, v)) on the path, decrease $c_f(a, b)$ by 1, increase $c_f(b, a)$ by 1. And then use BFS to find if there exists an augmenting path from s to t. If exists, then we can add the path to max-flow, but the value of max-flow will not change; otherwise we should decrease the value of max-flow by 1. It will run in $O(|V| + |E|)$.

Problem 4

Problem 5

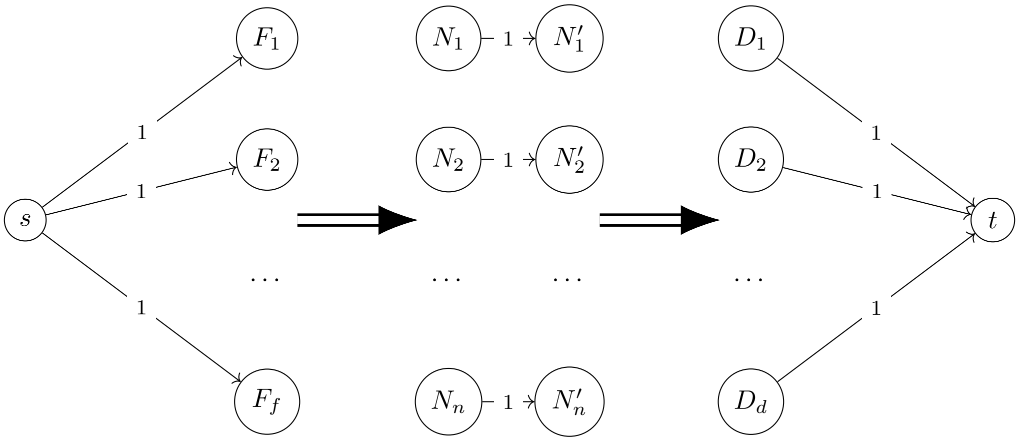
Let N to represent n cows, F represent f types of foods, D represent d types of drinks.

Create digraph $G = (\{s, t\} \cup F \cup G \cup N \cup N', E)$:

- Add source s, and unit capacity edges from s to each node in F.
- Add sink t, and unit capacity edges from each node in D to t.
- Direct edges from N_i to N'_i for every i, and assign unit capacity.
- Direct edges from F_j to N_i if food j on the cow i's list, and assign unit capacity.
- Direct edges from N'_i to D_j if drink j on the cow i's list, and assign unit capacity.

Every path from s to t: $s \to F_j \to N_i \to N'_i \to D_k \to t$, represent a choice that the cow i receives both food j and drink k in his preference lists. So this problem can convert to find the max-flow of G. The set N' is trying to prevent the occurrence of one cow receiving both two types of food and two types of drink at the same time.

Then use Ford-Fulkerson Algorithm to G, and find every path: $s \to F_j \to N_i \to N'_i \to D_k \to t$, then the set of {food j and drink k for cow i } is what we need.



Problem 6

Find a min-cut C, and the set E', where all edges cross C are in E'(Of course, every $e \in E'$ cross C). Then for every $e \in E'$, we can increase $c_f(e)$ by 1, and then find is there exists an augmenting path on the residual graph(The same as Problem 3(a)). So if for every $e \in E'$, there don't exist an augmenting path, the G has a unique min-cut, otherwise it hasn't.

For every $e \in E'$, find augmenting path run in $O(|V| + |E|)$, and $|E'| \leqslant |E|$. And finding a min-cut will cost polynomial time. So this algorithm is a polynomial time algorithm.