Network Flow Practice Problems

Please report any typos to Ben Chugg at `ben.chugg@alumni.ubc.ca`

1. There are n employees at a company, $E = \{e_1, \ldots, e_n\}$, and m tasks $T = \{t_1, \ldots, t_m\}$ each of which needs to be completed on a weekly basis. Furthermore, some tasks are specialized: only a subset of the employees may complete them. Formally, for employee e_i, let $T_i \subseteq T$ be the subset of tasks she may accomplish. Suppose the tasks are arduous enough that an employee can only complete one task per day[1]. Assume that only one employee can work on a given task.

   (a) For a given week, design an algorithm to determine if all the tasks can be accomplished.

   (b) Now suppose that the employees are moody and that they can only complete certain tasks on certain days. (Hint: Your answer shouldn't change much).

   (c) Let's now assume that some tasks need multiple days to complete. For task t_i, let d_i denote the number of days required to complete it. Different employees can help complete the task, but we still require that no two employees work on the same tasks on a single day. Design an algorithm which determines if all the tasks can be completed. (Hint: Add vertices to the flow network from (b)).

2. Let $G = (V, E)$ be a graph, and let $s, t \in V$. An *st* path is a path in G starting from s and ending at t.

   (a) Two *st* paths are *edge disjoint* if they share no edges (aptly named). Design an algorithm to determine the number of edge disjoint *st* paths if G is undirected. (Hint: If G is directed this is easy — how can you simulate G being directed?).

   (b) Two *st* paths are *vertex disjoint* if they share no vertices besides s and t. Design an algorithm to determine the number of vertex disjoint *st* paths if G is directed.

3. (Courtesy of Kleinberg and Tardos via William Evans.) Let M be an $n \times n$ matrix with each entry equal to either 0 or 1. Let m_{ij} denote the entry in row i and column j. A diagonal entry is one of the form m_{ii} for some i. Swapping rows i and j of the matrix M means swapping m_{ik} with m_{jk} for all $k = 1 \ldots n$. A similar definition holds for swapping columns. M is rearrangeable if by swapping rows and/or swapping columns of M one can make all diagonal entries be 1.

   (a) Give an example of a matrix M that is not rearrangeable even though every column and every row contains at least one entry that equals 1.

   (b) Describe an efficient algorithm (running in polynomial time) that determines if M is rearrangeable.

4. A natural approach to solving Network flow is to try a greedy approach. Consider the following strategy: Pick the *st* path in G with the largest minimum capacity edge — say it has capacity k — push k units of flow down the path and then delete the saturated edge(s) from the graph. Repeat until no *st* path can be found. Prove or disprove the correctness of this algorithm.

5. Suppose we want to adapt a network flow algorithm such that it can incorporate vertex capacities as well as edge capacities. (A vertex capacity limits the amount of flow that can travel through a vertex). Emulate this scenario with a regular network flow.

[1]You may assume there are five days in a week, but it doesn't matter much

6. (Pennant Race Problem). Suppose we are avid baseball fans. We cheer for team A and there are n other teams in the league, $T_1, \ldots, T_n$. The team who has the highest number of wins at the end of the season will win the pennant. It is halfway through the season, and we are given the number of games already won by each team (say w_i for team i). We are also given a list of remaining games to be played (e.g., $(T_1, T_4), (A, T_2), (T_1, T_2)$, etc). Give an algorithm to determine if there is hope of team A winning the pennant.