

Solutions of Assignment #1 part 1, Total points: 55
(Course: CS 401)

Problem 1 (20 points): Prof. Smart thinks he is smarter than all the students in this CS 401 class. He has made the following claim to show how smart he is.

Claim made by Prof. Smart: Consider the stable matching problem as taught in class. Suppose that we have only two men, say m_1 and m_2 , and two women, say w_1 and w_2 , with their corresponding preference lists. Suppose also that the matching $m_1 - w_1$ and $m_2 - w_2$ is a stable matching. Prof. Smart claims that in that case the matching $m_1 - w_2$ and $m_2 - w_1$ can **never** be a stable matching.

Your task is to decide if Prof. Smart is indeed so smart. For this purpose, do the following.

- Either prove the claim made by Prof. Smart is indeed correct. Such a proof should work **no matter** what the preferences of the men and women are, as long as $m_1 - w_1$ and $m_2 - w_2$ is a stable matching.
- Or, prove the claim made by Prof. Smart is wrong by giving a counter-example. The counter-example should provide the preferences lists of every man and woman, and show that for these preference lists **both** $m_1 - w_1$, $m_2 - w_2$ and $m_1 - w_2$, $m_2 - w_1$ are indeed stable matchings.

Solution: The following preference lists constitute a counter-example since both both $m_1 - w_1$, $m_2 - w_2$ and $m_1 - w_2$, $m_2 - w_1$ are stable matchings.

	first	second
m_1	w_1	w_2
m_2	w_2	w_1
w_1	m_2	m_1
w_2	m_1	m_2

Problem 2 (35 points): Let $G = (V, E)$ be a **directed** graph and let s and t be two nodes of G . Let n and m be the number of nodes and edges of G , respectively. In the class we say how to decide if there is a path **from** s **to** t , namely, we start a (directed) BFS starting from s and check if t appears among the list of nodes that are visited during BFS. The purpose of the assignment is to decide if such a path exists under some **additional constraints**. Let u and v be two other nodes of G that are **not** s or t .

(i) [15 points] Decide if G has a path from s and t that **avoids** using **both** the nodes u and v .

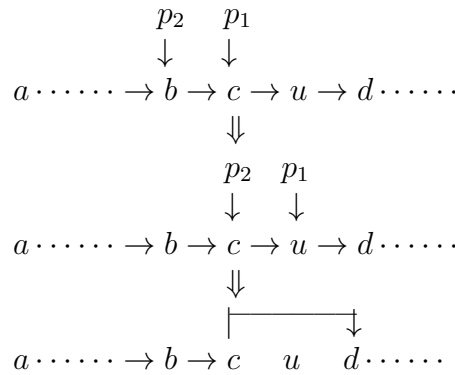
(ii) [20 points] Decide if G has a path from s and t that **uses both** the nodes u and v .

Both of your algorithms should run in $O(m + n)$ time. You may assume that the graph is given in its adjacency list representation. If you are using BFS, there is no need to give codes for it; simply saying “do a BFS starting at such-and-such node” will suffice.

Solution:

(i) We delete all the edges in-coming to nodes u and v , i. e., we delete all the edges of the form (x, u) or (x, v) where $x \in V - \{u, v\}$. Then we simply run a BFS starting at s and check if t can be reached.

To delete all the edges as mentioned above, we go through the adjacency list of every node except u and v . For each such list, we traverse the list keeping two pointers, pointer p_1 one at the current entry and pointer p_2 at the entry before the current entry. If the current entry is u or v , we change the link of entry to p_2 to skip u or v . Pictorially, it looks like as shown below for the adjacency list of node a :



(ii) We will use the notation $x \rightsquigarrow y$ to indicate a directed path from node x to node y . A path from s to t that uses both nodes u and v must be one of the following two types depending on whether node u is before or after node v :

(A) $s \rightsquigarrow u \rightsquigarrow v \rightsquigarrow t$

(B) $s \rightsquigarrow v \rightsquigarrow u \rightsquigarrow t$

For (A), we can use a BFS starting at s to find a path $s \rightsquigarrow u$, a BFS starting at u to find a path $u \rightsquigarrow v$, and a BFS starting at v to find a path $v \rightsquigarrow t$. (B) can be handled in a similar manner.