COMP 3711 – Honors Design and Analysis of Algorithms
2021 Spring Semester – Written Assignment # 5
Distributed April 28, 2021 – Due May 12, 2021
Typo corrected May 3, 2021

Your solutions should contain (i) your name, (ii) your student ID #, and (iii) your email address

Some Notes:

- Please write clearly and briefly. Your solutions should follow the guidelines given at
  https://canvas.ust.hk/courses/36180/pages/assignment-submission-guidelines

  In particular, your solutions should be written or printed on *clean* white paper with no watermarks, i.e., student society paper is not allowed.

- Please also follow the guidelines on doing your own work and avoiding plagiarism as described on the class home page.
  You must acknowledge individuals who assisted you, or sources where you found solutions. Failure to do so will be considered plagiarism.

- The term *Documented Pseudocode* means that you must include clear comments *inside* your pseudocode.

- Many questions ask you to explain things, e.g., what an algorithm is doing, why it is correct, etc. To receive full points, the explanation must also be *understandable* as well as correct.

- Please make a *copy* of your assignment before submitting it. If we can't find your submission, we will ask you to resubmit the copy.

- Submit a SOFTCOPY of your assignment to CASS by the deadline. The softcopy should be one PDF file (no word or jpegs permitted, nor multiple files).

  If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possibly denser.

- Typo corrected, May 3, 2021
  On page 13, Google's 3rd choice was listed as "z" but no "z" exists. This should have been "j". This has now been corrected here and in the distributed template.

P1: Max-Flow and Graph Reliability [35 pts]

Let $G = (V, E)$ be a directed graph with two specified vertices $s, t \in V$. Assume that G contains at least one s-t path.

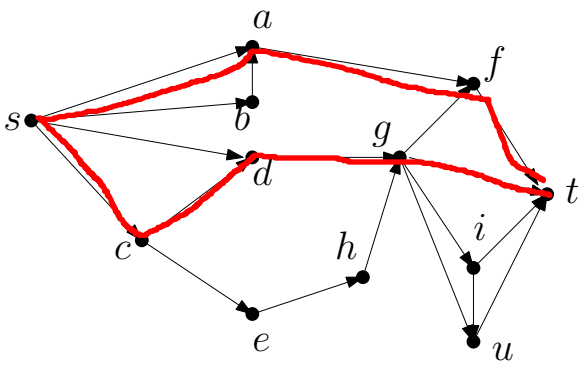
A subset of edges $E' \subseteq E$ is a *separating edge set* if, after removing E' from G, no s-t path exists.

A subset $V' \subseteq V - \{s, t\}$ of vertices is a *separating vertex set* if, after removing V' and all edges connected to V' from G, no s-t path exists. If $(s, t) \notin E$, a separating vertex set always exists.

E' is a *smallest separating edge set* if it contains the smallest number of edges among all separating edge sets. V' is a *smallest separating vertex set* if it contains the smallest number of vertices among all separating vertex sets.

Note that a smallest separating edge set E' might not be unique. A smallest separating vertex set V' also might not be unique.

In the graph below $\{(a, f), (d, g), (e, h)\}$ is a smallest separating edge set, but there are others as well. $\{g, f\}$ and $\{g, a\}$ are the only smallest separating vertex sets.



Smallest separating sets are indicators of failure points in a network and are therefore used in studies of network reliability.

In what follows you may use any facts or algorithms taught in class as long as you explicitly reference them (which means including their statement and where in the class or tutorial notes you found them).

(a) Give an $O(|E|^2)$ time algorithm for finding a smallest separating edge set for G.

(b) Assume $(s, t) \notin E$. Give an $O(|V||E|)$ time algorithm for finding a smallest separating vertex set for G.

Both (a) and (b) should contain 3 parts. Part (i) should describe your algorithm clearly (code is not necessary). Part (ii) should prove its correctness.

Part (iii) should derive its running time. Your proof of correctness must contain a section that *EXPLICITLY* justifies why your output is a smallest separating edge or vertex set.

Hints, Comments and Recommendations:

- For (a), consider the algorithm for finding the maximum number of edge disjoint s-t paths taught in class. How can you modify that to solve this problem?

  Hints. Let S, T be ANY cut. Then the set of edges crossing between S, T is a separating edge set.

  Consider the Min-Cut you get when solving the edge disjoint s-t path problem. The statement above implies that the max number of edge disjoint paths is an upper bound on the size of the smallest separating edge set (why?). Can you show that these two values must be equal?

- For (b), modify G to create a new graph $G' = (V', E')$.

  (i) For every vertex $v \in V$, create two vertices v_ℓ, v_r in V'.

  (ii) For every vertex $v \in V - \{s, t\}$, Create edge (v_ℓ, v_r) in E'.

  (iii) For every edge $(u, w) \in E$, create edge (u_r, w_ℓ) in E'.

  (iv) Remove vertices s_ℓ and t_r from V'.

  Note that if P is an $s - t$ path in G that passes through vertex v, the corresponding path P' in G' is an $s_r - t_\ell$ path that passes through edge (v_ℓ, v_r).

- The modified graph G' contains two very different type of edges:

$$\begin{aligned} E'_1 &= \{(v_\ell, v_r) : v \in V - \{s, t\}\}, \\ E'_2 &= \{(u_r, v_\ell) : (u, v) \in E\}. \end{aligned}$$

  A separating vertex set in G corresponds to a separating edge set in G' in which all edges are in E'_1 and vice-versa.

  So solving (b) corresponds to finding a smallest separating edge set in which all edges are in E'_1 in G'.

  The idea is to appropriately modify the algorithm for part (a) to find such a set.

- Suppose you assign capacity 1 to all edges in E'_1 and capacity 2 to all edges in E'_2. Prove that the edges crossing a Min-Cut must all be of type E'_1.

  (Proving this requires understanding the structure of the min-cut in the proof of the correctness of the Ford-Fulkerson algorithm.)

- Use the observations above and (a modified version of) the result of part (a) to solve (b).

P2: [25 pts] **Max Flow**

Modified from Kleinberg and Tardos, Chapter 7, Exercise 16.

Google makes money by selling targeted advertising, using what it knows about you.

When you do a search Google shows you an ad. A specific ad is shown to you because an advertiser requested that this ad be shown to a person in a demographic group to which you belong.

In this problem you will design an algorithm to decide whether Google can satisfy a collection of targeted requests made by advertisers.

- **U** is the universe of people currently online: $U = \{p_1, \ldots, p_n\}$.
- $\mathbf{X}_i$, $j = 1 \ldots m$ are subsets of U that *cover* U, i.e., $\forall j$, $\mathbf{X}_j \subseteq U$ and $\bigcup_j \mathbf{X}_j = U$.
  The $\mathbf{X}_i$ are *demographic groups*. For example, $\mathbf{X}_1$ might be all middle-aged professors with two children. The $\mathbf{X}_i$ are defined by inputs $D_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq m$:

$$D_{i,j} = \begin{cases} 1 & \text{If } p_i \in \mathbf{X}_j, \text{ i.e., person } p_i \text{ is in Demographic Group } j, \\ 0 & \text{If } p_i \notin \mathbf{X}_j. \end{cases}$$

- There are r Advertisers (companies). Advertiser k wants its ads shown to at least P_k people.
- Furthermore, each advertiser only wants to show ads to people within specific targeted collections of demographic groups. We denote these constraints as follows. The collections are defined by variables $Y_{j,k} \in \{0, 1\}$, $1 \leq j \leq m$, $1 \leq k \leq r$. Set

$$\mathbf{Z}_k = \bigcup_{\{j\,:\,Y_{j,k}=1\}} \mathbf{X}_j$$

  Advertiser k wants its advertisements only to be shown to people in $\mathbf{Z}_k$.
- Each Advertiser provides Google with one advertisement. Each person can only be shown one advertisement, i.e., from one Advertiser.

The inputs to the problem are the values of $D_{i,j}$, $Y_{j,k}$ and P_k.

The problem is now to find out if Google can devise an advertising policy satisfying all of the advertisers requests.

That is, can Google show each person at most one ad so that exactly P_k people see the ad from Advertiser k and these P_k people are all in the set $\mathbf{Z}_k$.

Your algorithm should run in $O\left(\left(\sum_k P_k\right) mn\right)$ time.

(a) Design an $O\left(\left(\sum_k P_k\right)mn\right)$ time algorithm for deciding if it is possible for Google to satisfy all of the requests.

Your algorithm does not need to be in pseudocode. It does need to be understandable, though. Do not write it as one paragraph.

Instead, present the algorithm in numbered point notation, putting each step in a different numbered point.

If satisfaction is possible, you should output which advertiser's ad each person is shown (or if a person is not shown an ad). That is, you should output n pairs (i, k) denoting that p_i is shown advertisement k (where $k = 0$ if p_i is not shown any ad).

Your algorithm may use any algorithm shown in class as a subroutine.

(b) Explain why your algorithm is correct.

If your explanation invoves an "if and only if" statement, then you need to explicitly prove the correctness of both directions of the "if and only if" statement (as in the proof of correctness of the solution of the maximum distinct paths problem taught in class).

(c) Explain why your algorithm runs in $O\left(\left(\sum_k P_k\right) mn\right)$ time.

Hints, Comments and Recommendations:

- You may assume that $1 \leq \min(m, n, r)$, that all $P_i \geq 1$ and that $r \leq \sum_k P_k \leq n$. This is because each person sees only one advertisement.
- An example: let $n = 11$. Set
  - $\mathbf{X}_1 = \{p_1, p_3, p_4\}$, $\mathbf{X}_2 = \{p_3, p_4, p_5, p_6, p_7, p_8\}$, $\mathbf{X}_3 = \{p_1, p_2, p_7, p_8, p_9\}$ $\mathbf{X}_4 = \{p_{10}, p_{11}\}$
  - $(P_1, P_2, P_3, P_4, P_5) = (1, 2, 3, 3, 2)$.
  - $Z_1 = \mathbf{X}_1 \cup \mathbf{X}_2$, $Z_2 = \mathbf{X}_2 \cup \mathbf{X}_3$, $Z_3 = \mathbf{X}_3 \cup \mathbf{X}_4$, $Z_4 = \mathbf{X}_1 \cup \mathbf{X}_2 \cup \mathbf{X}_3$, $Z_5 = \mathbf{X}_4$.

  Let A_i be the ad for company i. Then

  $$A_1 \leftarrow p_1, \quad A_2 \leftarrow p_5, p_6, \quad A_3 \leftarrow p_7, p_8, p_9, \quad A_4 \leftarrow p_2, p_3, p_4, \quad A_5 \leftarrow p_{10}, p_{11}.$$

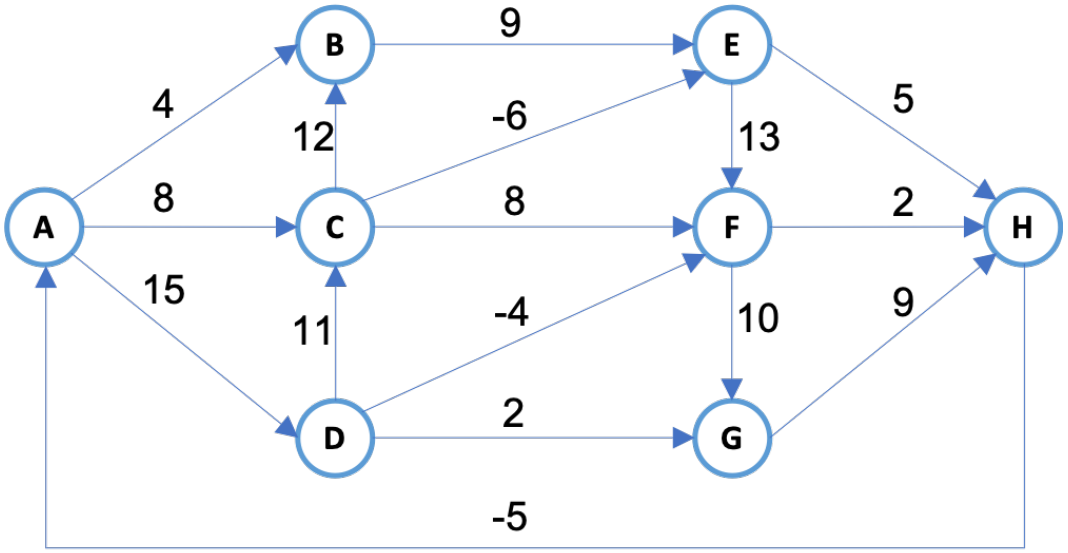
  satisfies the constraints. That is

  $$(1, 1),\ (2, 4),\ (3, 4),\ (4, 4),\ (5, 2),\ (6, 2),\ (7, 3),\ (8, 3),\ (9, 3),\ (10, 5),\ (11, 5).$$

  Now set $(P_1, P_2, P_3, P_4, P_5) = (1, 2, 2, 3, 3)$ but keep everything else the same. There is no solution to this modified problem.

P3: [16 pts] **All-Pairs Shortest Paths**

Solve the all-pairs shortest path problem on the given directed graph G. W is the weighted adjacency matrix of G.



$W =$

	A	B	C	D	E	F	G	H
A	0	4	8	15	∞	∞	∞	∞
B	∞	0	∞	∞	9	∞	∞	∞
C	∞	12	0	∞	-6	8	∞	∞
D	∞	∞	11	0	∞	-4	2	∞
E	∞	∞	∞	∞	0	13	∞	5
F	∞	∞	∞	∞	∞	0	10	2
G	∞	∞	∞	∞	∞	∞	0	9
H	-5	∞	∞	∞	∞	∞	∞	0

(A) Apply the 2nd dynamic programming solution on Page 37, 17_Shortest_Path, using the recurrence of $d_{ij}^{(2S)} = \min_{1 \le k \le n} \{d_{ik}^{(S)} + d_{kj}^{(S)}\}$.
To answer this question, fill in the following $D^{(S)}$ matrices for each step. *Note that* $D^{(1)} = W$.

$D^{(1)}$ =

	A	B	C	D	E	F	G	H
A	0	4	8	15	∞	∞	∞	∞
B	∞	0	∞	∞	9	∞	∞	∞
C	∞	12	0	∞	-6	8	∞	∞
D	∞	∞	11	0	∞	-4	2	∞
E	∞	∞	∞	∞	0	13	∞	5
F	∞	∞	∞	∞	∞	0	10	2
G	∞	∞	∞	∞	∞	∞	0	9
H	-5	∞	∞	∞	∞	∞	∞	0

$D^{(2)}$ =

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(4)}$ =

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(8)}$ =

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

(B) Apply the Floyd-Warshall algorithm on Page 45, 17_Shortest_Path,using the recurrence $d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$.

To answer this question, fill in the $D^{(k)}$ matrices for each step. *Note that* $D^{(0)} = W$.

$D^{(1)} =$

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(2)} =$

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(3)} =$

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(4)} =$

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(5)} =$

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(6)} =$

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(7)} =$

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

$D^{(8)} =$

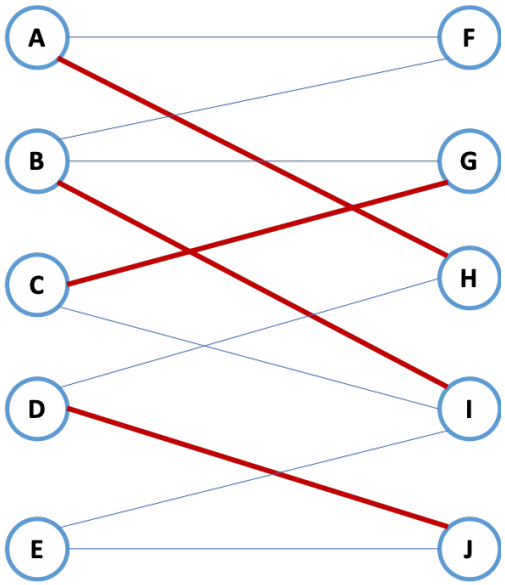
	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

Hints, Comments and Recommendations:

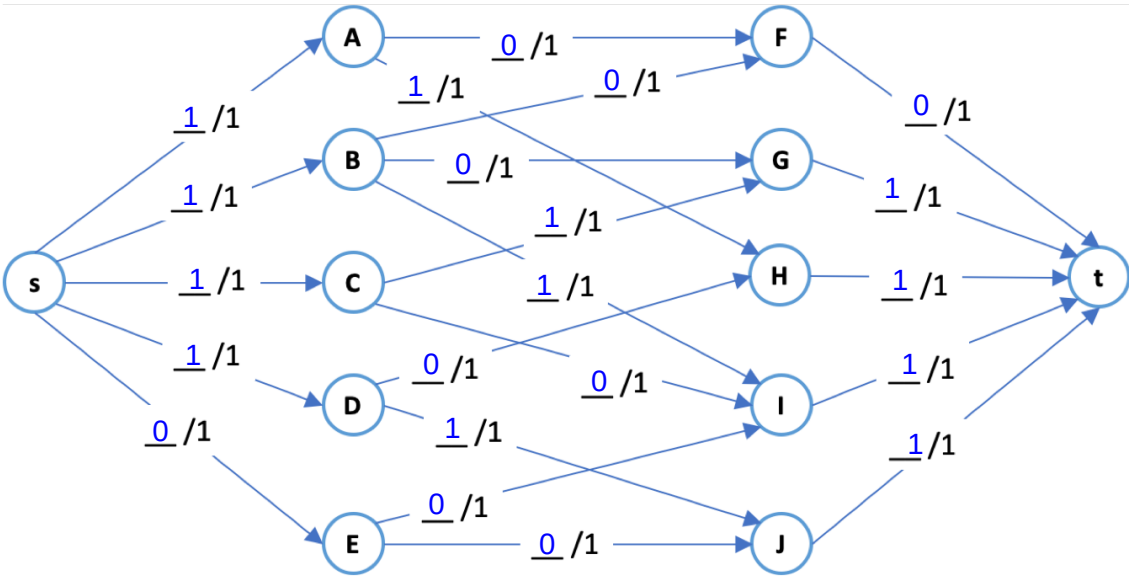
- It is perfectly fine to write code to generate the table entries. For marking purposes, though, we want you to present the results using our templates.
- This problem is self checking. From what we taught in class, matrix $D^{(8)}$ from part (A) should be identical to matrix $D^{(8)}$ in part (B).

P4: [16 pts] **Maximum Bipartite Matching**

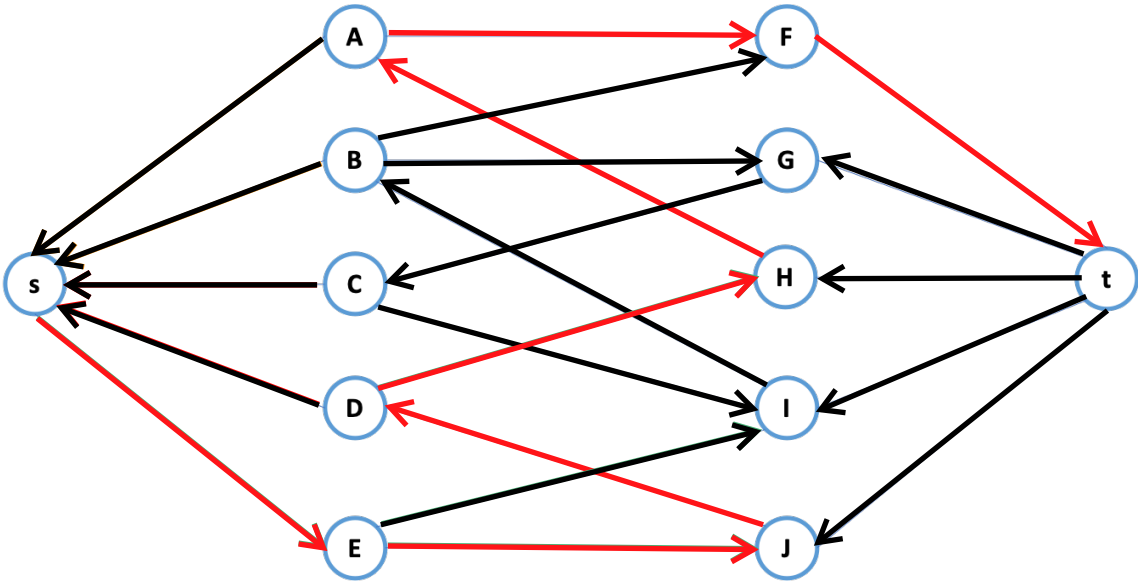
Given a matching M (edges highlighted in red) on a bipartite graph G below.



(A) Please draw the flow corresponding to M in the flow network of G. To do that, please specify the flow value on each edge in the template below, in the format of $f(e)/c(e)$. Note that $c(e)$ is already given.



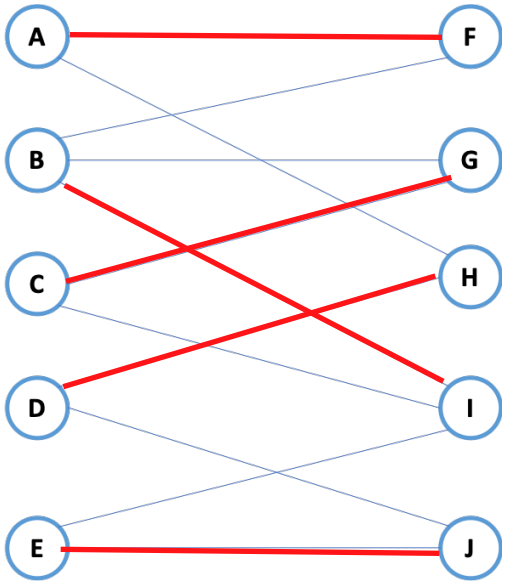
(B) Please draw the residual graph corresponding to M on the template below. Note that you only need to specify the direction of the edges (omitting the residual edge capacities $c_f(e)$, as they are all 1).



(C) Is M a maximum bipartite matching of G?

If yes, specify the corresponding min $S-T$ cut in the residual graph you have drawn in (B) by highlighting all the nodes in S.

If no, find an $s-t$ path P in the residual graph you have drawn in (B) by highlighting all the edges in P. Then provide the resulting matching M' by highlighting the selected edges in G using the following template.



Hints, Comments and Recommendations:

- To clarify: the result of part (B) and the s-t path (if it exists) in the residual graph requested by part (C) should BOTH be drawn on the graph template in part (B).

P5: [8 pts] **Stable Matching**

There are five companies and five CEO candidates. The companies' preferences of CEOs are given in the left table below, and the CEOs' preferences of companies are in the right table.

	1st	2nd	3rd	4th	5th
Apple (A)	k	t	j	b	s
Facebook (F)	k	b	t	j	s
Google (G)	s	k	j	t	b
Microsoft (M)	b	k	s	j	t
Amazon (Z)	b	j	s	t	k

	1st	2nd	3rd	4th	5th
Bill (b)	G	Z	F	A	M
Jeff (j)	Z	F	A	M	G
Mark (k)	G	M	Z	F	A
Sundar (s)	A	M	G	Z	F
Tim (t)	F	A	M	G	Z

(A) Run the Propose-And-Reject Algorithm (Page 9 of lecture note 20_Stable_Marriage) based on the companies' preferences.

(1) Please follow the tutorial problem MM4_Stable_Matching. Provide the detailed action and outcome of each step, and highlight the corresponding items in the table.

(2) Provide the final company optimal solution (as a set of matchings).

(B) Run the Propose-And-Reject Algorithm (Page 9 of lecture note 20_Stable_Marriage) based on the CEOs' preferences.

(1) Please follow the tutorial problem MM4_Stable_Matching. Provide the detailed action and outcome of each step, and highlight the corresponding items in the table.

(2) Provide the final CEO optimal solution (as a set of matchings).

Requirements, Hints, Comments and Recommendations:

- When running the Propose-And-Reject Algorithm, always choose the free company/CEO based on the order given in the tables.

  The Gale-Shapely algorithm technically lets you choose ANY free company/CEO to propose. For the purpose of assignment marking we are requiring you to always choose the free company/CEO that is the highest in the table.