Project 1 (v1.0)

Released: 15 Feb 2020 (Sat, Week 6)
Deadline: 29 Feb 2020 (Sat, Week 8) before 11:00pm

Always check "Announcements" at the server for errata and project-related information

General Instructions:

- There are 3 fun questions in this project to be attempted individually.
- This assignment constitutes 10% of your final grade for this course.
- For each question, you may use **p1q<x>_main.py** to check that your functions have been written correctly. But do not submit the main files.
- **WARNING**: Plagiarism is strictly not tolerated and plagiarism cases will be referred to the university's disciplinary committee. You MUST acknowledge all third-party contributions (including assistance acquired) in your write-up and list all sources in a reference list there.
- The following 3rd party modules have been installed on red server: **numpy**, **pandas**, **scipy** and **sklearn**[1]. You may use them by inserting the relevant import statements into **p1q<X>.py**. No additional 3rd party modules will be installed on red. You are allowed to use other 3rd party code in your solutions (just upload the code in a separate .py file or include the relevant functions in **p1q<X>.py**), and you must acknowledge and reference all usage of 3rd party code in your write-up.
- Only python code can constitute part of your solution. Do not include binary files or code from other languages. Your code should not communicate with any other servers/programs. You are also not allowed to use multi-threaded code to improve performance.

Q1: Sum of Numbers [2 marks]

The input data is represented as a 2D list of integers as follows:

2	1	3
4	9	8
6	2	7

The matrix above is represented like this in Python:

```
m = [[2, 1, 3],
     [4, 9, 8],
     [6, 2, 7]]
```

You need to produce an output 2D list of integers, with each cell in the output corresponding to the cell at the same column and row in the input, as explained below.

For each valid cell:

(a) Identify the numbers to the right (across).
(b) Identify the numbers below (down).
(c) Compute the sum of all these identified numbers

[1] To use **sklearn**: besides importing **sklearn**, you need to insert this statement at the top of your py file: **from pandas import ***

e.g. 1: for the cell in the first row and first column (2), the sum for 2 across is 2 + 1 + 3 = 6. The sum for 2 down is 2 + 4 + 6 = 12. Add across (6) and down (12) and store the value 18 in the corresponding cell in the output.

e.g. 2: for the cell in the last row and first column (6), the sum for 6 across is 6 + 2 + 7 = 15. The sum for 6 down is 6, because there is no further number. Add across (15) and down (6) and store the value 21 in the corresponding cell in the output.

The function should return the following output 2D list.

```
output = [[18, 16, 21],
          [31, 28, 23],
          [21, 11, 14]]
```

Requirements:

Edit the function **q1_recursive(m, output, row, col)** in **p1q1.py** that takes in a 2D list (**m**), and returns the corresponding output 2D list. You can use a combination of recursion and iteration to solve the problem but recursion must be used as part of the solution. Note: do not edit the **q1**(m) function in **p1q1.py**.

Q2: The Stable Marriage Problem (Again) [2 marks]

During week 1's lecture, you were briefly introduced to the stable marriage problem as well as how it can be applied to practical situations such as kidney-matching. This is a well-documented problem, and you are required to find out more about the stable marriage problem from online sources yourself.

In this context, **n** males and **n** females rank each member of the opposite sex from 1 (most preferred) to **n** (least preferred). For example, the following tables show the preferences of 4 couples (**n**=4).

Ranking by Males:

	F1	F2	F3	F4
M1	1	3	2	4
M2	1	2	4	3
M3	1	2	3	4
M4	2	3	1	4

e.g. M1 prefers F1 the most, followed by F3, F2 and F4.

Ranking by Females:

	M1	M2	M3	M4
F1	2	1	3	4
F2	4	3	2	1
F3	1	2	3	4
F4	3	4	2	1

e.g. F2 prefers M4 the most, followed by M3, M2 and M1.

The ranked preferences are represented in the form of a 2D list. The following shows how the two tables above are represented:

```
[[m1, 1, 3, 2, 4], [m2, 1, 2, 4, 3], [m3, 1, 2, 3, 4], [m4, 2, 3, 1, 4],
[f1, 2, 1, 3, 4], [f2, 4, 3, 2, 1], [f3, 1, 2, 3, 4], [f4, 3, 4, 2, 1]]
```

For these preferences, solution (a) below is unstable, and solution (b) is stable:

(a) (M1-F1), (M2-F3), (M3-F2), (M4-F4) – unstable solution
(b) (M1-F3), (M2-F1), (M3-F4), (M4-F2) – stable solution

Proposed solutions to the problem can also be represented as 2D lists. The two solutions above are represented like this:

Solution (a): `[[m1, f1], [m2, f3], [m3, f2], [m4, f4]]`
Solution (b): `[[m1, f3], [m2, f1], [m3, f4], [m4, f2]]`

Assumptions:

- The preferences are always valid. i.e. there are no “ties”, and each female is only ranked once by each male (and vice versa). You will not see invalid preferences such as `[[m1, 1, 1, 2, 4],…]` or `[[m1, 1, 3, 5, 4],…]` or `[[m1, 0, 1, 2, 3],…]`
- The preferences may be in any order. E.g. the preferences of f3 may appear before the preferences of f2. So this is possible: `[[f3, 1, 2, 3, 4], [m1, 1, 3, 2, 4], [m3, 1, 2, 3, 4],…]`
- In the solution list, the males will always come before the female partner. You will not see this: [[f1, m1],…]. However, the pairs may be in any order. So this is possible: [[m2, f1], [m1, f3], [m3, f4], [m4, f2]]
- In the solution, each member will only appear once, and males will always be paired with females.

For this question, you are not required to come up with a correct solution to the problem (in fact, existing solutions such as the Gale-Shapley algorithm are well known and well documented). Instead, you are required to verify if a particular proposed solution to the stable marriage problem is correct. A correct solution is one in which all pairs in the solution are stable.

Requirements:

Edit the function **is_stable** in **p1q2.py**. **is_stable** takes in **n**, the preferences, and a proposed solution to the problem. It is supposed to return **True** (if the solution is stable) or **False** (if not).

e.g.

```
>>> pref = [[m1, 1, 3, 2, 4], [m2, 1, 2, 4, 3], [m3, 1, 2, 3, 4], [m4, 2, 3, 1, 4], [f1, 2, 1, 3, 4], [f2, 4, 3, 2, 1], [f3, 1, 2, 3, 4], [f4, 3, 4, 2, 1]]
>>> solution_a = [[m1, f1], [m2, f3], [m3, f2], [m4, f4]]
>>> solution_b = [[m1, f3], [m2, f1], [m3, f4], [m4, f2]]
>>> is_stable(4, pref, solution_a)
False
>>> is_stable(4, pref, solution_b)
True
```

Q3: Lab 4 Extended! [5 marks]

This is an extension of lab 4, so you must be familiar with the question in lab 4. For this question, each Twitter user is given two additional attributes: a cost (**c**) and value (**v**). **c** is how much a user is asking to be paid in order to tweet your advertisement, and **v** is the approximated spending ability of the user based on his profile and income. We prefer users with high **v** to receive your advertisement.

Instead of selecting 5 users to tweet your advertisement, you are now given a budget to pay your advertisers. You can select any number of users to be your advertisers as long as their total **c** does not exceed the budget. Also, the objective of your marketing campaign has changed: in lab 4, you wanted to maximize the number of users who will get the message. For this question, you want to maximize the sum of **v** for all the users who will get your message.

e.g. 1: (n=11) Budget allocated = 10.

User ID	User IDs of Direct Followers	Cost	Value
0	2	1	9
1	0, 3	2	4
2	0, 1	1	7
3	1, 2, 4, 5	2	5
4	1, 6, 10	2	6
5	-	1	2
6	7, 8	7	5
7	-	2	2
8	-	2	2
9	8	5	2
10	9	3	4

e.g. 1(a):

This is an "incorrect" solution: **[3, 4, 6, 10]**. Total cost of these 4 users

= $c_3 + c_4 + c_6 + c_{10}$

= 2 + 2 + 7 + 3

= 14, which is > the allocated budget of 10.

Solutions which bust the allocated budget are considered incorrect.

e.g. 1(b):

This is a "correct" solution: **[0, 1, 2, 3, 4]**. Total cost of these 5 users

= $c_0 + c_1 + c_2 + c_3 + c_4$

= 1 + 2 + 1 + 2 + 2

= 8, which is <= the allocated budget of 10.

The additional users who will get the message are users 5, 6 and 10, making it a total of 5+3, or 8 users who will get the message. Remember that for this question, the actual number of users who get the message is irrelevant. We are interested in the total value of the users who get the message. For this case, total value =

= $(v_0 + v_1 + v_2 + v_3 + v_4) + (v_5 + v_6 + v_{10})$

= 9 + 4 + 7 + 5 + 6 + 2 + 5 + 4

= 42

The quality score for your solution will be the total value, and you strive to maximize the quality score.

Requirements:

Edit the function **select_advertisers** in **p1q3.py**. **select_ advertisers** takes in **budget**, **followers** (as for lab 4), **costs** (a list of costs for each user), and **values** (a list of values for each user). It is supposed to return a list of user IDs selected by your algorithm.

e.g.
```
>>> followers = [[2], [0,3], [0,1], [1,2,4,5], [1,6,10], [],
[7,8], [], [], [8], [9]]

>>> c = [1, 2, 1, 2, 2, 1, 7, 2, 2, 5, 3]
>>> v = [9, 4, 7, 5, 6, 2, 5, 2, 2, 2, 4]

>>> s = select_advertisers(10, followers, c, v)     # budget is 10
```

There are other functions used in **p1q3_main.py** that may be helpful. Check them out yourself.

Requirements for Written Report:

- You are required to submit ONE written report (in PDF format only) for the whole assignment to eLearn by the same deadline. Name your report **<Your name>.pdf**.
- Explain your algorithms adopted to approach Q2 and Q3. If possible, compute the time complexity of your algorithm and show how this value is derived. Besides your complexity calculation, you will be given credit for clarity of explanation, novelty of approach and evidence of analytical thinking. Use diagrams and examples where relevant. You are not required to explain Q1 in your report.
- The algorithms described in your report must be synchronized with the latest version of code submitted to the server.
- Your report must not be longer than 1 page (excluding the cover page, references and appendices, if any). It is one page for both Q2 and Q3; not one page per question. Content beyond 1 page may not be marked.
- Plagiarism is an offence. If you have used an idea from any source or obtained help from any person, you must acknowledge all sources and assistance in a reference section. Include URLs if relevant.

Scoring

This assignment is worth 10% of your final grade:

- Algorithm: Q1 (2%)
- Algorithm: Q2 (2%)
- Algorithm: Q3 (5%)
- Written report (1%)

Your algorithms will be scored in 3 ways:

(a) Correctness: your solution works as required and returns correct solutions for different test cases. Each function must each complete running within a fixed amount of time on the server (this time limit will be posted at red's "Announcement"). Additional test cases will be executed on the server after the deadline. Correctness is of highest importance. Only correct algorithms will be marked for speed and quality.

(b) Speed: the time taken for your function to complete. Speed is not taken into consideration for Q1 and Q2, as long as your function does not exceed the time limit on the server.
For Q3, you may be given credit for algorithms that have a good time complexity (and hence takes lesser time to complete for a large data set). The "time taken" on the scoreboard at red is merely a guide; after the deadline, your code may be executed using different data sets to derive the time taken.

(c) Quality is not taken into consideration for Q1 and Q2 (You will always see a quality score of "1.0" on the server if your solution is correct for Q1 and Q2).
For Q3, quality is important, and will make up the bulk of your score for Q3.

For all questions, an attempt is better than no submission. Attempts submitted to eLearn (together with your written report) may be manually marked if they do not show up as "correct" at red.

To Submit:

- Always refer to the "Announcements" at http://red.smu.edu.sg; clarifications, additional test cases and bug reports will be posted there.
- Submit to red:
  - You need to submit **p1q1.py**, **p1q2.py** and **p1q3.py** to red for automatic marking.
  - You can submit your solutions to red as many times as you wish, **but the final submission on the deadline will be taken as your final submission.**
  - **Do check all 3 scoreboards after your last submission to check that your name is on the scoreboards**. Your name will appear if you have submitted a correct answer.
- Submit to eLearn (Assignments):
  - Written report (PDF)
  - Identical copies of your final versions of **p1q1.py**, **p1q2.py** and **p1q3.py**.
- Late submissions will be penalized:
  - The time stamp on eLearn will be used to determine if a submission is late. You are supposed to submit BEFORE 11:00pm, so a time-stamp of 11:00pm is considered late.
  - The server will be closed for submission at the deadline. As long as you **submit** your code to red before the deadline, your submission will be considered "on time". Submissions that are in the server queue will be processed eventually.
  - If you are submitting late, you will have to contact Kwan Chin or Mok who will submit your solution on your behalf.
  - Penalty:
    - <1 hour late: penalty of 10%.
    - Submissions later than 1 hour will not be marked (you will get zero for that question).

~End