ASSIGNMENT - Week 6

COMP-204, Fall 2021, Section 001

Due: Oct **23th**, 2021 (23:59)

Please read the entire PDF before starting. You must do this assignment individually.

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. Following closely the instructions will allow the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. :)

- **Important:** All of the work you submit must be done by only you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. For Comp204, we will use software that compares programs for evidence of similar code. This software is very effective and it is able to identify similarities in the code even if you change the name of your variables and the position of your functions. The time that you will spend modifying a copied code, would be better invested in creating an original solution.

  Please don't copy. We want you to succeed and are here to help. Here are a couple of general guidelines to help you avoid plagiarism:

  Never look at another assignment solution, whether it is on paper or on the computer screen. Never share your assignment solution with another student. This applies to all drafts of a solution and to incomplete solutions. If you find code on the web, or get code from a private tutor, that solves part or all of an assignment, do not use or submit any part of it! A large percentage of the academic offenses in CS involve students who have never met, and who just happened to find the same solution online, or work with the same tutor. If you find a solution, someone else will too. The easiest way to avoid plagiarism is to only discuss a piece of work with the Comp204 TAs, the CS Help Centre TAs, or the Comp204 instructor.
- Your solution must be submitted electronically on Ed Lessons.
- To some extent, collaborations are allowed. These collaborations should not go as far as sharing code or giving away the answer. At the end of this file, I am provided a check-list with the activities and/or behaviours allowed during the development of this assignment. You must indicate on your assignments (i.e. as a comment at the beginning of your python source file) the names of the people with whom you collaborated or discussed your assignments (including members of the course staff). If you did not collaborate with anyone, you write "No collaborators". If asked, you should be able to orally explain your solution to a member of the course staff.
- This assignment is due on October 23nd at 11h59:59 pm. It is your responsibility to guarantee that your assignment is submitted on time. We do not cover technical issues or unexpected difficulties you may encounter. Last minute submissions are at your own risk.
- This assignment includes a programming component, which counts for 72% of the grade, and an "coding style" component (i.e., naming of variables, comments, clarity of your code etc), which counts for 28%.

- Multiple submissions are allowed before the deadline. Therefore, we encourage you to submit as early as possible a preliminary version of your solution to avoid any last minute issue. We will only grade the last submitted file. It is your responsibility to guarantee that your last submission is the right one (i.e., the one that you wanted us to mark).
- Late submissions can be submitted for 24 hours after the deadline, and will receive a flat penalty of 20%. We will not accept any submission more than 24 hours after the deadline. The submission site will be closed, and there will be no exceptions, except medical.
- In exceptional circumstances, we can grant a small extension of the deadline (e.g. 24h) for medical/personal reasons only. However, such request must be submitted before the deadline via e-mail to the instructor.
- Violation of any of the rules above may result in penalties or even absence of grading. If anything is unclear, it is up to you to clarify it by asking either directly the course staff during office hours, by email at the instructor or on the discussion board on Ed (recommended). Please, note that we reserve the right to make specific/targeted announcements affecting/extending these rules in class and/or on the website. It is your responsibility to monitor our channels of communications for announcements.
- The course staff will answer questions about the assignment during office hours or in the online forum on week days. We urge you to ask your questions as early as possible. We cannot guarantee that questions asked less than 24h before the submission deadline will be answered in time. In particular, we will not answer individual emails about the assignment that are sent the day of the deadline.
- You are provided some starter code that you should fill in as requested. Add your code only where you are instructed to do so. You can add some helper methods. Do not modify the code in any other way and in particular, do not change the methods that are already given to you, do not import extra code and do not touch the method headers. The format that you see on the provided code is the only format accepted for programming questions. **Any failure to comply with these rules will result in an automatic 0.**
- Public tests cases are available on Ed Lessons. You can run them on your code at any time. If your code fails those tests, it means that there is a mistake somewhere.
- Your code should be properly commented and indented.
- **Do not change or alter the name of the files you must submit, or the method headers in these files**. Files with the wrong name will not be graded. Make sure you are not changing file names by duplicating them. For example, diagnostic (2).py will not be graded.
- **You will automatically get 0 if the files you submitted on ed - Lessons do not compile, since you can ensure yourself that they do**.
- During the development of the assignment, you should only use techniques and/or topics that have been already explained during the lectures.

To get great marks, you must:

- Follow all directions above
- Code your solution using the template provided in mycourses.
- Make sure that your code compiles and it is correct.
  - Non-compiling code (code with syntax error) will receive a very low mark
- Write your name and student number in a comment in each .py file you hand in
- Write the names of the people with whom you collaborated or discussed your assignments
- Indent your code properly
- Name your variables appropriately

  - The purpose of each variable should be obvious from the name
  - Use a docstring in each function to comment and validate your program
- Comment your code properly
- Submit your code in ed - Lessons
  - Make sure that your code passes ALL the given test cases.

Part 0

Before starting the assignment, it is important that you complete the following steps. These steps are optional (and will not be graded), but we highly recommend you to go through them.

- For this assignment, we will use the online platform called Ed Lessons to facilitate the automatic grade of your work and to provide high-quality feedback on your programming work. The following are the basic steps that you will have to perform in ed Lessons.

  1. **Accessing Ed - Lessons:** Please make sure that you can access Ed Lessons. In order to login, you must use the link in the "Content" section of the "Discussion Board + Autograder" tab located in mycourses. Please notice that this is the same software that we are using as discussion board platform. In Ed, you must click in the "Lessons" button. Then, you need to click in the "Assignment_week4" link located inside the Module called "Assignments'.

  2. **Submitting work to Ed - Lessons:** Once you have finished the coding part of the assignment in your local machine, you will need to upload your .py file to ed - Lessons. We have already done that during our Laboratory of week 2. Please watch again the video of the laboratory (mycourses → Content → Friday Laboratory → Week2) if you need more information about it.
     For this assignment, you will be able to receive automated test results with your submission. Please double check that your code passes all the test cases.

Part 1

The questions in this part of the assignment will be graded.

The diagnostic (i.e., the process of determining which disease explain a person's symptoms) is a complex transition process that begins with the patient's individual symptoms and culminates in a result that can be classified (i.e., the disease). For the medical personnel, diagnosis is usually very challenging because a symptom can be the consequence of different diseases. The information required for diagnosis is typically collected from medical guidelines, and histories of people who have seek medical care before. As a Comp204 student, you have been commissioned (in assignment 6) to write a program that will help doctors make disease diagnostics based on a patient's symptom. This will be achieved by comparing a new patient's symptoms to a database of patients containing both their symptoms and their diagnostic. Given a patient X with a certain set of symptoms, your diagnostic will be obtained by identifying, among the patients in the database, those whose symptoms are most similar to those of patient X.

Data Representation

- Each patient is identified using an integer identifier (e.g. 56374).
- Each symptom is a string such as "headache" or "fever".
- Each diagnostic is a string such as "cold" or "meningitis".
- Symptoms for a given patient are stored in a `tuple` of two `sets`: the first `set` contains the symptoms that are present in the patient; the second `set` contains the symptoms that are observed not to be present (i.e. absent).
  - For example, a patient (called Yang) with coughing, runny nose, and sneezing, but no headache and no fever would be represented as:
    `({''coughing'', ''runny_nose'', ''sneezing''}, {''headache'',''fever''})`.
    Note that for a given patient, information may be missing about whether or not a patient has a symptom. For example, in the example above we don't know if the patient has a sore throat. Please see below for the data of three test patients.

```
# Three test patients
 yang = ({"coughing", "runny_nose", "sneezing"},{"headache","fever"})
 maria = ({"coughing", "fever", "sore_throat", "sneezing"},{"muscle_pain"})
```

```
    jaspal = ({"headache"},{"sneezing"})
```

- The set of symptoms of all the patients in our database is represented using a dictionary, whose keys are the patient identifiers, and values are the tuples of symptoms. For example:

```
  #A small dictionary of patients symptoms
   all_patients_symptoms = {
       56374: ({"headache","fever"}, {"coughing", "runny_nose","sneezing"}),
       45437: ({"coughing", "runny_nose"},{"headache","fever"}),
       16372: ({"coughing", "sore_throat"},{"fever"}),
       54324: ({"vomiting", "coughing","stomach_pain"},{"fever"}),
       73454: ({"coughing", "runny_nose"},{"headache","fever"}),
       35249: ({"sore_throat", "coughing","fever"},{"stomach_pain", "runny_nose"}),
       44274: ({"fever", "headache"},{"stomach_pain", "runny_nose","sore_throat",
           "coughing",}),
       74821: ({"vomiting", "fever"},{"headache"}),
       94231: ({"stomach_pain", "fever","sore_throat","coughing","headache"},{"vomiting"}),
    }
```

- The diagnostic given to each patient in our database (e.g., patients who has visited the hospital before) is stored in another dictionary, with keys equal to the patient identifiers and values corresponding to a string. For example:

```
   # A small dictionary of patients diagnostics
   all_patients_diagnostics = {45437: "cold", 56374:"meningitis", 54324:"food_poisoning",
       16372:"cold", 73454: "cold", 35249:"pharyngitis", 44274:"meningitis",
       74821:"food_poisoning", 94231:"unknown"}
```

The protocol: The diagnostic protocol will follow these steps:

1. (Question 1) To create a function called `symptom_similarity` to compute a similarity measure between the set of symptoms (and absent of symptoms) of two patients.
2. (Question 2) To create a function called `similarity_to_patients` which uses the function `symptom_similarity` to select the patients (from the entire population of patients in our database) with the highest similarity between the their symptoms and the symptoms of one patient.
3. (Question 3) To create a function called `getting_diagnostics` that finds the most frequent diagnostic from the patients with the highest symptoms similarity returned by the function `similarity_to_patients`.

For this assignment, your work will be to implement the aforementioned three functions. Please note that the functions must be implemented in the given order because the dependencies between them. Also note that the docstring of the functions in the following section contains the type-contract, the description of the function and one example (using the data of the section "Data Representation") to test your function. You are invited to create more examples to test your functions.

Question 1: symptom_similarity (25 points)

Complete the `symptom_similarity` function, which measures the similarity between the symptoms of two patients. See below for an explanation of how the similarity is computed.

```
def symptom_similarity(symptoms_A: Tuple[Set], symptoms_B: Tuple[Set]) -> int:
    """
    Returns the similarity between symptoms_A and symptoms_B. symptoms_A and
    symptoms_B are tuples of a set of symptoms present and a set of symptoms absent.
    The similarity measure is computed by the following equations:
    present_present + absent_absent - present_absent - absent_present
    where:
    present_present is the number of symptoms present in both patients
    absent_absent is the number of symptoms absent in both patients
    present_absent is the number of symptoms present in patientA and absent in patientB
    absent_present is the number of symptoms absent in patientA and present in patientB

    >>>symptom_similarity(yang, maria)
    1

    """
```

Please note that the call of the function `symptom_similarity(yang, maria)` returns 1. The symptoms of yang (i.e., `symptoms_A` in the `symptom_similarity` function) is represented by the `tuple` `({"coughing", "runny_nose", "sneezing"},{"headache","fever"})`. The symptoms of maria (i.e., `symptoms_B` in the `symptom_similarity` function) is represented by the `tuple` `({"coughing", "fever", "sore_throat", "sneezing"},{"muscle_pain"})`. Then, for this example, the variable `present_present` will be equal to 2 because there are two symptoms (i.e., `"coughing"` and `"sneezing"`) that are present in both patients. The variables `absent_absent` and `present_absent` will refer to 0 because there are no symptoms absent in both patients, and there are no symptoms present in patientA and absent in patientB, respectively. The variable `absent_present` will point to the value 1 because there is one symptom (i.e., `"fever"`) that is absent in patientA and present in patientB. Then, the equation to measure the similarity of symptoms (i.e. `present_present + absent_absent - present_absent - absent_present`) will be equal to (i.e. `2 + 0 - 0 - 1 = 1`) one (i.e., the returned value).

Question 2: similarity_to_patients (25 points)

Complete the `similarity_to_patients` function, which uses the function `symptom_similarity` to select the patients (from the entire population of patients in our database) with the highest similarity between the their symptoms and the symptoms of one patient. See below for an explanation of exactly what is expected.

```
def similarity_to_patients(my_symptoms : Tuple[Set], all_patients: Dict[str, Tuple[Set]]) ->
     Set[int]:
    """
    returns a set of the patient IDs with the highest similarity between my_symptoms
    (which is a tuple of symptoms present and absent) and the symptoms of
    all the patients stored in the dictionary all_patients.

    >>> similarity_to_patients(yang, all_patients_symptoms)
    {45437, 73454}

    """
```

Please note (in the example shown in the docstring) that comparing the symptoms of the patient `yang` (see "Data Representation") with the data base `all_patients_symptoms` (see "Data Representation") returns a `Set` comprised by the two patients IDs (`{45437, 73454}`) with the highest symptoms similarity measure (there is a tie between the values of those two patients).

Question 3: getting_diagnostics (25 points)

Complete the `getting_diagnostics` function, which reports the most frequent diagnostic from the patients with the highest symptoms similarity returned by the function `similarity_to_patients`. See below for an explanation of exactly what is expected.

```
def getting_diagnostics(patient_set : Set[int], diagnostic_by_patient: Dict[int, str]) -> str:
    """
    Returns a string representing the most frequent diagnostic from
    the set of diagnostics (stored in the dictionary diagnostic_by_patient)
    of the patient_set (that is a set of ID patients with high similar symptoms)
    >>> getting_diagnostics({45437, 73454}, all_patients_diagnostics)
    'cold'

    """

    #To initialize an empty dictionary. diagnostic_frequencies will have as key the diagnostic
        (i.e., disease) and as value an integer representing the frequency count (i.e., the
        number of patients in the set with the same diagnostic)
    diagnostic_frequencies = {}
    #To loop over all the patients with high similarity in their symptoms.
    for ID in patient_set:
        #To extract the diagnostic of each patient
        diag = diagnostic_by_patient[ID]
        #If it is the first time that I see this specific diagnostic
        if diag not in diagnostic_frequencies:
            #Initialize in one my counter
            diagnostic_frequencies[diag] = 1
        else:
            #else, increase my counter
            diagnostic_frequencies[diag] += 1
    #Now, you will need to extract from the diagnostic_frequencies dictionary the key with the
        highest frequency
    #Missing code starts here!!!!!!!
```

Please note (in the example shown in the docstring) that `cold` is the most frequent diagnostic given the two patients IDs ({`45437, 73454`}) (Please see that both patients have a diagnostic of `cold` in the `all_patients_diagnostics dictionary`) who correspond to the highest symptoms similarity measure when comparing the symptoms of the patient `yang` with the data base `all_patients_symptoms`. Also, please note that there is some code already provided in this function. I started working on the function; however, I got busy and I could not finish the coding. So far, I created a dictionary called `diagnostic_frequencies`, which has as `key` the diagnostic (i.e., disease) and as `value` an integer representing its frequency count (i.e., the number of patients in the set with the same diagnostic). Your job for this question is to complete the coding by extracting and returning from the `diagnostic_frequencies` dictionary the key with the highest frequency.

Question 4: protocol (0 points)

The function `protocol` implements the functionality of the diagnostic protocol followed in this assignment (see Section "The protocol:"). The main idea of this function is to call the functions that you have already implemented in the previous questions. See below for an explanation of exactly what the function does.

```
def protocol(my_symptoms: Tuple[Set], all_patients_symptoms: Dict[str, Tuple[Set]],
     all_patients_diagnostics: Dict[int, str]) -> str:
    '''Return the diagnostic for my_symptoms based on the dictionary of patients symptoms
        (all_patients_symptoms) and the dictionary of patients diagnostics
        (all_patients_diagnostics).
     >>>protocol(yang, all_patients_symptoms, all_patients_diagnostics)
    'cold'
    '''
    #The following statement selects the patients with the highest similarity between the their
        symptoms and my_symptoms (by calling the function similarity_to_patients)
    similiar_patients = similarity_to_patients(my_symptoms, all_patients_symptoms)
    #Then, it reports the the most frequent diagnostic from the patients with the highest
        symptoms similarity (by calling the function getting_diagnostics)
    diagnostic = getting_diagnostics(similiar_patients, all_patients_diagnostics)
    return diagnostic
```

Please notice that we have already implemented the function for you. You do not have to do anything for this question :) (it is there only to understand the full logic of the program and to be able to test it).

Question 5: my_test (0 points)
The function `my_test` implements the following data structures.

1. A small dictionary of patient's symptoms
2. A small dictionary of patient's diagnostics
3. Three test patients

We have also implemented some calls for the functions (i.e., the test open cases in Ed-Lessons) to give you a starting point to test your functions. Please note that this function will not be graded, and it is there only to make sure that you understand what every function is expected to do and to test your own code. Note: In order for you to test your functions one at a time, comment out the portions of the `my_test()` function that call functions you have not yet written.

What To Submit?

Attached to this assignment is a file called `diagnostic.py`. PLEASE note that the code outside the body of the function must not be modified. You have to submit only this `diagnostic.py` file. Please DO NOT zip (or rar) your files, and do not submit any other files.

Where To Submit?

You need to submit your assignment in ed - Lessons. Please review the Section 0 if you still have questions about how to do that. Please note that you do not need to submit anything to myCourses.

When To Submit?

Please do not wait until the last minute to submit your assignment. You never know what could go wrong during the last moment. Please also remember that you are allowed to have multiple submission. Then, submit your partial work early and you will be able to upload updated versions later (as far as they are submitted before the deadline). If you could not finish the assignment, also submit what you have, beside of having partial marks, you will also get feedback from your T.A to avoid the same errors in future submissions.

How will this assignment be graded?

Each student will receive an overall score for this assignment. This score is called the "Combined score" and it is the combination of the correctness and style scores. For this assignment, the correctness and style scores have a weight of 72% and 28%, respectively.

- **Correctness score:** This score is between 0% and 72% depending on how many test cases your code passed. Question 1 to 3 have a value of 24% each. For each question, we have created 6 test cases (3 open cases and 3 blind cases) each with a weight of 4%. The open cases will be run with-in your submissions and you will receive automated test results (i.e., the autograder output) for them. You MUST guarantee that your code passes these cases. The open cases for this assignment correspond to the function calls found in the function `my_test`. The blind test cases are inputs that you have not seen and they will test the correctness of your algorithm on those inputs once the deadline of the assignment is over.

- **Style score:** This score is between 0% and 28%. Your submission will be reviewed by your assigned T.A. The T.A will evaluate your coding style (e.g., indentation, variable naming, comments, coding style etc.) and they will give you feedback on it.

Student Code of Conduct Assignment Checklist

The instructor provides this checklist with each assignment. The instructor checks the boxes to items that will be permitted to occur in this assignment. If an item is not checked or not present in the list, then that item is not allowed. The instructor may edit this list for their case. A student cannot assume they can do something if it is not listed in this checklist, it is the responsibility of the student to ask the professor (not the TA).

Instructor's checklist of permitted student activities for an assignment:

Understanding the assignment:

- ☒ Read assignment with your classmates
- ☒ Discuss the meaning of the assignment with your classmates
- ☒ Consult the notes, slides, textbook, and the links to websites provided by the professor(s) and TA(s) with your classmates (do not visit other websites)
- ☐ Use flowcharts when discussing the assignment with classmates.
- ☒ Ask the professor(s) and TA(s) for clarification on assignment meaning and coding ideas.
- ☐ Discuss solution use code
- ☐ Discuss solution use pseudo-code
- ☐ Discuss solution use diagrams
- ☐ Can discuss the meaning of the assignment with tutors and other people outside of the course.
- ☐ Look for partial solutions in public repositories

Doing the assignment:

- Writing
    - ☒ Write the solution code on your own
    - ☒ Write your name at the top of every source file with the date
    - ☐ Provide references to copied code as comments in the source code (e.g. teacher's notes, websites, etc.)
    - ☒ Copied code is not permitted at all, even with references
    - ☐ Permitted to store partial solutions in a public repository
- Debugging
    - ☒ Debug the code on your own
    - ☒ Debugging code with the professor
    - ☒ Debugging code with the TA
    - ☒ Debugging code with the help desk
    - ☐ Debugging code with the Internet
    - ☐ You can debug code with a classmate
    - ☐ You can debug code with a tutor or other people outside of the course
- Validation
    - ☒ Share test cases with your classmates
- Internet
    - ☐ Visit stack-overflow (or similar)
    - ☐ Visit Chegg (or similar)

- Collaboration
    - [ ] Show your code with classmates
    - [ ] Sharing partial solutions with other people in the class
    - [ ] Can post code screenshots on the course discussion board
    - [ ] Can show code to help desk

Submitting and cleaning up after the assignment:

- [ ] Backup your code to a public repository/service like github without the express written permission from the professor (this is not plagiarism, but it may not be permitted)
- [ ] Let people peek at your files
- [ ] Share your files with anyone
- [ ] ZIP your files and upload to the submission box
- [x] Treat your work as private
- [ ] Make public the solutions to an assignment
- [ ] Discuss solutions in a public forum after the assignment is completed