Student name: **Thuong Phan**
Netid: **thphan2**
Credit hour: **4**

IS452 – Final Project

1. Project Overview

In the final project, I want to build a Python program which can provide different functionalities for a paper collection such as building the collection automatically add paper, update paper, delete paper and search for paper in the collection as well as extract data out of the collection.

2. Module requirement

List of built in modules in PyCharm required for the project: **os, csv, re.**

```
import os
import csv
import re
```

List of modules need to be installed in PyCharm so that the program can be run: **matplotlib, networkx.**

```
import networkx as nx
import matplotlib.pyplot as plt
```

3. Datasets

The datasets used for the final project contain 8 ASIST conference papers belonging to Library and Information Science Source downloaded from EBSCOhost http://search.ebscohost.com
Each paper is downloaded both fulltext in pdf format (saved in **Datasets/pdf/**) and csv file (saved in **Datasets/csv/**) containing basic information about the paper such as Author, Article Title, Publication Date, ISBN, ISSN,…
The pdf fulltext is then copied into txt file (saved in **Datasets/txt/**) for processing in the Python program.
Both txt and csv files should be named similarly; for example, Chong2016.txt and Chong2016.csv.

4. Program design

The below flowchart shows the design of the final project.

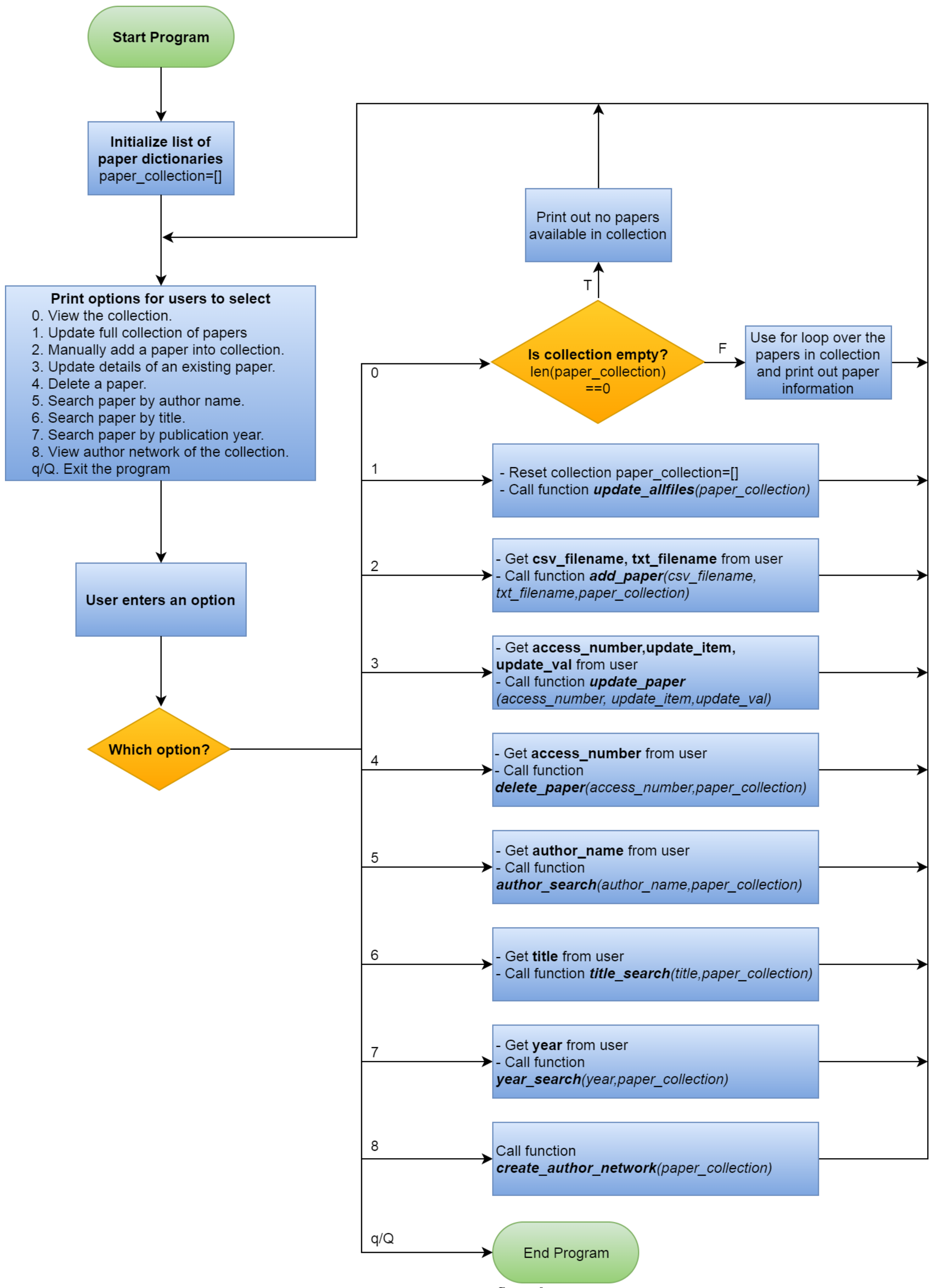
Start Program
Initialize list of paper dictionaries paper_collection=[]
Print options for users to select
0. View the collection.
1. Update full collection of papers
2. Manually add a paper into collection.
3. Update details of an existing paper.
4. Delete a paper.
5. Search paper by author name.
6. Search paper by title.
7. Search paper by publication year.
8. View author network of the collection.
q/Q. Exit the program
User enters an option
Which option?
0
Is collection empty? len(paper_collection) ==0
T
Print out no papers available in collection
F
Use for loop over the papers in collection and print out paper information
1
- Reset collection paper_collection=[]
- Call function update_allfiles(paper_collection)
2
- Get csv_filename, txt_filename from user
- Call function add_paper(csv_filename, txt_filename,paper_collection)
3
- Get access_number,update_item, update_val from user
- Call function update_paper (access_number, update_item,update_val)
4
- Get access_number from user
- Call function delete_paper(access_number,paper_collection)
5
- Get author_name from user
- Call function author_search(author_name,paper_collection)
6
- Get title from user
- Call function title_search(title,paper_collection)
7
- Get year from user
- Call function year_search(year,paper_collection)
8
Call function create_author_network(paper_collection)
q/Q
End Program

Figure 1. Project flowchart

5. Implementation

5.1. Small functions

These functions are used to build up bigger functions used for each user choice, they include tasks such as processing csv file, processing txt file, check whether a paper is available, find a paper based on a key (paper title, author name or publication year) and plot the author connection for a specific author.

a. process_csv(filename)

The csv file in our dataset acts like a metadata file, providing basic information about a paper such as Article Title, Author, Publication date, Accession Number, Abstract, …
process_csv() function takes *filename* as a parameter and the purpose of this function is to return a dictionary with the keys as the column names in the csv file and the key values as the respective column values.
To achieve this purpose, first we read in the file using *csv.reader()*; the first line (column names) stored in a list *keys_list*. We do some refinement here to remove the special character "ï»¿". Then read in the next line which contains the column values and stored them in a list *vals_list*.
A dictionary *paper_dict* is initialized as an empty dictionary
For each pair of items in *keys_list* and *vals_list*, we add them into dictionary with the key as element in *keys_list* and value as the corresponding element in *vals_list.* Because for columns/keys "Publication Date" and "ISSN", the values are displayed in format *=“<value>”*; hence, we also remove = and *“* characters so that only the <*value*> left by using regex sub() function to replace = or *“* with an empty string.
Finally the dictionary *paper_dict* is returned.

b. process_txt(filename, paper_dict)

The process_txt() function takes 2 parameters – txt filename and the dictionary created after processing csv file. This function is used to process txt file (or the fulltext of the paper) and extract several parts of the paper and add them in the dictionary created after csv file processed.
The txt file is read in using read() to create a string containing the whole content of the paper and store it in a string lines. There are 5 parts we want to extract out of the paper including INTRODUCTION, METHODOLOGY, RESULTS, CONCLUSION and REFERENCES. Although each paper might have different naming convention for the above parts (Eg: PROPOSED METHOD, METHODS and METHOD all refer to METHODOLOGY), we can group those similar names in a synonym list and replace them with a general name using re.sub().
The difficult part of processing txt file containing full text if each paper is determining the start position and end position of the above parts in the paper.
In the paper collection used in the project, all the papers have similar structure with the order of INTRODUCTION, LITERATURE REVIEW, METHODOLOGY,RESULTS, CONCLUSION then REFERENCES. With this order, we can specify the content of each part; for example, start position of INTRODUCTION is the index return by *find()* function of the string lines and end position of INTRODUCTION or the start position of LITERATURE REVIEW is the index turn by *find()* for the string "LITERATURE REVIEW" in the string lines. We apply similar method for other parts of the paper except some difference in the last part – REFERENCES. The end position for REFERENCES should be the index of the string *“Copyright of Proceedings of the Association").*
We add above parts as the keys of the *paper_dict* dictionary and by slicing the original string *lines* using the start and end indexes for each part, we can find the content for each part and add them into the dictionary as the values of the keys.

```
paper_dict["INTRODUCTION"]=lines[intro_start:intro_end]
paper_dict["METHODOLOGY"]=lines[method_start:result_start]
paper_dict["RESULTS"]=lines[result_start:conclusion_start]
paper_dict["CONCLUSION"]=lines[conclusion_start:ref_start]
paper_dict["REFERENCES"]=lines[ref_start:ref_end]"
```

We don't need to return *paper_dict* in this function as it's a parameter of the function and its value automatically updated after the function is executed. This function can be considered as a call back function.

c. **check_avail(access_number,paper_collection)**
check_avail() function takes 2 parameters: accession number of a paper *access_number*, and the current paper collection *paper_collection.*
The purpose of this function is to check if an accession number is already available in the whole current collection; on the other, whether a paper is already available in the collection. First, we get the list of accession numbers of the current collection by getting the value for the key "Accession Number" for each dictionary (or paper) in the collection *paper_collection.*
The function returns True if the accession number is already in the accession number list; otherwise, returns False.

d. **find_paper(paper_collection, key, value)**
find_paper() takes 3 parameters: the current collection *paper_collection*, key name and its value. This function is used to find a paper based on a key value. For example, to find a paper having the key "Accession Number" with value of "12345" we use:
find_paper(paper_collection, "Accession Number", "12345")
This function returns a subset of the paper collection (a list of dictionaries) in which each paper has value of the mentioned key name contains the input key value (case insensitive).

e. **plot_network(author,all_edges)**
This function takes 2 parameters, an author name and the list of all tuples; each tuple *(a1, a2)* contains 2 authors *a1* and *a2* connected to each other by the relationship *"a1 cited a2's paper"*.This function is used to get the list of authors connected to the input author then plot a network graph to show their connection (who cited who)
First, create the *subset_edges* as filtering the original *all_edges* by the condition that first author in an edge (citing author - *a1*) equal to the input author or the second author in an edge (cited author – *a2*) equal to the input author.
We create a network directed graph object G. To get the nodes required for network graph, we need to get a list of authors related to the input author and the input author himself without any duplication. To do that, we can initialize an empty *subset_nodes*; for each pair/tuple of citing author and cited author in *all_edges*, add both the citing and cited author into *subset_nodes*. After the loop is executed, add the set of the *subset_nodes* as the graph nodes by function *G.add_nodes_from()*. Also add the *subset_edges* into graph edges by function *G.add_edges_from()*. Finally, use *nx.draw_networkx()* to plot the network graph.
Below is an example of the author network for the author Blake, C. (position number: 16)
The graph shows that Blake C.'s paper was cited by Cohen, D., Stoytcheva, S. and even herself in another paper.

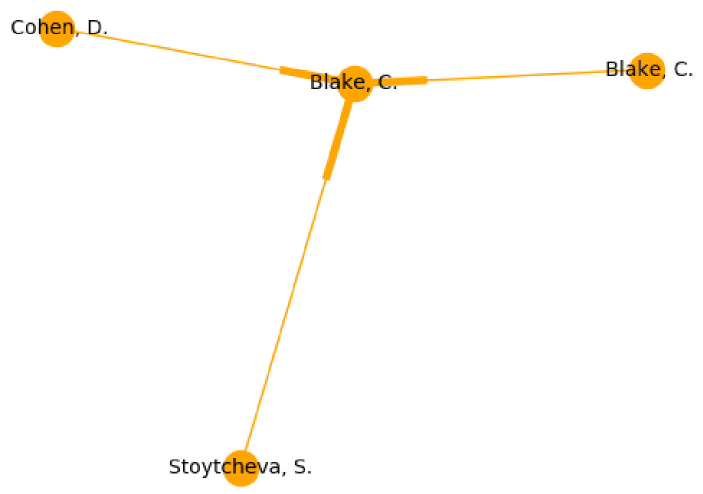

Figure 2. Author Network for "Blake, C."

5.2. Big functions

These are the functions we call directly for each option of the program such as creating the whole collection of paper, adding a new paper, updating an existing paper, deleting an existing paper, searching for paper based on some specific key and creating author network for a specific author.

a. update_allfiles(paper_collection) function

This function is used to update the whole paper collection.
The paper collection is stored in a list of dictionaries paper_collection. Each dictionary corresponds to a paper containing 25 below keys and their related values.

- 'Article Title': the title of the article/paper
- 'Author': author name(s). In case there are more than 1 author, they are separated by colon ";"
- 'Journal Title'
- 'ISSN'
- 'ISBN'
- 'Publication Date'
- 'Volume'
- 'Issue'
- 'First Page'
- 'Page Count'
- 'Accession Number': a unique number assigned for each paper
- 'DOI'
- 'Publisher'
- 'Doctype'
- 'Subjects'
- 'Keywords'
- 'Abstract'
- 'PLink'
- 'INTRODUCTION': introduction part of a paper
- 'METHODOLOGY': method/methods/methodology/proposed method part of a paper
- 'RESULTS': results and discussion/preliminary results/experiment and results analysis/experiment/result of a paper

- 'CONCLUSION': conclusion/conclusions/conclusion and future work of a paper
- 'REFERENCES': reference sources of a paper

This function takes *paper_collection* as a parameter and is used like a call back function. When a user wants to update all the papers in the collection, *paper_collection* is reset to an empty list and this function is called. First, a for loop to iterate over the csv files in Datasets/csv folder is created. For each csv file, *process_csv(filename)* is called to process the csv file and return a dictionary of a paper *paper_dict*. From the csv filename, get the name of txt file (Note that both csv and txt should have same names, just different file type) and use the generated txt filename and the above returned dictionary *paper_dict* to pass in the function *process_txt(filename, paper_dict)* to update *paper_dict* with new keys and key values. Finally, *paper_dict* is appended into the list of paper dictionaries *paper_collection*.

b. add_paper(csv_filename,txt_filename,paper_collection)

This function is used to add a new paper into the collection and takes 3 parameter: csv filename, txt filename and the list of paper dictionaries *paper_collection.*

Before adding the paper into current collection, we need to check if the paper is already available in the collection or not using function *check_avail(paper_collection)*. If *check_avail()* function returns True, inform the user that the paper is already in the collection; otherwise, start adding it in the collection.

To add a paper into the collection, it's quite similar to the step to update the full collection; we use *process_csv(csv_filename)* to create the dictionary for the paper, then use *process_txt(txt_filename,paper_dict)* to update the generated paper dictionary with extra keys and values. However, the *csv_filename* and *txt_filenames* do not require having same name or in Datasets/csv and Datasets/txt folders, we can specify it manually; but it is recommended to follow the naming convention to maintain the consistency. Finally, append the paper dictionary into *paper_collection* list.

c. update_paper(access_number,update_item,update_val)

This function is used to update value for a specific key of a paper in the paper collection. It requires 3 parameters – accession number of the paper to be updated *access_number*, the key name *update_item*, and the value of the key to be updated *update_val.*

First, we also need to check if the paper is available in the collection by passing the accession number into check_avail() function and also if the key name *update_item* is available in the list of keys of each dictionary in the paper collection. If the check condition returns False, inform the user that the paper or the required key is not available in the collection for update. Else, use a for loop to iterate over the papers within the collection, if the accession number of the paper equals to the input accession number, set the value of the input key *update_item* of that paper to the input *update_val.*

d. delete_paper(access_number,paper_collection)

This function is used to delete an existing paper from the current paper collection. It requires 2 parameters – accession number of the paper to be deleted *access_number*, the current paper collection *paper_collection*.

First, we also need to check if the paper is available in the collection by passing the accession number into *check_avail()* function; if the check condition returns False, inform the user that the paper is not available in the collection for delete. Else, use a list comprehension to update the *paper_collection* with the list of paper in original *paper_collection* but excluding the one having accession number equal to the input accession number. Finally, return *paper_collection* so that we get the updated collection after deleting a paper or the original one if no changes applied.

e. author_search(author_name,paper_collection)
title_search(title,paper_collection)
year_search(year,paper_collection)

These three functions are quite similar to each other as they all call the small function *find_paper(paper_collection, key, value)*. Key parameter changes for each case; search by author uses *key=‘Author’*, search by article title uses *key=‘Article Title’*, search by publication year use *key=‘Publication Date’*
Before calling the 3 functions, the user should provide the key value they want to search for. For example, to find the list of papers written by author name ‘Chong’, `author_search(“Chong”,paper_collection)` is used. Then within this function, the function `find_paper(paper_collection, “Author”, “Chong”)` is called and returns the filtered list of dictionaries that ‘Chong’ is included in the key ‘Author’ of each paper dictionary.

f. create_author_network(paper_collection)

The function requires the list of paper dictionaries *paper_collection* as its parameter. From the *paper_collection* we create a list of edges(each edge is a tuple of 2 authors *(citing author, cited author)*). We also need a list of all authors so that the users will key in the author’s position number in the list to select the author they want to see the network.
To do this, we first initialize two empty list *author_nodes* and *author_edges*. For each paper in the paper collection (by using for loop over *paper_collection* with variable *paper*), get the list of cited authors *cited_authors* by extracting it from *REFERENCES* key value with the help of regex pattern `"([A-Z][a-z]+, ([A-Z]\.\s?)+)"`. Next, use a for loop to iterate over the list of cited authors, within the loop body, append the cited author into *author_nodes*. Then create another for loop for each author in the list of paper authors (citing authors) - `paper["Author"].split(";")`. Because the author name of the paper is given in full name format instead of the short form *“<Lastname>,<Initial>”*, we need to clean it up so that the format of citing authors and cited authors match. This also prevents us to create duplicates; for example, Blake, Catherine and Blake, C. should refer to same person. The citing author name is transformed by getting only the uppercase character after commas character “,” in the full name using *re.findall()* for regex pattern `"[A-Z]"` and store the list of initials in *upper_chars*. Then the updated author name is formed by the substring in original name from beginning to the “,” character + the string created by joining “.” With above *upper_chars* (For example: Blake, Catherine → Blake, C.). After getting the updated author name, we append it into *author_nodes*. Next, create an inner for loop to iterate over the cited authors and add the pair of *(citing author, cited author)* into *author_edges* list.
After all paper in *paper_collection* are processed above steps, we have the *author_nodes* and *author_edges* created. However, *author_nodes* might have some duplications; hence, we use *set()* to get only the unique author names, make it a list again with *list()* then order it in alphabetical order. Finally print out all the author option with their position number into screen for the user to enter the author position number and store it in *selected_author*. Use

function `plot_network(author_nodes[selected_author],author_edges)` to generate the author network for the selected author by the user.

6. References

IS452 course materials
EBSCOhost - *http://search.ebscohost.com*
draw_networkx documentation - *https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.drawing.nx_pylab.draw_networkx.html*
Flowchart Maker - *https://www.draw.io/*