CS 1026B – Assignment 3

Due: March 22, 2022 – 6:00 PM

Overview

Write a program using data structures, functions and reading data from a file to simulate a preferential voting system.

Reminders

- Your code must be done individually.
- Your code will be graded in part by an automated system.
- Your code may be compared to other submissions using computer software.
- You can submit your code up to 48 hours late, with a deduction of 0.5% per hour (or part of an hour) that the assignment is late.

Background

Canadian provincial and federal elections use a 'first-past-the-post' system, where every voter casts a vote for one candidate in their riding and the candidate with the most votes wins. In this way, every riding elects one representative and the party that elects the most representatives (typically) forms the government. When there are more than two candidates in a riding, it is possible that the candidate who wins has not received votes from the majority of voters.

Figure 1: hypothetical ranked choice ballot (from https://commons.wikimedia.org/wiki/File:Preferential_ballot.svg)

Other systems for elections with more than two candidates exist to attempt to represent the preferences of voters. One system is called "Instant Runoff Voting" (IRV). IRV may also be called other names like Ranked Choice Voting or Alternative Voting.

In this system, each voter is given a ballot where they can rank as many of the candidates as they like with numbers. The candidate a voter marks with 1 in their most preferred: that voter most wants the candidate marked with 1 to be elected. Lower numbers represent lower preferences: "If my first-choice candidate is not elected, then I would prefer this candidate as my second choice". A hypothetical example of a filled ranked ballot is shown in Figure 1. As we see, the voter does not need to rank all candidates. Any candidate not ranked by a voter is ignored.

IRV is used in several elections worldwide: Australia uses IRV (and similar variants) in federal and state elections, political parties in Canada use it to elect leaders and the City of London used it to elect

councillors in 2018. For a short video introduction to IRV (where it is called *Alternative Voting*), here's a 5-minute video.

To count votes in an IRV election the following process is used:

1. The number of *first place* votes is counted for all the candidates.
2. If any candidate has more than 50% of the first-place votes, that candidate is declared the winner of the election.
3. Otherwise, the candidate **C** with the *least* number of first place votes is eliminated.
    a. When doing so, for every voter **V** that preferred candidate **C** (whether in first place or another place), those votes are eliminated, and any candidate that voter **V** preferred less than candidate **C** gets their preferences increased to replace the eliminated candidate **C**.
    b. For example, if a voter ranked a candidate **C** in 3rd place and candidate **C** was eliminated, then the candidate they ranked in 4th place is now their 3rd place candidate, and the same occurs for their old 4th place candidate, 5th place candidate, etc.
    c. If a tie exists between the candidates with the least number of votes exists, then we break that tie (our rules will be described in the section Program below)
4. After eliminating candidate **C** and all their votes, the number of first place votes for all the remaining candidates is recalculated. We then return to step 2.
5. The only outcomes are:
    a. There is one candidate that remains, with more than 50% of the vote. That candidate is declared the winner.
    b. There are exactly two candidates remaining, both with exactly 50% of the vote. We break this tie to declare the winner (our rules will be described in the section Program below).

Input Files

Your program to read a data file that represents anonymous votes for candidates in an IRV system. The file gives information on all votes cast in a riding.

Candidates in the voting system are represented by a number (their ID) rather than their name. The candidates in the election are all the candidates that exist in at least one vote in the election (we assume every candidate is ranked by at least one candidate).

The file will be a CSV file where each row of the file represents one vote. The information in a row is a sequence of numbers representing the preferences of this vote, in order (left-to-right) from highest preference (first choice) to lowest preference. Votes may not rank all candidates, but each vote will rank at least one candidate.

For example, if a vote was for the candidates with ids 35, 42, 104 and 16 (in that order), then the row of the file would be

35,42,104,16

You can assume that there are no errors in the files. The data will consist only of numbers that represent IDs. You do not need to verify whether the data consists of valid integers. You can also assume that each

vote ranks at least one valid candidate and that each vote does not rank the same candidate more than once (so a row like 3,4,3,1 will never appear in the input file).

Program

Your program should perform the following tasks:

1. Ask the user for the name of the file, using the prompt "Enter the name of the file: " (with a space after the colon).
    a. You can assume that all files are in the same directory as the code.
    b. You can assume the file exists, and contains valid data. You do not have to have any code to deal with files that do not exist, etc.
2. Determine the winner of the election. Use the process described above, but with two tiebreaking methods:
    a. If two or more candidates have the lowest number of first place votes, then the candidate with the **highest** id is eliminated.
    b. If two candidates are tied at the end of the process, the candidate with the **lower** id is declared the winner.

    We assume that this will be a fair process as each candidate will be randomly assigned IDs, so this process will be the same as a coin toss to break ties.

3. After determining the winter, print the order of elimination for all the candidates, from first to last. In particular,
    a. The first candidate eliminated should be printed first, followed by the rest in order of elimination.
    b. If multiple candidates are eliminated in one step (e.g., one candidate has more than 50% and all other remaining candidates are eliminated), then the candidates eliminated at one step should be listed in order of **increasing** IDs (i.e., smallest ID eliminated should be printed first).
    c. The last candidate shown should be the winner of the election.
    d. The format for the output should be as follows:

        ```
        Elimination order: ID1, ID2, ID3, …, IDn
        ```

        where IDx is a number of a candidate eliminated. There is a single space after each comma and the colon. IDn is the candidate that is eliminated.

Data Structures

There are no formal requirements for the data structures in the assignment beyond using data structures to manage your data.

There are several ways to store the data in this assignment. Here are some suggestions to help you organize your code

1. A vote can be represented as a list of numbers (the IDs of the candidates that have been ranked by that vote).

2. The information for the votes in the election can be represented as a list of lists, that is, the list of all votes that have been cast. This information can be updated (by removing as the process of determining the winner continues.
3. The current status of vote counting can be represented as a dictionary, where the key is the candidate ID and the value is the percentage of first-place votes that candidate has.

Functions

You are required to break your program into functions. This includes these functions:

1. A function to read the input from the file and store it in a data structure. (Parameters: file name. Return value: data structure with all votes)
2. A function to determine which candidate has should be eliminated in a round, which incorporates the tiebreaking method.
3. A function to update the votes in the election by eliminating one candidate.
4. A function to determine the order of elimination of candidates in an election.
5. A main function.

For each of these, you should have appropriate parameters and return values. You should not use global values in your functions: all data needed by a function should be communicated with parameters and return values. Except for main, all functions can be given any suitable name.

Code Requirements

Your code must satisfy the following requirements in addition to producing correct output.

1. Your code must be documented appropriately. Do not document every line of code, but major portions of your code must be commented.
2. Your code must include a comment at the top of program that includes your name and describes the overall function of the program.
3. Your code should use appropriate structures, such as loops, lists and functions. Functions must be used based on the requirements above. You should also use a main() function.
4. You should give a docstring for each function you write (except for main). See the programming standards for more details.
5. You should review the programming standards document for information on commenting, variable naming, documentation of functions, and other issues.

Submission Details

- You must submit your code to gradescope using the instructions on owl.
- You must name your code Assign3.py.
- You must follow the format of the output shown in the example below, including the prompt.

Assignment Marking

- Your code will be marked by an automated tool. The testing program assumes that
    - The code is saved as a file called Assign3.py
    - You are using a python3 version (recommended python3.9 or higher). If you use python2, you will likely lose marks.

If your code has to be marked manually due to failure to follow these instructions, you will receive a deduction from your final mark.

- Your code will be evaluated by the Teaching Assistants for the code requirements described above.

Example

Here is an example of how the algorithm works and how the candidates are ordered. Suppose the input file looks like this:

1	2	3	4	5
2	3	4	5	
1	3	2	4	5
2	4	3	1	5
2	1	3	4	5
1	3	2	4	5
5				
5	1	2	4	3
4	1	2	5	3
1	2	3	4	
2	3	4	5	1
1	4	2		
3	4	2		
4	3			

That is, there are 14 votes cast in this election. Each row of the CSV represents one vote. The candidates in this example have IDs from 1 to 5. As an example, the preferences of the first voter are candidate 1 (most preferred), then candidate 2, then 3, 4 and 5. The final voter prefers candidate 4 and then 3. The remaining candidates are not ranked by this voter (which is allowed).

Given this, we initially calculate the number of first place votes for each candidate. The percentages (all percentages are displayed rounded to two decimal places, but you should always use the non-rounded values in the assignment) are:

candidate 1: 35.71
candidate 2: 28.57
candidate 3: 7.14
candidate 4: 14.29
candidate 5: 14.29

We observe that no candidates have the sufficient votes to win (more than 50%), so we eliminate the candidate with the least number of votes, which is candidate 3. The votes now look like this:

1	2	4	5	
2	4	5		

1	2	4	5	
2	4	1	5	
2	1	4	5	
1	2	4	5	
5				
5	1	2	4	
4	1	2	5	
1	2	4		
2	4	5	1	
1	4	2		
4	2			
4				

We can recalculate the percentages of first place votes for the remaining candidates:

candidate 1: 35.71
candidate 2: 28.57
candidate 4: 21.43
candidate 5: 14.29

Again, no candidate has enough votes to win. We eliminate the candidate with the least first place votes, which is candidate 5. After eliminating the candidate, the votes look like this:

1	2	4		
2	4			
1	2	4		
2	4	1		
2	1	4		
1	2	4		
1	2	4		
4	1	2		
1	2	4		
2	4	1		
1	4	2		
4	2			
4				

Note that one vote is now completely exhausted – it has no preferences left. That's ok. We ignore this vote from now on (so there are only 13 valid votes left). The updated percentages of first place votes are:

candidate 1: 46.15

candidate 2: 30.77
candidate 4: 23.07

We eliminate candidate 4, giving these votes.

1	2			
2				
1	2			
2	1			
2	1			
1	2			
1	2			
1	2			
1	2			
2	1			
1	2			
2				

We only have two candidates left. Their first-place vote percentages are:

candidate 1: 58.33
candidate 2: 41.66

At this point, candidate 1 has enough votes to be declared the winner.

Sample Output

Assume that the votes in the previous example were in a csv file called votes1.csv. Then the output for a run of the program would look like this:

```
Enter the name of the file: votes1.csv
Elimination order: 3, 5, 4, 2, 1
```

Gradescope Test Cases

There are 13 gradescope tests. They are available on owl to review, along with the reason for each test case. If you are failing a test, you should review the test first to identify the problem that is being tested, to help debug your code. You can debug with the test case or a similar case that tests a similar input.