CS112: Discovering Computer Science

Voting Project

1 Project Overview

Elections are complicated. Even something as simple as a majority vote between two candidates can be complicated (see various recent US presidential elections). During the 17th century, a number of mathematicians began to systematically study voting and elections to determine if there are multiple ways to conduct them (there are) and which methods are more fair than others. In this project we will learn and then implement a handful of the most popular voting methods to arise during this period of study.

This is a group project. Your team should work on this project together; part of the learning experience is to evenly distribute the work among members of the team. You should not consult other students, in or outside the class. You should not work with your private tutors on this project. You may seek help from the class TAs and from your professor.

This is a scaffolded project. Below is a list of different levels. Your team earns points by completing each level. The levels increase in difficulty and challenge, so that your team can decide which levels match their ability.

2 Voting Theory

American citizens are most familiar with a voting system called *majority rules*. In this system, each citizen votes for one candidate and the candidate with the most number of votes wins the election. If there are two candidates, then the candidate that wins will have more than half the number of votes. If there are more than two candidates, the winner may have fewer than half the votes – this is called a *plurality winner*.

There are other voting methods besides majority/plurality. We will also study the Borda Method, Approval Voting, and the Condorcet Method.

2.1 Preference Ballots

In a typical US election, voters vote for just one candidate. We will change things around slightly. Each voter will rank ALL the candidates from first to last in order of that voter's preference for each candidate. This listing is called a *preference order*. For example, consider an election with five candidates whom we will call A, B, C, D and E. One voter might rank the candidates this way:

```
D A B C E
```

This listing is the voter's preference order of the candidates. The preference order ballot provides *more* information about that voter's wishes. We can assume that the candidate listed first is

the voter's first choice (D) but now we also learn how the voter feels about the other candidates in the election. Most of the methods rely on a preference order ballot from each voter.

2.2 Plurality

In a plurality election, we count the number of first place rankings for each candidate (number of times the candidate appears first on the ballots). The candidate with the most number of first place votes is the winner.

In the event of a tie, multiple winners are declared. Another method may be used to break the tie (including coin flips).

2.3 Majority

A majority election is similar to a plurality election in that we count the number of first-place rankings for each candidate. A candidate is a winner if they receive more than half (a majority) of the first place rankings. If no candidate receives more than half of the first place votes, then no winner is declared.

Note how this differs from Plurality. In plurality the winner receives the most first place votes. For this same candidate to be a majority winner, more than half of the voters must have ranked them first. It is possible to have a plurality winner and not have a majority winner using the same election data.

2.4 Borda

In the *Borda Method*, each candidate receives points based on how they are ranked on the ballot. Consider an election with n candidates. Every time a candidate shows up first on a ballot, they receive n points. Every time a candidate shows up second on a ballot, they receive $n-1$ points. And so on, such that every time a candidate appears last on a ballot, they receive 1 point.

In general, a candidate receives $n-k$ points when they are ranked in k^{th} place on a ballot (where we count k starting at 0 for first place, 1 for second, and so on).

The candidate with the most number of total points (from all ballots) is declared the winner.

2.5 Head to Head

In a head to head election we temporarily consider two candidates. We compare them on each ballot to see which candidate is ranked higher. In this head to head election, a candidate receives a point for being ranked higher than their competitor. A candidate wins the head to head contest if they receive more points (a ranked higher than the other on more ballots).

Now we consider every possible pair of candidates and put them in a head to head election. We are looking for a single candidate that beats every other candidate in a head to head election. Such

a candidate is called a *Condorcet winner* (named after a French mathematician who invented this method).

Many elections fail to produce a Condorcet winner even though the same data would produce a winner with other voting methods.

2.6 Approval Voting

This method is a bit different. Instead of providing a preference ranking, each voter produces a list of candidates for whom they approve. The order of the candidates on that list does not matter; the voter is simply providing a list of candidates that are acceptable to them.

A candidate wins an election under the *Approval Method* if they receive the most approval votes from the voters. In order to implement an approval method, we cannot use preference ballots.

3 Example Election

Here we give an example of each method. For all the methods (except approval voting) we use the following data. Consider an election with three candidates (A, B, and C). There are six voters who provide these preference rankings:

```
A B C
A C B
A B C
B A C
B C A
C B A
```

3.1 Plurality

We tally the number of first place votes for each candidate and declare A as the winner.

Candidate	Points
A	3
B	2
C	1

3.2 Majority

The same data fails to produce a majority winner since A has exactly half of the first place votes. A candidate needs more than half to be a winner.

3.3 Borda

Candidate A gets 3 first place votes, each worth 3 points. A also gets 1 second place vote, worth 2 more points, A also gets 2 third place votes, each worth 1 point.

$$\text{A}: \quad 3 \cdot 3 + 1 \cdot 2 + 2 \cdot 1 = 13$$

We do the same calculations for B and C to obtain:

Candidate	Points
A	13
B	13
C	10

The Borda method produces a tie between A and B.

3.4 Head to Head

Let's first compare B to C. In this head to head election, B gets 4 points and C gets 2 (there are 4 ballots that place B in front of C, 2 ballots where C is in front of B). So B beats C in a head to head competition. Since C has lost, they are no longer in consideration as a Condorcet winner.

Let's compare A to B. A gets 3, B gets 3. So this is a tie. Neither A nor B can be a Condorcet winner. We have eliminated all three candidates as possible Condorcet winners, thus no candidate is a Condorcet winner.

3.5 Approval Voting

Since approval voting does not utilize a preference ranking, we need to ask each of our 6 voters to provide an approval list. Let's suppose the list below is the approval list from each voter.

```
A
A C
A B C
B
B C
C B
```

We simply count the number of times each candidate is on the approval list (anywhere) to arrive at:

Candidate	Points
A	3
B	4
C	4

Thus B and C tie as approval winners.

4 Program Structure

The following section provides the requirements for the project. These are divided into levels so that your team can tackle them one part at a time. Each level is assigned the total cumulative points for the assignment that your team can earn by completing the project up through that level. You must tackle the levels in the order specified.

4.1 Level 1: Read Data (25)

Write a function to read the preference ballots from the data file. Always use `data.txt` as the filename for the data. Your team will have to think of a suitable data structure to store the information and return it from this function to your main program.

4.2 Level 2: Plurality (40)

Write a function which takes the preference ballot data as input and returns the counts for a plurality election. Your team will have to think of a suitable data structure to hold and return the results of the plurality election. In your main function, use the return results to create and print a table displaying the results.

4.3 Level 3: Majority (50)

Create a function which takes the results of the plurality election and determines if there is a majority winner. Your function should return the majority winner if there is one, or return None if no such winner exists. In your main function, print the Majority winner or indicate if no winner exists.

4.4 Level 4: Borda (65)

Write a function which takes the preference ranking data as an input parameter. The function should compute the Borda score for each candidate and return the scores in a suitable data structure. In the main function, use these returned results to print a table showing the results of the Borda election.

4.5 Level 5: Approval (75)

You will need to write two functions. One function will read the approval data from `approval.txt`. These are a list of approval ballots from the voters. The second function will take this data as a parameter and return a data structure holding the approval scores for each candidate. Have your main function call both of these two other functions and then print a table showing the approval scores for each candidate.

4.6 Level 5: Data Independent (88)

This level asks you to go back and make changes to your program. You want to make your program as flexible as possible to work with different data sets. Do the following:

- Be sure your program works with any candidate names. Do not assume your dataset has 'A' ... 'E' as candidates. Have your program find the candidate names from the `data.txt` file and use these. If you do this step correctly, you will not see any 'A' or 'B' or ... 'E' anywhere in your code. You should also be able to edit the `data.txt` file and change all the A's to X's and your program should still work correctly.

- Be sure your program works with any number of candidates. Do not assume there will be exactly five. Have your program determine the number of candidates from the `data.txt` file. You should not see any 5 anywhere in your program code. You should be able to create a new dataset in `data.txt` that has 3 or 4 or 6 candidates and your program should run correctly without any changes to the code.

- Use the same read function to read both the preference data in `data.txt` and the approval data in `approval.txt`. You will then obviously need to make sure the data structure you create and return here will work for both kinds of data.

4.7 Level 7: Condorcet (100)

In this final section you will compute the Condorcet winner, if any exists. I recommend writing two functions. The first function

```
isWinner(data,c1,c2)
```

will return True if candidate c1 beats candidates c2 in a head to head election using the data provided, returns False otherwise. This function is useful for determining the outcome of a head to head election between just two candidates.

The second function takes the preference data as a parameter and determines if there is a Condorcet winner. It will try all head to head competitions and determine if one candidate beats all the others. Return the candidate who is the Condorcet winner or None if no winner exists. Have your main function print the winner or indicate if none exists.

I really want your team to work on solving this one by themselves. I am asking the TAs to not give a lot of help here. Completing this level raises the group grade from a B+ to an A. To earn an "outstanding" grade on this project, your team should be able to solve this part of the problem largely on their own. Think of this level as the bonus part of the project.

5 Sample

I have provided for you a sample `data.txt` file and a sample `approval.txt` file. You can and should create other samples for approval and preference ballots with different number of candidates and different number of voters. For these two sample files, here is the output of my program:

```
Plurality
D                4
E                5
C                0
B                0
A                7

Majority
no majority winner

Borda
D               54
E               48
C               48
B               46
A               44

Approval
D                8
E               10
A                9
B                9
C                9

Condorcet
D
```

Your program should produce similar output on these two sample files.