Homework 3: Political Decision Trees

Your job is to create a decision tree to determine whether a member of the House of Representatives is a Democrat or a Republican.

Download the data file `voting-data.tsv`. The file looks like this:

```
Rep-1   R     +++--+-+++
Rep-2   D     +++----+++
Rep-3   R     ++++-+++--
Rep-4   D     ++++--+++-
Rep-5   R     ++++-++++-
```

Each row represents one member. First is that member's ID, then a tab, then a label (`D` or `R`) indicating if the member is a Democrat or Republican. Then there is a another tab, and the representative's voting record on ten issues, which will be designated as "Issue A" through "Issue J". Each vote has one of three values: a `+` for a "yea", a `-` for a "nay", or a `.` if the member voted "present" or otherwise abstained. There is data for 430 representatives.

You should use the decision tree algorithm that we discussed in class, using the "information gain" metric in order to determine what is the best question to ask at each stage. Some other implementation details to consider are:

- If two issues have the same information gain, use the issue with the earlier letter.
- If you reach a point where you still have both Democrats and Republicans, but they all have the exact same voting record, create a leaf that classifies them as the majority left.
- If you reach a point where there are still Democrats and Republicans, but they have different voting records even though the best information gain is 0, keep adding to the tree. Another issue may now be able to split them.
- If a branch has 0 reps, or a tied number of indistinguishable reps, classify based on the majority at its parent node. If this is tied too, keep going back up the tree until you have a majority.

Remember that you will not use the entire data set to induce your decision tree. The data must be properly divided into the *training set* used to create the tree, the *tuning set* used to prune the tree to make it more accurate, and the *test set* used to estimate its accuracy. How you partition the data into these three sets will depend on the task you are trying to perform.

First, you must create a decision tree and print it. To do this, there will be *no test set*. To create your tuning set, separate out every fourth element starting with the first. (So your tuning set will be element 0, element 4, element 8, and so on.) Pruning the tree is done by assessing your entire tree's current accuracy on the tuning set, and its accuracy when each internal node is pruned. (A pruned node is one that is replaced by a leaf that classifies a rep based on the majority of representatives in the training set who would have reached that node.) If the best pruned tree is at least as good as the overall tree, making the prune permanent and repeat the process. If two possible prunings are tied for best, use the pruning that eliminates more nodes. Keep pruning until every possible chop would make your tree perform worse on the tuning set.

Your output should appear like the following:

```
Issue C:
  + Issue A:
    + D
    - R
    . D
  - Issue F:
    + R
    - Issue L:
      + D
      - D
      . R
    . D
  . R
```

In this tree, to classify a new representative we would first ask how s/he voted on Issue C. If the rep voted "yea", ask about Issue A. Here a vote of "yea" or an abstention means a Democrat, while a vote of "nay" means a Republican. But if the rep voted "nay" on Issue C, ask about Issue F. Here a "yea" mean Republican, while on a "nay" you need to ask about Issue L...and so on.

Second, you must estimate your decision tree's accuracy. To do this, you will use *leave-one-out-cross-validation.* Loop through every datum. For that datum, exclude it from the calculation and create a new decision tree as was done above. Then after it has been trained and tuned, test it on the left-out datum. Do this for all your data. (Note that when taking out every fourth datum to be the tuning set, you should do this as if the testing datum never existed.) Print out the accuracy on the left-out testing data as the estimate of your tree's data. *Please do not print out the hundreds of trees you create for this step.*

Your file should be called `tree-inducer.py`. Its single command-line argument will be the data file to use.