22C:16 Homework 6

Due via ICON on Friday, April 13th, 4:59 pm

What to submit: Your submission for this homework will consist of 6 files: (i) `fasterListBuild.py` for Problem 1(c). (ii) `makeRandomList.py` for Problem 2(a), (iii) `fasterMakeRandomList.py` for Problem 2(b), (iv) `mostFrequent.py` for Problem 3, (v) `extractFrequentWords.py` for Problem 4, and (vi) `homework7.pdf` for the rest of the problems. Each of these Python files should start also with your name, section number and student ID appearing at the top of the file as Python comments. Also make sure that your Python code is appropriately commented. The pdf file should also start with your name, section numbers, and student ID appearing at the beginning of the file. You will get no credit for this homework if your files are named differently, have a different format, or if your files are missing your information.

1. Copy and save the Python program below in a file called `slowListBuild.py`.

```
import time
startTime = time.time()
n = 500000
L = []
for i in range(n):
    L.insert(0, i)
endTime = time.time()
print endTime - startTime
```

   (a) Run `slowListBuild.py` and report the amount of time (in seconds) that the program took to complete.

   (b) Suppose you double the value of `n` from 500,000 to a million. Do you think the program will take roughly twice as much time (yes or no)? Explain your answer in 1-2 sentences.

   (c) Rewrite the program using `append` rather than insert and report the time the program took. Make sure your new program does exactly what the original did, i.e., it should build the same list as before. Call your new program `fasterListBuild.py` and submit it as part of your solution.

   (d) Explain in 1-2 sentences why there is such a big difference in running times of the two versions of the programs.

   (e) Now rewrite the program using `insert`, but make sure to insert at the end of the list rather than at the front In other words, you are being asked to just perform the `append` operation, but by calling `insert` instead of `append`. Make sure your new program does exactly what the original did, i.e., it should build the same list as before. Report on the running time of your new program. Explain in 1-2 sentences why this new program has such a different running time relative to the original program. You do not have to submit your program for this part.

2. Your task is to write a program that builds a list of 500,000 distinct, randomly chosen numbers in the range 1 through a million (inclusive of 1 and a million). The algorithm you are required to use is this: (i) start with an empty list called `L`, (ii) repeatedly pick a number at random in the range 1 through 1000000, check if the number is in `L` and if not append it to `L`. Your loop should terminate when the `L` contained 500,000 numbers.

(a) Implement the program described above, save it as `makeRandomList.py`, and submit it as part of your solution. Using the `time()` function from the `time` module to compute and report the running time of your program.

(b) The program can be made much more efficient if you use dictionaries rather than lists to store the generated numbers. Reimplement the algorithm using a dictionary rather than list. Save your new program in a file called `fasterMakeRandomList.py`. Report on the running time of the program.

3. Write a function called `mostFrequent` that takes as a parameter a dictionary. The keys of the dictionary are strings with corresponding values being frequencies (possibly indicating how frequently these strings occured in some file). The function should return a list of 50 most frequent keys in a list. The list should be ordered in decreasing order of frequency, i.e., with the most frequent key appearing first. If two keys have the same frequency it does not matter what order they appear in. It is possible that the dictionary contains fewer than 50 keys. In this case your function should return a list with all the keys in the dictionary, in appropriate order. Save your function in a file called `mostFrequent.py` and submit as part of your solution.

4. Write a program to find the 50 most frequent "words" in the posted English translation of Leo Tolstoy's "War and Peace." Recall that we wrote a program called `makeDictionary4.py` that extracted words from this text (this program is posted under "Weekly Topics and Links to Lecture Notes", Week 9). Your task in the current problem is quite similar and you should modify `makeDictionary4.py` by replacing the list `wordList` by a dictionary. The keys in this dictionary would be words and the associated values should be word frequencies. After the dictionary has been built, you can simply call the function `mostFrequent` you defined in the previous problem. Your program should produce as output the 50 most frequent words along with their frequencies, one word-frequency pair per line. These should be in decreasing order of frequency — if two words have the same frequency, it does not matter which order they appear in. Save your program in a file called `extractFrequentWords.py` and submit as part of your solution.

   Note: The program `makeDictionary4.py` used a smaller version of the novel because it was somewhat inefficient. Your new program will not have this problem and should use the entire novel.
