CARNEGIE MELLON UNIVERSITY 10-607

HOMEWORK 3

DUE: Friday, Dec. 3, 2021

https://www.cs.cmu.edu/~10607

INSTRUCTIONS

- **Format:** Use the provided LaTeX template to write your answers in the appropriate locations within the *.tex files and then compile a pdf for submission. We try to mark these areas with STUDENT SOLUTION HERE comments. Make sure that you don't change the size or location of any of the answer boxes and that your answers are within the dedicated regions for each question/part. If you do not follow this format, we may deduct points.

  You may also type you answer or write by hand on the digital or printed pdf. Illegible handwriting will lead to lost points. However, we suggest that try to do at least some of your work directly in LaTeX.

  Programming components

- **How to submit written component:** Submit to Gradescope a pdf with your answers. Again, make sure your answer boxes are aligned with the original pdf template.
- **How to submit programming component:** See section Programming Submission for details on how to submit to the Gradescope autograder.
- **Policy:** See the course website for homework policies, including late policy, and academic integrity policies.

Name	
Andrew ID	
Hours to complete all components (nearest hour)	

1 Complexity, Recursion, and Induction [20 pts]

1. **[5 pts]** The formula for the sum of n terms of a geometric series a, ar, $ar^2 \ldots ar^{n-1}$ is $\sum_{i=0}^{n-1} ar^i = \frac{ar^n - a}{r-1}$ Prove this formula by induction on n. Note that you must assume r $\neq$ 1 for the formula to hold.

Proof

2. **[5 pts]** The formula for the sum of an arithmetic series with first term a and increment b, that is, a,(a + b),(a + 2b),. . . , a + (n - 1)b

$\sum_{i=0}^{n-1} a + bi = \frac{n(2a+(n-1)b)}{2}$.
Prove this formula by induction on n.

Proof

3. You are given the algorithm for recursive selection sort as follows:

```
def recSS(A, i, n):
'''
@param A: A list of integers
@param i: start_index
@param n : length of array
'''
if i < n-1:
small = i
for j in range(i+1, n):
    if A[j] < A[small]:
        small = j
temp = A[small]
A[small] = A[i]
A[i] = temp
recSS(A, i+1, n)
```

(a) **[3 pts]** Suppose that we call `recSS([10, 13, 4, 7, 11], 0, 5)`. What are the contents of the array `A` just before each recursive call to `recSS`. You should work through this by hand rather than actually running the code to find the answer.

Contents of A

(b) **[5 pts]** Prove by induction that the running time of this algorithm is O(n^2).

Hint: Write an expression for the number of times the `if` statement is executed for a given n.

Proof

4. **[2 pts]** What is the runtime complexity of the following code in terms of n. Give the simplest and tightest bound.

```
def f(n):
    result = 0
    j = 0
    while j < n
        k = 0
        while k<n:
            result += 1
            k *= 2
        j++
```

Complexity

2 Vector Implementation (Programming) [15 pts]

In this questions we'll be using the vector dot product to analyze the difference in performance between for loop implementations and implementations using NumPy. We'll also take a look at sparse representations of vectors and they're affect on performance. The following Jupyter notebook contains the instructions for implementing the three dot products functions and running three preformance experiments:

https://drive.google.com/file/d/1Bxi6cL0aqpcLUKud_o2LYfWgbCiOV-9q

Note: To make it more likely that we are all running our code in similar environments, if possible, please use Google Colab to run the performance experiments in the above file.

After running the performance experiments, provide your plots and answer the questions below.

1. **[5 pts]** Experiment 1: Get an empirical estimate of the computational complexity of the NumPy implementation with respect to the length of the vectors.

Plot

(a) What is the (simplest and tightest) time complexity that you would expect for a dot product implementation with respect to the vector length N?

◯ $O(1)$ ◯ $O(\log N)$ ◯ $O(N)$ ◯ $O(N \log N)$ ◯ $O(N^2)$

(b) Do the empirical results in your plot seem to support this complexity? Explain your reasoning in one sentence. (Note that we, of course, can't measure the actually complexity from just these empirical results with a finite N).

2. **[5 pts]** Experiment 2: Compare the performance of the for-loop and NumPy implementations as the length of the vectors changes.

Plot

(a) Do the empirical results for Experiments 1 and 2 support the claim that for-loop and NumPy implementations are in the same complexity class? Explain your reasoning in one or two sentences.

(Again, we, of course, can't measure the actually complexity from just these empirical results with a finite N).

3. **[5 pts]** Experiment 3: Compare the performance of the NumPy and sparse-for-loop implementations as the length of the vectors changes.

Plot

(a) What time complexity for the sparse-for-loop dot product implementation with respect to the vector length N and the number of nonzeros K? Give your answer in big-O notation with with the simplest and tightest bound. Note: this is not an empirical estimate, you should analyze your code.

(b) Based on your Experiment 3 plot, give an estimate for the density of nonzeros, K/N, below which it may be more efficient to use the sparse-for-loop implementation. Give your answer as a decimal value between 0.0 and 1.0 (not as a fraction).

4. Programming Submission

   Please submit the file `vector.ipynb` (in addition to `tree.py`) directly to Gradescope under the HW3 (programming) assignment. Note there are no autograder tests associated with this submitted file on Gradescope. We just need to see the code for your dot product implementations and your experiments.

3 Tree (Programming) [1 pts]

In the accompanying file `tree.py`, fill out code to complete the provided functions to implement a tree data structure that recursively splits a data array.

1. Task 1

   In `tree.py`, implement the following functions based on the descriptions in the code comments:

   - `getChildren`
   - `getData`
   - `getID`
   - `buildTree`

2. Task 2

   In `tree.py`, implement the following functions based on the descriptions in the code comments:

   - `printTreeBF`
   - `printTreeDF`

3. **[1 pts]** Programming Submission

   Please submit the file `tree.py` (in addition to `vector.ipynb`) directly to Gradescope under the HW3 (programming) assignment.

   Your code file submissions should follow the provided code template files. In particular, they should NOT include any additional Python imports such as `matplotlib` which might make it fail on the autograder.

   Please maintain any code for creating your plots in separate files. You do not have to submit your plotting code on Gradescope.

   You may submit your code as many times as you like before the deadline.

   Have you made the code submission on Gradescope?

   Select one:

   ◯ Yes ◯ No

4 Collaboration Policy

After you have completed all other components of this assignment, report your answers to the following collaboration questions.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details including names of people who helped you and the exact nature of help you received.

2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details including names of people you helped and the exact nature of help you offered.

3. Did you find or come across code that implements any part of this assignment? If so, include full details including the source of the code and how you used it in the assignment.