CS372 Homework 1: Introduction to Algorithmic Analysis and recurrence

DUE: Thursday, Sept 12th, at 8AM

The purpose of this assignment is to give you practice with pseudocode, practice on mathematical foundations, and concepts from the first few lectures of class.

The assignment consists of 3 parts

A. Practice empirically supporting your results
B. Application of topics
C. Conceptual Questions

Coding [50pt]

Coding algorithms from pseudocode (or translating an algorithm to a new language) is an important skill. And the purpose of the coding section is to get you accustomed to reading and using pseudocode. Empirically proving the run time of your algorithm is also important to show you have implemented an algorithm correctly.

Quick sort runs in $O(n \lg(n))$ average-case time and heap sort runs in $\Theta(n \lg(n))$, BUT the constant factors and computer architecture will affect the run time. To empirically prove that these sorts do have different run times, perform the following tasks with the starter code given on D2L under the same module. You may alter the starter code, but not the output, menu format, or "GRADING" tag. Several items are incomplete in the code and are marked with TODO.

1. CODE the **book's** version of heapsort and quick sort. Use the last index as the pivot and sort integers. Use a time (AKA random) seed. Mark these with comments in the format; "GRADING: HEAP" and "GRADING: QUICK", respectively.
2. Make a menu to run correctness tests, accept user input, and run time tests. I must continue until the user chooses to quit. You must have the following format, which is given to you in the starting code:

   ```
   Enter T to run tests
   Enter U to accept user input
   Enter E to run empirical tests
   Enter Q to quit
   ```

   User Input must be in the format:

   ```
   Input: # # # # ... -999
   ```

   -999 will act as a flag and should not be sorted.
   For the user input, you also must print the results in

   ```
   Quick: #, #, #, ...
   Heap: #, #, #, ...
   ```

   format.
   The menu must loop until Q is selected.

3. ADD built in tests to confirm your algorithms works. Since this is the first time you will write test cases, I'm giving you the sets I want you to add, and a template is provided in the starter code. Please review the testing standard for this class. The tests are:
   - Test 1: sort {2, 1} (both sorts)
   - Test 2: sort {1, 3, 2} (both sorts)
   - Test 3: sort {2, 2, 1, 3, 5, 4, 8, 0 } (both sorts)
   - Test 4: 9 random sort {#, #, #, #, #, #, #, #, # } (both sorts)

   The output format must be:

```
Test #: <text name>
Original: <original list separated with , )
Quick Result: <result list>
<pass/fail result>
Heap Result: <result list>
<pass/fail result>
```

You may add more tests if desired.
These WILL be checked that these can return false if the answer is not correct.

<u>You may NOT use OS or computer specific C++ libraries (e.g. no Linux/Mac/Win MACROs or new libraries you may have installed). This MUST compile on MSVS 2019 (I have not had any trouble with MSVS 2015-2017 importing to 2019).</u>

4. RECORD the run time for different, increasing, number of elements. Since this is quick sort, you must average the run times for each array size with a minimum of three runs. You must choose at least 10 sizes to test, and select a range that clearly shows the difference in both small array sizes and very large array sizes (the run times may cross!). The range necessary to show the difference **will vary** based on your computer and algorithm design, but {50, 100, 200, 500, 1000, 2000, 10000, 50000, 100000, 200000} will be a good start.

   Use the following format for the table (you may list individual runs if you wish):

size	Average for Quick	Average for Quick
10		
20		
…		

5. PLOT the results for both the sorts on the same graph. The plot should be titled and labeled, and *size* should be on the horizontal axis, and time with its unit should be on the vertical axis. You may use whatever software you wish to plot. A tutorial on plotting in Excel is available in the same module as the homework.
6. Submit the written results at the top of the remaining application and concept questions.

Common errors:

1. *Giving a sorted list to quicksort…you may get a stack overflow*
2. *Counting the copy time as part of your run time.*
3. *Just outputting a pass for all tests, regardless of the result*

Application

Mathematical foundations

1) [8pt] Prove which has a greater, or equivalent, asymptotic run time for each of the follow formula pairs. You may not plug in large values of n to prove the run time. You may use l'Hospital's or Stirlings approximation. Do not stop until there is a constant or you are in a standard family.

a: $\sqrt{2}^{\lg n} = -10n$ b: n *(hint: use log properties)*

a: 1: n^n b: $\sqrt{n!}$

2) [12 pt] Determine the run time (big-O) for the following recurrence formula using the tree or substitution method. You may use the master method only to check your answer.

$$T(n) = \begin{cases} 1 & n = 1 \\ 4T(n/3) + n & n > 1 \end{cases}$$

3) [12 pt] Determine the run time (big-O) for the following recurrence formula using the tree or substitution method (Hint: we jump by *powers* of b if we have T(n/b), we jump by *multiples* if we subtract by b). Note: you will need to use < when simplifying the summation.

$$T(n) = \begin{cases} 0 & n = 0 \\ T(n-5) + n & n > 0 \end{cases}$$

As an example

$$T(n) = 2T(n-1) + 1, T(0) = 0$$

Yields:

$T(n) = 2T(n-1) + \mathbf{1}$

$T(n) = 2(2T(n-2) + 1) + 1$

$= 4T(n-2) + \mathbf{2} + \mathbf{1}$

$T(n) = 4(2T(n-3) + 1) + 2 + 1$

$= 8T(n-3) + \mathbf{4} + \mathbf{2} + \mathbf{1}$

$T(n) = 8(2T(n-4) + 1) + 4 + 2 + 1$

$= 16T(n-4) + \mathbf{8} + \mathbf{4} + \mathbf{2} + \mathbf{1}$

$\dots$

$$T(n) = \sum_{i=0}^{n} 2^i$$

Reminder: the following is the Geometric series of x^i when x > 1.

$$T(n) = \sum_{i=0}^{n} 2^i = \frac{2^{n+1}-1}{1-2} = \Theta(2^n)$$

We got *lucky* here with a canned formula, otherwise we would need to prove by induction with the base case, and assumption steps as well.

4) [6pt] Determine which case of the Mater Theorem applies for the following recurrences Include the values of a, b, and k (and ideally b^k) as proof of your selection. Also, include the final big-theta formula. You also have the option of a recurrence relation that cannot use the master method as described in class.

a. $T(n) = 4T\left(\frac{n}{2}\right) + \sqrt{n}$

b. $T(n) = 2T\left(\frac{n}{2}\right) + n^2$

c. $T(n) = T\left(\frac{n}{3}\right) + \left(\frac{n}{4}\right) + n^3$

Concepts

5) *OPTIONAL* Prove the correctness of a binary search using the loop invariant technique. Only worry about the "found" case.

6) [6pt] For the following problem, what is the best sort among the given list, and why?
Problem: *You are analyzing wind speeds across South Dakota. You need to be able to list the wind speed in ascending order for each location which is also in ascending order. You may assume there is an average of a wind speed station every 20 miles.*

7) [12pt] Write the resulting recurrence relation for the following pseudocode where A and B are arrays of integers:

```
GOO(A, B)
  If A.length < 5
        avg = average of A
        for i = 0 to B.length
             print B[i] * avg
        return
  sep = ⌊A.length / 3⌋
  x = copy A[1] to A[sep]
  y = copy A[ sep + 1 ] to A[ A.length]
  GOO(x, B)
  GOO(y, B)
```

Submission instructions

You may upload as many times as you please, but only the last submission will be graded.

1. Double check you are not in violation of any additional deductions as posted in the main assignment tab.
2. Put your name, class, and homework number at the top of the PDF.
3. Delete any build folders in your MSVS solution.
4. Zip up your project along with **one** PDF of your graph, run times, and answers to the application questions. Name and **TITLE** your PDF lastname_firstname (so mine would be rebenitsch_lisa). If the compressed folder opens with 7-zip, you're good.
   You may submit the PDF portion on paper by either giving it to me, sliding it under may door, or putting it in my mailbox.
5. Submit this to D2L under the associated folder.

If you have any trouble with D2L, please email me (with your submission if necessary).

Rubric

Task	Points
Code	50 Total
Book's Heap sort (1/2 credit if between 50-99% done or not the books version)	8
Book's Quick sort (1/2 credit if between 50-99% done, or not the books version)	8
Correctness test, and can fail (2pt each)	8
Run time test implemented (1/2 credit if between 50-99% done)	8
Run times recorded (1/2 credit if between 50-99% done)	6
Plot results (4 point each)	8
Graph format (units, location, etc, -2 each)	4
Application	36 Total
1) growth relations (4 each)	8
2) Recurrence 1 (points given only up to last correct subitem) a. 3 for base + initial expansion b. 3 for sum formula c. 4 for simplification d. 2 for solution	12
3) Recurrence 2 (points given only up to last correct subitem) a. 3 for base + initial expansion b. 3 for sum formula c. 4 for simplification d. 2 for solution	12
4) Master method (2 each)	6
Concepts	18 Total
5) Select sort (2pt selection, 4pt reasoning)	6
6) Determine Recursion relation (-2pt per incorrect term)	12
Total	100