Data Structures
Assignment #2

Due: Oct 27, 2020

1. Describe the standard algorithm for finding the binary representation of a positive decimal integer. One can use a string as the output.
   (a) in English (or Chinese)
   (b) in a pseudocode using iterative approach
   (c) in a pseudocode using recursive approach
2. Please show the following statements.
   (a) $f(x) = 4x^2 + 2x + 1 = \Theta(x^2)$
   (b) $f(x) = x \log x + \sqrt(x) = \Theta(x \log x)$
3. Please verify the following statements. If it is true, please show it; otherwise, give an argument to show the incorrectness.
   (a) $2^{n+1} = O(2^n)$
   (b) $2^{2n} = O(2^n)$
4. A ***permutation*** P of size n is a way of arranging the numbers between 1 and n, where each number must be used and can only be used once. For example, 5 4 1 2 3 is a permutation of 1 2 3 4 5 where $P(5) = 1$, $P(4) = 2$, $P(1) = 3$, $P(2) = 4$, and $P(3) = 5$. An ***inversion*** in a permutation P is a pair of numbers i and $j \in \{1, \ldots, n\}$ with $i < j$ such that the larger number appears to the left of the smaller one in the permutation, that is $P(j) < P(i)$. In the example, the pair of (3, 5) is an inversion in P. The ***inversion number*** of a permutation is the total number of inversions and can be thought of as a measure of how "out of order" a permutation is. Please consider the following questions:
   (a) What is the maximum inversion number in a permutation of size n? Please provide the case when the maximum inversion number occurs.
   (b) Given a permutation P, an interesting problem is to find the inversion number of P. A straightforward approach is to check each number in P one by one, starting from the leftmost one of P. At each position (number), count how many smaller numbers are to the right, and then sum them up. Please provide an *iterative* version for this approach in pseudo-code and analyze its time complexity.
   (c) Please provide a *linear recursive* version for the approach mentioned in 4b with pseudo-code and analyze its time complexity.
   (d) Recall the binary recursion discussed in class. Please provide a *binary recursion* version for the approach mentioned in 4b with pseudo-code and analyze its time complexity.

5. (**Programming problem 1**)
   Consider the problem of finding the binary representation of a positive decimal integer Problem 1. Please use Python to implement the pseudo-code you provide for solving the problem. You should provide the iterative and recursive functions named with `Dec_to_Bin_Ite()` and `Dec_to_Bin_Rec()` respectively. Then, perform a comparison on these two versions by measuring the execution time. Hence, you need to

   (a) implement the iterative approach as a function named as `S_iterative()`,

   (b) provide the recursive version for the approach named as `S_recursive()`, and

   (c) compare these two function with the same input in terms of running time and write what you have observed.

   We will use an in-built python library `timeit` and the module function `timeit.timeit()` for measuring the running time respective. In you observation, please indicate the ways to measure the time and this will be provided in the template.

   Note: For the programming problems, please do some experiments by yourself to observe the running time and the number of recursive calls. You need to report what you have observed from the experiments when you submit your homework.

6. (**Programming problem 2**)
   In problem 4, there are three versions for deriving the inversion number of a given permutation. Please implement these versions in Python and compare them. You need to present the results and what you observed after the experiments on the performance. The things to be provided are as below.

   (a) Implement the iterative approach using Python and name the function as `inversion_number_iterative()`.

   (b) Please implement the linear recursive version you provide in problem 4(c) with the function name of `inversion_number_recursive()`.

   (c) Write the binary recursive version you provide in problem 4(d) with the function named as `inversion_number_tworecurs()`.

   (d) Compare these functions with the same input in terms of running time and write what you have observed.