CSC 6013 – Summer 2022

Week 7 Coding Assignment

Create a recursive, decrease-and-conquer algorithm to solve each of these two problems. Implement your algorithm in Python code and run it on the given problem instances. Submit your work as a docx or pdf file through Blackboard.

1) A "decrease-by-a constant amount" algorithm - Mean of an array

a) Write a recursive algorithm to determine the mean of the numbers in a non-empty array using the strategy that is illustrated by the following two examples with an array of 5 numbers (notice the recursive call with one less entry in the array) and an "array" of 1 number (think base case of the recursive algorithm):

$$mean(\{40, 50, 60, 70, 80\}) = \frac{4}{5} * mean(\{40, 50, 60, 70\}) + \frac{1}{5} * 80$$

$$mean(\{40\}) = 40$$

Your function should have <u>two parameters</u> – the array of numbers and an integer indicating how many of the array values should be included in the calculations. For example, given five numbers stored in array A (in slots A[0]..A[4]), the function call

Mean(A, 5)

would cause a return of

4/5*Mean(A, 4) + 1/5*A[4].

Of course, the parameter in your code for the definition of the function would be n; the 4s and 5s would not show up explicitly in the code.

b) Run your code on the problem instances:

i) A1 = <12, 23, 37, 45, 63, 82, 47, 75, 91, 88, 102> 11 entries

ii) A2 = <-1.7, 6.5, 8.2, 0.0, 4.7, 6.3, 9.5, 12.2, 37.9, 53.2> 10 entries

2) A "decrease-by-a constant factor" algorithm – Binary search

a) Write a recursive algorithm to determine the location in a sorted array where a specified searchkey is found. Unlike the algorithm in the class notes, this algorithm works with an array that is sorted into DESCENDING order and the code should print out all the subscripts in the array that were examined during the search.

b) Run your code on the problem instances:

In array [100, 87, 85, 80, 72, 67, 55, 50, 48, 42, 40, 31, 25, 22, 18]

search for 87

search for 48

search for 33

search for 10

3) A "decrease-by-a variable amount" algorithm - Euclidean Algorithm for Greatest Common Divisor (GCD)

In his famous math book *Elements,* Euclid included an algorithm (that he might or might not have invented) to find the GCD of two positive integers. The basic idea is as follows:

$$GCD(a,b) = \begin{cases} a & if\ b = 0 \\ GCD(b, a\ mod\ b) & otherwise \end{cases}$$

For example, GCD (144, 42) = GCD (42, 18) = GCD (18, 6) = GCD (6, 0) = 6

a) Code this algorithm in Python. Use the appropriate Python operator to implement the modular arithmetic operation.

Before each recursive call, have your algorithm print out the two integers being passed as parameters, make the call to find their GCD, and print out the result of the call.

b) Run your code on the problem instances:

i) GCD (2468, 1357)

ii) GCD (111, 378)

iii) GCD (123456789, 987654321)