COMP161
Exam 6 Study Problems

Spring 2017

Practice Problems for the last Exam. Each problem can/should be solved using basic recursive and iterative design. Practice writing good documentation and tests along with the actual implementation. This is particularly helpful when dealing with the helper procedures that come along with recursive design. It might be help to classify each problem as a map, filter, or reduction. Recall that maps transform a vector into another vector by applying some function to each element. A filter selects elements that meet some criteria. A reduction combines elements of a vector in some consistent way.

1. **Complexity**

   (a) Provide names for the following classes (be ready to do Names to Big-O as well)

   $O(n \log n)$ ____________________

   $O(n)$ ____________________

   $O(n!)$ ____________________

   $O(1)$ ____________________

   $O(n^2)$ ____________________

   $O(2^n)$ ____________________

   $O(\log n)$ ____________________

   $O(n^3)$ ____________________

   (b) What are the complexity classes of the following algorithms? What does the complexity class tell you, *specifically*, about the algorithm? What do you not know by knowing the complexity class?

   Linear Search ____________________

   Insertion Sort ____________________

(c) Fill in the blank with $<$, $>$, or $=$ whichever is correct for the given complexity classes. (Know the ordering given in the lecture notes)

$O(n^2)$	____________________	$O(n)$
Logarithmic	____________________	Linearithmic
$O(1)$	____________________	Factorial
$O(n^2)$	____________________	$O(2^n)$
$O(n^2)$	____________________	$O(n \log n)$

2. Iterative and Recursive Design (Classic Problems)

   (a) Search: Return a bool or return the location

   (b) Sort (Insert and Sort)

   (c) Min/Max: Return a bool or return the location

   (d) Exclusive Prefix Sum

   The exclusive prefix sum procedure would take a vector of numbers and compute for each number in that vector the sum of all the numbers proceeding it in the vector. For example, the vector containing $\{3, 5, 8, 2, 7\}$ has an exclusive prefix sum of $\{0, 3, 8, 16, 18\}$ where the fourth element is 16 because the sum of the first three numbers in the original vector is 16. This problem could be solved functionally (return the exclusive prefix sum) or as a mutator (modify the original vector).

   (e) Inclusive Prefix Sum

   Like the exclusive prefix sum but include the i^{th} element of the original vector in the i^{th} sum. The fourth element of the inclusive prefix sum example was 16, in the case of exclusive prefix sum we'd include the fourth element of the original vector, 2, to get 18. The complete exclusive prefix sum of $\{3, 5, 8, 2, 7\}$ is $\{3, 8, 16, 18, 25\}$

3. Iterative and Recursive Design (Arbitrary Problems)

   (a) A function to compute a vector of booleans given a vector of strings such that the returned vector contains true at index i if the string at index i of the argument vector contains the letter e. (Could generalize to a given character or rewrite for different specific letters)

(b) A mutator that removes all the multiples of 5 from a given vector of integers. (This could also be a function that returns the vector containing numbers that are not a multiple of 5)

(c) A function that counts the number of times a given string occurs in a vector of strings.

(d) Compute a weighted average given a vector of scores and and vector of weights where the weight at index i goes with the score at index i.

(e) A predicate (function that returns a boolean) that returns true if a vector of doubles contains only positive numbers.

(f) A predicate (function that returns a boolean) that returns true if a vector of characters contains only alphabetic letters.