PDS Tutorial Questions

Tutorial/Prep 2

Aug. 7, 2024

Information

- This document comprises tutorial questions for CS2700, which includes both conceptual/theory questions (relevant for CS2700) and programming questions (relevant for both CS2700 and CS2710; these questions can also be thought of as preparatory/practice programming questions for CS2710 Lab 2, and so also referred to as Prep 2).

Conceptual questions (for CS2700)

1. To prove that a Hoare triple involving a sequence of assignment statements is true, we always proceed in the REVERSE direction and see if the proof is consistent! That is, we start from the desired post-condition Q, then work our way back by applying the Hoare assignment rule, and then check if we get the desired pre-condition P. This would imply that the Hoare rules can be applied in the forward direction to start from P and derive Q.

   Now, verify if you've understood this proof strategy in the swap(a,b) code seen in class. Apply this proof strategy to complete the proof of "$j == 2^n$" code seen in class.

2. Use Hoare logic to prove that the program to compute factorial given in the class slides correctly computes n!. Specify the pre-condition and post-condition of each (simple/compound) statement and show how the Hoare triplet rules are applied to prove that these pre/post-conditions are indeed satisfied by the program.

3. In the Hoare logic proof of the program to compute "$j == 2^n$" seen in class, why is "$\{n > 0\}$" pre-condition required? That is, where exactly is this pre-condition used in the proof that the program correctly computes 2^n?

Programming questions (for CS2710 Lab 2 preparation/practice, as well as for CS2700)

4. [Two Sum] Given an sorted array A of integers, of size N and an integer $Target$, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution.
   Try solving the problem with $\Theta(N^2)$, $\Theta(N \log N)$ and O(N) complexity.

   Input Format:
   The first line will contain integer N (the size of array) and integer $Target$.
   The next line has N elements $A_1, A_2, ..., A_N$ in non-decreasing order.

Output Format:
Print two integer representing indexes of the numbers that add up to target.

Sample Input 1
4 13
2 7 11 15

Sample Output 1
0 2

5. [MAJORITY ELEMENT] A majority element in an array, A, of size N is an element that appears more than $N/2$ times. For example, the array $[3,4,4,2,3,4,2,4,4]$ has a majority element (4), whereas the array $[3,4,4,2,3,4,2,4]$ does not have any majority element. If there is no majority element, your program should print -1.
   Try solving the problem with O(N^2) , O($NlogN$) and O(N).

   Input Format:
   The first line will contain integer N, the number of elements in array.
   The next line has N elements $A_1, A_2, ..., A_N$.

   Output Format:
   Print the majority element of array. (If there is no majority element, print -1)

   Sample Input 1
   9
   3 4 4 2 3 4 2 4 4

   Sample Output 1
   4

   Sample Input 2
   8
   3 4 4 2 3 4 2 4

   Sample Output 2
   -1

6. [FREQUENT WORDS PROBLEM] Given a DNA string S consisting of alphabets A, C, G, T, find the most frequent k-mers in the given string. Frequent k-mers in a given DNA sequence or genome sequence may hint at regions of the DNA/genome that are hiding some biologically interesting message.
   k-mer is a length-k substring of a string.
   What is the running time complexity of your code? (consider length of string S = n)

   Input Format:
   The first line will contain integer k.
   The next line will contain string S, over the alphabet {A,C,G,T}.

   Output Format:
   Return all most frequent k-mers in S (if more than one, list them in lexicographic (alphabetical) order).

Sample Input 1
3
ACGTAACG

Sample Output 1
$[''ACG'']$

Explanation
All possible substring of length 3 = $[ACG, CGT, GTA, TAA, AAC, ACG]$
Most frequently occuring 3-mer is $['ACG']$, since it has frequency 2 in the above multi-set, and all other 3-mers have frequency 2.

Note:
Try to solve problem using Array/Vector only. (Don't use any other data structure).
You can use built-in sort() if required.