Data Structures and Algorithms

Practice sheet

Proof of correctness of iterative algorithm

We discussed extensively the method of proving the correctness of an iterative algorithm. Recall that each such proof required stating a suitable assertion $P(i)$ that captures the *progress* of the algorithm at the end of the ith iteration. After that, we establish its correctness by giving a proof of induction on the number of iterations. In this proof, first we establish the validity of the assertion before entering the loop (this is the base case). Then we use the induction hypothesis (validity of $P(i-1)$) and the body of the loop to establish meticulously the validity of $P(i)$. In this practice sheet, first you are given 2 standard iterative algorithms and you will have to establish their proof of correctness. Then you are given a problem for which you need to design an efficient iterative algorithm, write its pseudocode, and then prove its correctness.

1. Write a neat pseudocode of the iterative Binary Search and prove its correctness.
   Important advice: If you really wish to learn from this problem, then do not search for the pseudocode of binary search anywhere (book, web, friend, ...). Instead, you must come up with the pseudocode on your own. Interestingly, this will help you to derive its proof of correctness as well.

2. Recall the following algorithm for GCD.

```
GCD(a,b)    // here a is greater than or equal to b.
{
   while b <> 0
   {
         t <- b
         b <- a mod b
         a <- t
   }
   return a
}
```

   You have to establish the correctness of this algorithm.

3. You are given a number x and an array A storing n numbers in the increasing order. You need to design an $O(n)$ time algorithm to determine if there exist any 2 distinct indices i and j such that $A[i] + A[j] = x$. Prove the correctness of the pseudocode.

Note: Hints are given on the next page.

1. It should be clear from the pseudocode of binary search that if binary search is successful, then the element is indeed in the array. So you have to establish that if the outcome of binary search is failure, then the search element is nowhere in the array. Use this observation to come up with the appropriate assertion. The proof of correctness of the iterative algorithm for local minima in array might be useful. So go through that proof carefully before you attempt the current proof.

2. You may, if you wish, use the following facts from elementary number theory. If p divides q and p divides r, then p divides $(xq + yr)$ for every integer x and y.

3. *Hint:* There is a simple $O(n \log n)$ time algorithm to solve this problem by executing n calls of binary search. In order to achieve $O(n)$ time, you need to bypass binary search. Start as follows. First determine whether $A[0] + A[n-1] = x$. If $A[0] + A[n-1] > n$, what do you infer ? If $A[0] + A[n-1] < n$, what do you infer ? Can you discard at least one of 0 and $n-1$ if $A[0] + A[n-1] \neq 0$? Proceed along these lines. Try to use these observations in the proof of correctness as well.