Algorithms: Solutions 2

Problem 1

Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an *inversion*. Write an algorithm that determines the number of inversions in $A[1..n]$.

INVERSIONS(A, n)
counter $\leftarrow 0$
for $i \leftarrow 1$ **to** $n - 1$
    do for $j \leftarrow i + 1$ **to** n
            do if $A[i] > A[j]$
                  then *counter* $\leftarrow$ *counter* $+1$
return *counter*

The time complexity of the INVERSIONS algorithm is $\Theta(n^2)$.

Problem 2

Let $A[1..n]$ be a *sorted* array of n distinct numbers. Write an algorithm that finds a given value k in the array $A[1..n]$ and returns its index. If the array does not include k, the algorithm returns 0.

BINARY-SEARCH(A, n, k)
$p \leftarrow 1$
$r \leftarrow n$
while $p < r$
    do $q = \lfloor (p + r)/2 \rfloor$
        if $k \leq A[q]$
            then $r \leftarrow q$
            else $p \leftarrow q + 1$
if $k = A[p]$
   then return p
   else return 0

The time complexity of this binary search is $\Theta(\lg n)$.