Test 2: Divide and Conquer

For **#1, #2(i), #3(i), #4, and #5** below, you need to describe **(i)** the divide-and-conquer algorithm you propose for solving the problem, **(ii)** the resulting recurrence relation in the form of $T(n)= a*T(n/b) + n^c$ regarding the time complexity $T(n)$ and **(iii)** the time complexity $T(n)$ in closed form according to the master method in the handout and the actual coefficients a, b and c, in the recurrence relation in (ii).

1. Given an array A of n elements $A[1], A[2] ,..., A[n]$ and you are also told the values of these n elements have the following V shape property: there is a unique (untold) index k in the range of [1,n] such that A[i]>A[i+1] for every index i in the range of [1, k-1] and A[i]<A[i+1] for every index i in the range of [k, n-1]. Obviously A[k] is the minimum among all the values in the entire array. **Devise a good divide-and-conquer algorithm to find this minimal value. The time complexity of the algorithm should be better than $\Theta(n)$. 10 points.**

2.

(i) Given an array A of n elements $A[1], A[2] ,..., A[n]$ storing arbitrarily large integers, the maximum value gap in this array is the maximum of <u>$A[i]-A[j]$</u> over all possible index pairs i and j **where $i>j$**. **Devise a good divide-and-conquer algorithm to find the maximum value gap. Your algorithm should be more efficient than an Θ (n^2) brute force search to find the maximum value gap. 10 points.**

(ii) Implement your algorithm as a running program that can ask the user to enter the n values and then determine the maximum value gap. **5 points.**

3.

(i) Given an array A of n elements $A[1], A[2] ,..., A[n]$ storing arbitrarily large integers, we say a pair of values of $A[i]$ and A[j] is a significant inversion if and only if $i < j$ and $A[i]> 2*A[j]$. **Devise a good divide-and-conquer to find the total number of significant inversion pairs in A. Your algorithm should be more efficient than an $\Theta(n^2)$ brute force search that examine all the $\Theta(n^2)$ pairs to find the number of significant inversions. 10 points.**

(ii) Implement your algorithm as a running program that can ask the user to enter the n values and then determine the number of significant inversion pairs. **5 points.**

4. Consider a two dimensional array A of ***n*** elements $A[i][j]$ where i and j are indices in the range of $[1,n^{1/2}]$. All the elements $A[i][j]$ are **distinct** integers. Two elements $A[i][j]$ and $A[k][l]$ are neighbors if and only if $|i-k|+|j-l|$ equals 1. An element $A[i][j]$ is a local minimum if and only if the integer stored in $A[i][j]$ is smaller than all of its neighbors. **Devise a good divide-and-conquer to find a local minimum. Your algorithm should be more efficient than an $\Theta(n)$ brute force search that examine all the n elements and their neighbors. 10 points.**

5. **Problem for bonus points.** Given an array *A* of *n* elements *A[1], A[2] ,…, A[n]* storing arbitrarily large integers, we say an integer k is the majority value in this array if and only if more than n/2 of the elements in A store this particular value k. Note that there may or may not be a majority value. Assuming that checking whether two integers are equal takes only constant time and n is a power of 2. **Devise a good divide-and-conquer algorithm that uses no more than *2n* comparisons to determine whether the majority value does exist in the array and if it does exist find out this majority value.** **Bonus 10 points.**