CS1713 – ALGORITHM DESIGN AND ANALYSIS

Sort Algorithms

SelectionSort(A[0……n – 1]) (Page 99)

```
// Sorts a given array by selection sort
// Input: An array A[0……n – 1] of orderable elements
// Output: Array A[0……n – 1] sorted in ascending order
for i = 0 to n – 2 do
   min = i
   for j = i + 1 to n – 1 do
      if A[j] < A[min] min = j
   swap A[i] and A[min]
```

InsertionSort(A[0……n – 1]) (Page 134)

```
// Sorts a given array by insertion sort
// Input: An array A [0……n – 1] of n orderable elements
// Output: Array A [0……n – 1] sorted in nondecreasing order
for i = 1 to n – 1 do
   v = A[i]
   j = i – 1
   while j >= 0 and A[j] < v
      A[j + 1] = A[j]
      j = j – 1
   A[j + 1] = v
```

BubbleSort(A[0……n – 1]) (Page 100)

```
// Sorts a given array by bubble sort
// Input: An array A[O……n – 1] of n orderable elements
// Output: Array A [O……n – 1] sorted in nondecreasing order

for i = 0 to n – 2 do
   for j = 0 to n – 2 - i do
      if A[j + 1] < A[j] swap A[j] and A[j + 1]
```

ImprovedBubbleSort(A[0……n – 1])

```
// Sorts a given array by bubble sort
// This version ends as soon as the array is sorted
// Input: An array A[0……n – 1] of n orderable elements
// Output: Array A[0……n – 1] sorted in nondecreasing order

for i = 0 to n – 2 do
   noSwaps = true
   for j = 0 to n – 2 - i do
      if A[j + 1] < A[j]
         swap A[j] and A[j + 1]
         noSwaps = false
   if noSwaps  // array is sorted
```

Mergesort(A[0……n – 1]) (Page 172)

```
// Sorts array by recursive mergesort
// Input: An array A[0……n – 1] of n orderable elements
// Output: Array A[0……n – 1] sorted in nondecreasing order

if n > 1
   copy A[0……⌊n/2⌋ - 1] to B[0……⌊n/2⌋ - 1]
   copy A[⌊n/2⌋……n - 1] to C[0……⌊n/2⌋ - 1]
   Mergesort(B[0……⌊n/2⌋ - 1])
   Mergesort(C[0……⌊n/2⌋ - 1])
   Merge(B, C, A)    // Merge B and C into A
```

Quicksort(A[l…r]) (Page 176)

```
// Sorts a subarray by recursive quicksort
// Input: Subarray of array A[O……n – 1] defined by its
//    left and right indices l and r
// Output: Subarray A[l……r] sorted in nondecreasing order

if l < r
   s = Partition(A[l……r])   // s is the pivot (split) position
   Quicksort(A[l……s - 1])
   Quicksort(A[s + 1……r])
```

RunTimes (= Number of Comparisons)

Sort Algorithm	Best	Worst	Average
Selection			
Insertion			
Bubble			
Improved Bubble			
Merge			
QuickSort with regular Partition			
QuickSort with Partition that uses median of 3			