Antonio Rosado
PreQuiz 5

Sequential search in unordered collection
T(n) = T(n-1) + 1
T(0) = 0
T(n) = n
O(n)

Modified sequential search I
T(n) = T(n-1) + 2
T(0) = 0, T(1) = 1
T(n) = n
O(n)

Modified sequential search II
T(n) = T(n/2) + 2
T(0) = 0, T(1) = 1
T(n) = sqrt(n)
O(sqrt(n))

Modified sequential search III
T(n) = T(n/2) + 1
T(0) = 0, T(1) = 1
T(n) = log2(n)
O(log n)

Binary search I
T(n) = T(n/2) + 1
T(1) = 1
T(n) = log2(n)
O(log n)

Binary search II
T(n) = T(n/2) + 1 or T(n/2) + 2
T(1) = 1
T(n) = log2(n)
O(log n)

Binary search III
T(n) = T(n/2) + 1 or T(n/2) + 2
T(1) = 1
T(n) = log2(n)+1
O(log n)

Algorithm Name:
Recurrence Relation (worst **case**) Recurrence Relation (best **case**)
Base Case and Cost
Closed Form (worst **case**) Closed Form (best **case**)
Time Complexity (worst **case**) Time Complexity (best **case**)

Bubble Sort
T(n) = T(n-1) + T(n-2) + ... + 1
T(n) = T(n-1) + T(n-2) + ... + 1
T(1) = 0
T(n) = n^2
T(n) = n^2
O(n^2) O(n^2)

Improved Bubble Sort
T(n) = T(n-1) + T(n-2) + ... + 1
T(n) = 1
T(1) = 0
T(n) = n^2
T(n) = n
O(n^2) O(n)

Selection Sort
T(n) = T(n-1) + T(n-2) + ... + 1
T(n) = T(n-1) + T(n-2) + ... + 1
T(1) = 0
T(n) = n^2
T(n) = n^2
O(n^2) O(n^2)

Improved Selection Sort
T(n) = T(n-1) + T(n-2) + ... + 1
T(n) = T(n-1) + T(n-2) + ... + 1
T(1) = 0
T(n) = n^2
T(n) = n^2
O(n^2) O(n^2)

Insertion Sort
T(n) = T(n-1) + T(n-2) + ... + 1
T(n) = 1
T(1) = 0
T(n) = n^2
T(n) = n
O(n^2) O(n)

Improved Insertion Sort
T(n) = T(n-1) + T(n-2) + ... + 1
T(n) = 1
T(1) = 0
T(n) = n^2
T(n) = n
O(n^2) O(n)

Merge Sort
T(n) = 2T(n/2) + n
T(n) = 1
T(1) = 0
T(n) = n log n
T(n) = n log n
O(n log n) O(n log n)

Quick Sort
T(n) = T(k) + T(n-k-1) + n
T(n) = T(k) + T(n-k-1) + n
T(1) = 0
T(n) = n^2
T(n) = n log n
O(n^2) O(n log n)

Bubble Sort - Compares pairs and largest item \221bubbles\222 to the last index.

Improved Bubble Sort - Same as bubble sort, where it differs is it stops when sorted rather than decreasing size of subarray and repeating.

Selection Sort - Utilizes index comparisons to select the largest item and move it to the last index.

Improved Selection Sort – Same as selection, where it differs is it stops **if** the collection is sorted.

Insertion Sort – Begins with a somewhat sorted subarray of size 1, then inserts a **new** item into the collection by using sequential search to identify position, and inserting accordingly.

Improved Insertion Sort – Same as Insertion, where it differs is it uses Binary Search instead of Sequential Search.

Merge Sort – Starts with singletons, then merges into sorted subarrays until the whole array is sorted; reaching completion.

Quick Sort – Partitions array into subarrays and sorts it as partitions, and then combines into a single sorted array.