Analysis of Algorithms

*Defend all answers based on specific references to the code. Do not count return statements or initialization of method arguments. You are* ***strongly*** *encouraged to walk through algorithms in the debugger and to add statement-counting code to given methods to test and refine your analysis.*

# Algorithm: find()

# Minimum Statements:

# How many statements would be executed in a call to find() when the array size is zero (n == 0)?

# Best Case Scenario:

# Under what conditions would the minimum number of statements be executed for an array where n is large?

# Where would the target element be located in the array?

# What is the growth function under these conditions?

# Worst Case Scenario:

# Under what conditions would the maximum number of statements be executed for an array where n is large?

# Where would the target element be located?

# What is the growth function under these conditions?

# Expected Average Case Scenario:

# Assuming a random array of unique elements and the target element is in the array, where would a target element be located on average?

# What is the expected average number of statements (the expected growth function) for a call to find()?

# What is the runtime order (big-O) of find() based on the above growth functions?

# Algorithm: replaceAll()

# Minimum Statements:

# How many statements would be executed in a call to replaceAll() when the array size is zero (n == 0)?

# Best Case Scenario:

# Under what conditions would the minimum number of statements be executed for an array where n is large?

# How many occurrences of the oldValue element would be in the array?

# Where would it/they be located in the array?

# What is the growth function under these conditions?

# Worst Case Scenario:

# Assuming newValue and oldValue are not equal, under what conditions would the maximum number of statements be executed for an array where n is large?

# How many occurrences of oldValue are in the array?

# Where would it/they be located?

# What is the growth function under these conditions?

# Expected Average Case Scenario:

# Assuming a random array of unique elements and oldValue is a value in the array, what is the average number of statements (the expected growth function) for a call to replaceAll()?

# What is the runtime order (big-O) of replaceAll() based on the above growth functions?

# Algorithm: sortIt()

# Minimum Statements:

# How many statements would be executed in a call to sortIt() when the array size is zero (n == 0) or one (n == 1)?

# Best Case Scenario:

# Under what conditions would the minimum number of statements be executed for an array where n is large?

# Would the algorithm execute a different number of statements if the elements in the array were already in sorted order? Reverse order? Random order? All the same value?

# What is the growth function under the best case conditions?

# Worst Case Scenario:

# Under what conditions would the maximum number of statements be executed for an array where n is large? (Already in some kind of sorted order? Duplicates?)

# What is the growth function under the worst case conditions?

# Expected Average Case Scenario:

# Assuming a random array of unique elements, what is the expected average number of statements (the expected growth function) for a call to sortIt()?

# What is the runtime order (big-O) of sortIt() based on the above growth functions?