Metropolitan State University

ICS 140 Computational Thinking with Programming

Class Exercise 11

**Lecture Section**

1. What is the name of the search algorithm that loops over every item in the list to check if it contains the sought item?

Seqential sorting

1. The binary search method is faster than searching all items but has an additional constraint. What is required of a list to use binary search?

Needs to be sorted

1. What does big O notation represent?

Time needed/ Complexity/ scalability

1. Describe the Selection Sort algorithm.

Keeps looking until it finds the next lowest value and moves it to the correct location 1 at a time

1. Describe the Insertion Sort algorithm.

Essentially sorts left to right 1 item at a time

1. Describe the Bubble Sort algorithm.

Looking at 2 values at a time slipping accordingly, rinse and repeat

1. Describe the Merge Sort algorithm.

Divides the list in half until it gets singular values and reconstructs the list sorting the values along the way

1. For the sort algorithms above, which one will generally be the most efficient as lists grown larger?

Merge Sort

1. In addition to the increase in processing required as list grow, name another resource constraint to watch for.

Memory

1. Coding sorting algorithms requires a way to swap values between 2 places in a list. Choose one of the methods that can be used and write the python code for swapping values below.

Generic Way

Temp = myList[0]

myList[0] = myList[1]

myList[1] = temp

**Coding Search Algorithms**

Write python function that implements the sequential and binary search algorithms. Use the provided search\_algorithms.py file provided to complete this exercise. In the search\_algorithms.py file, you will find some functions already built that will help test out the performance of the algorithms. I have also copied the pseudocode from the assigned reading to help simplify the coding process. Write functions with the names and inputs of the routines in the commented pseudocode. I would recommend copying the comments for each algorithm and then go through it line by line converting the pseudocode to python.

Note: Some lines of the pseudocode are not relevant to python. For example, there is no end if statement like there is in other programming languages. Lines like this can be deleted or commented out.

The output should like something like this:

Text

Description automatically generated

Smaller lists will be very close in speed. If we test out a larger number, we should see the binary search pull away with dramatically.

Text

Description automatically generated

Once you have coded the functions, paste the code for the 2 functions in their designated area below and take a few screenshots of your program running with different sizes of lists similar to the screenshots above.

**Selection Search Function code here**

def sequential\_search(mylist, target):

    found = False

    index = 0

    while index < len(mylist) and not found:

        if mylist[index] == target:

            found = True

        else:

            index = index + 1

    return found

**Binary Search Function code here**

def binary\_search(mylist, target):

    low = 0

    high = len(mylist) - 1

    found = False

    while low <= high and not found:

        mid = (low + high) // 2

        if mylist[mid] == target:

            found = True

        else:

            if mylist[mid] < target:

                low = mid + 1

            else:

                high = mid - 1

    return found

**Screenshot of Test here**

**Text

Description automatically generated**