**Spring 2018 - Sec. 2**

**Weekly Assignment #11**

**RUNNING TIME OF ALGORITHMS, BIG OH & RECURSION**

**DUE by midnight on 180416**

**Why:**

There are numerous searching and sorting algorithms. How do we decide which algorithm to use and when? One way of deciding which algorithm to use is by comparing the running time (or work required) of the various algorithms in relation to the number of inputs, n. We can also use the same concepts to help us develop more efficient code.

Recursive code is often more elegant than recursive code; in some cases the recursive code is the only way to solve a problem. Recursion is used in cases of playing games, drawing graphics, and searching and sorting.

**Learning Objectives:**

1. Compare the amount of work of a sequential search, binary search, and selection sort in relation to n.
2. Understand how this relates to Big O notation.
3. To learn to write a recursive binary search method
4. To learn to write a recursive sequential search method.
5. To learn an appreciation for using recursion for solving problems, playing games and displaying graphics.

**Performance criteria:**

*Criterion #1*: *Be able to compare the amount of work of a sequential search, binary search, and selection sort.*

1. Be able to determine the amount of work in a sequential search.
2. Be able to determine the amount of work in a selection sort.
3. Be able to determine the amount of work in a binary search.

*Criterion #2: Understand how to use this information to obtain Big O notation for each algorithm.*

1. Demonstrate knowledge of how to obtain Big O notation for the sequential search.
2. Demonstrate knowledge of how to obtain Big O notation for the selection sort.
3. Demonstrate knowledge of how to obtain Big O notation for the binary search.

*Criterion #3*: To learn to write a recursive binary search method

1. Demonstrate a working recursive binary search method.

*Criterion #4*: To learn to write a recursive sequential search method.

*a*. Demonstrate a working recursive sequential search method.

*Criterion #5*: To learn an appreciation for using recursion for solving problems, playing games and displaying graphics.

a. Use an online search engine to find a recursive graphics program online (many “fill” algorithms are implemented recursively!)

b. Use an online search engine to find a recursive gaming program online.

c. Use an online search engine to find a recursive Towers of Hanoi problem.

**Problem:**

You have been able to search and sort a list user-defined objects (one of the class definitions you have coded). Is it more efficient to sort the data before searching OR is it better to just leave the data unsorted and do a sequential search?

**Solution:**

In previous labs you used the sequentialSearch() , selection sort(), and binSearch() methods to manipulate your list of objects. You can use this code to determine and compare the amount of work needed to accomplish each of these tasks. You should have all of these methods in your last version of the lab which used seqSearch(), selectSort() and binSearch(). How much work is it to sort the object list then use binSearch() rather than leave the list unsorted and use the sequential search method? To answer this question you can add a loop counter to the *inner loop* of the selectSort () method to determine how much work is being done to sort the object list an input of n=20. The loop counter will increment each time the inner loop is executed. You can also add a counter to both the seqSearch() method loop and the binSearch() method loop. The searches should count the number of iterations of the loop required before the searchkey is found. Be sure to print out the values of each of the counters so you can easily see the amount of work performed for each method. You should fill your object list then use sequential search to find the search data in the left column; you should record the counter and where found for each data item. Sort the data using selectSort() (count how much work it takes to sort the data) then use the binSearch() to again find each of the searchkeys; record your results. Be sure you place your counters so you will count each iteration and print before the return from the method occurs!

**Exercises:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Number of comparisons:**  **Perform these on an array of 20 objects** | | |
| **Use a user-defined object that you have coded** | *Sequential search()* | *SelectSort()* | *binSearch()* |
| Object not found | 20 | 89 | 4 |
| Object is found with 1st try(object at 0) | 1 | 89 | 1 |
| **Average # of above:** | **10.5** | **89** | **2.5** |

Note: Your selectSort() method will not show any differences since a search is not being performed.

**Summary of above table:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Sequential search()* | *SelectSort()* | *binSearch()* |
| Best case | 1 | 0 | 1 |
| Worst Case | 20 | 400 | log(20) ; base 2 |
| Average case | 10 | 200 | log(20)/2 ; base 2 |
| Expected amount of work compared to N using BigO notation | O(n) | O(n^2) | O(log(n)) ; base 2 |

1. What is the value of n for the exercises? \_\_\_\_\_The number of iteration of comparison that is performed before search key is founded \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. What is the expected number of comparisons for selectSort for the exercises? \_\_\_\_\_\_On the worst case, it is expected to be 400 or less iterations\_\_\_\_\_\_\_\_
3. What is (logn) for the exercises? \_\_\_\_This is the big o notation for binary search\_\_\_\_\_\_\_\_
4. Does the Big O notation best approximate the best, worst, or average case for the sorts/search?\_\_\_\_\_\_\_It approximates the worst case.\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_
5. When would it be more efficient to selectSort data then use binary search versus using sequential search? \_\_\_\_\_\_\_\_It would be when you have a large data base and you have to frequently search for objects in that data base.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Use the above knowledge to predict how many comparisons will be required for each of the below searches/sorts.

**Prediction:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Number of comparisons:**  **Perform these on an array of 40 objects** | | |
| **Use a user-defined object that you have coded** | *Sequential search()* | *SelectSort()* | *binSearch()* |
| Object not found | 40 | 1600 | log(40) ; base 2 |
| Object is found with 1st try | 1 | 1600 | 1 |
| **Average # of above:** | **20.5** | **1600** | **3.16** |

Can you confirm your predictions? Yes, I’ll just need to do the same thing as the exercise with 20 objects.

**Recursion**:

#1:Use the lab from a previous week where you used binary search to “find” Books. Add a recursive binary search method to your application-specific subclass code.

#2: Add a recursive sequential search method to your application-specific subclass code from above.

Fill in the below table:

**Exercises: record search results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use a user-defined object that you have coded** | *binSearch()* | *rBinSearch()* | *sequentialSearch()* | *rSequentialSearc()* |
| Object not found | 4 | 4 | 20 | 20 |
| Object found(object 7) | 3 | 3 | 7 | 7 |
| Object found**(object 8)** | **4** | **4** | **8** | **8** |

Use an online search engine to find the following:

1. Demonstration of recursive Towers of Hanoi problem. Record the link below:

\_\_\_\_\_\_\_\_\_\_\_\_\_ https://www.youtube.com/watch?v=YstLjLCGmgg\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Demonstration of a recursive gaming program. Record the link below:

\_\_\_\_\_\_\_\_\_ <https://www.youtube.com/watch?v=0II_uz5UwAc&t=17s____________>

- the defile card is a recursive function

1. Demonstration of a recursive graphics program (many fill programs are recursive!). Record the link below.

\_\_\_\_\_\_\_ <https://www.youtube.com/watch?v=p_7GWRup-nQ________>

The recursive scanning of hypergan

**Grading:**

Exercises: 3 points (1pt./row)

Summary & Big O notation 4 points (1 pt./row)

Answers to 5 questions: 5 points (1 pt/question)

Prediction table 3 points (1 pt/row)

Recursive binary search method 1 point

Recursive sequential search method 1 point

Recursion table 3 points

TOTAL 20 points

Note Your code should not match any one else’s code.