Homework Assignment 5

Peer Reviewed

Due Date: Thursday, April 21st, 2016 BEFORE CLASS

Objectives: Upon completing this assignment, you will be able to empirically and theoretically analyze sorting algorithms.

Tasks: (Total Points Possible: 20)

1. **Programming (6pts)**: You are given C++ code for **`quicksort`** and another (very bad) sorting algorithm, called **`badsort`**. Modify the given code, so that
   a. It will count the number of comparisons between objects in the input array for each sorting algorithm.
   b. It will calculate the *real execution time* of each sorting algorithm.
   (*Hint*: use the function call **`time(0)`** to record the current time. Function **`time()`** is defined in the **`ctime`** library.)
   c. It allows the user to enter the length of an input array to be sorted, and choose either to enter values or to generate random numbers for the array elements. It also allows the user to view the randomly generated array if he/she wants to. Make sure that it will generate different sequences of random numbers at different runs of the program.
   d. It will call the two sorting algorithms on the same input array obtained from step (c) above, and display both the number of object comparisons and the real execution time for each sorting algorithm call. It also allows the user to view the output array for each sorting algorithm call if he/she wants to.

   See pages 3-5 below for sample runs of this program.

2. **Empirical Analysis (6pts)**:
   a. Run your program to sort an input array of n random integers for **10** different values of n in the range [10, 100]. Then report the number of comparisons of each sorting algorithm call in a table of the same format as Table 1 below.
   b. Run your program to sort an input array of n random integers for **5** different values of n in the range [10000, 100000]. These runs must be done on the same computer. Then report the real execution time of each sorting algorithm call in a table of the same format as Table 2 below.
   c. Draw a line graph (using Excel) that shows the number of comparisons of each sorting algorithm call from your experiments in part (2a) as well as the two functions n^2 and $n\log(n)$. So, there will be four lines in this graph.
   d. Draw a conclusion regarding the number of comparisons based on the graph, and draw another conclusion regarding the real execution time from your data collected in part 2b.

Table 1: Number of Object Comparisons					Table 2: Real Execution Time		
n	quicksort	badsort	n^2	$n\log(n)$	n	quicksort	badsort
10					10000		
20							
…							

3. **Theoretical Analysis (8pts)**:
    a. Analyze the space complexity for the **`badsort`** algorithm.
    b. Analyze the worst-case time complexity for the **`badsort`** algorithm. For this part, you will have to *estimate* the total number of object comparisons made by badsort:
        i. Try to find a good asymptotic upper bound on the total number of object comparisons made by badsort on an input array of length n
        ii. Try to find a good asymptotic lower bound on the total number of object comparisons made by badsort on an input array of length n in *reverse order*.

Submission:

- Write your program for Task 1 in a single cpp file. Save your answers to Tasks 2 and 3 as **.pdf** files.
- Upload the **.cpp** file of your source code and the **.pdf** files on Canvas.
- *No submission by email will be accepted.*
- *No late submission will be accepted*.

Peer Review:
This assignment will be reviewed and graded by your peers. Each student who has submitted their work will be assigned to review the work of a random student in the class. Students who fail to submit their work will not be assigned work for peer review.

To complete the review, the reviewer needs to write at least one comment and fill out the provided rubric on Canvas. Detailed guidelines for peer review will be posted after the due date of this assignment. The deadline for submitting peer review is **Tuesday, April 26th, 2016 BEFORE CLASS**. If you don't submit your peer review on time, you just lose your opportunity of reviewing your peer's work.

Students who do not agree with their reviewer should contact the instructor within one week of the date the grade is posted. In any case, *the instructor reserves the right to double-check and regrade any peer-reviewed work.*

Sample Runs for the Program in Homework 5

In these sample runs, the text in bold is the input entered by the user.

Sample 1:

```
Enter the length of the array to be sorted: 5
Do you want to enter values for the input array? (y/n): y
Enter values for the input array:
12 23 4 5 10

Sorting the array using quicksort...
Time taken by quicksort to sort 5 numbers is 0 seconds.
Total number of input object comparisons made by quicksort is 19
Would you like to see the sorted array? (y/n): y
The array after being sorted is:
4       5       10      12      23

Sorting the array using badsort...
Time taken by badsort to sort 5 numbers is 0 seconds.
Total number of input object comparisons made by badsort is 19
Would you like to see the sorted array? (y/n): y
The array after being sorted is:
4       5       10      12      23

Press any key to continue . . .
```

Sample 2:

```
Enter the length of the array to be sorted: 9
Do you want to enter values for the input array? (y/n): y
Enter values for the input array:
23 15 3 24 28 16 30 12 28

Sorting the array using quicksort...
Time taken by quicksort to sort 9 numbers is 0 seconds.
Total number of input object comparisons made by quicksort is 44
Would you like to see the sorted array? (y/n): y
The array after being sorted is:
3       12      15      16      23      24      28      28      30

Sorting the array using badsort...
Time taken by badsort to sort 9 numbers is 0 seconds.
Total number of input object comparisons made by badsort is 59
Would you like to see the sorted array? (y/n): y
The array after being sorted is:
3       12      15      16      23      24      28      28      30

Press any key to continue . . .
```

Sample 3:

```
Enter the length of the array to be sorted: 20
Do you want to enter values for the input array? (y/n): n
We have generated 20 random numbers for the input array.
Would you like to see the generated array? (y/n): y
The array generated is:
2709    15021   2575    32215   14821   18422   32197   2115    16701   14434
10856   21735   32486   17121   13748   21217   17564   21782   28248   3958



Sorting the array using quicksort...
Time taken by quicksort to sort 20 numbers is 0 seconds.
Total number of input object comparisons made by quicksort is 150
Would you like to see the sorted array? (y/n): y
The array after being sorted is:
2115    2575    2709    3958    10856   13748   14434   14821   15021   16701
17121   17564   18422   21217   21735   21782   28248   32197   32215   32486


Sorting the array using badsort...
Time taken by badsort to sort 20 numbers is 0 seconds.
Total number of input object comparisons made by badsort is 766
Would you like to see the sorted array? (y/n): y
The array after being sorted is:
2115    2575    2709    3958    10856   13748   14434   14821   15021   16701
17121   17564   18422   21217   21735   21782   28248   32197   32215   32486


Press any key to continue . . .
```

Sample 4:

```
Enter the length of the array to be sorted: 25
Do you want to enter values for the input array? (y/n): n
We have generated 25 random numbers for the input array.
Would you like to see the generated array? (y/n): y
The array generated is:
16320   1870    31926   1401    2221    23881   24769   16033   26519   29342
5823    5614    25630   11013   17368   27381   12877   12777   20255   3218
6500    22040   7793    4664    14299


Sorting the array using quicksort...
Time taken by quicksort to sort 25 numbers is 0 seconds.
Total number of input object comparisons made by quicksort is 182
Would you like to see the sorted array? (y/n): n

Sorting the array using badsort...
Time taken by badsort to sort 25 numbers is 0 seconds.
Total number of input object comparisons made by badsort is 1968
Would you like to see the sorted array? (y/n): n

Press any key to continue . . .
```

Sample 5:

```
Enter the length of the array to be sorted: 5000
Do you want to enter values for the input array? (y/n): n
We have generated 5000 random numbers for the input array.
Would you like to see the generated array? (y/n): n


Sorting the array using quicksort...
Time taken by quicksort to sort 5000 numbers is 0 seconds.
Total number of input object comparisons made by quicksort is 96248
Would you like to see the sorted array? (y/n): n

Sorting the array using badsort...
Time taken by badsort to sort 5000 numbers is 43 seconds.
Total number of input object comparisons made by badsort is 1016104700
Would you like to see the sorted array? (y/n): n

Press any key to continue . . .
```

Sample 6:

```
Enter the length of the array to be sorted: 10000
Do you want to enter values for the input array? (y/n): n
We have generated 10000 random numbers for the input array.
Would you like to see the generated array? (y/n): n


Sorting the array using quicksort...
Time taken by quicksort to sort 10000 numbers is 0 seconds.
Total number of input object comparisons made by quicksort is 208294
Would you like to see the sorted array? (y/n): n

Sorting the array using badsort...
Time taken by badsort to sort 10000 numbers is 348 seconds.
Total number of input object comparisons made by badsort is -1123891535
Would you like to see the sorted array? (y/n): n

Press any key to continue . . .
```