HW5: Analysis and Benchmarking! 100 pts

Due: Tuesday, November 27th, at 11:59:59 pm

Overview

1. In this assignment you will review and analyse the complexity of different algorithms.
2. You will execute these algorithms and experience the how the analysis would be applicable to the code you run and benchmark the algorithms.

As you get started, please pay attention to the following:

- Please read the ENTIRE write-up before getting started.

Starter files:

- Entry.java
- Helpers.java
- Improve.java
- ImproveTest.java
- Search.java
- SearchTest.java
- Sort.java
- SortTest.java

You will submit the following files for this assignment:

- A report with the run time analysis and the run time plots for the respective algorithms. (You will be using gradescope for this)

Where to find the starter code

- **From ieng6 server at**
  - **<accountname>@ieng6.ucsd.edu:/home/linux/ieng6/cs12f/public/pa5/***

Part 1 - Analysis of search algorithms (30 points)

The problem of search is one of the most important problems in the field of computer science. Can you imagine a world without Google right now? It takes great effort and algorithmic innovation to provide such search results so fast. In this assignment you will experience how different algorithms affect the time taken to search for objects. How the time taken varies as the number of objects we are searching among varies. How the time taken varies when the cost of different operations vary. And how the theoretical analysis translates to practical effects.

In this assignment we have a task of searching through a set of values called 'Entry' corresponding to a person's last name. The entry also contains much more information about the person. For the sake of this assignment that value is just an integer.
The task at hand is to find a corresponding entry in a data-structure and return the integer associated with it. THIS HAS ALREADY BEEN IMPLEMENTED. Your task is to analyse and benchmark the algorithms which do this.

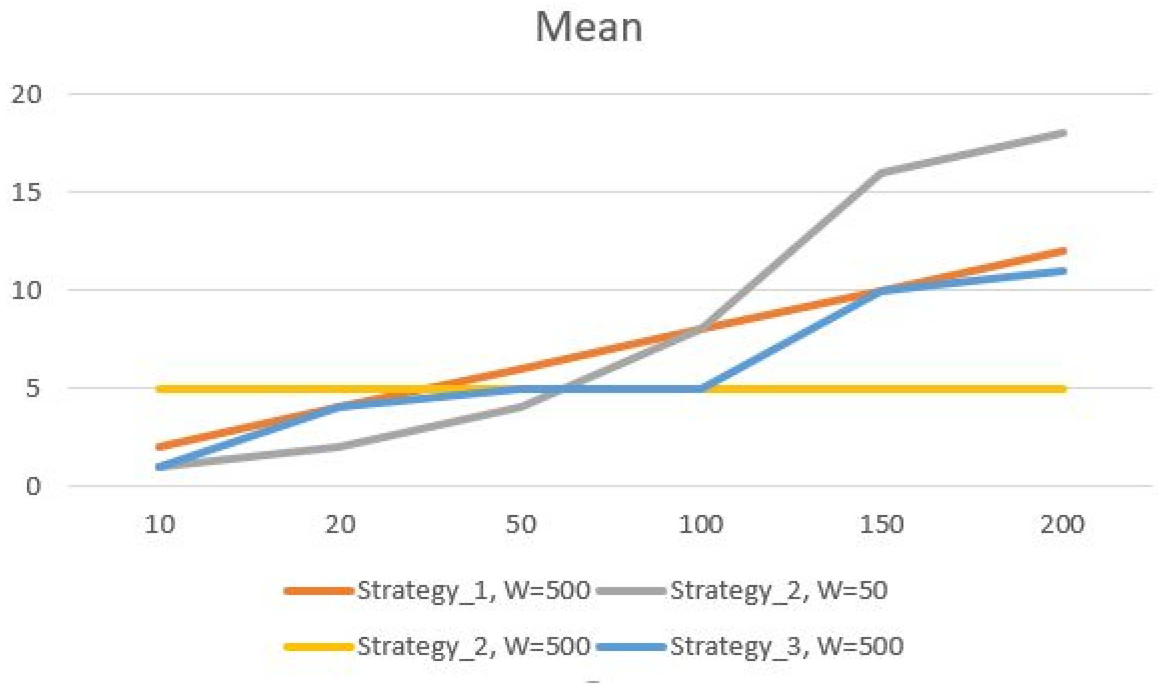
Q1. There are 3 strategies to to the above task in 'Search.java'. Go through each of them, analyse and prove the time complexity of each of the provided strategies. Provide answers in Big-O Notation, Omega Notation and Theta Notation wherever applicable. Provide the steps for your analysis. (**4pts each**)

Q2.

1. The function 'work()' in represents a constant and the most significant unit of work. In the current problem of search we assume that, this is the comparison to check if the element is the element we want. In order to highlight the significance of the algorithm run_times we have chosen to give a reasonable amount of delay which is representative of it being expensive.
2. The file 'Search' has a list of Entries in 'elements' through which it will search for the corresponding entry. The constructor will also take.
    a. Work_time - The constant unit of time taken to do a unit of work.
    b. Hash_time - The constant unit of time taken to run a hash function.
    c. N - Number of elements to search across.
2. The constructor loads the first N entries from a file called 'data.txt'. Feel free to open and inspect the elements in this.
3. You are expected to
    a. Write test cases in order to check the run time for each strategy. The file 'SearchTest.java' has a demo test case showing the use of time commands to inspect the time taken to execute a piece of code. Make use of it as a template to build your own test cases.
    b. Write test cases to check the time taken by each strategy for N = 10, 20, 50, 100, 200.
    c. For each of the above N, also vary the work_time between 10, 20, 50.
    d. For each ot the above cases try with a hash_time of 10, 500.
    e. As the algorithms performance might vary depending on the elements being searched for, randomly choose 10 different words from the data.txt, Provide the average, maximum and the minimum times among those. (Choose the words wisely and choose different words for each N)

NOTE: Please don’t write individual test cases for all of the cases. Make use of loops.

4. Plot a graph of total time taken for each of the strategy. With N as the X axis and time taken as the Y axis. Plot 3 graphs, one each (Mean, Minimum, Maximum) with all of the corresponding cases in the same graph. Use colored pen/pencils if you are doing it by hand instead of a software. Label each of the lines according to the configuration that generated the the corresponding entries. Along with the graphs with all the strategies, please attach individual graphs with the 3 strategies for each configuration.(**4pts each**) A sample graph is as shown below. (The values here are FAKE)



NOTE: Do your development work with a lesser work_time and finally run the code with the correct work_time. This will save your time!

Graphs needed:

Strategy	Work_Time	Hash_Time	Type
All	All	All	One Each for (Min, Max, Mean)
All	20	500	One Each for (Min, Max, Mean)
Strategy 1	10, 20, 50	10	One Each for (Min, Max, Mean)
Strategy 3, Strategy 2	10, 50	500, 10	One Each for (Min, Max, Mean)

Feel free to add any other graphs.

Q3. Make some Predictions about what values you would be getting for a particular scenario according to your theoretical analysis. (**2pts**)
Q4. Write about any of the patterns you noticed in your graph. Explain How its coherent with your analysis and prediction. Its ok if its not in sync with your prediction. Write all your interpretations from the graph. (If you notice something completely against your prediction, still write it down and say that it was surprising) (**3pts**)
Q5. Give a comparative statement about which algorithm is the best / most suitable and in what cases. (**1pt**)

Part 2 - Analysis of sort algorithms (30 points)

NOTE: Part 2 to is similar to Part 1, but with slight variations. If you are bored of reading and are confident that you could figure out the variations while going through the code, please feel free to skip reading this the following part. HOWEVER DO LOOK AT THE BOLDED PARTS.

The problem of sorting is another important problem in the field of computer science. It is used as a basic block for many important algorithms. In this assignment you will experience how different algorithms affect the time taken to sort the objects. How the time taken varies as the number of objects we are searching among varies. How the time taken varies when the cost of different operations vary. And how the theoretical analysis translates to practical effects.

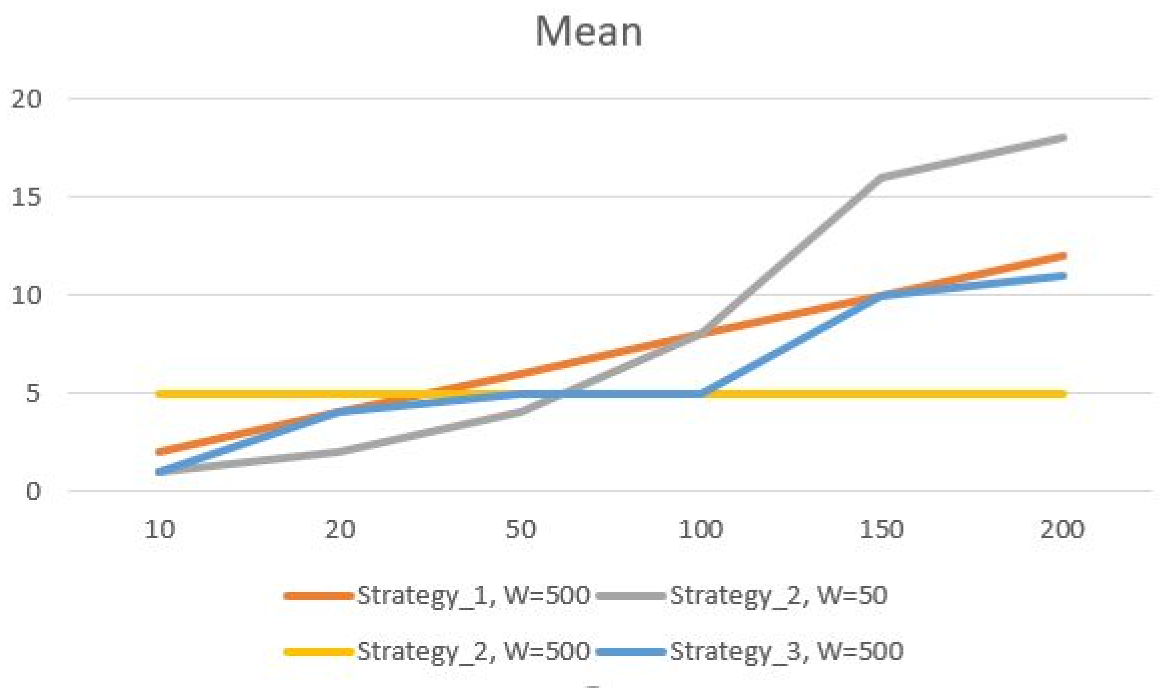
In this assignment we have a task of searching through a set of values called 'Entry' corresponding to a person's last name. The entry also contains many more information about the person. For the sake of this assignment that value is just an integer.
The task at hand is to sort the entries based on the key(Lastname). THIS HAS ALREADY BEEN IMPLEMENTED. Your task is to analyse and benchmark the algorithms which do this.

Q1. There are 3 strategies to to the above task in '`Sort.java`'. Go through each of them, analyse and prove the time complexity of each of the provided strategies. Provide answers in Big-O Notation, Omega Notation and Theta Notation wherever applicable. Provide the steps for your analysis. (**4pts each**)

Q2.

3. The function 'work()' in represents a constant and the most significant unit of work. In the current problem of search we assume that, this is the comparison to check if the element is the element we want. In order to highlight the significance of the algorithm run_times we have chosen to give a reasonable amount of delay which is representative of it being expensive.

4. The file ‘Search’ has a list of Entries in ‘elements’ through which it will search for the corresponding entry. The constructor will also take.
    a. Work_time - The constant unit of time taken to do a unit of work.
    b. N - Number of elements to search across.
    c. Offset - Offset while loading the data.
5. The constructor loads the first N entries from a file called ‘data.txt’. You are free to open and inspect the elements in this.
6. You are expected to
    a. Write test cases in order to check the run time for each strategy. The file ‘`SortTest.java`’ has a demo test case showing the use of time commands to inspect the time taken to execute a piece of code. Make use of it as a template to build your own test cases.
    b. Write test cases to check the time taken by each strategy for N = 10, 20, 50, 100, 200.
    c. For each of the above N, also vary the work_time between 10, 20, 50.
    d. As the algorithms’ performance might vary depending on the elements being sorted. In order to account for this. **We will use an offset and load the values from (offset to offset+N) in the file data.txt**. **To do this In the constructor set the offset value to 0, 200, 400, 600 .. 2000.**, for, Provide the average, maximum and the minimum times among those. (Choose the words wisely and choose different words for each N)
    NOTE: Please don’t write individual test cases for all of the cases. Make use of loops.
7. Plot a graph of total time taken for each of the strategy. With N as the X axis and time taken as the Y axis. Plot 3 graphs, one each (Mean, Minimum, Maximum) with all of the corresponding cases in the same graph. Use colored pen/pencils if you are doing it by hand instead of a software. Label each of the lines according to the configuration that generated the the corresponding entries. Along with the graphs with all the strategies, please attach individual graphs with the 3 strategies for each configuration. (**4pts each**) A sample graph is as shown below. (The values here are FAKE)



Graphs Needed:

Strategy	Work_Time	Type
All	All	One Each for (Min, Max, Mean)
All	5	One Each for (Min, Max, Mean)
Strategy 1	1, 2, 5	One Each for (Min, Max, Mean)
Strategy 3, Strategy 2	1, 5	One Each for (Min, Max, Mean)

Feel free to add any other graphs.

Q3. Make some Predictions about what value you would be getting for a particular scenario according to your theoretical analysis. (**2pts**)
Q4. Write about any of the patterns you noticed in your graph. Explain How its coherent with your analysis and prediction. Its ok if its not in sync with your prediction. Write all your interpretations from the graph. (If you notice something completely against your prediction, still write it down and say that it was surprising) (**3pts**)
Q5. Give a comparative statement about which algorithm is the best / most suitable and in what cases. (**1pt**)

Part 3 - Analysis of Algorithms (24 points)

Q1. Observe the function ‘not_really_a_mystery’ in the file Improve.java
Analyse the run time complexity of this function. (**10pts**)

Q2. Is this the most efficient way to do the task it is accomplishing? (If the task is still a mystery observe the test case of the task to get a clue). (**14 points**)

If Yes, Explain where the inefficiency arises from. Suggest a better way to do that. Prove that your method is better by analysing it’s complexity and comparing it with the earlier analysis.

If No, Argue why the current algorithm cannot be improved.

Part 4 - More practice on theory (4x4 = 16 points)

Analyse the upper bound of running time (Big-O) in terms of n.

Q1.

```
num = 0;
for (i = 1; i <= n; i = i*2)
    Num++;
```

Q2.

```
p = 10;
num = 0;
plimit = 100000;
for (i = p; i <= plimit; i++)
    for (j = 1; j <= i; j++)
        num = num + 1;
```

Q3.

```
for (i = 0; i < n; i++) {
    smallest = i;
    for (j = i+1; j <= n; j++) {
        if (a[j] < a[smallest])
            smallest = j;
        }
    swap(a, i, smallest); // has three instructions
}
```

Q4.

```
    num = 0;
    i = 0;
    while (i < n) {
        j = 0;
        while (j < 100) {
        // constant time operations
            J++;
        }
        I++;
    }
```

Final Submission

The final submission for this homework is on **November 27, 11:59:59 pm**.

A report with the answers to the question is to be submitted for the Final Submission. (Please make the graph with all plots big so it's easy to look at. Also if you are generating it using a computer. Try to add an additional one with a log scale on the Y axis). This as as courtesy for the graders. It will not affect your grade.

Submit via Gradescope.