1 Introduction

In this experiment, you will analyze different algorithms and compare their running times. You are expected to measure running times of the algorithms which are listed below (in the Problem section), and comment about the results.

2 Background

Analysis of algorithms is the area of computer science that provides tools to analyze the efficiency of different methods of solutions. Efficiency of an algorithm is depending on different parameters; how much time, memory space, disk space etc. it requires. Analysis of algorithms is necessary for lots of reasons but mainly used to predict performance and compare algorithms that are developed for the same task. Also it provides guarantees for performance and helps to understand theoretical basis.

A complete analysis of the running time of an algorithm involves the following steps:

- Implement the algorithm completely.
- Determine the time required for each basic operation.
- Identify unknown quantities that can be used to describe the frequency of execution of the basic operations.
- Develop a realistic model for the input to the program.
- Analyze the unknown quantities, assuming the modelled input.
- Calculate the total running time by multiplying the time by the frequency for each operation, then adding all the products.

**On these experiments, you will measure time requirements of the algorithms and compare their time complexities.** A time complexity analysis should focus on gross differences in the efficiency of algorithms that are likely to dominate the overall cost of a solution. You can analyze the example below:

<table>
<thead>
<tr>
<th>Unit Cost</th>
<th>Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>i=1;</td>
<td>c1</td>
</tr>
<tr>
<td>sum = 0;</td>
<td>c2</td>
</tr>
<tr>
<td>while (i ≤ n) {</td>
<td>c3</td>
</tr>
<tr>
<td>j=1;</td>
<td>c4</td>
</tr>
<tr>
<td>while (j ≤ n) {</td>
<td>c5</td>
</tr>
<tr>
<td>sum = sum + i;</td>
<td>c6</td>
</tr>
<tr>
<td>j = j + 1;</td>
<td>c7</td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>i = i + 1;</td>
<td>c8</td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

Total Cost = c1 + c2 + (n+1)*c3 + n*c4 + n*(n+1)*c5+n*n*c6 + n*n*c7 + n*c8. The time required for this algorithm is proportional to $n^2$ which is determined as growth rate and it is usually denoted as $O(n^2)$.

The details of the notations are not given in this assignment paper, you should research and use it for your comments.

### 3 Problem

You are given 5 different algorithms for different purposes and their pseudocodes that are listed below.

1. Matrix multiplication: Pseudocode for matrix multiplication of two square matrices,

   ```plaintext
   for i = 1 to N
     for j = 1 to N
       c(i, j) = 0
       for k = 1 to N
         c(i, j) = c(i, j) + a(i, k) * b(k, j)
       end
     end
   end
   ```

2. Finding maximum element
3. Bubble sort algorithm: An optimized version of bubble sort algorithm’s pseudocode,

```python
func BubbleSort(var a as array)
    for i from 1 to N
        swaps = 0
        for j from 0 to N - i
            if a[j] > a[j + 1]
                swap(a[j], a[j + 1])
                swaps = swaps + 1
            if swaps = 0
                break
    end func

4. Merge sort algorithm

func mergesort(var a as array)
    if ( n == 1 ) return a
    var l1 as array = a[0] ... a[n/2]
    var l2 as array = a[n/2+1] ... a[n]
    l1 = mergesort(l1)
    l2 = mergesort(l2)
    return merge(l1, l2)
end func

func merge(var a as array, var b as array)
    var c as array
    while ( a and b have elements )
        if ( a[0] > b[0] )
            add b[0] to the end of c
            remove b[0] from b
        else
            add a[0] to the end of c
            remove a[0] from a
        while ( a has elements )
            add a[0] to the end of c
            remove a[0] from a
        while ( b has elements )
            add b[0] to the end of c
            remove b[0] from b
    return c
end func
```
5. Binary search algorithm: Pseudocode of iterative binary search algorithm,

```java
func BinarySearch(a, value, left, right)
while left <= right
    mid = floor((right - left)/2) + left
    if a[mid] == value
        return mid
    if value < a[mid]
        right = mid - 1
    else
        left = mid + 1
return not_found
end func
```

You must execute this algorithm on randomly generated integer numbers. For this purpose, you must carry out the following main steps.

- Generate \( n \) random integer numbers. For matrix multiplication, you will generate a \( n \times n \) matrix.
- Run given algorithms. Binary search algorithm must be executed on sorted numbers that are obtained by Quick or Bubble sort algorithms and searched number must be determined randomly.
- Construct a table and save execution time for each algorithm. (See Table 1)
- You must carry out these steps for different \( n \) number.
- Finally, you will plot a graph that will show relation between \( n \) and execution time

<table>
<thead>
<tr>
<th>Algorithms / ( n )</th>
<th>100</th>
<th>300</th>
<th>500</th>
<th>700</th>
<th>1100</th>
<th>1300</th>
<th>1500</th>
<th>1700</th>
<th>1900</th>
<th>2100</th>
<th>2300</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix multiplication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bubble sort</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finding maximum element</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge sort algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary search algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Exemplary Table

**Reports**

1. You must argue the algorithms in terms of time complexity and memory requirements
2. You must show your analysis in detail for each algorithm in your report
3. Your report will include table, graph and your analysis for the experiment.
4. There are no special requirements for the report. If you use a document, code, solution, etc., you have to refer related documents.
Notes

Give necessary details in your report. Save all your work until the assignment is graded. You can ask your questions about the experiment on Piazza.

You will only submit your report file in the given format below

studentid.zip  
   — — <report>
   — — — — report.pdf
   — — — <src>
   — — — — main.java
   — — — — * .java

Your assignment will not be marked if you do not prepare a table and graph.

Late submission will not be accepted!

Policy

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else’s work (from internet), in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.