COMP3506/7505 Homework Task 2

Due Fri 23 Aug 2019, 5:00pm

10 marks total

Overview

The goal of this problem set is to become an expert at writing non-trivial algorithms and analysing the runtime efficiency of these algorithms.

You have been supplied with the file `Algorithms.java`. You are required to implement two methods - *sortQueue* and *FindMissingNumber*. **Your implementations of both these algorithms should be as efficient as possible.**

sortQueue

The *sortQueue* algorithm is described as follows:

Given a queue of elements implementing the Comparable interface, sort the elements of the queue using $O(1)$ extra space.

Dequeing the elements of the sorted queue should yield each element in ascending order. Your solution should take a `java.util.Queue` as input (`https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html`). You are not allowed to convert this queue to another data structure.

Example:

```
Input:  Queue(1, 3, 5, 4, 2)
Output:  Queue(1, 2, 3, 4, 5)

Input:  Queue("a", "b", "c", "b", "a")
Output:  Queue("a", "a", "b", "b", "c")
```

findMissingNumber

The *findMissingNumber* algorithm is described as follows:

Given an array of integers representing an arithmetic sequence, find the missing element.

You may assume that the missing element will not be at the start or end of the sequence, that the input array will contain at least two elements, and that exactly one element will be missing from the array.

Example:

```
Input:  [2, 4, 6, 8, 10, 12, 14, 18, 20]
Output:  16

Input:  [4, 1, -5]
Output:  -2
```

Your Task

1. (3 marks) Write an *iterative* implementation of the *sortQueue* algorithm described above. You may write one or more helper methods as a part of your solution. Your algorithm *must* use $O(1)$ extra space (that is, it must run in-place) and your solution should be as efficient as possible.

2. (2 marks) Represent the worst-case running time of your *sortQueue* algorithm as a mathematical function. Then, use the mathematical definition of big-O to determine an asymptotic bound on this function (that is, you should compute the values of c and n_0 to show that the bound exists). You may assume the runtime complexities of the standard queue methods are as described in lectures.

3. (3 marks) Write a *recursive* implementation of the *findMissingNumber* algorithm described above. You may write one or more helper methods as a part of your solution. Your algorithm should be as efficient as possible.

4. (2 marks) Express the worst-case running time of your *findMissingNumber* algorithm as a mathematical recurrence. State an asymptotic bound in big-O notation for this recurrence. Explain how you determined this bound.

Constraints

- You may only add private helper methods to `Algorithms.java`
- You may not add any member variables to `Algorithms.java`
- You may not use anything from the Java Collections Framework. The only exception to this is using `java.lang.Comparable` and `java.util.Queue` in *sortQueue*.
- Your implementation should only use basic Java programming constructs and not other libraries (apart from the ones described above)

Failure to adhere to these constraints will result in no marks for this exercise.

Submission and Marking

Submit two files as a part of your submission. Your solution to questions 1 and 3 should be in `Algorithms.java`. Your answers to questions 2 and 4 should be in a PDF file named `README.pdf`. Do not submit any other files or directories. To preserve anonymity, please do not include your name in either of these files (it is okay to include your student number).

Questions 1 and 3 will be marked by an automated test suite with timeouts present on each of the tests. A sample test suite has been provided in `AlgorithmsTest.java`. This test suite is not comprehensive - there is no guarantee that passing these will ensure passing the tests used during marking. It is recommended, but not required, that you write your own tests for your algorithms. Your *sortQueue* algorithm must be iterative and use $O(1)$ extra space, and your *findMissingNumber* algorithm must be recursive to be eligible for any marks for these questions. Marks may be deducted for poor coding style and/or inefficient algorithms.

You should submit `README.pdf` through Turnitin - this will be manually marked by a tutor. Any asymptotic bounds should be as tight as possible. Your solution should be electronically processed - hand-written answers will not be accepted. If you are comfortable with the LaTeX typesetting system, it is recommended that you write your answers using it, however it is not a requirement.

Late Submissions and Extensions

Late submissions will *not* be accepted. It is your responsibility to ensure you have submitted your work well in advance of the deadline (taking into account the possibility of computer or internet issues). See the ECP for information about extensions.

Academic Misconduct

Students are reminded of the University's policy on student misconduct, including plagiarism. See the course profile and the School web page:
`http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism`.