City St George's, University of London IN1002 Introduction to Algorithms Coursework Specification

March 2025

Module details	
Module Code	IN1002
Module Title	Introduction to Algorithms

Assessment weighting, type and contact details		
Title	iAnalytics	
Weighting	50 %	
Mode of working for assessment task	Individual - there should be no collusion or collaboration. Your solutions should be your own work, not copied from elsewhere nor created using generative AI.	
Module Leader	Professor Jerry Shen	Contact details: jerry.shen@city.ac.uk

Submission details	
How to submit your work.	Moodle > Coursework submission.
Submission date/s and times	6th April 2025

1. Coursework Objective

The primary objective of this coursework is to assess your proficiency in algorithm development and implementation, both of which are fundamental skills for any computer scientist. As this course progresses into the second semester, we expect you to have already acquired certain necessary skills and knowledge from your first semester. These include:

- Ability to write concise Java programs.
- Proficiency in writing Java programs capable of manipulating integers.

2. Background Introduction

Building upon these foundational skills, consider the scenario where you have been recently employed by *XData.com*, a company that heavily relies on data analytics for its business growth and development strategies. Your role entails the development and implementation of various data analytics algorithms using Java. These algorithms are crucial for the company's success.

In addition to implementing data analytics algorithms, you are tasked with analysing their time complexity to ensure efficient processing of large volumes of data. Understanding the time complexity of these algorithms is essential for optimizing performance and scalability, especially when dealing with extensive datasets common in real-world applications.

In this context, your coursework serves a dual purpose: reinforcing your existing Java programming skills and equipping you with the ability to apply these skills in real-world scenarios within a professional setting. By working on data analytics tasks (integer array), you gain hands-on experience in algorithm design/analysis and solving practical problems commonly encountered.

3. Coursework Task

Please read the instructions carefully. You will lose marks if you do not follow them.

Please download `iAnalytics.zip` from Moodle. The zip file contains several data files for testing and two incomplete Java source files (iAnalytics.java and iTest.java).

The following methods will be implemented by you in the class iAnalytics. For each method, add a comment giving the time complexity on the method in big-O notation. You must not use Java's built-in collections (HashMap, PriorityQueue etc) or built-in algorithms (sort, binary search, etc) in your implementation.

Part A Basic Operations

- Task 1: public int countUnique (int [] arr)

Assume arr is sorted (in non-decreasing order). Return the number of unique values in the array arr. For example, if arr=[1,1,2,4,4,4,5,6,7,7], the answer is 6 (because the unique values are 1, 2, 4, 5, 6 and 7).

- Task 2: public int leastFrequent (int [] arr)

Assume arr is sorted (in non-decreasing order). Return the value that occurs the least frequently in the array. If two or more are equally least frequent, return the smallest of them. For example, if arr=[1,1,2,4,4,4,5,6,7,7], the answer is 2 (values 2, 5 and 6 occur only once and 2 is the smallest among them).

- Task 3: public int countLess (int []arr, int num)

Assume arr is sorted (in non-decreasing order). Return the number of elements of the array that are smaller than `num`. For example, if arr=[1,1,2,4,4,4,5,6,7,7] and num=4, the answer is 3 (the elements [1,1,2]).

- Task 4: public int countBetween (int [] arr, int low, int high)

Assume arr is sorted (in non-decreasing order). Return the number of elements of the array with values between the values low and high (inclusive). For example, if arr=[1,1,2,4,4,4,5,5,6,7,7], low=2 and high=5, the answer is 6 (the elements [2,4,4,4,5,5]).

- Task 5: public int [] topKFrequent (int [] arr, int k)

Assume arr is sorted (in non-decreasing order). Return the k most frequent values in the array arr, in descending order of frequency. If the array has fewer than k distinct values, return them all. If there is a tie for the last position, chose the smallest value. For example, if arr=[1,1,2,4,4,4,4,5,6,7,7,7,8,8,8] and k=2, the answer is [4,7].

Part B Advanced Operations

- Task 6: public int [] longestAscSubarray(int[] arr)

Return the longest contiguous subarray in which the elements are in strictly ascending order (note that you should not assume a sorted array here). If two or more equally longest contiguous subarrays are detected, return the one that comes earliest. For example, if arr=[2,3,3,4,4,8,3,5,7,8,9,2,5,7,9,11], the answer is [3,5,7,8,9].

- Task 7: public int maxSubarraySum(int[] arr, int k)

Find the maximum sum of a contiguous subarray with exactly k elements (again you should not assume a sorted array here). For example, if arr=[-4,-2,8,9,4,-3,6,-10,3] and k=4, the answer is 19 (the sum of the subarray [-2,8,9,4], which has the largest sum). If the array has fewer than k elements, return the sum of the whole array.

4. Marking Scheme

The marks (total 100%) will be divided as follows, with 90% of the marks for implementation (7 tasks), 10% of the marks for algorithm and code quality and creativity.

- **Algorithm Implementation (90%)**

  Each of the methods will be assessed for correctness and efficiency. This criterion assesses the accuracy and integrity of the algorithm implemented in the project. The algorithm should effectively solve the problem statement, producing correct results under various test cases and scenarios. Be sure to consider boundary cases. The marks allocated to each method are:

Task	Function	Mark %
1	countUnique	10%
2	leastFrequent	10%
3	countLess	10%
4	countBetween	10%
5	topKFrequent	10%
6	longestAsSubarray	20%
7	maxSubarrayAtMostK	20%

- **Algorithm and code quality (10%)**

Criteria	Mark%	Remark
Code quality and creativity	10%	• The overall quality of the codebase - its readability, maintainability, and adherence to coding standards and best practices. • Marks will be awarded for well-structured, organized, and properly documented code that is easy to understand and maintain. • Creativity in code design and implementation will also be considered, rewarding innovative approaches, efficient algorithms, and elegant solutions.

5. Testing

Please rigorously test your solutions, ensuring comprehensive coverage of potential boundary cases. The distribution includes four public test files: `tiny.txt`, `small.txt`, `medium.txt`, and `large.txt`. Use these files to verify the accuracy and reliability of your code's outputs.

6. Submission

You are to submit a single file called iAnalytics.java as described above by

6th April 2025 17:00

All submissions will be compiled and run by an automated process, so your class **must** build with the classes I have supplied. Do not submit versions of the other files; your class must compile with my originals. You do not need to submit anything else. Do not submit .class files or your test files.

If circumstances beyond your control prevent you from completing a coursework on time, you must inform us of your extenuating circumstances a soon as possible as detailed on the Student Hub. (see https://studenthub.citystgeorges.ac.uk/academic-resources/learning-and-study-support/extenuating-circumstances).

If you encounter issues with Moodle at the deadline, you should email your submission to your Course Officer (ug.cs@city.ac.uk) and the relevant Module Leader – Professor Jerry Shen (email: jerry.shen@city.ac.uk) If you encounter issues with Moodle prior to the deadline, you must raise a ticket with IT Services (https://cityuni.service-now.com/sp?id=index).

7. Reference

Anany V. Levitkin, *Introduction to the Design & Analysis of Algorithms* (3rd edition), Pearson, 2011.