EST 1892 LSBU

Coursework 2 Specification

CW2_Specification_CSI-4-DSA_21-22

Read this coursework specification carefully, it tells you how you are going to be assessed, how to submit your coursework on-time and how (and when) you'll receive your marks and feedback.

Module Code	CSI-4-DSA
Module Title	Data Structures and Algorithms
Lecturer	Mike Child
% of Module Mark	50%
Distributed	
Submission Method	Submit online via this Module's Moodle site
Submission Deadline	Friday 13/05/22 at 16:00
Release of Feedback & Marks	Feedback and provisional marks will be available in the Gradebook on Moodle from [30/06/22]

Coursework Aim:

Complete a programming task involving the use of the several data structures that have been explored in this module (see *Assignment Task*, below).

Coursework Details:

Type:	Report
Word Count:	No formal word count is required but for general guidance no more than 2000 words would be expected. The word count will always be ambiguous because the requirement for embedded code will skew it. There are no specific penalties for exceeding or failing to reach the word count guidance, but your submission must properly address the task in hand.
Presentation:	▪ Work must be submitted as a word processor document (odt/doc/docx) or a PDF ▪ Your student number must appear at the front of the coursework. Your name must **not** be on your coursework.
Referencing:	While there is no specific requirement for references in this assignment, Harvard Referencing should be used for any you supply, see your Library Subject Guide for guides and tips on referencing.
Regulations:	Make sure you understand the University Regulations on expected academic practice and academic misconduct. Note in particular: ▪ Your work must be your own. Markers will be attentive to both the plausibility of the sources provided as well as the consistency and approach to writing of the work. Simply, if you do the research and reading, and then write it up on your own, giving the reference to sources, you will approach the work in the appropriate way and will cause not give markers reason to question the authenticity of the work. ▪ All quotations must be credited and properly referenced. Paraphrasing is still regarded as plagiarism if you fail to acknowledge the source for the ideas being expressed. **TURNITIN:** When you upload your work to the Moodle site it will be checked by anti-plagiarism software.

Learning Outcomes

This coursework will fully or partially assess the following learning outcomes for this module.

- Describe different algorithms and the means used to measure their performance.
- Analyse programming problems and identify appropriate algorithmic solutions.
- Develop software to solve relatively complex problems and assess alternative solutions.
- Apply critical and analytic reasoning to tasks.

Assignment Task

You are provided with a Java application that loads each image file in a given directory and performs some statistical analysis of the colours used in that image. The provided version is incomplete and you need to add some code to make it work.

Provided files:

StudentWork.java - this is the main program that you can run. You should not modify the contents of the main method in this class (except as mentioned below, to temporarily turn off some of the functionality), but the other methods in this class are where you need to make your changes.

CColor.java, **LoadedImage.java** and **Util.java** - these classes do not need to be changed to complete the assignment. The first is a class that represents an RGB defined colour, the second is a class that holds a loaded image and provides access to the colours of its pixels and the third is a class that contains a few utility methods that are used by other parts of the program.

charts.xlsx - this is an excel spreadsheet with a space to paste the output from the program into that is linked to two charts that will display the results you obtained. You can use this to help you in your discussion of the performance characteristics of the program, to contrast theory and the observed behaviour. The charts in this spreadsheet are preconfigured with suitable trendlines and do not need any adjustments to use.

You should create a new IDE project for this assignment and create a Java package called ***dsa22.first*** (that is, a package called ***first*** inside a package called ***dsa22***) and add all the provided classes to it. Alternatively, add the provided classes to a package of your choice and correct the package declaration line at the top of each file to match the name of your chosen package.

Task 1

You must add code to the *StudentWork* class in this program so that it creates a collection of all the unique colours used in the image. To do this you must iterate over all the pixels in the image and get its colour as a *Color* object by calling *image.getColor(x, y)*. Then make a collection that contains each unique *Color* object only once.

Please note that you are given a working implementation that does this inefficiently using a list and you can use this code as the basis of your more efficient versions.

The program is designed to do this in three different ways, in three different methods provided for the purpose. The first of these methods is complete and working and provided for you as an example. The other two methods you must complete yourself, but they will be quite similar to the first in many regards.

1. The method *distinctColoursUsingList*(LoadedImage image) uses an ArrayList that is initialised to contain enough space so that it does not need resizing (the image width times height) and then checks whether it contains each Color object already, and if not adds it.

2. In the method *distinctColoursUsingTreeSet*(LoadedImage image) you must use a TreeSet that will automatically ignore attempts to add duplicate objects, so you can just add all the Color objects to it one after another.

3. In the method *distinctColoursUsingHashSet*(LoadedImage image) you must use a HashSet that will also automatically ignore attempts to add duplicate objects, so you can just add all the Color objects to it one after another.

Important: The Color object returned by *image.getColor(x, y)* is safe to use as a value in both a HashSet and TreeSet, and safe to use as a key in both a HashMap and TreeMap.

The existing main method reads a list of the image files in the directory **dsacw_images** and for each image, calls each of these methods in turn. For each call it uses *System.currentTimeMillis()* to record the time before and after calling the method so that it can then display the length of time taken as well as the number of unique colours that has been found. The images are processed in increasing order of size (number of pixels).

All three methods should find the same number of colours for any given image - if not, they are not working properly.

You can run the program as it stands, but it will only use the *distinctColoursUsingList* method to find the number of unique colours. This method is ***very*** slow and the program will take a considerable time to run - expect ten minutes or more.

Once the program completes, it will print out a little summary of the times taken for each method.

Completing the *distinctColoursUsingTreeSet* and *distinctColoursUsingHashSet* methods does not require a lot of complicated code. They will be very similar to the existing *distinctColoursUsingList* and even more similar to each other. They will also run much, much faster than the existing method.

While developing your code for these methods, you may want to change the boolean variable found in the *main* method called *suppressListCalculation* to have the value **true**. This causes the program to skip doing the slow list calculation and will help you to test your new code without waiting for that to be done. Other than this, you should not change any of the code in the main method.

You need to explain how your code works and discuss the different times that your program reports; explaining the reasons behind the differences in timing in terms of the concepts we have studied in this module. You can copy and paste the summary outputs from your program into the green shaded region of the ***charts.xlsx*** spreadsheet to see the results plotted in the graphs there; this may help in your discussion of the performance of the different methods. Do note, however, that many factors such as runtime compiler optimisations, competition for CPU time with other processes, operating system memory

management and the number of unique colours actually present in the different images will obscure the theoretical performance of the data structures involved.

Task 2

You need to provide an implementation of the method *frequencyCountUsingMap*(LoadedImage image) so that it returns a map which has the distinct colours in the image as the keys and the number of pixels which are that colour as the values. The *main* method also displays the time this method takes and you should experiment with implementations based on both TreeMap and HashMap and discuss the code for each approach and the difference of the results. Once this method has been implemented, for the first image processed (only) the program displays the frequency map graphically in a bar chart showing each colour and how much it is used in the image. (Note that there are a lot of colours of very similar shades and the bar chart is very wide so does not give a very useful overview in practice).

Assignment Report

You must write a report in which you present all the code you have written (but not the code that is already given to you), properly formatted in a fixed width font and clearly readable. You should also include the summary output data your program produces, and if you have used this in the provided ***charts.xlsx*** spreadsheet a screenshot of that spreadsheet and its graphs might be useful to help you in your discussion of the performance of your code.

For each of the methods involved in each task (that is, *distinctColoursUsingList*, *distinctColoursUsingTreeSet*, *distinctColoursUsingHashSet* and *frequencyCountUsingMap*) you should clearly explain the code and how it works, making sure you explain what the data structure behind the code is based upon (binary search tree, hash table, array list etc.) and the corresponding big O performance characteristics you expect it to exhibit. You should then discuss the timings output by the program and to what extent these correspond to what theory predicts.

You must also quote any reference materials (including course materials) that you have used to help you complete the assignment.

Assessment Criteria and Weighting

LSBU marking criteria have been developed to help tutors give you clear and helpful feedback on your work. They will be applied to your work to help you understand what you have accomplished, how any mark given was arrived at, and how you can improve your work in future.

For this assignment the following criteria will be applied (also see rubric following).

Marking Criteria

This assignment will be marked using an adaptation of the University's standardised marking criteria. It is important that you pay attention to the criteria that will be applied and address them in the text of your report. A detailed rubric is shown on the next page, but the main criteria are as follows:

1. **Subject Knowledge (35%)**

   Understanding and application of subject knowledge. Contribution to subject debate.
   Assessed implicitly by your written explanation of the code, and explicitly by your ability to discuss it in terms of the theoretical content of the module and demonstrate understanding of data structures and the significance of the algorithms associated with them.
2. **Critical Analysis (15%)**

   Assessed mainly by your analysis of the output of the program and its relation to theory, but also by the rationale you give for your design approaches and their contrast to alternative approaches in the coding.
3. **Testing and Problem-Solving Skills (30%)**

   Design, implementation, testing and analysis of product / process / system / idea / solution(s) to practical or theoretical questions or problems.
   Assessed on the basis of the level of achievement of the coding you have developed and documented your understanding of, bearing in mind that code that you do not discuss in your narrative will be given very little credit (as the implication is that you do not understand what it is and what it does if you did not discuss it). However, your ability to explain the problems involved and the solutions they demanded, will also be considered here, whether or not you were able to solve them.
4. **Practical Competence (10%)**

   Skills to apply theory to practice or to test theory.
   Assessed on documented evidence of your use of technical documentation (for example tutorials and reference documentation and course materials) in working out how to accomplish the assignment.
5. **Personal and Professional Development (10%)**

   Management of learning through self-direction, planning and reflection
   Assessed on the basis of the quality of your submitted report, including clarity of writing, presentation, and properly addressing the assignment specification.

Please note the criteria weightings and general interpretation shown in bold capitals under each criteria.

Criteria	Outstanding 100-80%	Excellent 79-70%	Very good 69-60%	Good 59-50%	Satisfactory 49-40%	Inadequate 39-30%	Very poor 29-0%
Subject Knowledge Understanding and application of subject knowledge. Contribution to subject debate. **CODE EXPLANATION 35%**	Shows sustained breadth, accuracy and detail in understanding key aspects of subject. Contributes to subject debate. Awareness of ambiguities and limitations of knowledge.	Shows breadth, accuracy and detail in understanding key aspects of subject. Contributes to subject debate. Some awareness of ambiguities and limitations of knowledge.	Accurate and extensive understanding of key aspects of subject. Evidence of coherent knowledge.	Accurate understanding of key aspects of subject. Evidence of coherent knowledge.	Understanding of **key aspects** of subject. Some evidence of coherent knowledge.	Some evidence of superficial understanding of subject. Inaccuracies.	Little or no evidence of understanding of subject. Inaccuracies.
Critical Analysis Analysis and interpretation of sources, literature and/or results. Structuring of issues/debates. **ANALYSIS 15%**	Outstanding demonstration of critical analysis of the possible design strategies that could be used to meet the software requirements, and evaluation of the approaches chosen.	Excellent demonstration of critical analysis of the possible design strategies that could be used to meet the software requirements, and evaluation of the approaches chosen.	Very good demonstration of critical analysis of the possible design strategies that could be used to meet the software requirements, and evaluation of the approaches chosen.	Good demonstration of critical analysis of the possible design strategies that could be used to meet the software requirements, and evaluation of the approaches chosen.	Demonstration of critical analysis of the **key** possible design strategies that could be used to meet the software requirements, and evaluation of the approaches chosen.	Trivial demonstration of critical analysis of the possible design strategies that could be used to meet the software requirements, and evaluation of the approaches chosen.	Little or no critical analysis has been demonstrated.
Testing and Problem-Solving Skills Design, implementation, testing and analysis of **product/process/system/idea/solution(s)** to practical or theoretical questions or problems **IMPLEMENTATION AND DISCUSSION 30%**	Outstanding implementation of all required software, with near perfectly organised, formatted and documented source code, and documented demonstration of runtime behaviour, and outstanding comprehension evidenced.	Excellent implementation of all required software, with well organised, formatted and documented source code provided, and excellent comprehension evidenced.	Competent implementation of all required software, with well organised, formatted and documented source code, and documented demonstration of runtime behaviour, and very good comprehension evidenced.	Implementation of all required software, with well organised, formatted and documented source code, and documented demonstration of runtime behaviour, with some missing/incorrect functionality or poor quality. Good evidence of comprehension.	Implementation of most of the required software, with well organised, formatted and documented source code, and documented demonstration of runtime behaviour, with some missing/incorrect functionality or poor quality. Some evidence of comprehension.	Implementation of only part of the required software, with well organised, formatted and documented source code, and documented demonstration of runtime behaviour, with some missing/incorrect functionality or poor quality. Little evidence of comprehension.	Little or no functionality has been implemented.
Practical Competence Skills to apply theory to practice or to test theory **USE OF REFERENCE MATERIAL 10%**	Outstanding descriptions of factual information, programming techniques or theoretical explanations being found in technical or theoretical reference material.	Excellent explicit descriptions of all factual information, programming techniques or theoretical explanations that were found in technical or theoretical reference material.	Good explicit descriptions of all factual information, programming techniques or theoretical explanations that were found in technical or theoretical reference material.	Reasonable descriptions of most factual information, programming techniques or theoretical explanations that were found in technical or theoretical reference material.	Basic examples of the **main** factual information, programming techniques or theoretical explanations that were found in technical or theoretical reference material.	Some trivial examples of factual information, programming techniques or theoretical explanations being found in technical or theoretical reference material.	Little or no evidence of factual information, programming techniques or theoretical explanations being found in technical or theoretical reference material.
Personal and Professional Development Management of learning through self-direction, planning and reflection **REPORT QUALITY 10%**	Outstanding report organisation, structure, presentation, narrative voice and language.	Excellent report organisation, structure, presentation, narrative voice and language.	Very good report organisation, structure, presentation, narrative voice and language.	Good report organisation, structure, presentation, narrative voice and language.	Satisfactory report organisation, structure, presentation, narrative voice and language.	Poor report organisation, structure, presentation, narrative voice and language.	Report does not constitute a serious attempt at the assignment.

How to get help

We will discuss this Coursework Specification in class. However, if you have related questions, please contact Mike Child, childm@lsbu.ac.uk as soon as possible.

Resources

The main resources are the tutorial documents that you used with the Raspberry Pi activities and the activities themselves.

Quality assurance of coursework specifications

Coursework specifications within CSI division go through internal (for new modules with 100% coursework also through external) moderation. This is to ensure high quality, consistency and appropriateness of the coursework as well as to share best practice within the CSI division.

Details of the moderators for this coursework specification are below:

Moderated (internal)	Ioannis Iatropolis, Aarbaz Alam, Panagiotis Alefragkis
Moderated (CSI lead)	[Name, date]
Signed off by (HoD/DHoD)	[Name, date]

----------For Internal use by CSI lead only----------

Changes required to CW?	Yes, No *
Examples of good practice	

*** if changes are required, moderator to complete the below:**

List of changes required	[These needs to be met before signoff can be achieved]
ML Response	[ML response, date]
Moderator Response	[ML response, date]