|  |  |
| --- | --- |
| **1. Module number** | *SET09117* |
| **2. Module title** | *Algorithms and Data Structures* |
| **3. Module leader** | *Andrew Cumming* |
| **4.** **Tutor with responsibility for this Assessment**  Student’s first point of contact | *Neil Urquhart* |
| **5. Assessment** | *Vehicle Routing: Code and report* |
| **6. Weighting** | *40% of module assessment.* |
| **7. Size and/or time limits for assessment** | *Submitted code should be succinct and well commented. The indentation must be correct.* |
| **8. Deadline of submission**  Your attention is drawn to the penalties for late submission | *17th Nov 2014* |
| **9. Arrangements for submission** | *Hand in via moodle* |
| **10. Assessment Regulations**  All assessments are subject to the University Regulations**.** |  |
| **11. The requirements for the assessment** | *Complete the tasks given. Submit a report of your implementation. The report should include your source code.* |
| **12. Special instructions** | *The report must be submitted using the moodle page* |
| **13. Return of work** | *Your feedback and grade will be returned via moodle.* |
| **14. Assessment criteria** | *You will be assessed on*   * *your use of appropriate algorithms and data structures* * *your ability to conduct an experiment and present the results* |

# SET09117 Algorithms and Data Structures Coursework 2014

## Introduction

You are required to implement the Clarke-Wright vehicle routing algorithm.

You must demonstrate that your solution is correct.

You must measure the performance of your algorithm against data sets of various sizes.

## Provided Files

*Problem instances and solutions* –

contained in SET09117 CW 2014 Test data.zip:

|  |  |
| --- | --- |
| \*prob.csv | problem instances  First line gives cords of depot & max capacity of a truck  Subequent triples are coords of and the requirement |
| \*cwsn.csv | Solution as generated by our Clarke-Wright algorithm  Each line has a list of triples indicating a route |
| \*dmsn.csv | Poor but fast solution to the problem. These files are provided to give examples of solutions for you benchmark your work against. |
| \*.svg | A visualisation of a problem or solution |

*Java Source code*

|  |  |
| --- | --- |
| Customer.java | A class representing a customer (also misused for the depot) |
| Experiment.java | This runs the Dumb solution on a range of problem instances. To obtain timings. Each is run 50 times, Output is to the console – copy and paste the output into Excel for analysis. |
| MkProblem.java | Generates random problem instances. |
| VRProblem.java | Represents a single vehicle routing problem |
| VRSolution.java | Represents a single solution (or partial solution) to the VRP |
| VRTest.java | Examines a number of problem/solution pairs and checks they are valid solutions and gives the quality. |

**Hints to get you started.**

1. Unzip SET09117 CW 2014 Test data.zip, this will provide a large sample of problem instances and solutions as outlined above
2. Compile and run Experiment.java This will run the provided solver in VRSolution.oneRoutePerCustomerSolution() on a range of problem instances. Examine the source code and note how it creates a new instance of VRProblem for each instance to be solved, then creates an instance of VRSolution to solve each instance. Note that a solution file \*dmsn.csv is created for each instance solved.
3. You need to be able to check your solutions are valid (e.g. your vehicles are not over loaded, or customers missed out!). VRTests.Java allows you to examine a selection of problems and solutions and check their validity and determine their cost. The example code checks a range of problems, compile and run it. Examine the code, and note that for demonstration purposes two of the solutions examined are invalid. Note that this code demonstrates how to verify a solution, and to cost it (cost= total distance travelled), also shown is how to load an existing solution (\*dmsn.csv or \*cwsn.csv) and to generate .SVG files representing the solution
4. To get started, implement a new solver within the VRSolution class, e.g. (VRSolution.mySolver() ) and use it in VRTests.Java in place of VRSolution.oneRoutePerCustomerSolution(). Make sure that you have chosen a range of problem instances to work with and specified them within VRTests.Java . When you write your solution files, rename them along the lines of \*mySol.CSV to differentiate them from the supplied solutions.
5. To time your algorithm modify Experiment.Java .
6. If you implement a more complex algorithm, create additional classes as required, just make sure that they are called from VRSolution

## Report

Your report should have the following headings:

* **Introduction**
  + Problem summary including any limitations of your solution.
* **Method**
  + Description of the experiment that you conducted.
* **Results**
  + Tables and charts that show the performance of your solution (how long does it take to run your algorithm).
  + Demonstration that your solution is valid.
  + Demonstration of the quality of your solution (what is the cost).
* **Conclusions**
  + Summary of your results.
  + Reflection on your performance on this assessment.
* **Appendix**
  + Source code of your solution

Notes on the report:

* The reports must be written in third person.
* The result should presented as a table of figures *and* as a chart. Only averaged results should be shown. Charts must have a title and axis labels for x and y.
* A reasonable number of decimal places should be presented.

SET09117 Coursework assessment sheet.

|  |  |  |
| --- | --- | --- |
| **Element** | **Comments** | **Mark** |
| Introduction | Report includes:   * A description of the algorithm used with references * An analysis of the expected performance of the algorithm | /20 |
| Experimental method | Report includes:   * The methodology used * Steps taken to ensure the accuracy and repeatability of the results | /10 |
| Experimental result | Report includes:   * Tables showing the timings for a variety of data sets * Charts that illustrate the run time against data set size | /20 |
| Conclusions and reflections | Report includes:   * An explanation for the results obtained * Reflections on how reliable the results are * Reflections on the value of the new implementation | /20 |
| Source code  (Appendix) | Implementation includes:   * Use of appropriate data structures * Useful comments | /30 |

NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATRIC: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TOTAL: \_\_\_\_\_\_\_\_\_\_\_