RMIT UNIVERSITY

Algorithms and Analysis
COSC 1285
Assignment 2: Hangman

Assessment Type	Group assignment. Submit online via Canvas → Assignments → Assignment 2. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/assignment FAQ/relevant discussion forums.
Due Date	Week 12, Friday 16th October 2020, 11:59pm
Marks	30

Assignment Teams

Please read all the following information before attempting your assignment. This is a *group* assignment. You may not collude with any other people outside of your group, and plagiarise their work. Each group is expected to present the results of his/her/their own thinking and writing. Never copy other student's work (even if they "explain it to you first") and never give your written work to others. Keep any conversation high-level and never show your solution to others. Never copy from the Web or any other resource. Remember you are meant to generate the solution to the questions by yourself. Suspected collusion or plagiarism will be dealt with according to RMIT policy.

In the submission (your PDF file for the report of tasks C and D) you will be required to certify that the submitted solution *represents your own work only* by agreeing to the following statement:

> *I (or We) certify that this is all my (our) own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in my (our) submission. I (We) will show we I (we) agree to this honor code by typing "Yes":*

A sample format for this requirement is provided, and please find it in `Canvas` → `Assignments` → `Assignment2`.

Introduction

Hangman is a popular children 2-player game, where one player selects a word, phrase or sentence and the other player guesses this by making a number of single letter guesses. Each time the guessing player makes an incorrect guess (either letter or the whole word/phrase/sentence), a part is added to a diagram (usually a hangman). The game ends when either the guessing player guesses all the letters of the word correctly, guesses the word/phrase/sentence correctly, or the diagram is complete. In this assignment you will implement algorithms for the guessing player, that can solve hangman games and its variants.

1 Learning Outcomes

This assessment relates to 3 learning outcomes of the course which are:

- CLO 1: Compare, contrast, and apply the key algorithmic design paradigms: brute force, divide and conquer, decrease and conquer, transform and conquer, greedy, dynamic programming and iterative improvement;
- CLO 3: Define, compare, analyse, and solve general algorithmic problem types: sorting, searching, graphs and geometric; and
- CLO 5: Implement, empirically compare, and apply fundamental algorithms and data structures to real-world problems.

Background

Hangman

In Hangman, the guessing player is given a word, phrase or sentence to guess. The length of the word, phrase and sentence are given, and the guessing player makes a number of single letter guesses to guess that unknown word, phrase and sentence. If a guess is correct, the guessed letter in the unknown word will be revealed. If the guess is incorrect, a piece is added to a drawing (typically a hangman), and if the hangman drawing is complete, the guessing player loses. If all the letters of the word etc are correctly guessed and revealed, then the guessing player wins. For more details about Hangman, please see `https://en.wikipedia.org/wiki/Hangman_(game)`.

Example app to see Hangman in action: `https://thewordsearch.com/hangman`

Hangman Strategies

In this assignment, we will implement a few different known strategies for the guessing player and to explore and design algorithms for two variants of Hangman.

Random Guessing Strategy

This is the most basic strategy, but a good way to start the consideration of designing a good Hangman guessing algorithm. In this strategy, the algorithm makes random guesses of letters that it hasn't guessed before. Consider it as sampling without replacement type of guessing.

Dictionary Aware Statistics Strategy

This is a more advanced strategy. When we know the dictionary of words that Hangman is generated from, we can compute statistics per letter of who many words they appear in. We then guess according to the letter that appears in most words in the dictionary, as that, if words are uniformly sampled from the dictionary, are most likely to result in a correct letter guess.

Subsequently, the set of possible words are narrowed down according to each correct guess. From the remaining possible words, we compute the frequency statistics again and select the letter that is more frequent among the remaining possible words.

Hangman Variants

Two word Hangman

In this variant, instead of guessing one word, we guess two words, both from a known dictionary.

Wheel of Fortune

In the Wheel of Fortune variant (yes, same as the popular TV show `https://en.wikipedia.org/wiki/Wheel_of_Fortune_(Australian_game_show)`), instead of guessing a word, we guess a phrase or sentence. The previous dictionary aware approach may not work in this case - we want you to think about why that might be so, and devise an approach that can solve this Wheel of Fortune variant faster than the previously mentioned strategies.

This concludes the background. In the following, we will describe the tasks of this assignment.

2 Tasks

The assignment is broken up into a number of tasks. Apart from Task A that should be completed initially, all other tasks can be completed in an order you are more comfortable with. Task D is considered a high distinction task and hence we suggest to tackle this after you have completed the other tasks.

Task A: Implement Random guessing (5 marks)

To help to understand the problem and the challenges involved, the first task is to develop a random guessing approach to Hangman game.

Task B: Implement Dictionary-Aware Guessing (8 marks)

In this task, you'll implement the dictionary-aware guessing strategy.

Task C: Design and Implement Dictionary-Aware Guessing Strategy for Two Word Hangman (5 marks)

In this task, you'll design and implement a guessing strategy/algorithm for the two word Hangman variant.

Task D: Design and Implement an Algorithm for Wheel of Fortune Variant (6 marks)

In this task, you will implement an algorithm for the Wheel of Fortune Variant. We want you to explore an algorithm for this.

3 Details for all tasks

To help you get started and to provide a framework for testing, you are provided with skeleton code that implements some of the mechanics of the Hangman program. The main class (RmitHangman.java) implements the main IO components the assignment, parsing parameters and read in the details of the game. The list of main java files provided are listed in Table 1.

file	description
`RmitHangman.java`	Class implementing basic IO and processing code. *Do not modify, as we will use this to do testing.*
`solver/HangmanSolver.java`	Abstract class for Hangman guessing algorithms. *Can add to, but don't modify existing methods.*
`solver/RandomGuessSolver.java`	Class implementing the random guessing strategy (task A). *Complete the implementation, particularly for IMPLEMENT ME! parts*
`solver/DictAwareSolver.java`	Class implementing the dictionary-ware guessing strategy (task B). *Complete the implementation, particularly for IMPLEMENT ME! parts*
`solver/TwoWordHangmanGuessSolver.java`	Class implementing the two word hangman variant guessing strategy (task C). *Complete the implementation, particularly for IMPLEMENT ME! parts*
`solver/WheelOfFortuneGuessSolver.java`	Class implementing the Wheel of Fortune variant guessing strategy (task D). *Complete the implementation, particularly for IMPLEMENT ME! parts*
`game/HangmanGame.java`	Class implementing the mechanicisms to run a hangman game. *Do not modify, as we will use this to do testing.*
`trace/HangmanGameTracer.java`	Class to trace/log games. *Do not modify, as we will use this to do testing.*

Table 1: Table of supplied Java files.

We also strongly suggest to avoid modifying the "Do not modify" ones, as they form the interface between the main class and algorithms. You may add java files and methods, but it should be within the structure of the skeleton code, i.e., keep the same directory structure and you must ensure they compile. Ensure your code and any modifications you made to the structure compiles and runs on the **core teaching servers**. Note that the onus is on you to ensure correct compilation and behaviour on the core teaching servers

before submission.

As a friendly reminder, remember how packages work and IDE like Eclipse will automatically add the package qualifiers to files created in their environments. This is a large source of compile errors, so remove these package qualifiers when testing on the core teaching servers and before submission.

Compiling and Executing

To compile the files, run the following command from the root directory (the directory that RmitHangman.java is in):

```
javac *.java
```

Note we expect to be able to compile your implementation using the same command on the core teaching server.
To run the framework:

```
java RmitHangman <hangman solver type> <dictionary filename> <maximum
incorrect number> <word(s) to guess> [(optional) game trace fileName]
```

where

- hangman solver type: type of Hangman solver, one of [random, dict, twoword, wheel]
- dictionary filename: file of dictionary to use.
- maximum incorrect number: maximum number of incorrect guesses before game is over.
- word/phrase to guess: word or phrase (in double quotes) to guess.
- game trace filename: Output file name of the trace of guessing.

The dictionary file should have one word per line (we provide a sample dictionary, ausDict.txt, to get you started).

The word(s) to guess should be inputed in double quotes, and multiple words separated by a space. For example, the following are valid command inputs:

```
java RmitHangman random ausDict.txt 7 "hello"
```

```
java RmitHangman twoword ausDict.txt 9 "hello bye" traceOutput.txt
```

3.1 Clarification to Specifications

Please periodically check the assignment FAQ for further clarifications about specifications. In addition, the lecturer will go through different aspects of the assignment each week (typically via video recordings), so make sure to check the course material page on Canvas to see if there are additional notes posted.

4 Assessment

The assignment will be marked out of 30.

The assessment in this assignment will be broken down into a number of components. The following criteria will be considered when allocating marks. All evaluation will be done on the core teaching servers.

Task A (**5/30**):

For this task, we will evaluate your implementation and algorithm on whether:

1. Implementation and Approach: It implements the random strategy to guess Hangman games.

2. Correctness: Over a large number of games, whether it has the same performance (win/loss ratios, number of guesses for wins) as the standard implementation of random guessing strategy.

Task B (**8/30**):

For this task, we will evaluate your implementation and algorithm on whether:

1. Implementation and Approach: It implements the dictionary-aware strategy to guess Hangman games.

2. Correctness: Over a large number of games, whether it has the same performance (win/loss ratios, number of guesses for wins) as the standard implementation of dictionary aware guessing strategy. Note this is a deterministic algorithm, so performance should be the similar.

3. Efficiency: As part of correctness, your implementation should not take excessively long to solve a puzzle. We will benchmark the running time against our non-optimised solution and add a substantial margin on top, and solutions taking longer than this will be timed out.

Task C (**5/30**):

For this task, we will evaluate your implementation and algorithm on whether:

1. Implementation and Approach: It takes a reasonable approach and can solve two word Hangman games.

2. Correctness: Over a large number of games, whether statistically it has at least an equal performance, in terms of win/loss ratios and number of guesses for wins, as the standard implementations of two-word dictionary-aware guessing strategy.

3. Efficiency: As part of correctness, your implementation should not take excessively long to solve a puzzle. We will benchmark the running time against our non-optimised solution and add a substantial margin on top, and solutions taking longer than this will be timed out.

4. Description of Approach: In addition to the code, you will be assessed on a description of your approach, which will help us to understand your approach (this will help us with assessing the Implementation and Approach of all tasks). Include

how you approached solving two word Hangman and your rationale behind it. You may cite other references you have researched upon and utilised the information within. This should be no longer than 1 page and submitted as a pdf as part of your submission. You will be assessed on the approach, its clarity and whether it reflects your code.

Task D (**6/30**):

For this task, we will evaluate your implementation and algorithm on whether:

1. Implementation and Approach: It takes a reasonable approach and can Wheel of Fortune Hangman games.

2. Performance: Over a large number of games, whether statistically it has equal or greater performance, in terms of win/loss ratios and number of guesses for wins, as our basic strategy for solving Wheel of Fortune games.

3. Efficiency: As part of correctness, your implementation should not take excessively long to solve a puzzle. We will benchmark the running time against our non-optimised solution and add a substantial margin on top, and solutions taking longer than this will be timed out.

4. Description of Approach: In addition to the code, you will be assessed on a description of your approach, which will help us to understand your approach (this will help us with assessing the Implementation and Approach of all tasks). Include how you approached solving uncertain dictionary Hangman and your rationale behind it. You may cite other references you have researched upon and utilised the information within. This should be no longer than 3 pages and submitted as a pdf as part of your submission. You will be assessed on the approach, its clarity and whether it reflects your code.

Coding style and Commenting (**3/30**):

You will be evaluated on your level of commenting, readability and modularity. This should be at least at the level expected of a first year masters student who has done some programming courses.

Online Interview (**3/30**):

You will be asked a number of questions, of which you will then upload a short recording of your response to them. The questions will be about the tasks in the assignment.

Note if there is either little comment or there are issues raised from the online interview, we may ask you to clarify your submission to us, so please take these two parts seriously.

4.1 Late Submissions

Late submissions will incur a 10% penalty on the total marks of the corresponding assessment task per day or part of day late, i.e, 3 marks per day. Submissions that are late by 5 days or more are not accepted and will be awarded zero, unless special consideration has been granted. Granted Special Considerations with new due date set after the results have been released (typically 2 weeks after the deadline) will automatically result in an

equivalent assessment in the form of a practical test, assessing the same knowledge and skills of the assignment (location and time to be arranged by the coordinator). Please ensure your submission is correct (all files are there, compiles etc), re-submissions after the due date and time will be considered as late submissions. The core teaching servers and Canvas can be slow, so please do double check ensure you have your assignments done and submitted a little before the submission deadline to avoid submitting late.

Assessment declaration: By submitting this assessment, all the members of the group agree to the assessment declaration - `https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/assessment-declaration`

5 Team Structure

This assignment is designed to be done in *pairs* (group of two). If you have difficulty in finding a partner, post on the discussion forum or contact your lecturer. If there are issues with work division and workload in your group, please contact your lecturer as soon as possible.

In addition, please submit what percentage each partner made to the assignment (a contribution sheet will be made available for you to fill in), and submit this sheet in your submission. The contributions of your group should add up to 100%. If the contribution percentages are not 50-50, the partner with less than 50% will have their marks reduced. Let student A has contribution X%, and student B has contribution Y%, and $X > Y$. The group is given a group mark of M. Student A will get M for assignment 1, but student B will get $\frac{M}{\frac{X}{Y}}$.

6 Submission

The final submission will consist of:

- The Java source code of your implementations, including the ones we provided. Keep the same folder structure as provided in skeleton (otherwise the packages won't work). Maintaining the folder structure, ensure all the java source files are within the folder tree structure. Rename the root folder as `Assign2-<partner 1 student number>-<partner 2 student number>`. Specifically, if your student numbers are s12345 and s67890, then all the source code files should be within the root folder Assign2-s12345-s67890 and its children folders.

- All folder (and files within) should be zipped up and named as `Assign2-<partner 1 student number>-<partner 2 student number>.zip`. E.g., if your student numbers are s12345 and s67890, then your submission file should be called `Assign2-s12345-s67890.zip`, and when we unzip that zip file, then all the submission files should be in the folder Assign2-s12345-s67890. You do not need to zip the code within your submission zip file.

- Your report of both tasks C and D approaches, called "assign2Report.pdf". Place this pdf within the Java source file root directory/folder, e.g., Assign2-s12345-s67890. Maximum of 1 page for task C, and maximum of 3 pages for task D.

- Your group's **contribution sheet**. See the previous 'Team Structure' section for more details. This sheet should also be placed within your root source file folder.

Note: submission of the zip file will be done via Canvas.

7 Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites. If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to the following: `https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity`.

8 Getting Help

There are multiple venues to get help. There are weekly consultation hours (see Canvas for time and location details). In addition, you are encouraged to discuss any issues you have with your Tutor or Lab Demonstrator. We will also be posting common questions on the assignment 2 Q&A section on Canvas and we encourage you to check and participate in the discussion forum on Canvas. However, please **refrain from posting solutions**, particularly as this assignment is focused on algorithmic and data structure design.