FIT3036 COMPUTER SCIENCE PROJECT

Computers Doing IQ Tests, and Pick the ‘Odd One Out’

Project Specification Proposal

Tutorial: Assoc. Prof. David L. Dowe

By: Kelvin Benzali

26445468

Table of Contents

1 Introduction

Artificial intelligence is a set of complex techniques and algorithm to create an intelligence to a machines. Artificial intelligence has been on a subject for over 50 years ago in the computer world. Despite the subject has been researched for a long time, artificial intelligence still has not yet recognized as a real intelligence. By the real intelligence means is to be able to rival the human intelligence. There were some tests to check whether the artificial intelligence level is enough to rival the human intelligence such as Turing Test in 1950. The test consist of two participants which are a computer as artificial intelligence and a human. The test purposes is to fool the judges when they are interact with each other or interrogated by the judges. The test is proven that the artificial intelligence has achieved human intelligence by fooling the judges that is believed the computer is the real human.

Based on the background, the effective way to pass the hurdle up to Turing Test is to make the computer pass the IQ test. IQ test is frequently used by companies and school to determine the people who have good or decent intelligence level. Usually the test consist of questions about picture, patterns, sequences, mathematics, and verbal questions. Therefore, the IQ tests can determine whether a program can pass the pass score of normal human intelligence.

This project is about developing a program of artificial intelligence that takes input of IQ test and solve the problems. The purposes of this project is to make a program that pass the minimum score of IQ test. In other words, the IQ tests can be solved by an artificial intelligence which can raise a question of the efficiency of the IQ test itself and the development progress in artificial intelligence field.

2 Project Requirements

2.1 Functional Requirements

Primary:

- The program runs and compiled in Windows system environment
- The program is compiled using Python programming language
- The program understand the meaning of the questions
- The program able to receive single input of IQ test from the user
- The program able to take a file containing a modified format of IQ tests consist of multiple
- The program shows the statistic of the answered questions for file input type
- The program is able to solve Insert missing number IQ type questions
- The program is able to solve Insert missing letters IQ type questions
- The program is able to solve Pick the odd one out IQ type questions
- The program is able to solve IQ questions involving directions
- The program must achieved at least 90 scores in IQ test targeted for adult human difficulty which can match the average normal adult human results

Secondary:

- The program is able to solve prefix and suffix questions
- The program is able to solve comparisons IQ type questions
- The program is able to solve pick the odd one out questions using machine learning technique
- The program is able to analyze other kinds of questions outside specification with its best ability

2.2 Non-functional Requirements

- The program can processed the questions in reasonable amount of time
- The program design need to be extensible for new features which requires the design to have good flexibility, extensibility, and well documented
- Ease of use user interface, simple and working
- The program can handle error or handle unknown solution method for a particular question

3 Project Plan

3.1 Overview

Project Objectives

The objectives of this project is to develop an application that can read an IQ tests, understand the questions, and solve the questions appropriately. The target for the application to be successful as an application that can beat IQ tests needs to be at least the same or higher than average normal human results. In details, the program need to have at least an IQ scores of 90 on IQ tests designed for adult human. The application does not need to solve all type of questions in IQ test since it will be unfeasible and need a lot of time to develop the application. Therefore, several type of IQ test questions are chosen as the program scope which consist of questions related to mathematical theory, and logical problem. The purposes of this project also served to contribute the development in understanding of IQ tests area, Artificial Intelligence field, and mathematical theory in computer science.

Constraints

The project scope constraint are limited to develop an artificial intelligence for solving IQ test questions by using mathematical theory and common problem solving algorithm. The main requirements need to be implemented by the deadline and secondary requirements are optional. The project main requirements consist of receiving inputs of IQ test question which the type of the questions are limited to the project scope requirements which are sequence of number, sequence of letter, odd one out problem, and directions problem. The scope of the program regarding user interface also necessary. The Graphic User Interface for the program just need to be simple of consisting input box or input file for input format and output console or text for output format. All main requirements need to be implemented before any other additional features. All other additional features outside the main requirements can be recognized as secondary requirement. The secondary requirements are not guaranteed to be implemented by the deadlines and it depends on the situation of schedule constraint.

The other scope of the project is the development of the program is restricted to Python programming language and Windows Operating System only. The additional features to run in other platform and languages is not possible at any cost. The features is just not feasible to be implemented in term of schedule constraint.

The project schedule constraint has a total time of 12 weeks to implement the whole program. Any additional or extension to schedule deadlines is not possible. Therefore, effectively the program should be working and running successfully by the twelfth week. The implementation of the program start at 9 April 2018 and finished at 25 May 2018. The allocation of the schedule consist of implementation of Artificial Intelligence algorithm on sixth week up to tenth week. The eleventh week is for testing the program with the test cases. Furthermore, the project must be complete with all the main requirements and the program alongside with the final report and presentation by 25 May 2018. The details of the schedule can be seen at Schedule context in this project specification.

3.2 Risk Analysis

No.	Risk	Description	Triggers	Root Cause	Potential Response
1	Errors and bugs	Errors and bugs appear in the program implementation phase	The output of the program is incorrect or the program stop working correctly	Lack of problem solving analysis and test runs	Debug the source of the problem or error and retest the program. Do the previous step until the problem solved
2	Failed Algorithm	Algorithm designed to solve the problem cannot solve the problem completely	The output for several test cases is incorrect despite multiple modifications of the algorithm	Lack the analysis to prove the algorithm completeness on the problem	Revise new algorithm for the problem and implement it, overwriting the old algorithm
3	Program time complexity	The time needed to solve a problem must be in reasonable time	The program takes a long time to solve a problem	The use of inefficient algorithm in the program. Bad design of the program modules might contribute the complexity of the program	Revisit and analyze the whole program design and algorithm used in the program. Remove or change inefficient algorithm or design modules
4	Platform problem	The probability of facing problems regarding the software or hardware used to develop this project	Software or hardware used in developing the program is not functioning properly. Halting the progress of the work completely	The lack of proper maintenance and updates of the software and hardware used in the project. No back up for the previous version of the program.	Find another alternative software and hardware. Back up the latest version of the program into safe place. Fix the broken software or hardware. Since it takes some time, need to beware of the deadline time
5	Unexpected events	The unexpected events occurs to the developer of the program. For example away for important events, sickness or accident	The developer is away for unexpected events rendering the progress of program implementation in halt	No backup plan for the risk or no backup additional developer available.	Reduce the scope of the project in order to fulfill the requirement within schedule and time.
6	Schedule overrun	The project does not meet its schedule constraint and meet its expected deadline with	The current schedule for the project does not match with the schedule table plan	The scope constraint was not reviewed properly and maintained regularly. Changes in scope	Reduce the scope of the project to meet the schedule constraint.

		all the main requirement		requirements too often	
7	Test Cases Completeness	Test cases sample and methods are not covering all boundary and bugs possibilities, leaving the program uncomplete	High number of bugs and error are still in the program after the testing phase finished	The lack of test cases sample, bad test plan, and lack analysis of test case method to cover all the boundary and bugs possibilities	Revisit the testing phase of the program to test the whole program from unit testing and system testing once more with more thorough test plan

Risk threat score table:

No.	**Risk**	**Rank**	**Probability**	**Impact**	**Score**	**Threat**
1	Error and bugs	7	10	1	10	Very Low
2	Failed algorithm	3	3	9	27	Moderate
3	Program time complexity	2	7	5	35	Moderate
4	Platform problem	5	2	9	18	Low
5	Unexpected events	6	2	7	14	Low
6	Schedule overrun	1	7	9	63	High
7	Test Cases Completeness	4	3	7	21	Moderate

Risk Mitigation

Based from the analysis in risk table, there are some risks that need to be noted and aware of. The risks that have a threat to affect the project are failed algorithm, program time complexity, and schedule overrun. The root causes need to be noted in the project plan and the trigger events of the risks need to be checked on weekly basis. If the risk occurs in the project, the potential response listed in this table need to be completed immediately to reduce the impact of the risk to the project

3.3 Resource Requirements

Minimum Hardware Requirements:

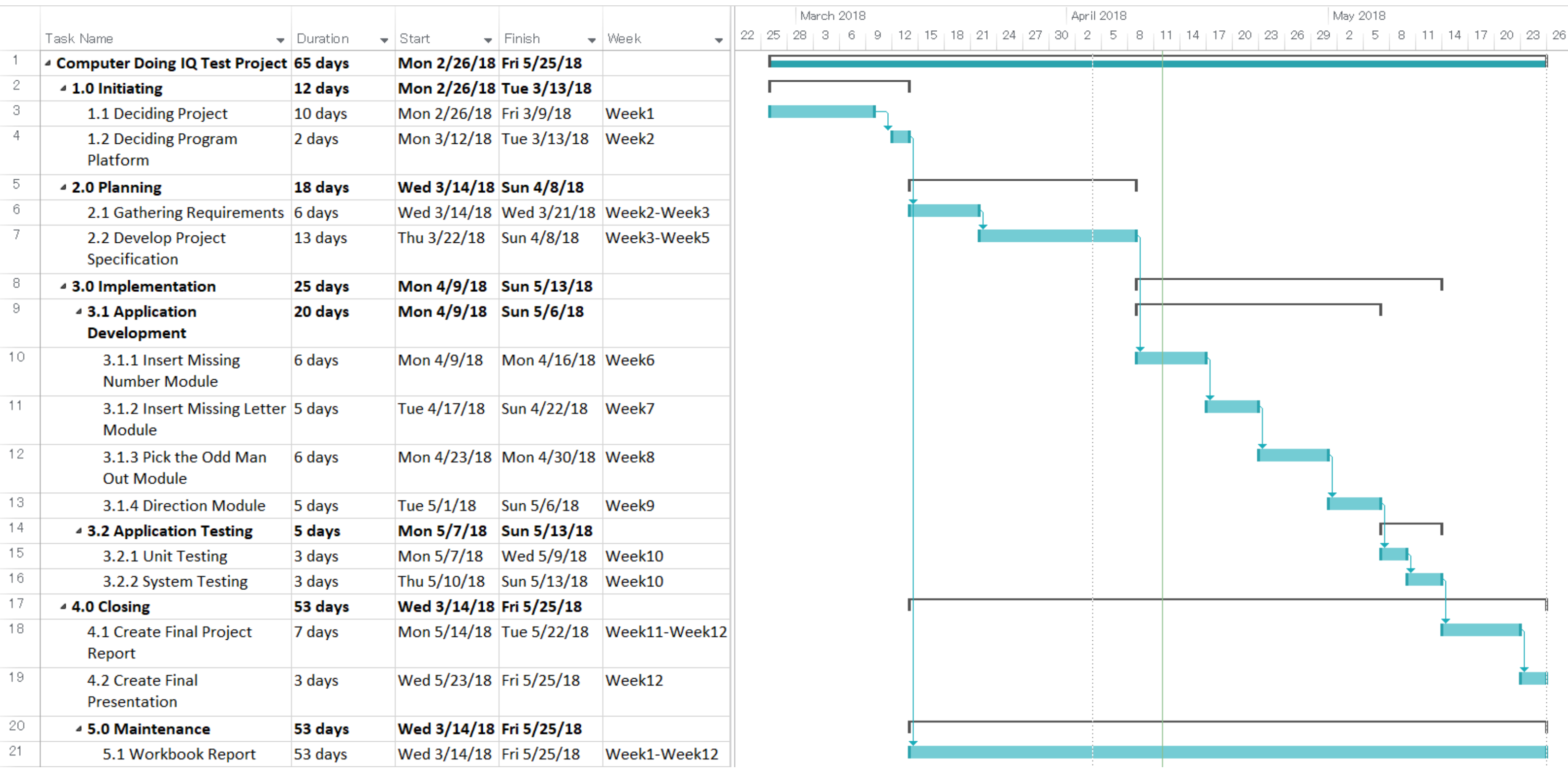
- 1 gigahertz (GHz) processor
- 1 gigabyte (GB) RAM (32-bit/64-bit)
- 1 GB available hard disk space
- Basic integrated video card graphic
- No internet connection required

Minimum Software Requirements:

- Windows 7 operating system (32-bit / 64-bit)
- Python programming language version 3.5.1
- Python libraries used:
  - Appjar library for GUI
  - abc module (Abstract Base Class for Python)

3.4 Schedule

Gantt chart:



	Task Name	Duration	Start	Finish	Week
1	**Computer Doing IQ Test Project**	**65 days**	**Mon 2/26/18**	**Fri 5/25/18**	
2	**1.0 Initiating**	**12 days**	**Mon 2/26/18**	**Tue 3/13/18**	
3	1.1 Deciding Project	10 days	Mon 2/26/18	Fri 3/9/18	Week1
4	1.2 Deciding Program Platform	2 days	Mon 3/12/18	Tue 3/13/18	Week2
5	**2.0 Planning**	**18 days**	**Wed 3/14/18**	**Sun 4/8/18**	
6	2.1 Gathering Requirements	6 days	Wed 3/14/18	Wed 3/21/18	Week2-Week3
7	2.2 Develop Project Specification	13 days	Thu 3/22/18	Sun 4/8/18	Week3-Week5
8	**3.0 Implementation**	**25 days**	**Mon 4/9/18**	**Sun 5/13/18**	
9	**3.1 Application Development**	**20 days**	**Mon 4/9/18**	**Sun 5/6/18**	
10	3.1.1 Insert Missing Number Module	6 days	Mon 4/9/18	Mon 4/16/18	Week6
11	3.1.2 Insert Missing Letter Module	5 days	Tue 4/17/18	Sun 4/22/18	Week7
12	3.1.3 Pick the Odd Man Out Module	6 days	Mon 4/23/18	Mon 4/30/18	Week8
13	3.1.4 Direction Module	5 days	Tue 5/1/18	Sun 5/6/18	Week9
14	**3.2 Application Testing**	**5 days**	**Mon 5/7/18**	**Sun 5/13/18**	
15	3.2.1 Unit Testing	3 days	Mon 5/7/18	Wed 5/9/18	Week10
16	3.2.2 System Testing	3 days	Thu 5/10/18	Sun 5/13/18	Week10
17	**4.0 Closing**	**53 days**	**Wed 3/14/18**	**Fri 5/25/18**	
18	4.1 Create Final Project Report	7 days	Mon 5/14/18	Tue 5/22/18	Week11-Week12
19	4.2 Create Final Presentation	3 days	Wed 5/23/18	Fri 5/25/18	Week12
20	**5.0 Maintenance**	**53 days**	**Wed 3/14/18**	**Fri 5/25/18**	
21	5.1 Workbook Report	53 days	Wed 3/14/18	Fri 5/25/18	Week1-Week12

4 External Design

4.1 User Interface

The program is planned to be in executable form. Hence, the program can be run by clicking the executable program and close the application with close button.

The program graphic user interface consist of one main window and two sub windows. The main window consist of two interactive buttons that each button would open different sub window. The purposes of each interactive element of the program in each window are as follow:

Main window:

Purpose: The first window to open when the program starts and offering user options to either use the AI for solving IQ questions or show the AI analytics results of solving IQ questions.

- Solve button: open the solve window and close the main window
- Analytic button: open the analytic window and close the main window
- Close button: exit the program

Solve window:

Purpose: solve sub window offers user to input the IQ question through the text field or attach a text file reformatted specially for the AI. Then the program would solve the question and output the result in result text field or output.txt if file is attached. Please note that user cannot choose to input text field and attach file at the same time.

- Input Text Field: text field of an IQ question including the multiple choices if any
- Output Text Field: unwritable text field showing the result of the IQ question inputted in input text field
- Attach file button: button to attach a file containing all the IQ questions specially formatted for the AI program to process
- Solve button: button to solve the IQ questions inputted in either input text field or attached file button. If neither are blank, then error message in outputted
- Back button: button to go back to the main window

Analytic Window:

Purpose: sub window that solve the IQ questions, show the results or solutions, and shows the descriptive analysis of the program performance such as correct and wrong ratio, percentage, questions solved based on the type, and unrecognized questions. The elements of this sub window are basically the same with solve sub window with an exceptions of no input text field and expanded output text field to show the descriptive statistics.

4.2 Performance

The time complexity of the program are unique based on the questions type. In other words, each question type has its own problem solving algorithm that might unique with each other. There are 4 types of question the program can handle based on the main requirements. Each time complexity for solving each types are as follows:

1.	Insert the missing number:	$O(n)$ linear complexity
2.	Insert the missing letters:	$O(n)$ linear complexity
3.	Directions:	$O(n)$ linear complexity
4.	Pick the odd one out:	$O(n^2)$ quadratic complexity

In overall, the time complexity based on the average of all the problem solving algorithm in the program is in linear complexity $O(n)$.

The space complexity of the program are also unique based on the question type since each type would have different problem solving technique. The space complexity of each question type are as follows:

1. Insert the missing number: $O(n)$ linear complexity
2. Insert the missing letters: $O(n)$ linear complexity
3. Directions: $O(n)$ linear complexity
4. Pick the odd one out: $O(n)$ linear complexity

All of the algorithm complexity is in linear complexity $O(n)$. Therefore, the space complexity of the program is proven to be effective and better than the time complexity.

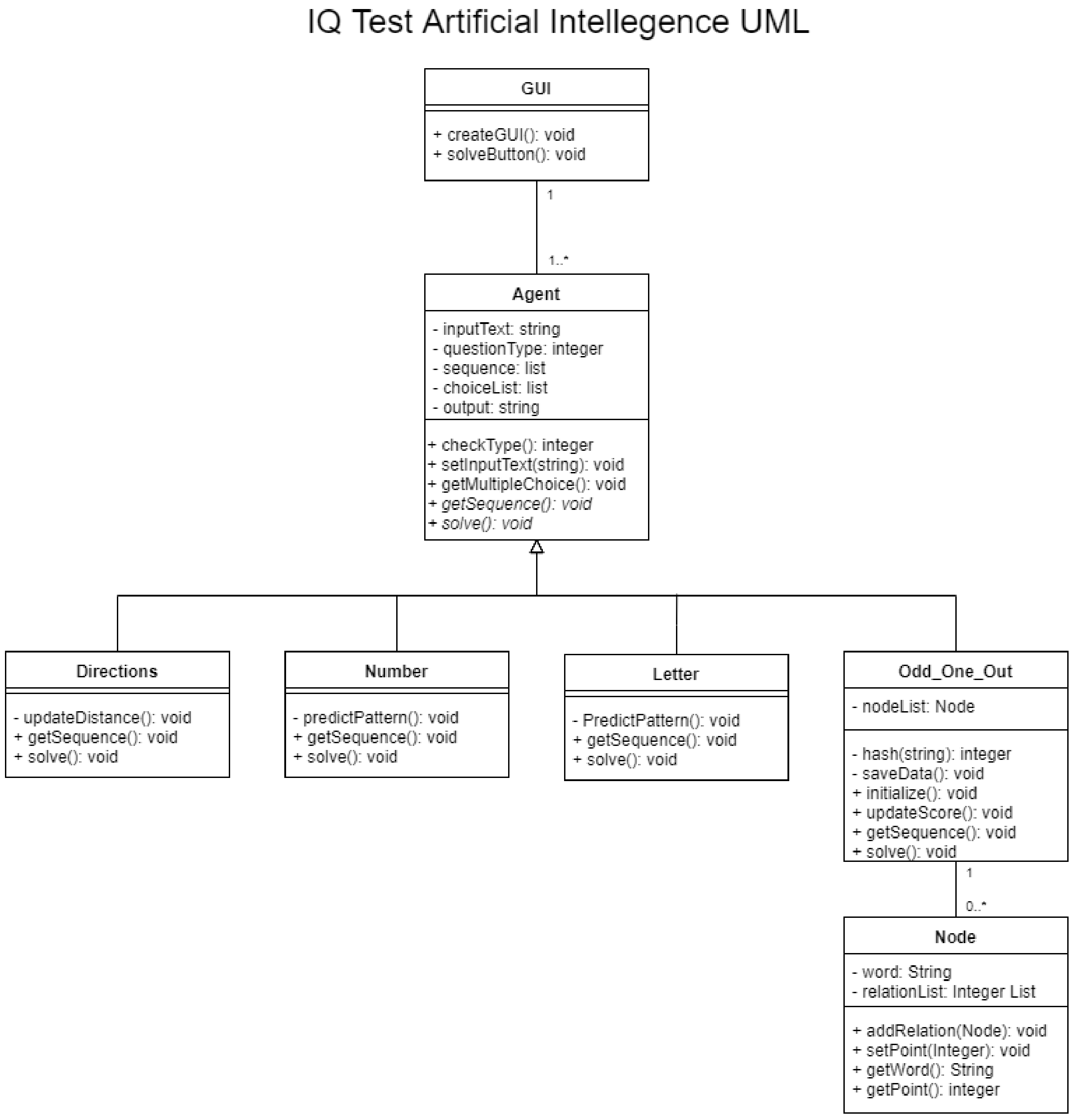
5 Internal Design

5.1 Use Case

Name: Inputting IQ Question into the program
Actors: Users
Purpose: Solve the IQ question using the AI system in the program

Typical Course of Events		
Step	**Actor Action**	**System Response**
1	User choose the solve button in the main window	
2		System show the solve sub window and hide the main window
3	User inputted the IQ question in input text field including the question and the multiple choice answers if any	
4	User click the solve button	
5		System process the inputted text and solve the question. System output the solution in the output text field
Alternative Courses		
5.1	System output an error message in output text field if the input text field or attached file is blank or not supported	

6 Software Architecture



The software is consist with GUI class that maintain the user interface and interaction between the user and the AI agent, and Agent class that solves the IQ questions. The Agent class has abstract method that are inherited and implemented into its subclass of Number, Letter, Directions, and Odd_Man_Out. These subclasses represent each problem solving technique for their respective question type. Since the Odd_Man_Out algorithm uses supervise learning, it needs Node class to represent each data of words and establish relationship with other words.

The relationship of GUI class and Agent class is one to many relationship which representing that the program can handle multiple types of questions in terms of many agent while the each agent must be assigned to particular program. The relationship between Odd_Man_Out class and Node class is one to many relationship which representing that the algorithm need a Node list to solve the problem and each Node must be assigned to an agent only.

7 Test Plan

No.	Requirement	Method	Test Case	Expected Result
1	Missing and sequence number module	Unit Testing, Integration Testing	The sample test case is collected based on references in IQ test book and questions made from mathematical questions. All the questions format inputted into the module is in array format.	The output of the program is stored in an array and checked immediately by automated testing. The console output shows that all the answer is correct
2	Odd one out module	Unit Testing, Integration Testing	Test case is inputted in array format into the module. For example, 4 words item in an array.	The output should be a word that is unrelated to the other 3 words.
3	Direction module	Unit Testing, Integration Testing	Test case is inputted in array format. Each item in array consist of number represent the distance and a letter represent the direction. For example 10S represent 10 to the south. 5R represents 5 to the right.	The output of the module is printed in number format represent the correct calculation for the distance
4	Graphic user interface functionality	Integration testing	Test case of GUI is specially treated manually by the developer by interacting the program with user extensively. The test cases include clicking buttons, typing text, and combination interaction between elements.	The program should behave in right manner for each user interaction.
5	The whole program testing on file input	System testing	Test case is prepared in text file format. The test case is inputted through the program final version and attached the file then click solve button.	The output solution of the program can be found in output.txt and compared in automated testing

			The test case contents are all the IQ questions gathered from IQ test book but the question type is limited to main requirement of the program	with the prepared solution text. The result of the output text should mostly correct.
6	Results and score satisfiability	Acceptance testing	The sample test case is the same format with system test case.	The output solution of the program has the total scoring of 90 or above in IQ score.

8 References

Carter, P. J. (2008). *Advanced IQ Tests : The Toughest Practice Questions to Test Your Lateral Thinking Problem Solving and Reasoning Skills*. London, United Kingdom: Kogan Page Ltd.

D. L. Dowe, H.-O. J. (2012, March-April). IQ tests are not for machines, yet. Intelligence, 40(2), 77-81. doi:10.1016/j.intell.2011.12.001, accepted Thu 22/Dec/2011, online Sat 21/Jan/2012

Erwin Hilton, Q. L. a. T. P. (2017). *Spatial IQ Test for AI*. Massachusetts Institute of Technology, Retrieved from https://dspace.mit.edu/handle/1721.1/113004

J. Hernandez-Orallo, F. Martinez-Plumed, U. Schmid, M. Siebers and D. L. Dowe (2016), ``*Computer models solving intelligence test problems: Progress and implications*'', Artificial Intelligence Journal (AIJ), Volume 230, Jan 2016, pp74-107 (was here). [doi:10.1016/j.artint.2015.09.011, accepted Sun 27/Sept/2015, online Thu 22/Oct/2015]

Martina, M. (2014). *Applying Inductive Programming to Solving Number Series Problems -- Comparing Performance of Igor with Humans*. (Master of Applied Computer Science), University of Bamberg

Phillip J. Carter, K. R. (2007). *The Ultimate IQ Test Book : 1000 Practice Test Questions to Boost Your Brain Power*. London, United Kingdom: Kogan Page Ltd.

P Sanghi, D. L. D. (2003, 13-17 July). *A Computer Program Capable of Passing I.Q. Tests*, in P P Slezak (ed), Proceedings of the Joint International Conference on Cognitive Science, 4th ICCS International Conference on Cognitive Science & 7th ASCS Australasian Society for Cognitive Science (ICCS/ASCS-2003), Sydney, NSW, Australia, pp 570-575

Richard S. Sutton, A. G. B. (2018). 1.1 Reinforcement Learning. In *Reinforcement Learning: An Introduction, Second Edition (Draft)* (pp. 548). Massachusetts: The MIT Press.