QJudge: Automated Testing of Students Solutions for Quantum Algorithms Courses

Open-Source Software

Mansur Ziiatdinov[*]

Abstract

We describe QJudge — an automated testing system tailored for problems with quantum circuit solutions.

QJudge allows you to prepare problems and make decisions in the form of quantum circuits. It can be convenient for use in the initial courses of quantum programming when students do not yet have sufficient knowledge of quantum programming libraries.

The paper compares QJudge with other testing systems and describes the types of problems that are available in QJudge. We discuss the architecture of the system, as well as the experience of using QJudge for teaching a course on quantum algorithms.

1 Introduction

Mastering something requires solving exercises. A lot of programming courses include exercises; many of them use automated testing systems to reduce the teacher's workload. However, these testing systems are not suitable for quantum programming courses, because they require to write code, and quantum algorithms course begins by describing the computation model of circuits.

There are several games about quantum informatics, but they usually teach about different gates and quantum mechanics. Moreover, there is no competitive programming system for quantum tasks that allows to draw circuits, and there is no middle ground between novices who learn about quantum optics and professionals who program in Qiskit.

QJudge aims to fill in this gap, and provides a familiar interface for student, who solves problems by drawing the same circuits that were described at the beginning of the course.

[*]gltronred@gmail.com

2 QJudge Overview

User interface of QJudge [1] is similar to other automated testing systems. It provides a list of problems (exercices), table of user standings, a list of user submissions etc. However, instead of form to write a code, or to upload a file with solution, QJudge shows an interface for drawing a circuit. User can drag and drop gates to define a circuit that solves the problem and submit it to the server.

QJudge uses a modified fork of Quirk [2] to draw circuits. QJudge converts solutions to Cirq [3] programs and runs them on tests predefined by problem authors.

There are several types of problems that QJudge currently supports. The test defines an input state and describes a correct output state. The output state description is a base state, amplitudes of all states, or an inequality that the probabilities of measurement results must satisfy.

3 Implementation Details

QJudge is an open-source [4] web application. The frontend part is written in Vue.js [5], and backend part consists of several services. One service is server that accepts submissions and puts them into Faktory [6] queue. Another service is worker that fetches the submission from the queue, runs it on tests, and writes the results into Sqlite database. The architecture allows to launch more workers if necessary, and balance the load.

While usual automated testing systems employ some containerization tools to compile and run user submissions, QJudge in the current state don't require it. User submission is simply a description of quantum circuit that has no side effects. It is compiled into Cirq code, and the compilation result also do not have side effects. However, it is possible to add containerization around the worker, would it be necessary.

4 Conclusions

Author used QJudge to test knowledge of students in the introductory course of quantum algorithms. QJudge can also be used in the introductory level quantum programming contests and other events, where it is required to test knowledge of quantum algorithms.

However, QJudge lacks the hosted instance that is available to everyone. Another drawback is the difficulty of defining problems: while tests are simply JSON-files, the real UI for then would be more convenient.

References

[1] Mansur Ziiatdinov. Hosted instance of QJudge, 2022. `https://qjudge.gltronred.info`.

[2] Craig Gidney. Quirk sources. `https://github.com/Strilanc/Quirk`.

[3] Cirq Developers. Cirq, April 2022. See full list of authors on Github: `https://github.com/quantumlib/Cirq/graphs/contributors`.

[4] Mansur Ziiatdinov. QJudge sources, 2022. `https://sr.ht/~rd/qjudge`.

[5] Vue.js Team. Vue.js. `https://vuejs.org`.

[6] Contribsys. Faktory. `https://contribsys.com/faktory/`.