QJudge

Automated Testing of Students Solutions for Quantum Algorithms Courses

Mansur Ziiatdinov

Kazan Federal University

May 30, 2023

Outline

Why?

Why?

Quantum Technologies competitions

- evaluating solutions in quantum algorithms

Why?

Quantum Technologies competitions

- evaluating solutions in quantum algorithms

KFU introductory course on quantum algorithms

- $\approx$ 10 students
- final year
- evaluating solutions in quantum algorithms

Automated Testing of Quantum Circuits

- no knowledge of Qiskit/Cirq/etc. expected

Automated Testing of Quantum Circuits

- no knowledge of Qiskit/Cirq/etc. expected
- sometimes students don't know Python

Automated Testing of Quantum Circuits

- no knowledge of Qiskit/Cirq/etc. expected
- sometimes students don't know Python
- quick feedback

Testing Systems

Automated Testing Platforms

- ejudge
- codeforces
- . . .

Testing Systems

Automated Testing Platforms

- ejudge
- codeforces
- ...

Usual Structure

- list of problems

Testing Systems

Automated Testing Platforms

- ejudge
- codeforces
- ...

Usual Structure

- list of problems
- solution is a program

Testing Systems

Automated Testing Platforms

- ejudge
- codeforces
- ...

Usual Structure

- list of problems
- solution is a program
- system compiles it

Testing Systems

Automated Testing Platforms

- ejudge
- codeforces
- . . .

Usual Structure

- list of problems
- solution is a program
- system compiles it
- . . . feeds predefined input to solution

Testing Systems

Automated Testing Platforms

- ejudge
- codeforces
- . . .

Usual Structure

- list of problems
- solution is a program
- system compiles it
- . . . feeds predefined input to solution
- . . . checks whether output is correct

Testing Systems

Automated Testing Platforms

- ejudge
- codeforces
- . . .

Usual Structure

- list of problems
- solution is a program
- system compiles it
- . . . feeds predefined input to solution
- . . . checks whether output is correct
- . . . repeats for all predefined inputs

Testing Systems

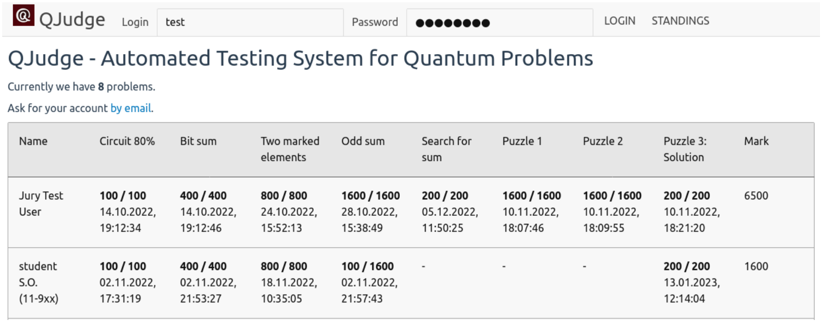
Automated Testing Platforms

- ejudge
- codeforces
- ...

Usual Structure

- list of problems
- solution is a program
- system compiles it
- ... feeds predefined input to solution
- ... checks whether output is correct
- ... repeats for all predefined inputs
- can't use circuit as solution

Brief look at the interface



QJudge Login test Password ●●●●●●●●● LOGIN STANDINGS

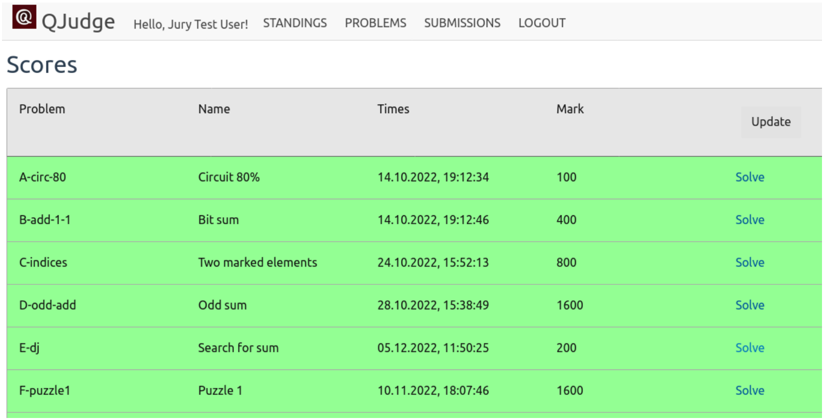
QJudge - Automated Testing System for Quantum Problems

Currently we have **8** problems.

Ask for your account by email.

Name	Circuit 80%	Bit sum	Two marked elements	Odd sum	Search for sum	Puzzle 1	Puzzle 2	Puzzle 3: Solution	Mark
Jury Test User	**100 / 100** 14.10.2022, 19:12:34	**400 / 400** 14.10.2022, 19:12:46	**800 / 800** 24.10.2022, 15:52:13	**1600 / 1600** 28.10.2022, 15:38:49	**200 / 200** 05.12.2022, 11:50:25	**1600 / 1600** 10.11.2022, 18:07:46	**1600 / 1600** 10.11.2022, 18:09:55	**200 / 200** 10.11.2022, 18:21:20	6500
student S.O. (11-9xx)	**100 / 100** 02.11.2022, 17:31:19	**400 / 400** 02.11.2022, 21:53:27	**800 / 800** 18.11.2022, 10:35:05	**100 / 1600** 02.11.2022, 21:57:43	-	-	-	**200 / 200** 13.01.2023, 12:14:04	1600

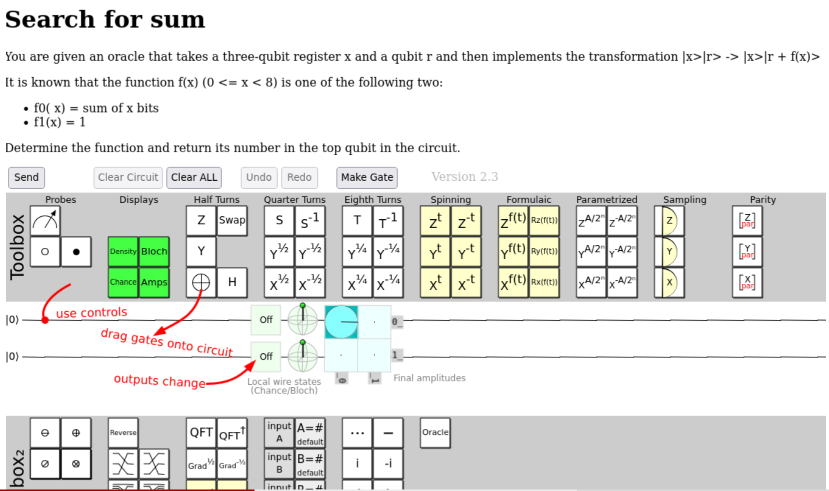
Brief look at the interface



Scores

Problem	Name	Times	Mark	
A-circ-80	Circuit 80%	14.10.2022, 19:12:34	100	Solve
B-add-1-1	Bit sum	14.10.2022, 19:12:46	400	Solve
C-indices	Two marked elements	24.10.2022, 15:52:13	800	Solve
D-odd-add	Odd sum	28.10.2022, 15:38:49	1600	Solve
E-dj	Search for sum	05.12.2022, 11:50:25	200	Solve
F-puzzle1	Puzzle 1	10.11.2022, 18:07:46	1600	Solve

Brief look at the interface



Search for sum

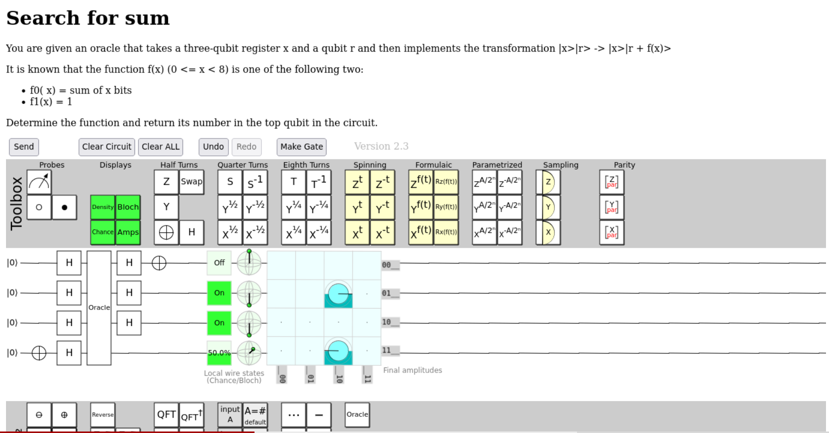
You are given an oracle that takes a three-qubit register x and a qubit r and then implements the transformation |x>|r> -> |x>|r + f(x)>

It is known that the function f(x) (0 <= x < 8) is one of the following two:

- f0(x) = sum of x bits
- f1(x) = 1

Determine the function and return its number in the top qubit in the circuit.

Brief look at the interface



Search for sum

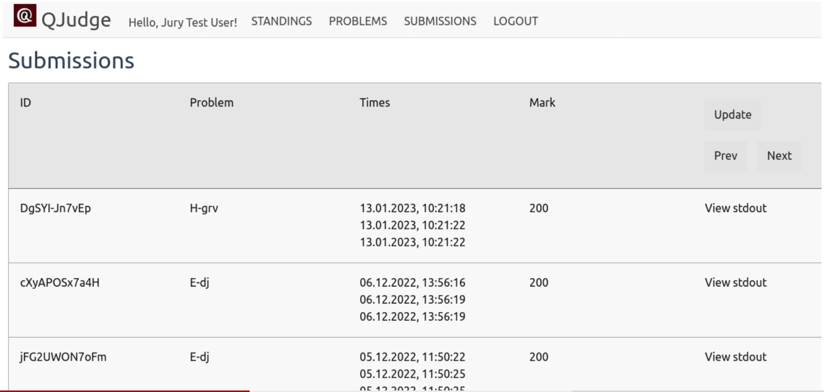
You are given an oracle that takes a three-qubit register x and a qubit r and then implements the transformation |x>|r> -> |x>|r + f(x)>

It is known that the function f(x) (0 <= x < 8) is one of the following two:

- f0(x) = sum of x bits
- f1(x) = 1

Determine the function and return its number in the top qubit in the circuit.

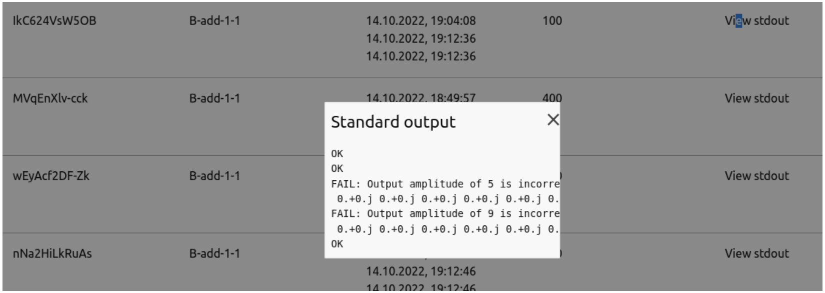
Brief look at the interface



Submissions

ID	Problem	Times	Mark	
DgSYI-Jn7vEp	H-grv	13.01.2023, 10:21:18 13.01.2023, 10:21:22 13.01.2023, 10:21:22	200	View stdout
cXyAPOSx7a4H	E-dj	06.12.2022, 13:56:16 06.12.2022, 13:56:19 06.12.2022, 13:56:19	200	View stdout
jFG2UWON7oFm	E-dj	05.12.2022, 11:50:22 05.12.2022, 11:50:25 05.12.2022, 11:50:25	200	View stdout

Brief look at the interface



IkC624VsW5OB	B-add-1-1	14.10.2022, 19:04:08 14.10.2022, 19:12:36 14.10.2022, 19:12:36	100	View stdout
MVqEnXlv-cck	B-add-1-1	14.10.2022, 18:49:57	400	View stdout
wEyAcf2DF-Zk	B-add-1-1			View stdout
nNa2HiLkRuAs	B-add-1-1	14.10.2022, 19:12:46 14.10.2022, 19:12:46		View stdout

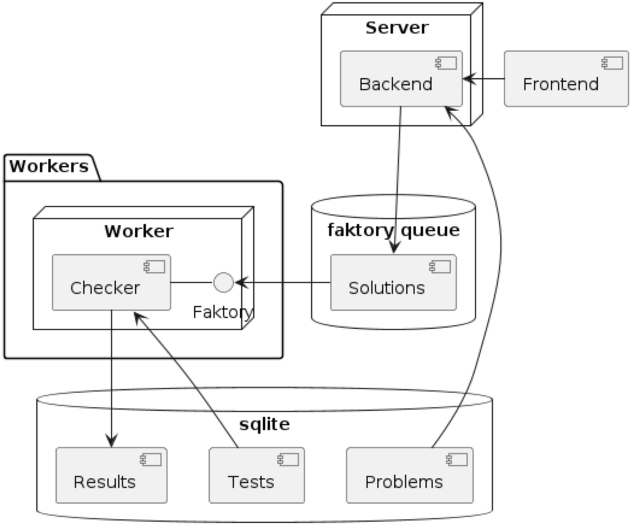
Standard output

```
OK
OK
FAIL: Output amplitude of 5 is incorre
 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.
FAIL: Output amplitude of 9 is incorre
 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.
OK
```

Outline

Architecture

- scalability
- easy deployment



Checking Solution

Frontend

- fork of Quirk

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
    - postselection

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
  - postselection
  - time dependence

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
    - postselection
    - time dependence
    - non-physical “gates”

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
    - postselection
    - time dependence
    - non-physical “gates”
    - . . .

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
    - postselection
    - time dependence
    - non-physical “gates”
    - . . .
- send JSON to backend

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
  - postselection
  - time dependence
  - non-physical “gates”
  - . . .
- send JSON to backend

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
  - postselection
  - time dependence
  - non-physical “gates”
  - ...
- send JSON to backend

Backend

- import to Cirq

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
  - postselection
  - time dependence
  - non-physical “gates”
  - . . .
- send JSON to backend

Backend

- import to Cirq
- no need to “jail” solution

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
  - postselection
  - time dependence
  - non-physical "gates"
  - ...
- send JSON to backend

Backend

- import to Cirq
- no need to "jail" solution
- run on predefined tests

Checking Solution

Frontend

- fork of Quirk
- remove gates not supported by Cirq:
  - postselection
  - time dependence
  - non-physical “gates”
  - ...
- send JSON to backend

Backend

- import to Cirq
- no need to “jail” solution
- run on predefined tests

Checking Solution

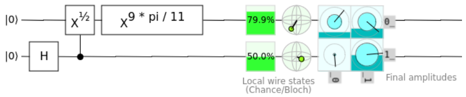
Frontend

- fork of Quirk
- remove gates not supported by Cirq:
  - postselection
  - time dependence
  - non-physical “gates”
  - ...
- send JSON to backend

Backend

- import to Cirq
- no need to “jail” solution
- run on predefined tests

Example



Checking Solution

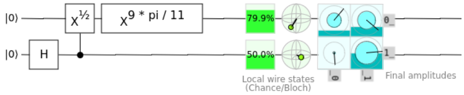
Frontend

- fork of Quirk
- remove gates not supported by Cirq:
    - postselection
    - time dependence
    - non-physical "gates"
    - ...
- send JSON to backend

Backend

- import to Cirq
- no need to "jail" solution
- run on predefined tests

Example



- `{"cols": [[1, "H"], ["X^½", "•"], [{"id": "X^ft", "arg": "9*pi/11"}]]}`

Outline

Base State to Base State

- $|001\rangle \mapsto |010\rangle, \ldots$

Base State to Base State

- $|001\rangle \mapsto |010\rangle, \ldots$
- e.g.: oracle $|x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$

Base State to Base State

- $|001\rangle \mapsto |010\rangle, \ldots$
- e.g.: oracle $|x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$
- `{"input": {"base": "001"}}`

Base State to Base State

- $|001\rangle \mapsto |010\rangle, \dots$
- e.g.: oracle $|x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$
- `{"input": {"base": "001"}}`
- `{"output": {"base": "010"}}`

Base State to Base State

- $|001\rangle \mapsto |010\rangle, \ldots$
- e.g.: oracle $|x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$
- `{"input": {"base": "001"}}`
- `{"output": {"base": "010"}}`
- special handling of ancilla qubits: e.g. three qubits and ancilla gives state $\alpha_0|0100\rangle + \alpha_1|0101\rangle$

Base State to Base State

- $|001\rangle \mapsto |010\rangle, \dots$
- e.g.: oracle $|x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$
- `{"input": {"base": "001"}}`
- `{"output": {"base": "010"}}`
- special handling of ancilla qubits: e.g. three qubits and ancilla gives state $\alpha_0|0100\rangle + \alpha_1|0101\rangle$
- `{"output": {"proj": "010"}}`

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$
- e.g.: phase oracle, single operations, ...

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$
- e.g.: phase oracle, single operations, . . .
- `{"output": {"ampl": [ [0,0], ..., [0,0], [-1,0], [0,0], ...]}}`

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$
- e.g.: phase oracle, single operations, ...
- `{"output": {"ampl": [ [0,0], ..., [0,0], [-1,0], [0,0], ...]}}`
- `[a,b]` means $a + bi$

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$
- e.g.: phase oracle, single operations, ...
- `{"output": {"ampl": [ [0,0], ..., [0,0], [-1,0], [0,0], ...]}}`
- `[a,b]` means $a + bi$
- comparing with precision ε

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$
- e.g.: phase oracle, single operations, ...
- `{"output": {"ampl": [ [0,0], ..., [0,0], [-1,0], [0,0], ...]}}`
- `[a,b]` means $a + bi$
- comparing with precision ε

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$
- e.g.: phase oracle, single operations, ...
- `{"output": {"ampl": [ [0,0], ..., [0,0], [-1,0], [0,0], ...]}}`
- `[a,b]` means $a + bi$
- comparing with precision ε

Subset of Amplitudes

- it is tiresome to list all amplitudes

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$
- e.g.: phase oracle, single operations, ...
- `{"output": {"ampl": [ [0,0], ..., [0,0], [-1,0], [0,0], ...]}}`
- `[a,b]` means $a + bi$
- comparing with precision ε

Subset of Amplitudes

- it is tiresome to list all amplitudes
- you can be interested only in some of them

Amplitudes

All Amplitudes

- $|000\rangle \mapsto \alpha_0|000\rangle + \alpha_1|001\rangle + \ldots + \alpha_7|111\rangle$
- e.g.: phase oracle, single operations, ...
- `{"output": {"ampl": [ [0,0], ..., [0,0], [-1,0], [0,0], ...]}}`
- `[a,b]` means $a + bi$
- comparing with precision ε

Subset of Amplitudes

- it is tiresome to list all amplitudes
- you can be interested only in some of them
- `{"output": {"sub_ampl": {"001": [0.707, 0.707], ...}}}`

Measurement Probabilities

- probability of measuring some set of states

Measurement Probabilities

- probability of measuring some set of states
- e.g. $\Pr[r \in \{|001\rangle, |010\rangle, |100\rangle, |111\rangle\}] \geq 3/4$

Measurement Probabilities

- probability of measuring some set of states
- e.g. $\Pr[r \in \{|001\rangle, |010\rangle, |100\rangle, |111\rangle\}] \geq 3/4$
- algorithms: Deutch-Jozsa, Grover etc.

Measurement Probabilities

- probability of measuring some set of states
- e.g. $\Pr[r \in \{|001\rangle, |010\rangle, |100\rangle, |111\rangle\}] \geq 3/4$
- algorithms: Deutch-Jozsa, Grover etc.
- `{"output": {"prob": [{"states": ["001","010","100","111"], "req": "GE", "prob": 0.75}, ...]}}`

Using Oracles

- oracles are special gates that are used in solution

Using Oracles

- oracles are special gates that are used in solution
- tests can define different oracles

Using Oracles

- oracles are special gates that are used in solution
- tests can define different oracles
- e.g. Deutch-Jozsa when function is balanced and when function is constant

Using Oracles

- oracles are special gates that are used in solution
- tests can define different oracles
- e.g. Deutch-Jozsa when function is balanced and when function is constant
- tests define list of oracles: `{"oracles":[ oracle1, ...  ]}`

Using Oracles

- oracles are special gates that are used in solution
- tests can define different oracles
- e.g. Deutch-Jozsa when function is balanced and when function is constant
- tests define list of oracles: `{"oracles":[ oracle1, ... ]}`
- each oracle is a circuit: `{"id":"~ib1j", "name":"Oracle", "circuit": {"cols": [["•","•","•","•",1,"X"], ["◦","◦","◦","◦","X"], [1,1,1,1,"X","•"], ["•","•","•","•",1,"X"]]}}`

Extending

- checker is Python script

Extending

- checker is Python script
- test descriptions are sent to standard input

Extending

- checker is Python script
- test descriptions are sent to standard input
- test verdicts are written to standard output

Outline

Current Drawbacks and Future Work

- UI for teacher

Current Drawbacks and Future Work

- UI for teacher
- UI for problem designer

Current Drawbacks and Future Work

- UI for teacher
- UI for problem designer
- automatic registration with confirmation by email

Current Drawbacks and Future Work

- UI for teacher
- UI for problem designer
- automatic registration with confirmation by email
- user-friendly error messages

Current Drawbacks and Future Work

- UI for teacher
- UI for problem designer
- automatic registration with confirmation by email
- user-friendly error messages
- UI for system administrator

Current Drawbacks and Future Work

- UI for teacher
- UI for problem designer
- automatic registration with confirmation by email
- user-friendly error messages
- UI for system administrator
- hosted instance

Current Drawbacks and Future Work

- UI for teacher
- UI for problem designer
- automatic registration with confirmation by email
- user-friendly error messages
- UI for system administrator
- hosted instance
- further testing

Outline

Demo

Thank you!

QJudge Links

- `https://qjudge.gltronred.info` (early alpha stage)
- ask me (Mansur Ziiatdinov) for account if you want to try
- source: `https://sr.ht/~rd/qjudge`