# CS239 Q# HW

Anirudh Veeraragavan, Vaibhav Aggarwal

June 7, 2019

## 1 Evaluation

1. We made numerous test cases for both programs that test the correctness of each algorithm. For Deutsch-Jozsa, we tested a few balanced and unbalanced functions up to $n = 23$ bits. For Bernstein-Vazirani, we tested various values of a for functions up to $n = 12$ bits. For Grover's, we tested functions up to $n = 7$ bits. For Simon's, we tested functions up to $n = 8$ bits.

   To run them, just run the command `dotnet test`. All the test cases pass so we are confident that our implementations are correct.

   For Deutsch-Jozsa, we see a spike in runtime if the function we are testing is $x\%2$, compared to both constant and balanced functions of the same $n$. The other balanced function we tested was splitting the input range into a lower half and upper half, and this ran faster than the mod 2 balanced function.

   For Bernstein-Vazirani, runtimes were the same across the board for different $a$ values if we held $n$ constant.

   For Grover's we first noticed that runtimes tended to have huge variability. Results widely differed between successive runs, and it made it challenging to infer any generalizations. Regardless it does appear that certain values of $x$ do affect the runtime.

   For Simon's, runtimes were the same across the board for different $s$ values if we held $n$ constant.

2. As $n$ grows, runtime seems to get exponentially slower for both algorithms. The upper limit of reasonably simulating these algorithms on a personal computer varies based on the algorithm of choice. A diagram is shown below of the results:

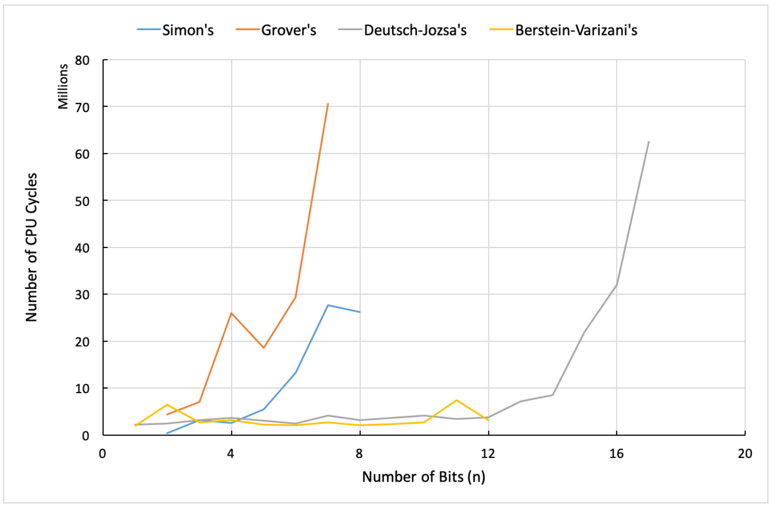

Figure 1: Scalability of $n$ for Bernstein-Vazirani, Deutsch-Jozsa, Simon's, and Grover's.

## 2 Instructions

The programs for Deutsch-Jozsa, Bernstein-Vazirani, and Grover are primarily implemented in Q and were tested/run through the unit test harness. The command for running the unit test harness in all cases is `dotnet test`. To run Deutsch-Jozsa our unit test harness has a function `AssertDJAlgorithmWorks` which takes four input parameters: the number of qubits, a $U_f$ to test Deutsch-Jozsa on, the expected output from Deutsch-Jozsa, and an error message in the event of failure. Thus the user simply has to input their function to this unit test, run the test harness, and ensure the tests pass as expected. To run Berstein-Vazirani a similar process is used, except the function used is called `BV` and it takes two input parameters: the value of a and the number of qubits. To run Grover the function used is called `Grover` which takes as input the value $x$ for which $f(x) = 1$ and the number of qubits. Again in either case simply run the test harness and ensure everything passes.

The process for running Simon's is slightly different. Simon's uses a C harness to run the unit tests, and within this harness the function `Test` is used to input various values of s for various number of qubits. Simply modify this function to run the specific instance you are interested in and run the harness.

Here is an example:

```
$ dotnet test
Microsoft (R) Test Execution Command Line Tool Version 16.1.0
Copyright (c) Microsoft Corporation.  All rights reserved.
```

```
Starting test execution, please wait...
Name: DJ_Algorithm_Constant_0_N_5_Test RunTime: 22177500

Test Run Successful.
```

## 3 Q#

1. It was easy to learn how to test the code, write oracles, and to integrate with C# code. It was a bit tricky to properly handle profiling, measurement, and the construction of $U_f$.

2. Q# had good support for quantum gates, for modularizing quantum programs, and for testing quantum code. It could use better support for profiling, making the Qubit handling process easier, and debugging with clearer stack traces.

3. If Q# could have better support for common classical computing operations in the language itself such as profiling, printing, and so on it would be a lot more convenient.

4. We had a rather negative experience with the Q# documentation. It was challenging to find references, there were limited examples, and the search bar worked poorly. On the positive side the documentation was thorough, the GitHub code was very clear, and there were useful explanation of Quantum Computing concepts.

5. Measurement $\rightarrow$ Result, Gates $\rightarrow$ Operations, Unitary Matrix $\rightarrow$ Operator Application

6. The simulator does enforce this. In our Grover's test harness we have the following code:

   ```
   using (register = Qubit[N]) {
       testOp(register);
       ...
       ResetAll(register);
   }
   ```

   `testOp` modifies register, and if we remove the `ResetAll` line then the code will fail to compile and throw an Qubit error message.