BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Menurut Hukum Moore, jumlah transistor pada sebuah *integrated circuit* (IC) akan berlipat ganda setiap 18 bulan, hal ini menyebabkan kecepatan pemrosesan semakin cepat (Kinnunen, 2015). Ukuran transistor yang semakin kecil mendekati ukuran atomnya menyebabkan elektron tidak lagi mengikuti prinsip-prinsip fisika klasik (Barde, dkk., 2011). Elektron dapat *shoot-through* dari transistor satu ke transistor lainnya yang disebut fenomena kuantum. Komputer kuantum diusulkan sebagai salah satu cara untuk mengatasi fenomena kuantum tersebut (Kanamori dkk, 2006).

Salah satu potensi komputer kuantum adalah meretas *public-key* kriptografi seperti RSA (Moller dan Vuik, 2017). Potensi ini didukung secara teori dengan ditemukannya algoritma kuantum Shor (1994) yang kompleksitasnya $O((\log N)^2(\log\log N)(\log\log\log N))$ memecahkan faktorisasi bilangan yang besar ke dalam faktor prima. Selain itu, komputer kuantum juga berpotensi dalam pencarian sebuah data pada *database* tidak terstruktur (Ying, 2010). Pencarian data ini dapat dilakukan dengan menggunakan algoritma kuantum Grover yang kompleksitasnya $O(\sqrt{N})$ (Ying, 2010).

Komputasi kuantum merupakan tantangan baru seperti diungkapkan oleh Moller dan Vuik (2017), yaitu "berlomba ke bulan" sehingga banyak perusahaan yang melakukan investasi terhadap komputer kuantum . Google telah meresmikan prosesor kuantum komputer dengan 72 *qubit processor,* yang dikenal dengan

Bristlecone, dimana telah melampaui milik IBM, 50 *qubit processor* (Greene, 2018). Selain itu, perusahaan lain seperti Microsoft dan Rigetti juga melakukan riset terhadap komputasi kuantum dari sisi komputer, *platform*, *library*, bahasa, hingga *tools.*

Salah satu produk dari Rigetti adalah *quantum cloud platform* yang memungkinkan mengakses *quantum hardware* Rigetti melalui *virtual development environment* (Rigetti Computing, 2019). Rigetti juga menyediakan Forest SDK yang berisikan Quantum Virtual Machine (QVM) didalamnya untuk melakukan pemrograman kuantum dan simulasi komputasi (Rigetti Computing, 2019).

Quil adalah bahasa berbasis intruksi untuk merepresentasikan komputasi *quantum* dengan kontrol dan *feedback* secara klasik (Smith dkk., 2017). Quil dapat digunakan untuk pemrograman *quantum*, sebagai format perantara dalam program klasik, atau target *compiler* untuk bahasa pemrograman kuantum (Smih dkk., 2017). *Library* pyQuil diperlukan untuk mengeksekusi program Quil pada *platform* Rigetti Forest (Rigetti Computing, 2019). Selain itu, Rigetti QVM juga dapat digunakan untuk simulasi program Quil dan Rigetti Quil Compiler untuk meng-*compile* dan mengoptimasi program Quil (Rigetti Computing, 2019). Selain Rigetti, IBM juga menyediakan *platform* berbasis *cloud* untuk belajar atau riset terhadap komputer kuantum pada IBM Research Lab bernama IBM Q Experience (IBM Research and The IBM QX team, 2017).

Penelitian terkait implementasi algoritma kuantum Shor dan Grover dapat ditemukan pada Quantum Computing Playground, yang merupakan WebGL Chrome Experiment berbasis web (Quantum Computing Playground, 2014).

Quantum Computing Playground diciptakan oleh sekelompok *engineer* Google pada tahun 2014 (Quantum Computing Playground, 2014). Simulasi komputer kuantum juga dapat dilakukan pada Quantum Computing Playground ini dengan jumlah *qubit* antara 6 sampai 22 *qubit* (Quantum Computing Playground, 2014). *Scripting language* dan *compiler* yang digunakan pada Quantum Computing Playground ini adalah QScript (Quantum Computing Playground, 2014).

Grover pada (1996) memaparkan algoritma kuantum untuk pencarian data dengan kompleksitas waktu $O(\sqrt{N})$ pada *database* yang tidak terstruktur. Algoritma ini juga dikenal dengan algoritma kuantum Grover. Mutiara dan Refianti (2010) berhasil melakukan simulasi algoritma Grover pada komputer klasik menggunakan salah satu bahasa pemrograman kuantum, yaitu Quantum Computer Language (QCL) dengan jumlah *qubit* yang digunakan antara 4 sampai 14 *qubit*.

Pada penelitian ini digunakan Rigetti Forest SDK untuk implementasi algoritma kuantum Grover. Metode pengujian yang digunakan adalah metode *white box*. Bahasa pemrograman yang digunakan adalah Python. Rigetti Forest SDK memerlukan *library* pyQuil, Quantum Virtual Machine (QVM), dan Quil Compiler (Rigetti Computing, 2019). Tampilan antarmuka program dibuat menggunakan Flask. Untuk evaluasi terhadap performanya, akan dibandingkan dengan performa algoritma kuantum Grover pada Quantum Computing Playground dari sisi *user time*. Evaluasi dilakukan pada dua perangkat keras yang berbeda. Perbedaan perangkat keras seperti *core processor* mempengaruhi kinerja program yang dijalankan (Hermann, dkk., 2010).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dijelaskan sebelumnya, maka dirumuskan masalah sebagai berikut.

1. Bagaimana mengimplementasikan algoritma kuantum Grover untuk pencarian data menggunakan Rigetti Forest SDK?
2. Bagaimana performa dari implementasi algoritma kuantum Grover menggunakan Rigetti Forest SDK diukur dari sisi waktu eksekusi *user time*?

1.3 Batasan Masalah

Berdasarkan rumusan masalah dan untuk menjaga penelitian tidak keluar jalur, maka batasan masalah pada penelitian ini sebagai berikut.

1. Implementasi algoritma kuantum Grover diterapkan menggunakan *quantum virtual machine* (QVM) Rigetti dan tidak menggunakan komputer kuantum yang sebenarnya.
2. Performa implementasi algoritma kuantum Grover diukur dari sisi waktu eksekusi program *user time.*
3. Fokus pada penelitian ini tidak tertuju pada antarmuka program.
4. Implementasi algoritma kuantum Grover tidak menggunakan ancilla bits.
5. Solusi dari algoritma kuantum Grover yang diimplementasikan hanya untuk satu kali eksekusi.
6. Ukuran setiap data kurang dari sama dengan 10 *qubits* dan data bersifat unik.
7. Tidak digunakan untuk mencari bilangan negatif.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang sudah dijelaskan, tujuan dari penelitian ini sebagai berikut.

1. Mengimplementasikan algoritma kuantum Grover menggunakan Rigetti Forest SDK.
2. Mengetahui performa dari implementasi algoritma kuantum Grover menggunakan Rigetti Forest SDK dari sisi waktu eksekusi.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini sebagai berikut.

1. Mengetahui tentang komputasi kuantum dan potensinya, khususnya pada pencarian data menggunakan algoritma kuantum Grover.
2. Menambah referensi implementasi algoritma kuantum Grover untuk penelitian selanjutnya.
3. Mengetahui korelasi antara waktu eksekusi program *user time*, jumlah *core processor* dan *clock speed* yang digunakan, serta banyak data dan nilai terbesar data yang dapat dilakukan.

1.6 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam laporan skripsi ini adalah sebagai berikut.

BAB I PENDAHULUAN

Bab ini berisikan latar belakang pemilihan judul skripsi "Implementasi Algoritma Kuantum Grover Menggunakan Rigetti Forest SDK", rumusan

masalah, batasan penelitian, tujuan penelitian, manfaat penelitian, dan sistematika penulisan skripsi.

BAB II LANDASAN TEORI

Bab ini berisi dasar-dasar teori yang digunakan dalam penelitian terkait permasalahan yang dibahas. Teori-teori yang digunakan dalam penelitian ini antara lain *quantum bit*, *quantum gate,* algoritma kuantum Grover, Quantum Computing Playground, dan Rigetti Forest SDK.

BAB III METODOLOGI DAN PERANCANGAN PROGRAM

Bab ini berisi metodologi penelitian antara lain telaah literatur, perancangan dan implementasi, pengujian dan evaluasi, dokumentasi, serta konsultasi dan penulisan. Bab ini juga berisi perancangan program yang dijelaskan melalui *flowchart* dan rancangan tampilan antarmuka.

BAB IV IMPLEMENTASI DAN ANALISIS

Bab ini berisi hasil implementasi program yang dibuat dan diikuti uji coba program menggunakan metode *white box.* Kemudian evaluasi performa program dilakukan dari sisi waktu eksekusi *user time*.

BAB V SIMPULAN DAN SARAN

Bab ini berisi simpulan dari hasil uji coba dan evaluasi yang telah dilakukan dalam penelitian, beserta saran untuk pengembangan lebih lanjut.