Optimasi Komputasi menggunakan Algoritma Quantum Grover dan Keunggulannya dalam Pemecahan Permasalahan Pencarian

Royyan Abdullah Dzakiy / 13515123
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515123@std.stei.itb.ac.id

***Abstrak*—Di digital era ini, penggunaan komputer untuk melakukan komputasi telah menjadi suatu bagian yang mendasar pada hampir seluruh aspek kehidupan manusia. Permasalahan yang harus diselesaikanpun menjadi semakin besar dan kompleks. Kini semakin banyak metode yang dilakukan untuk mengoptimasi komputasi pencarian, salah satunya dengan menggunakan algoritma quantum. Algoritma quantum yang telah ditemukan memiliki fungsi berbeda serta keunggulannya masing-masing terhadap metode komputasi yang sudah ada. Dalam makalah ini akan diulas algoritma quantum Grover's Theory, serta pengaplikasiannya dalam penyelesaian masalah pencarian.**

***Kata Kunci*—Komputasi, Kompleksitas Algoritma, Algoritma Pencarian, Algoritma Quantum Grover**

I. PENDAHULUAN

Komputer merupakan alat bantu yang dipakai unutk mengolah data menurut perintah yang telah dirumuskan. Menurut KBBI, komputer berarti [n] alat elektronik otomatis yang dapat menghitung atau mengolah data secara cermat menurut yg diinstruksikan, dan memberikan hasil pengolahan, serta dapat menjalankan sistem multimedia (film, musik, televisi, faksimile, dsb), biasanya terdiri atas unit pemasukan, unit pengeluaran, unit penyimpanan, serta unit pengontrolan. Kata komputer semula dipergunakan untuk memggambarkan seseorang yangpekerjanannya melaukan perhitugnan aritkmatika, namun di kemudian hari arti kata ini bergeser menjadi suatu nama mesin tersendiri. Penggunaannya komputer kini sudah semakin meluas dan telah menjadi bagian integral dari kehidupan manusia. Komputer digunakan untuk mencari nilai data pada database suatu perguruan tinggi, juga digunakan untuk memprediksi keadaan cuaca yang akan datang. Semua permasalahan ini hadir dengan data terkait yang harus di olah dan dijadikan suatu output yang dapat dipahami maknanya. Kini permasalahannya adalah data yang terkumpul demikian banyak. Data yang banyak tersebut harus mampu untuk diolah dengan efektif dan efisien sehingga dapat menghasilkan pengetahuan yang dibutuhkan. Untuk mengatasi masalah tersebut, optimasi terhadap proses komputasi ini terus dilakukan, salah satunya adalah menemukan algoritma yang mampu mengolah data dengan lebih cepat. Penulis ingin mencoba membandingkan algoritma pencarian sequensial dan mengangkat salah satu algoritma quantum. Pada algoritma quantum, langkah penyelesaian masalah dan instruksi di implementasikan pada komputer quantum. Algoritma quantum memanfaatkan fitur-fitur dasar dari komputasi quantum seperti quantum superposition atau quantum entanglement. Menurut penelitian yang telah dilakukan, penggunaan fitur-fitur ini telah mampu mengoptimasi pemrosesan data, terutama pengolahan data besar. Pada makalah ini akan di paparkan beberapa algoritma quantum dan keunggulannya terhadap metode komputasi konvensional.

II. DASAR TEORI

2.1 Komputasi

Menurut ACM Computing Curricula 2005[1], komputasi didefinisikan sebagai berikut

"Secara umum, kita dapat mendefinisikan komputasi sebagai setiap kegiatan yang berorientasi pada tujuan yang menggunakan, manfaat dari, atau membuat komputer, Komputasi termasuk juga merancang dan membangun perangkat keras dan perangkat lunak sistem untuk berbagai tujuan, pengolahan, penataan, dan pengelolaan berbagai macam informasi; melakukan penelitian ilmiah dengan menggunakan komputer; membuat sistem komputer berperilaku cerdas, menciptakan dan menggunakan komunikasi dan media hiburan; menemukan dan mengumpulkan informasi yang relevan dengan tujuan tertentu, dan lain sebagainya. Daftar ini hampir tak ada habisnya, dan kemungkinan masih sangat luas."

2.2 Kemangkusan Algoritma

Algoritma yang ideal adalah algoritma yang efisien atau bahasa lainya adalah mangkus. Kemangkusan ini dapat dikur dengan meminimumkan kebutuhan waktu untuk

menjalankan algoritma dan ruang simpan yang dibutuhkan. Kebutuhan waktu dan ruang suatu algoritma berganutng pada seberapa besar ukuran masukan atau berapa jumlah data yang harus diproses. Ukuran yang dimaksud seringkali dilambangkan dengan n. Misalkan untuk bisa mengurutkan 500 elemen baris, maka n adalah 500; menghitung 3! maka n=3. waktu atau ruang yang dibutuhkan oleh suatu algoritma disebut sebagai fungsi dari n. BIla n menngkat, maka kebutuhan waktu dan ruang akan ikut meningkat. Seberapa besar peningkatan kebkutuhan itu sesuai dengan masukan yang diberikan menjunjukkan apakah suatu algirtma mangkus (efisien) atau tidak.

2.3 Kompleksitas Algoritma

Kompleksitas algoritma adalah seberapa cepat atau seberapa lamban suatu algoritma berjalan. Komopleksitas algorita di definisikan dengan suatu fungsi numeric T(n), yaitu waktu dibandingkan dnegan ukuran masukan n. Kita inign mendifiniskan waktu penyelesaian masalah oleh suatu algoritma tanpa memasukkan ukuarn detail implementasi, atau tanpa memperhitungkan faktor implementasi. Hal tersebut dikarenakan sebuah algoritma akan membutuhkan waktu yang berbeda dalam penyelesaian masalah dengan inputan yang sama jika memperhitungkan fakgtor implementasi seperti: keceptaan proseser, susunana instruksi, kecepatan disk, compiler brand, dsb. Cara untuk menghitung efisiensi (kemangkusan) algoritma adalah memperkirakan kemangkusannya secara asimptotis. Kita akan mampu mengukur waktu T(n) sebagai jumlah "langkah" elementer, pada waktu yang konstan.

Mengenai langkah elementer, kita definisikan dengan mengambil contoh penambahan dua bilangan bulat. Ditambahkan dua bilangan bulat digit dengan digit, dan ini akan menjadi penentu "langkah" dalam model komputasi kali ini. Oleh karena itu, dikatakan bahwa penambahan dua bilangan bulat n-bit mengambil n langkah. Akibatnya, total waktu komputasi adalah T (n) = c * n, di mana c adalah waktu yang dibutuhkan oleh penambahan dua bit. Pada komputer yang berbeda, penambahan dari dua bit mungkin membutuhkan waktu yang berbeda, jika c1 dan c2, adalah penambahan dari dua bilangan bulat n-bit dan membutuhkan T (n) = c1 * n dan T (n) = c2 * n masing-masing. Hal ini menunjukkan bahwa mesin yang berbeda menghasilkan waktu yang berbeda, tapi kali T (n) tumbuh secara linear seiring ukuran input meningkat.[7]

2.4 Notasi Asimptotik

Tujuan dari kompleksitas komputasi adalah untuk mengklasifikasikan algoritma berdasarkan performanya. Kita akan coba merepresentasikan waktu T(n) menggunakan notasi "big-O" untuk mengekspresikan kompleksitas suatu algoritma.

$$T(n) = O(n^2)$$

Notasi diatas menyatakan bahwa suatu algoritma memiliki kompleksitas quadrat.

Definisi "Big Oh"

Untuk setiap fungsi monoton f (n) dan g (n) dari suatu bilangan bulat positif sampai suatu bilangan bulat positif, kita mengatakan bahwa f (n) = O (g (n)) ketika terdapat konstanta c> 0 dan n0> 0 sehingga,

$$f(n) \leq c * g(n), \text{ for all } n \geq n_0$$

Secara intuitif, ini berarti bahwa fungsi f (n) tidak tumbuh lebih cepat dari g (n), atau fungsi g (n) adalah batas atas untuk f (n), untuk semua n cukup besar → ∞

Notasi "O besar" tidak dinyatakan simetris: n = O (n^2) tapi $n^2 \neq O(n)$.

Berikut adalah beberapa teori ukuran kompleksitas berdasarkan waktu.

- Waktu Konstan: O (1)

Sebuah algoritma dikatakan berjalan dalam waktu yang konstan jika membutuhkan jumlah waktu yang sama terlepas dari ukuran masukan. contoh:

Array: mengakses setiap elemen
Stack ukuran tetap: metode push dan pop
Queue ukuran tetap: metode enqueue dan dequeue

- Waktu Lanjar: O (n)

Sebuah algoritma dikatakan berjalan dalam waktu linear jika waktu pelaksanaan adalah berbanding lurus dengan ukuran input, yaitu waktu tumbuh secara linear sebagai ukuran input meningkat. contoh:

Array: pencarian linear, melintasi, menemukan minimum
ArrayList: berisi metode
antrian: berisi metode

- Waktu logaritmik: O (log n)

Sebuah algoritma dikatakan berjalan dalam waktu logaritmik jika eksekusi waktu sebanding dengan logaritma dari ukuran masukan. Contoh: pencarian biner

- Waktu kuadrat: O (n2)

Sebuah algoritma dikatakan berjalan dalam waktu logaritmik jika waktu pelaksanaan adalah sebanding

dengan kuadrat dari ukuran masukan. contoh: bubble sort, selection sort, insertion sort.

Definisi "Omega besar"

Kita perlu notasi untuk batas bawah. Sebuah notasi Ω omega modal digunakan dalam kasus ini. Dkatakan bahwa f (n) = Ω (g (n)) ketika terdapat konstan c yang f (n) ≥ c * g (n) untuk semua n yang cukup besar. contoh:

n = Ω (1)
n2 = Ω (n)
n2 = Ω (n log (n))
2 n + 1 = O (n)

Definisi "Theta besar"

Untuk mengukur kompleksitas algoritma tertentu, berarti untuk menemukan batas atas dan bawah. Sebuah notasi baru digunakan dalam kasus ini. Kami mengatakan bahwa f (n) = Θ (g (n)) jika dan hanya f (n) = O (g (n)) dan f (n) = Ω (g (n)). contoh:

2 n = Θ (n)
n2 + 2 n + 1 = Θ (n2)

Analisis Algoritma

Istilah Analisis algoritma digunakan untuk menggambarkan pendekatan untuk mempelajari kinerja algoritma. Dalam makalah ini kita akan melakukan jenis analisis berikut:

- *kasus kompleksitas runtime-terburuk (worst-case)* dari algoritma adalah fungsi yang didefinisikan dengan jumlah maksimum langkah yang diambil pada setiap contoh dari ukuran.
- *kasus kompleksitas runtime-terbaik (best-case)* dari algoritma adalah fungsi yang didefinisikan dengan jumlah minimum langkah yang diambil pada setiap contoh dari ukuran.
- *kasus kompleksitas runtime-rata-rata (average-case)* algoritma adalah fungsi yang didefinisikan oleh jumlah rata-rata langkah yang diambil pada setiap contoh dari ukuran.

Contoh: Berikut adalah suatu algoritma pencarian sekuensial dalam ukuran array.n.

kasus kompleksitas runtime-Terburuk adalah O (n)

kasus kompleksitas runtime-Terbaik adalah O (1)

Kasus kompleksitas runtime-Rata-rata adalah O (n / 2) = O (n)

2.5 Komputer

Komputer merupakan alat bantu yang dipakai unutk mengolah data menurut perintah yang telah dirumuskan. Menurut KBBI, komputer berarti [n] alat elektronik otomatis yg dapat menghitung atau mengolah data secara cermat menurut yg diinstruksikan, dan memberikan hasil pengolahan, serta dapat menjalankan sistem multimedia (film, musik, televisi, faksimile, dsb), biasanya terdiri atas unit pemasukan, unit pengeluaran, unit penyimpanan, serta unit pengontrolan. Kata komputer semula dipergunakan untuk memggambarkan seseorang yangpekerjanannya melaukan perhitugnan aritkmatika, namun di kemudian hari arti kata ini bergeser menjadi suatu nama mesin tersendiri. Asal mulanya, pengolahan informasi lebih terfokus pada permasalah aritmatika saja, namun kelak computer modern dipakai untuk berbagai tugas yang tidak langsung berhubungan denagn matematika.

Teknologi yang menyusun komputer sempat berubah pada periode tertentu. Alat komputasi pertama adalah abacus, yang diestimasikan diciptakan di Babylon pada 2400 SM. Penggunaan awalnya adalah untuk menuliskan garis di pasir menggunakan kerikil. Teknologi komputer terus berkembang dari awalnya menggunakan Kini teknologi komputer modern telah banyak menggunakan elektronik digital terutama dalam penyelesaian operasi boolean algebra.

Charles Babbage mendesain salah satu mesin hitung pertama uang idsebut mesin alaritikal.

Saat ini teknologi yang idpakai pada computer digital sudah berganti secara dramats sejak computer pertamapada tahun 1940-an yang masih menggunakan arsitektur Von Neumann. Arsitektor Von Neumann membagi computer kepada 4 bagian: Unit Aritmatika dan Logis (ALU), memori , dan unit control dan pranala input/output (I/O devices).

2.6 Komputer Quantum

Komputer quantum memiliki fungsi kerja yang mirip dengan komputer klasik, namun terbentuk dan memiliki cara kerja yang berbeda. Komputer klasik memiliki memori yang tersusun atas bit-bit. Tiap bit ini tersusun atas 0 dan 1. Komputer qunatum terdiri dari sekumpulan qubits. Sebuah qubit bisa merepresentasikan nilai satu atau nol, atau berada pada quantum superposition dari dua state tersebut.

Sepasang qubit akan mampu berada pada 4 state quantum superposition, dan tiga qubit pada 4 state quantum superposition. Secara umum sebuah komputer kuantum dengan n qubit akan bisa berada pada 2^n state berbeda dalam waktu yang sama. Sedangkan pada komputer biasa hanya 1 state dari tiap 2^n state itu dalam satu waktu.

Sebuah komputer quantum beroperasi dengan dengan mengolah qubit sehingga mampu merepresentasikan masalah yang dihadapi menggunakan quantum logic gates. Serangkaian langkah pengolahan dengan memanfaat quantum logic gates disebut dengan algoritma quantum. Pengolahan itu akan berakhir dengan pengukuran, yaitu menghentikan superposisi qubit menjadi salah satu dari 2^n state yang bersesuaian, dimana tiap qubit itu akan berubah menjadi satu atau nol, kembali pada state

klasiknya.

Algoritma quantum cenderung bersifat probabilistik, artinya hasil yang bisa didapatkan cenderung akan tepat hingga suatu batas probabilitas tertentu.

Contoh implementasi dari qubit pada quantum computer adalah penggunaan partikel dengan 2 spin state, yaitu "atas" dan "bawah" (biasa direpresentasikan dengan |1> dan |0>). Atau bisa juga direpresentasikan dengan menggunakan sistem lain.

Operasi pada Komputer Quantum

Suatu komputer dengan 3 quantum-bit state mampu menampilkan vektor dengan 8 dimensi. Pengolahan dari qubit cukup berbeda dibanding pengolahan bit biasa. Pertama-tama sistem harus diinisialisasi. Komputasi quantum kemudian akan menggunakan operasi dengan unitary matrices berupa rotasi. Oleh karena rotasi dapat diulang dengan merotasi ke arah sebaliknya, maka komputasi quantum dapat dibalik.

Terakhir, pada terminasi dari algoritma, hasilnya masih harus dibaca. Pada kasus komputer klasik, kita dapat mengambil suatu sampel distribusi probabilitas pada register 3 bit sehingga menjadi suatu string 3 bit, misalkan '000', ini menghentikan state quantum yang sedang dialami.

III. PEMBAHASAN

A. Pencarian dengan Algortima Searching Sekuensial

```c
#include <stdio.h>

void SeqSearch(int a[], int n, int x) {
    int i = 0;
    int found = 0;
    int iterasi = 0;
    while (i<n && !found) {
        if (a[i] == x) {
            found = 1;
        }
        else {
            i++;
            iterasi++;
            printf("iterasi ke-%d\n", iterasi);
        }
    }
    printf("Dilakukan iterasi sebanyak %d-kali\n", iterasi);
}
```

Gambar 3.a.1: Metode Searching menggunakan sekuensial search

Terdapat suatu permasalahan dimana ada 4 kartu yang salah satunya bergambar queen. Keempat kartu kemudian ditutup dan dicoba dicari yang mana dari 4 kartu tersebut merupakan kartu queen. Masalah tersebut dapat disimulasikan dengan adanya 2 bit yang merepresentasikan nilai yang diinginkan. Pada umumnya, permasalahan ini dapat diselesaikan dengan best case 1/4 peluang kali pengambilan kartu dengan tepat, sedangkan rata-rata pengambilan adalah 2/4 kali percobaan. Maka worstcase adalah harus dilakukan iterasi sebanyak n kasus, dalam hal ini 4.

```c
int main() {
    int i;
    int n=4;
    int a[n];
    // inisialisasi array
    for (i=0;i<n;i++) {
        a[i] = i;
    }

    // menjalankan search
    printf("cari angka 4 pada suatu array sebesar n=4\n");
    SeqSearch(a,n,4);
    return 0;
}
```

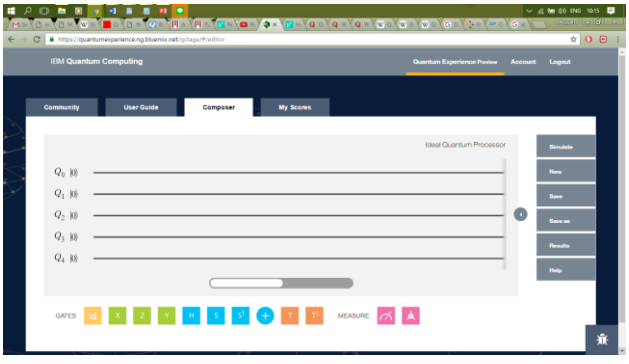
```
G:\Project Coding\C>tes
cari angka 4 pada suatu array sebesar n=4
iterasi ke-1
iterasi ke-2
iterasi ke-3
iterasi ke-4
dilakukan iterasi sebanyak 4-kali
```

Gambar 3.a.2: hasil implementasi menggunakan sequensial search

B. Implementasi Algoritma Grover untuk Searching

Pada impelementasi kali ini, penulis tidak mencoba menurunkan algoritma grover secara simbolis, tapi mencoba memanfaatkan suatu tool yant telah disediakan oleh IBM yaitu IBM Quantum Computing Composer. Tool ini dapat diakses online melalui laman (https://quantumexperience.ng.bluemix.net/qstage). Dengan ini penulis akan membandingkan hasil penjalanan algoritma grover dengan hasil algoritma search sekuensial.

Berikut adalah test-implementasi penerapan Algoritma Grover pada komputer quantum yang disediakan secara bebas oleh IBM Quantum Computing untuk digunakan secara online.



Gambar 3.b.1 : Composer Quantum Gate serta Elemen yang bisa dimanipulasi[7]

Pada sub-menu composer terdapat area untuk menyusun algoritma yang diinginkan, serta elemen yang dapat digunakan untuk menyusun algoritma.

Berikut adalah penjelasan singkat mengenai elemen yang disediakan pada composer.

Id: Identitiy Gate, digunakan untuk melakukan operasi idle kepada suatu qubit untuk waktu yang setara dengan durasi gate satu qubit

X: Pauli X gate adalah sebuah rotasi π disekitar X axis dan memiliki properti dimana X→X, Z→-Z. Juga direfer sebagai bit-flip.

Z: Pauli Z gate adalah gate untuk melakukan rotasi π disekitar Z axis dan memiliki properti X→−X, Z→Z. Juga disebut sebgai phase-flip.

Y: Pauli Y gate adalah gate untuk melakukan rotasi π disekitar Y axis dan memiliki properti X→−X, Z→-Z. Ini adalah bit-flip dan phase-flip yang memenuhi Y=XZ.

H: Hadamard gate adalah sebuah property yang memetakan X→Z, Z→X. Gate ini diperlukan untuk membuat superposisi.

S: Phase gate adalah √Z dan memiliki properti untuk memetakan X→Z, Z→X. Gate ini memperluas gate H untuk membuat superposisi kompleks.

S^t: Phase gate yang ditranspose konjugasikan terhadap S dan memiliki properti untuk memetakan X→-Y, Z→Z.

+: Controlled-NOT gate: sebuah gate 2 qubit yang membalikkan target qubit (menerapkan Pauli X) jika state kontrol adalah 1. Gate ini dibutuhkan untuk mengenerate entanglement.

T: Phase gate dimana √S sebagai rotasi π/4 disekitar Z axis. Gate ini dibutuhkan untuk kontrol universal.

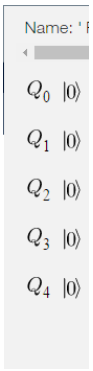
T^t: Phase gate yang di transpose konjugasi terhadap T.

Measure 1: Pengukur basis komputasi Z

Measure 2: Pengukur Bloch: Tomografi dari qubit

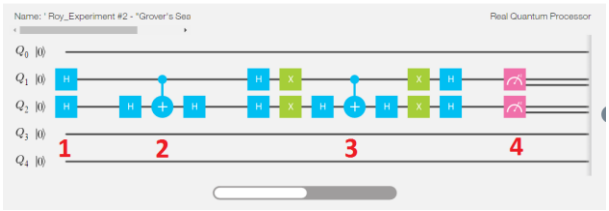
Berikut adalah simulasi grover algorithm yang dapat dilakukan pada simulator quantum computing pada IBM Quantum Computing. Algoritma yang dibentuk adalah sebagai berikut:

Gunakan tab composer untuk menyusun algoritma yang diinginkan



Gambar 3.b.2 : Qubit yang dapat diolah[7]

Susunan qubit yang bisa kita gunakan. Cukup gunakan 2 untuk persoalan kali ini.



Gambar 3.b.3 : Algoritma Grover yang telah disusun pada composer[7]

Berikut susunan quantum logic-gate yang diperlukan untuk membentuk algoritma searching grover.

bagian 1: menjadikan qubit berada pada bentuk superposisi

bagian 2: fungsi oracle yang berguna untuk menyembunyikan "queen" atau barang yang dicari dari 4 kemungkinan tempat, dan memosisikannya pada slot keempat, atau posisi '11'.

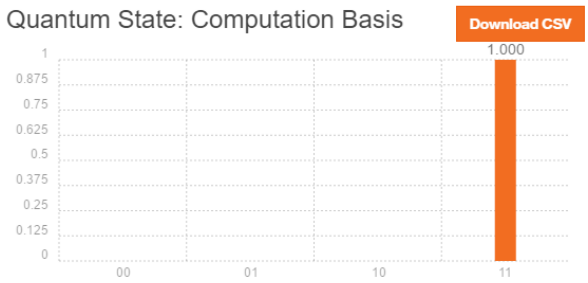
bagian 3: selanjutnya adalah step quantum inversion. Ia mengambil state entangled dari 2 qubit kemudian mengolahnya menjadi sesuatu yang dapat di ukur dan pahami.

bagian 4: pengamatan/pengukuran qubit setelah qubit keluar dari state superposisi.

Dengan algoritma yang telah disusun, maka program dijalankan dengan menggunakan Run. Maka algoritma yang telah disusun akan dijalankan pada prosesor quantum computer yang disediakan IBM. Dan hasilnya akan keluar setelah usai program dijalankan atau di proses.

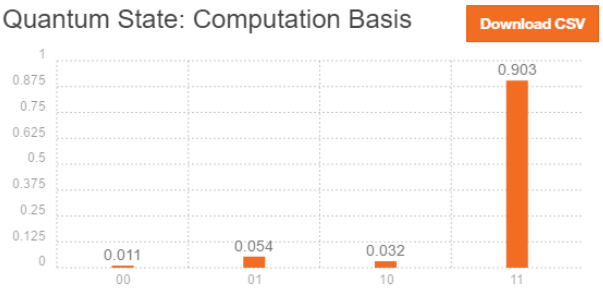
Gambar 3.b.4 : Algoritma masih di proses[7]

Gambar 3.b.5 : Algoritma selesai di proses[7]



Gambar 3.b.6 : Hasil pengolahan data setelah percobaan input 1 kali[7]

Berikut hasil dari algoritma berupa data dimana dengan sekali saja menjalankan algoritma langsung didapati hasil yang sesuai, yaitu didapati kartu yang bersesuaian ada pada slot '11' atau posisi keempat.



Gambar 3.b.7 : Hasil pengolahan data setelah input 1024 kali[7]

C. Analisis

Setelah melakukan percobaan dengan mengimplementasikan Algoritma Pencarian Grover dengan menggunakan composer Quantum Experience IBM, didapatkan hasil dimana yang dicari langsung ditemukan pada satu langkah saja, bahkan pada worstcase (untuk n=4). Sedangkan ketika dicoba lagi dengan mengulang percobaan sebanyak 1024 kali, didapatkan hasil pencarian tidak tepat 100%, namun sekitar 90%. Hal ini dapat disebabkan oleh hardware yang menjalankan algoritma terkait, yaitu komputer quantum yang menjalankannya. Hal ini sangat mungkin terjadi dilihat dari kompleksnya sistem di skala quantum serta kemungkinan miss akibat ketidakpastian yang ada.

Tentunya pada pengunaan quantum algorithm yang sebenarnya, daftar yang dicari akan berjumlah lebih dari empat item. Untuk melakukannya, algoritma mungkin mengulangi tiga operasi kuantum berkali-kali, menggeser sistem menuju state yang diinginkan setiap kali melewati loop. Apa yang membuat pencarian quantum begitu kuat adalah bahwa, untuk daftar N item, algoritma hanya membutuhkan sekitar akar kuadrat dari langkah N dibandingkan metode sequensialnya yang membutuhkan rata-rata N / 2 langkah dari pencarian. Sehingga komputer kuantum bisa melakukan pencarian pada buku telepon dengan satu juta nama hanya dengan 1.000 iterasi ketimbang 500.000 kali. Semakin panjang suatu daftar, semakin tampak keunggulan dari algoritma kuantum dibandingkan metode klasik.

Kelak dengan berkembangnya teknologi quantum computer, dapat dilakukan simulasi serupa dengan jumlah qubit lebih banyak sehingga didapati percepatan signifikan untuk penyelesaian masalah pencarian yang jauh lebih rumit dan banyak. Kelak penggunaan algoritma quantum lain juga akan terus berkembang dalam rangka penyelesaian masalah-masalah berbeda dan menjadi ujung tombak optimasi komputasi di masa depan.

IV. KESIMPULAN

Semakin kompleksnya permasalahan akibat banyaknya data yang harus diolah. Kini sudah tidak bisa lagi bergantung pada metode komputasi yang sudah ada

(komputer konvensional) dibutuhkan metode komputasi yang lebih cepat. Untuk kasus pencarian algoritma quantum Grover dapat mengoptimasi pemecahan masalah pencarian. Dalam menghadapi kasus yang berbeda tentu dibutuhkan metode yang berbeda pula dalam penyelesaiannya, maka algoritma quantum lainnya dapat digunakan seperti Grover's Algorithm, Deutsch's Algorithm, Shor's Algorithm, Quantum Factoring, dsb. Tentu penggunaan algoritma ini tak terlepas dari terobosan baru berupa teknologi komputer quantum dan kekuatannya dalam melakukan komputasi.

DAFTAR PUSTAKA

[1] James, Russell, "Computing Curricula 2005", ACM: United States of America, September 2015

[2] http://kamusbahasaindonesia.org/komputer, diakses pada 7 Desember 2016

[3] https://www.cs.cmu.edu/~adamchik/15-121/lectures/Algorithmic%20Complexity/complexity.html, Diakses pada 7 Desember 2016

[4] http://ffden-2.phys.uaf.edu/211.web.stuff/Almeida/visual.html, Diakses pada 7 Desember 2016

[5] https://www.quora.com/Quantum-Computation-How-do-you-explain-the-Deutsch-Jozsa-algorithm, Diakses pada 7 Desember 2016

[6] "Experience - IBM Research", https://www.youtube.com/watch?v=pYD6bvKLI_c - Running an experiment in the IBM Quantum , Diakses pada 7 Desember 2016

[7] https://quantumexperience.ng.bluemix.net/qstage, Diakses pada 7 Desember 2016

[8] http://www.tomshardware.com/reviews/upgrade-repair-pc,3000-2.html, Diakses pada 7 Desember 2016

[9] https://cryptome.org/qc-grover.htm, Diakses pada 7 Desember 2016

[10] Bennett C.H.; Bernstein E.; Brassard G.; Vazirani U. (1997). "The strengths and weaknesses of quantum computation". SIAM Journal on Computing.

[11] Vamanan Ramesh, "Quantum computing for big data analysis", Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya University, Maret 2015

[12] Munir Rinaldi, "Matematika Diskrit Rivisi Keenam", Informatika Bandung, September 2016

[13] David Deutsch and Richard Jozsa (1992). "Rapid solutions of problems by quantum computation". Proceedings of the Royal Society of London A

[14] Lanzagorta, Marco; Uhlmann, Jeffrey K. (2009-01-01). Quantum Computer Science. Morgan & Claypool Publishers

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Desember 2016

Royyan Abdullah Dzakiy / 13515123