Pengoptimalan Kompleksitas Waktu Komputasi dengan Menggunakan Sistem dan Algoritma dari Komputer Kuantum

Aditias Alif Mardiano - 13518039
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518039@std.stei.itb.ac.id

***Abstrak*—Pada zaman sekarang, dimana terjadinya informasi *overload*, teknologi yang dimiliki manusia, komputer klasik, tidak bisa memenuhi kebutuhan manusia akan teknologi yang dapat membantu perkembangan peradaban manusia. Komputer Kuantum dapat menjawab permasalahan dengan pemrosesan secara kuantum dengan pertumbuhan secara lanjar dan tidak eksponensial yang dapat memotong kompleksitas waktu yang dimiliki dari suatu pemrosesan. Algoritma kuantum memiliki adalah komputasi yang memproses data secara kuantum. Dalam makalah ini akan diulas Sistem dan Algoritma dari Komputer Kuantum.**

***Keywords*—Kombinatorial, Kuantum, Kompleksitas, Algoritma**

I. PENGENALAN

Perkembanhgan teknologi yang terjadi di dunia berkembang dengan pesat sejak kejadian Revolusi Industri. Pada zaman perang sejak munculnya konsep dari mesin untuk melakukan komputasi sebagai pemecah sandi dari pihak musuh yang merupakan invensi ke mesin-mesin untuk membantu pekerjaan manusia dan segala kebutuhan akan teknologi yang lebih maju, maka Komputer lahir sebagai salah satu jawaban dari permasalahan tersebut.

Komputer adalah salah satu alat komputasi atau alat hitung yang dapat melakukan pemrosesan aritmatika. Komputer adalah satu entitas yang terdiri dari komponen-komponen sederhana yang mampu mengubungkan komponen-komponen yang ada di dalamnya sehingga dapat menghasilkan informasi yang sudah diolah selama pemrosesan. Merepresentasikan data, memprosesnya, dan membuat mekanisme kontrol dapat dilakukan dengan mudah oleh komputer.

Dengan memanfaatkan komputer kita dapat melakukan hal yang simpel seperti perhitungan sederhana layaknya kalkulator, melakukan komunikasi dengan orang banyak, menyimpan data, media hiburan, mencari informasi, hingga proses yang sulit seperti mendaratkan astronot ke bulan. Hal-hal besar yang dapat dilakukan oleh komputer ini dapat membantu perkembangan teknologi lain sehingga dapat ikut berkembang dengan cepat untuk memudahkan pekerjaan manusia.

Kekuatan dari mesin komputer telah berkembang secara eksponensial, membuat komputer yang ada berkembang dengan pesat mengikuti zaman, dengan ukuran komputer yang makin kecil tetapi semakin kuat dalam waktu yang bersamaan. Namun dengan teknologi yang semakin berkembang, muncul juga masalah baru yang tidak dapat diselesaikan dengan teknologi konvensional seperi komputer klasik yang kita punyai sekarang. Perkembangan dari mesin komputer ini memuncak pada komputer kuantum, dengan teknologi mutakhir dan optimalnya, dapat membantu permasalahan kompleks yang ada. Dalam komputasi kuantum, algoritman kuantum adalah algoritman yang berjalan sesuai dengan model dari model komputasi kuantum, tidak seperti komputer klasik yang memproses satu demi satu, komoputer kuantum dapat memeceah tugas yang dijalankan dengan fitur-fitur seperti quantum superposition dan quantum entanglement.

II. DASAR TEORI

2.1. Kompleksitas Algoritma

a. Kemangkusan Algoritma

Algoritma yang bagus adalah algoritma yang mangkus. Kemangkusan algoritma diukur dari berapa jumlah waktu dan ruang (space) memori yang dibutuhkan untuk menjalankan. Algoritma yang mangkus ialah algoritma yang meminimumkan kebutuhan waktu dan ruang. Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan, yang secara khas adalah jumlah data, yang diproses. Ukuran masukan itu disimbolkan dengan n. Misalnya, bila mengurutkan 1000 buah elemen larik, maka n adalah 1000 menghitung 6! maka n = 6 clan lain-lain. Waktu/ruang yang dibutuhkan oleh algoritma dinyatakan sebagai fungsi dari n. Bila n meningkat, maka sumberdaya waktu/ruang yang dibutuhkan juga meningkat. Seberapa besar peningkatan sumberdaya itu menentukan apakah algoritmanya mangkus atau tidak.

b. Kompleksitas Waktu dan Ruang

Secara teoritis, model abstrak pen gukuran waktu/ruang harus independen dari pertimbangan mesin dan compiler apapun . Model abstrak seperti itu dapat dipakai untuk membandingkan algoritma yang berbeda. Besaran yang dipakai untuk menerangkan

model abstrak pengukuran waktu/ruang ini adalah kompleksitas algoritma. Ada dua macam kompleksitas algoritma, yaitu kompleksitas waktu dan kompleksitas ruang. Kompleksitas waktu diekspresikan sebagai jumlah tahapan komputasi yang dibutuhkan untuk menj alankan algoritma sebagai fungsi dari ukuran masukan n. Kompleksitas ruang diekspre sikan sebagai jumlah memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari uk:uran masukan n. Dengan menggunakan besaran kompleksitas waktu/ruang algoritma, kita dapat menentukan laju peningkatan waktu (ruang) yang diperlukan algoritma dengan meningkatnya ukuran masukan n.

c. Terminologi Kompleksitas Waktu dan Ruang

Terminologi yang diperlukan dalam membahas kompleksitas waktu dan kompleksitas ruang suatu algoritma adalah:

i. Ukuran besar masukan data untuk suatu algoritma, n.
ii. Kompleksitas waktu, T(n), adalah jumlah operasi yang dilakukan untuk melaksanakan algoritma sebagai fungsi dari ukuran masukan n.
iii. Kompleksitas ruang, S(n), adalah ruang memori yang dibutuhkan algoritma sebagai fungsi dari ukuran masukan n.

d. **Kompleksitas Waktu**

Pada algoritma pengurutan, operasi dasar adalah operasi perbandingan elemen-elemen larik dan operasi pertukaran elemen-elemen. Jadi, kita cukup menghitung berapa kali operasi perbandingan dan berapa kali operasi pertukaran elemen di dalam algoritma pengurutan. Kedua operasi dasar ini dihitung secara terpisah, karena di dalam algoritma pengurutan jumlah operasi perbandingan tidak sama dengan jumlah operasi pertukaran. Pada algoritma perkalian dua buah matriks A x B yang masing-masing berukuran n x n, operasi dasar yang bagus untuk dipilih adalah operasi penjumlahan dan perkalian. Jadi, kita cukup menghitung berapa kali operasi penjumlahan dan berapa kali operasi perkalian pada algoritma perkalian dua buah matriks. Kedua operasi dasar ini dihitung secara terpisah. Pada algoritma evaluasi polinom, $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$, operasi operasi dasar di dalam algoritmanya juga operasi penjumlahan dan perkalian. Jadi, kita cukup menghitung berapa kali operasi penjumlahan dan berapa kali operasi perkalian pada algoritma evaluasi polinom.
kompleksitas waktu dibedakan atas tiga
macam :

i. $T_{max}(n)$: kompleksitas waktu untuk kasus terburuk (*worst case*). yaitu kebutuhan waktu maksimum yang diperlukan sebuah algoritma sebagai fungsi dari n.

ii. $T_{min}(n)$: kompleksitas waktu untuk kasus terbaik (*best case*). yaitu kebutuhan waktu minimum yang diperlukan sebuah algoritma sebagai fungsi dari n.

iii. $T_{avg}(n)$: kompleksitas waktu unt:uk kasus rata-rata (*average case*) yaitu kebutuhan waktu rata-rata yang diperlukan algoritma sebagai fungsi dari n. Untuk kasus rata-rata ini. biasanya dibuat asumsi bahwa semua barisan masukan bersifat sama.

2.2. Kompleksitas Waktu Asimptotik

Dengan pemrosesan n data dengan n yang merupakan angka kecil maka akan susah untuk melihat dari perbedaan waktu pemrosesannya. Juga misal terlihat dari suatu persamaan kompleksitas waktu terburuk dari sebuah algoritma adalah $T(n) = 2n^2 + 6n + 1$, dimana jika n semakin bertumbuh, maka nilai dari 6n+1 semakin tidak akan berarti. Maka dari itu dibutuhkan suatu notasi yang dapat merepresentasikan kompleksitas waktu ini secara asimtotik

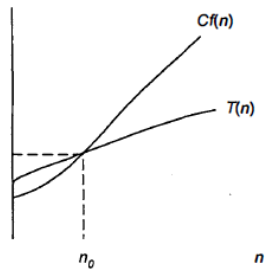
a. Notasi *O*-Besar

$T(n) = O(f(n))$ (dibaca "T(n) adalah $O(f(n))$" yang artinya $T(n)$ berorde paling besar $f(n)$ bila terdapat konstanta C dan n_o sedemikian, sehingga.

$$T(n) <= C \, . f(n)$$

untuk n >= n0.

Makna notasi *O*-Besar adalah jika sebuah algoritma mempunyai waktu asimptotik $O(f(n))$ maka jika n dibuat semakin besar, waktu yang dibutuhkannya tidak akan pemah melebihi suatu konstanta C dikali dengan $f(n)$. Jadi $f(n)$ adalah batas lebih atas (*upper bound*) dari T(n) untuk n yang besar. Kita katakan T(n) berorde palingbesar $f(n)$.



Gambar 2.2.a.1 : Ilustrasi "*O* besar"
(Sumber : Matematika Diskrit, Ed.3)

Gambar 2.2.a.1 memperlihatkan tafsiran geometri "*O* besar". Meskipun T(n) pada mulanya berada di atas $Cf(n)$, tapi setelah $n = n_0$ ia selalu berada di bawah $Cf(n)$. Contoh nyatanya adalah $T(n) = n^2 + 5n$, meskipun pada awalnya kurva berada di atas $2n^2$, tetapi untuk $n >= 5$

$$n^2 + 5n <= 2n^2$$

Karena itu kita mengambil $C = 2$ dan $n_0 = 5$ untuk memperlihatkan bahwa

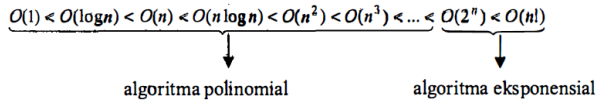
$$n^2 + 5n = O(n^2)$$

Dari definisi *O*-Besar jelas menuliskan $T(n) = O(f(n))$ tidak sama dengan $O(f(n)) = T(n)$. Lagi pula, tidaklah bermakna apa-apa menyatakan $O(f(n)) = T(n)$.. Penggunaan simbol "=" tidak menguntungkan karena simbol ini sudah umum menyatakan "kesamaan". Kebingungan yang timbul dari penggunaan simbol ini dapat dihindari dengan membaca simbol "=" sebagai "adalah" dan bukan "sama dengan". Untuk

menunjukkan bahwa $T(n) = O(f(n))$ kita hanya perlu menemukan pasangan C dan n_0 sedemikian sehingga $T(n) <= C(f(n))$. Tetapi, perlu diingat bahwa pasangan C dan no yang memenuhi definisi di atas tidak unik. Ada banyak C dan n_0 yang memenuhi definisi ini. Contoh-contoh berikut memperlihatkan cara memperoleh notasi kompleksitas asimptotik untuk bermacam-macam $T(n)$.

Tiap-tiap algoritma mempunyai kompleksitas waktu asimptotik masing-masing. Kompleksitas waktu asimptotik ini dapat digunakan untuk mengelompokkan algoritma sesuai dengan.

Urutan spektrum kompleksitas waktu algoritma adalah:



Kelompok Algoritma	Nama
$O(1)$	konstan
$O(\log n)$	logaritmik
$O(n)$	lanjar
$O(n \log n)$	$n \log n$
$O(n^2)$	kuadratik
$O(n^3)$	kubik
$O(2^n)$	eksponensial
$O(n!)$	faktorial

Tabel 2.2.a.1 : Kompleksitas Algoritma berdasarkan waktu asimptotiknya

b. Definisi " Ω 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)

c. 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)

2.3. Komputasi

Komputasi adalah kalkulasi suatu cara untuk menemukan solusi dari permasalahan menggunakan algoritma. Pengaplikasian dari komputasi sendiri sudah dimulai sejak masa lampau dimana masayarakat pada zaman dahulu menggunakan batu atau media tulis serta ingatan mereka sendiri untuk memikirkan suatu alogoritma dari pemecahan suatu masalah. Secara umum, komputasi adalah konsep pemahaman penyusunan model matematika dan teknik penyelesaian numerik serta penggunaan alat komputasi untuk menganalisis dan memecahkan permasalahan-permasalahan di bidang sains. Dalam ilmu alam, pendekatan ilmu komputasi dapat memberikan berbagai pemahaman baru, melalui penerapan model-model matematika dalam program komputer berdasarkan landasan teori yang telah berkembang, untuk menyelesaikan masalah-masalah nyata dalam ilmu tersebut.

2.4. Komputer Klasik

Menurut Robicomp, komputer didefinisikan sebagai berikut "Komputer merupakan seperangkat elektronik yang saling terkait dan dapat bekerja sama dalam penerimaan data (*input*), pengolahan data (proses) serta pemberian informasi (*output*) yang mana secara keseluruhan terkoordinasi menurut kontrol program yang tersimpan pada memorinya."

Komputer kelompok angkatan pertama berkembang pada tahun 1940 hingga 1959 yang bernama Electronic Numerical Integrator and Computer (ENIAC) yang dikembangkan oleh angkatan darat Amerika Serikat untuk kepentingan menghitung tabel tembakan senjata di Laboratorium Persenjataan. Kemampuan ENIAC saat itu hanya terbatas pada memecahkan satu masalah dalam satu waktu dan menggunakan bahasa yang hanya bisa dimengerti oleh mesin komputer.

Perbedaan dari Komputer klasik angkatan pertama dengan apa yang kita gunakan sekarang memiliki banyak perbedaan dari segi manfaat, mulai dari caranya merepresentasikan, mengolah, hingga mengontrol data dengan cara memanipulasi bits dalam biner yang ada di dalam komputer menggunakan komputasi.

Komputasi klasik adalah nama lain dari komputasi biner. Dengan pendekatan tradisional ini, infromasi disimpan dalam bentuk bits yang direpresentasikan secara logikal dengan 0 (mati) atau 1 (nyala) dengan kondisi dari suatu bit hanya 0 atau 1 tetapi tidak keduanya menyebabkan kalkulasi yang ditimbulkan tidak akan berubah tergantung *input* yang dimasukkan. Prosesor yang kita punya hari ini seperti x86 dan ARM processors adalah salah satu bentuk dari komputer klasik.

Tidak seperti komputer kuantum, komputer klasik menjalankan *task* dari suatu program secara teratur dan bekerja mengikuti hukum fisika klasik. Namun, dengan begitu untuk pemrosesan data yang sedikit dan tidak kompleks, komputer klasik dapat bekerja dengan baik tanpa harus dibandingkan dengan komputer kuantum. Namun untuk pencarian *database* dan permodelan dari suatu bentuk fisik yang kompleks akan memakan waktu yang panjang dan tidak akurat dikarenakan data yang diproses banyak dan komputer klasik akan mengecek satu-satu kemungkinan yang ada membuat proses ini tidak efisien.

2.5. Komputer Kuantum

Komputer kuantum adalah teknologi yang paling mutakhir dengan salah satu keuntungan dari komputer kuantum adalah Qubits atau Quantum Bits yang merupakan bits, tetapi perbedaanya terletak dari kondisi bits itu sendiri. Qubits dapat berada di dua kondisi yang sama di saat bersamaam dengan probabilitas yang tidak menentu.

Komputer kuantum menggunakan qubits atau quantum bits yang meruapakan bits yang bisa di atur nilainya menjadi satu yaitu 0 atau 1. Namun, dalam dunia kuantum qubit tidak perlu berada dalam salah satu kondisi tersebut, qubit dapat berada dalam kedua kondisi dalam satu waktu. Inilah yang

dinamakan superposisi. Namun seketika kita mengukur nilainya, superposisi runtuh dan nilainya akan muncul tergantung kemungkinan dari kondisi qubit itu sendiri. Bayangkan seperti koin yang diputar, kita tidak dapat menentukan kondisi dari koin itu kepala atau ekor, lalu saat diberhentikan secara paksa, kita dapat mengecek apakah koin dalam kondisi kepala atau ekor.

Keuntungan yang dimiliki dari superposisi ini berada pada kasus perhitungan bit. Kombinasi dari satu bit sendiri adalah 2 sehingga kombinasi dari *n*-bit dapat direpresentasikan dengan 2^n. Jika kita mempunyai 4 bit klasik yang memiliki 16 macam kombinasi dan dari 16 kombinasi tersebut kita hanya butuh salah satunya. 4 qubit yang dalam superposisi dapat berada dalam semua 16 kemungkinan tersebut dalam satu waktu, membuat kombinasi yang dilakukan dari *n*-qubit dapat direpresentasikan dengan *n*.

Dengan pertumbuhan secara eksponensial, kompleksitas waktu akan semakin terpotong memungkinkan untuk waktu pemrosesan yang cepat dan efisien.

Salah satu kelebihan dari komputer kuantum sendiri adalah quantum entanglement atau belitan kuantum. Quantum Entanglement adalah koneksi dari masing-masing qubit yang berekasi dengan perubahan kondisi yang terjadi dari qubit lainnya yang dilakukan secara instan, tidak peduli seberapa jauh jaraknya. Dengan artian, satu qubit yang ter-*entangle* satu sama lain dapat menentukan kondisi dari pasangannya tanpa perlu dicek.

Cara kerja dari komputer kuantum untuk mengecek nilai dari kombinasi qubit berbeda dengan gerbang logika normal yang mengkonfigurasi input dan menghasilkan satu output yang pasti, gerbang kuantum memanipulasi input dari superposisi, memutar probabilitas, dan memproduksi superposisi lain sebagai outputnya.

Komputer quantum bekerja dari mengkonfigurasi qubit, menyetel gerbang kuantum, membelitkannya, memanipulasi probabilitasnya, dan terakhir menghitung keluarannya, meruntuhkan semua superposisi dan mengubahnya menjadi 0 atau 1.

Jadi walaupun komputer kuantum tidak akan menggantikan komputer klasik untuk kerja yang simpel, dalam hal lain, Kompuer kuantum akan memberikan dampak perubahan yang besar pada pencarian database dengan algoritma khususnya sehingga membutuhkan akar dari waktu pemrosesan pada komputer klasik.

Namun, dengan teknologi ini, Keamanan IT dapat terganggu. Dengan waktu pemrosesan yang singkat, Dekripsi dari email, data Bank, dan kata sandi dapat dengan mudah dilakukan.

Simulasi juga dapat dilakukan dengan mudah dengan komputer kuantum, seperti simulasi molekul, karena dengan menggunakan komputer klasik simulasi ini akan memakan waktu yang lama dan tidak akurat, hal ini tentu akan membantu perkembangan di dunia pengobatan.

2.6. Algoritma Kuantum

Dalam komputasi kuantum, algoritma kuantum adalah algoritma yang berjalan pada model realistis perhitungan kuantum, model yang paling umum digunakan adalah model komputasi rangkaian kuantum. Algoritma klasik (atau non-kuantum) memiliki urutan instruksi yang terbatas, atau prosedur langkah demi langkah untuk menyelesaikan masalah, di mana setiap langkah atau instruksi dapat dilakukan pada komputer klasik. Demikian pula, algoritma kuantum adalah prosedur langkah-demi-langkah, setiap langkah dapat dilakukan pada komputer kuantum. Algoritma kuantum biasanya digunakan untuk algoritme yang tampaknya inheren kuantum, atau menggunakan beberapa fitur penting dari perhitungan kuantum seperti superposisi kuantum atau keterikatan kuantum.

Algoritma kuantum yang ada diantaranya yaitu :

- Shor's Algorithm

  Algoritma Shor adalah algoritma kuantum untuk memfaktorkan angka N dalam ruang O ((log N) 3) dan ruang O (log N), dinamai menurut Peter Shor.

  Algoritma ini penting karena menyiratkan bahwa kriptografi kunci publik mungkin mudah rusak, mengingat komputer kuantum yang cukup besar. RSA, misalnya, menggunakan kunci publik N yang merupakan produk dari dua bilangan prima besar. Salah satu cara untuk memecahkan enkripsi RSA adalah dengan memfaktorkan N, tetapi dengan algoritma klasik, anjak piutang menjadi semakin memakan waktu ketika N bertambah besar; lebih khusus, tidak ada algoritma klasik yang diketahui yang dapat memperhitungkan waktu O ((log N) k) untuk setiap k. Sebaliknya, algoritma Shor dapat memecahkan RSA dalam waktu polinomial. Itu juga telah diperluas untuk menyerang banyak cryptosystem kunci publik lainnya.

  Seperti semua algoritma komputer kuantum, algoritma Shor adalah probabilistik: memberikan jawaban yang benar dengan probabilitas tinggi, dan probabilitas kegagalan dapat dikurangi dengan mengulangi algoritma.

  Algoritma Shor didemonstrasikan pada tahun 2001 oleh sebuah kelompok di IBM, yang memasukkan 15 menjadi 3 dan 5, menggunakan komputer kuantum dengan 7 qubit.

- Grover Algorithm

  Algoritma Grover adalah algoritma kuantum yang menemukan dengan probabilitas tinggi input unik ke fungsi kotak hitam yang menghasilkan nilai output tertentu, hanya menggunakan $O(\sqrt{N})$ evaluasi fungsi, di mana N adalah ukuran domain fungsi.

  Masalah analog dalam perhitungan klasik tidak dapat diselesaikan dalam kurang dari $O(N)$ evaluasi (karena, dalam kasus terburuk, urutan ke-N anggota domain mungkin benar anggota). Pada waktu yang hampir bersamaan ketika Grover menerbitkan algoritmanya, Bennett, Bernstein, Brassard, dan Vazirani membuktikan bahwa setiap solusi kuantum untuk masalah perlu mengevaluasi fungsi $\Omega(\sqrt{N})$ kali, jadi algoritme Grover bekerja optimal secara asimptotik.

  Telah ditunjukkan bahwa komputer kuantum variabel tersembunyi non-lokal dapat mengimplementasikan pencarian database sebanyak N paling banyak $O(\sqrt[3]{N})$ langkah-langkah. Ini lebih cepat daripada langkah $O(\sqrt{N})$ yang diambil oleh algoritma Grover. Tidak ada metode pencarian yang

memungkinkan komputer kuantum untuk menyelesaikan masalah NP-Complete dalam waktu polinomial.

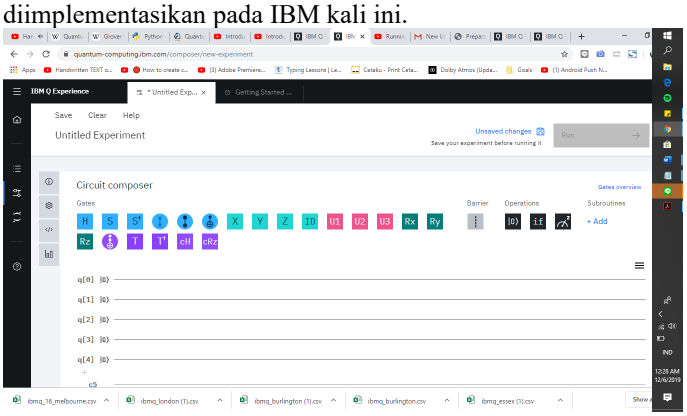
Tidak seperti algoritma kuantum lainnya, yang dapat memberikan peningkatan eksponensial atas rekan-rekan klasiknya, algoritma Grover hanya menyediakan peningkatan kuadratik. Namun, bahkan percepatan kuadrat cukup besar ketika *N* besar. Algoritma Grover dapat memaksa kunci kriptografi simetris 128-bit dalam sekitar 264 iterasi, atau kunci 256-bit dalam sekitar 2128 iterasi. Akibatnya, kadang-kadang panjang kunci simetris digandakan untuk melindungi terhadap serangan kuantum yang akan datang.

Seperti banyak algoritma kuantum, algoritma Grover adalah probabilistik dalam arti bahwa ia memberikan jawaban yang benar dengan probabilitas kurang dari 1. Meskipun secara teknis tidak ada batas atas jumlah pengulangan yang mungkin diperlukan sebelum jawaban yang benar diperoleh, jumlah pengulangan yang diharapkan adalah faktor konstan yang tidak tumbuh dengan *N*. Makalah asli Grover menggambarkan algoritma sebagai algoritma pencarian basis data, dan deskripsi ini masih umum. Basis data dalam analogi ini adalah tabel dari semua output fungsi, diindeks oleh input yang sesuai.

III. PEMBAHASAN

A. Percobaan Penggunaan IBM Quantum Computing Composer dalam menguji Algoritma Kuantum

Implementasi kali ini menggunakan IBM Quantum Computing yang dapat diakses pada https://quantum-computing.ibm.com/, penulis tidak menunjukkan kerja langsung dari komputer kuantum karena penulis tidak mempunyai akses untuk komputer kuantum itu sendiri, maka dari itu Algoritma kuantum diimplementasikan pada IBM kali ini.



Kali ini saya cantumkan percobaan yang dilakukan pada website tersebut dengan menggunakan Grover's Algorithm yang dilakukan oleh Royyan A Dzakiy.

Gambar 3.a.1 : Composer Sirkuit Kuantum pada serta Elemen yang bisa dimanipulasi

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 $\sqrt{Z}$ dan memiliki properti untuk memetakan X→Z, Z→X. Gate ini memperluas gate H untuk membuat superposisi kompleks.

St: 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 $\sqrt{S}$ sebagai rotasi $\pi/4$ disekitar Z axis. Gate ini dibutuhkan untuk kontrol universal.

Tt: Phase gate yang di transpose konjugasi terhadap T.

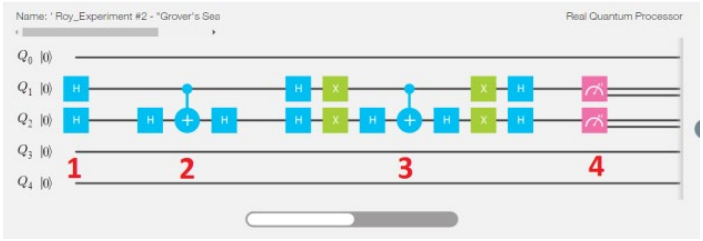
Measure 1: Pengukur basis komputasi Z

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

Susunan qubit yang bisa kita gunakan. Untuk persoalan Grover's Algorithm



Gambar 3.a.2 : Algoritma Grover yang telah disusun pada composer

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

bagian 1: menjadikan qubit berada pada bentuk superposisi

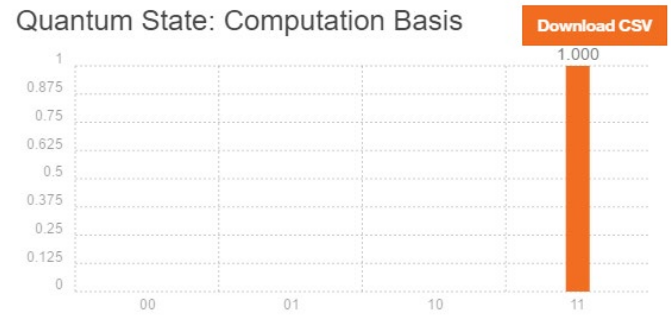
bagian 2: fungsi oracle yang berguna untuk menyembunyikan "queen" atau barang yang dicari dari 4 kemungkinan tempat, dan memosisikannya pada slot keempat, atau posisi '11'.

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.

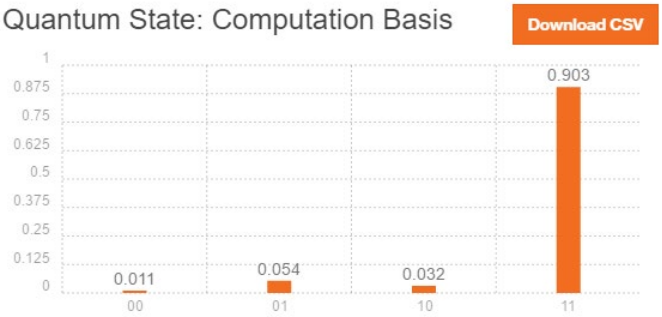
Dengan algoritma yang telah disusun, maka program dijalankan dengan menggunakan Run. Maka algoritma yang telah disusun akan dijalankan pada prosesor quantum computer yang disediakan IBM. Dan hasilnya akan keluar setelah usai program dijalankan atau di proses.

Gambar 3.b.3 : Algoritma selesai di proses



Gambar 3.b.4 : Hasil pengolahan data setelah percobaan input 1 kali

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.5 : Hasil pengolahan data setelah input 1024 kali

B. Analisis

Percobaan yang dialkukan dengan menggunaan mengimplementasikan Grover's Algorithm 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 disebabkan oleh kesalahan operator yang ada di dalam mesin tersebut, bukan kesalahan kalkulasi atau akurasi. Hal tersebut sangat mungkin terjadi mengingat komputer kuantum yang memiliki struktur dan arsitektur yang kompleks sehingga akan ada kemungkinan galat di perangkat kerasnya.

Algoritma kuantum ini sangat kuat seperti yang sudah dijelaskan pada bab II, karena kuantum menggunakan qubit yang memotong kompleksitas waktu yang merupakan eksponensial (2^n) menjadi lanjar(n). Sehingga komputer kuantum bisa melakukan pencarian pada buku telepon dengan satu juta nama hanya dengan 1.000 iterasi ketimbang 2^{1000} kali.

IV. KESIMPULAN

Dengan zaman sekarang yang merupakan zaman data overload, dibutuhkan suatu sistem yang dapat memproses dengan efisien dan cepat serta akurat data-data tersebut. Komputasi klasik yang ada sekarang tidaklah cukup untuk menghadapi masa depan yang akan menanti peradaban manusia kelak.

Komputer Kuantum dapat dengan algoritmanya seperti Grover's Algotihm dan Shor's Algorithm dapat menjawab permasalahan manusia seperi dekripsi dan perbaharuan sistem keamanan IT, juga dengan komputer kuantum kita dapat melakukan permodelan molekul yang dapat membantu dunia pengobatan, dan juga membantu memahami isi dari dunia itu sendiri.

REFERENSI

[1] https://www.maxmanroe.com/vid/teknologi/komputer/pengertian-komputer.html, diakses pada 4 Desember 2019
[2] https://www.robicomp.com/perbandingan-komputer-klasik-dan-komputer-kuantum.html, diakses pada 4 Desember 2019
[3] https://whatis.techtarget.com/definition/classical-computing
[4] http://www.klientsolutech.com/uses-of-computer-in-our-daily-life/, diakses pada 4 Desember 2019
[5] https://plato.stanford.edu/entries/computation-physicalsystems/, diakses pada 4 Desember 2019
[6] https://salamadian.com/fungsi-komponen-pengertian-komputer/, diakses pada 4 Desember 2019
[7] http://www.opi.lipi.go.id/situs/mki/, diakses pada 4 Desember 2019
[8] http://www.opi.lipi.go.id/situs/gfti/, diakses pada 4 Desember 2019

[9] http://www.klientsolutech.com/uses-of-computer-in-our-daily-life/, diakses pada 4 Desember 2019
[10] https://whatis.techtarget.com/definition/classical-computing, diakses pada 4 Desember 2019
[11] https://www.sciencedaily.com/terms/quantum_computer.html, diakses pada 4 Desember 2019
[12] https://www.quantiki.org/wiki/shors-factoring-algorithm. O. Young, diakses pada 4 Desember 2019
[13] https://quantum-computing.ibm.com/composer/new-experiment, diakses pada 5 Desember 2019
[14] Munir Rinaldi, "Matematika Diskrit, Ed 3", Informatika Bandung, 2010
[15] Dzakiy, A Royyan, “Optimasi Komputasi menggunakan Algoritma Quantum Grover dan Keunggulannya dalam Pemecahan Permasalahan Pencarian”, Informatika Bandung 2016

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, 6 Desember 2019

)

Aditias Alif Mardiano | 13518039