Analisis Kompleksitas Algoritma dalam Generasi Dunia di *Minecraft*

Christophorus Dharma Winata - 13521009
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13521009@std.stei.itb.ac.id

***Abstract*—Pada makalah ini akan dibahas tentang apa itu kompleksitas algoritma, bagaimana menotasikan kompleksitas dari suatu algoritma, contoh dari kompleksitas algoritma dalam kehidupan sehari-hari dan dalam program Minecraft, Perlin Noise serta implementasinya dalam game Minecraft.**

***Keywords*—Minecraft, Kompleksitas, Algoritma, Perlin**

I. PENDAHULUAN

Pada suatu program seperti video gim, data-data yang ada pada game tersebut akan disimpan ke dalam penyimpanan lokal pada komputer. Bisa terlihat pada gim yang ada di zaman sekarang, mayoritas dari program yang ada menghabiskan banyak ruang dan memakan banyak waktu dalam pemrosesannya. Ditambah dengan sebagaimana maraknya gim video di masa sekarang, semakin banyak orang yang frustasi dan terganggu dengan semakin besarnya ukuran-ukuran game yang ada sekarang.

Video game, terutama game online sudah menjadi menjadi bagian yang erat dalam kehidupan sehari-hari kita, terutama untuk anak-anak muda. Ditambah dengan semakin majunya teknologi video game sudah menjadi hal yang terjangkau bagi mayoritas orang.

Namun ditemukan lagi masalah yang baru. Jika ukuran game semakin besar, seberapa besar ukuran game di masa depan nanti? Untuk menjalankan program, kebutuhan ruang dan waktu prosesnya semakin lama semakin tinggi sehingga semakin berat untuk membuat suatu dengan kualitas yang baik secara konsumsi dan baik juga dalam aspek operasional.

Disini akan diberi hasil analisis bahwa untuk memberikan suatu game yang berkualitas baik dapat dilakukan tanpa menuntut banyak kapasitas memori dan waktu yang tinggi, contoh dari game tersebut adalah Minecraft.

II. LANDASAN TEORI

A. Algoritma

Algoritma adalah sekumpulan instruksi dalam menyelesaikan suatu tugas. Sebelumnya algoritma terdengar seperti hal yang kompleks, tetapi pada kenyataannya ia adalah suatu hal yang sederhana, bahkan sesederhana makan dan minum.

Di kehidupan sehari-hari, algoritma yang setiap hari kita terapkan tidak selalu berupa langkah-langkah yang efisien. Contohnya dalam hal memakan makanan, secara teknis makan dengan tangan masih memenuhi tujuan proses makan makanan, namun jika dibandingkan dengan kita menggunakan sendok dan garpu saat makan, proses yang dilakukan akan menjadi jauh lebih efisien. Sama halnya dengan program, suatu algoritma dapat menyelesaikan tugasnya dengan berbagai macam cara, namun cara yang satu dengan yang lain dapat menghasilkan efisiensi yang berbeda, dan disitulah dimana kompleksitas dari suatu algoritma harus diperhatikan.

```c
int indexOf(List l, ElType val)
/* I.S. l, val terdefinisi */
/* F.S. Mencari apakah ada elemen list l yang bernilai val */
/* Jika ada, mengembalikan indeks elemen pertama l yang bernilai val */
/* Mengembalikan IDX_UNDEF jika tidak ditemukan */
{
    List p = l;
    boolean found = false;
    int idx = 0;
    while (p!=NULL && !found) {
        if (INFO(p) == val) {
            found = true;
        } else {
            p = NEXT(p);
            idx++;
        }
    }
    if (found) {
        return idx;
    } else {
        return IDX_UNDEF;
    }
}
```

Gambar 2.1. Algoritma dalam mencari index list pada bahasa C

B. Kompleksitas algoritma

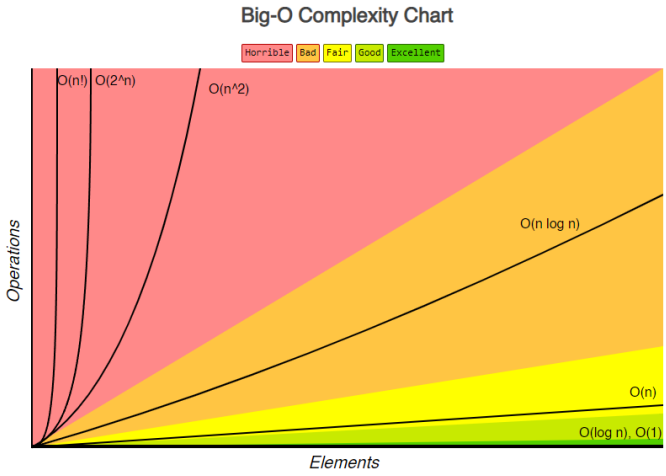
Pada suatu algoritma, algoritma tersebut selain benar juga harus efisien dalam menjalankan tugasnya. Algoritma dinilai baik ketika ia dapat mengeluarkan hasil yang benar dan juga efisien dalam prosesnya. Untuk menentukan efisiensi suatu algoritma kita dapat mengukur jumlah ruang dan waktu yang diperlukan dalam menjalankannya. Kita tidak bisa hanya menyebut suatu algoritma memiliki kompleksitas 10 langkah atau 300 langkah. Hal ini disebabkan suatu algoritma dapat berubah kecepatannya bergantung pada banyak memori yang dibutuhkan dalam menjalankan algoritma (atau ruang yang dipakai), perangkat yang menjalankan algoritma tersebut, dan data yang menjadi masukan dari algoritma tersebut.

Maka dari itu, cara standar untuk menetapkan suatu notasi kompleksitas algoritma adalah dengan notasi *Big O* atau umumnya disebut dengan *Big O Notation.* Notasi ini

melambangkan suatu kompleksitas algoritma dengan suatu notasi fungsi seperti O(n) untuk menjelaskan suatu kompleksitas algoritma yang menerima masukan data sebanyak n buah lalu melakukan langkah algoritma sebanyak n kali. Contoh lainnya, pada algoritma dengan kompleksitas $O(n^2)$, algoritma tersebut menerima masukan data sebanyak n buah, lalu melakukan langkah sebanyak n^2 kali untuk menyelesaikan tugasnya.

Dengan telah mengetahui notasi *big O*, kita dapat memperkirakan suatu kompleksitas dari suatu algoritma. Sebelum itu, dalam pembahasan kompleksitas algoritma, seharusnya ada notasi lain selain notasi *big O* untuk menggambarkan kompleksitas dari suatu algoritma. Notasi yang dimaksud yaitu notasi *big* Ω, umumnya disebut *big Omega*, yang merepresentasikan batas bawah dari kompleksitas suatu algoritma. Sedangkan jika dibandingkan dengan notasi big O, notasi big O menggambarkan batas atas dari suatu algoritma.

Yang dimaksud dari batas atas dan batas bawah adalah dalam sekumpulan data, ada kemungkinan data-data tersebut akan sudah dalam bentuk akhir setelah melewati algoritma atau sudah mendekati bentuk akhir tersebut, sehingga saat program harus menjalankan algoritma, lama pemrosesannya bisa lebih cepat dari biasanya. Hal ini umumnya disebut *best case*, atau kasus terbaik bagi suatu algoritma dimana performanya adalah performa yang terbaik dari semua kemungkinan yang ada, inilah yang dimaksud batas bawah suatu algoritma. Sebaliknya ada juga saat dimana susunan data masukan bersifat paling merugikan bagi algoritma. Kasus disebut sebagai *worst case* atau kasus terburuk dimana performa algoritma adalah yang terburuk dari kasus-kasus yang lain, ini yang merupakan batas atas algoritma. Selain kasus terburuk dan kasus terbaik, terdapat kasus lain yaitu kasus rata-rata, atau disebut *average case*. Kasus ini adalah saat data-data yang diberikan memiliki frekuensi dan pola-pola yang tidak seburuk kasus terburuk, ataupun sebaik kasus terbaik, dan pola data serta lama proses algoritmanya terdapat pada kisaran yang tidak jauh berbeda dengan kasus lain pada umumnya.



Gambar 2.2. Bagan representasi kompleksitas big-O

Sebagai pedoman singkat, pada notasi big O atau big Omega, semakin tinggi n akan merepresentasikan langkah algoritma yang semakin banyak, jadi algoritma yang semakin tinggi nilai n nya akan semakin lambat.

C. Minecraft

Gambar 2.3. Minecraft

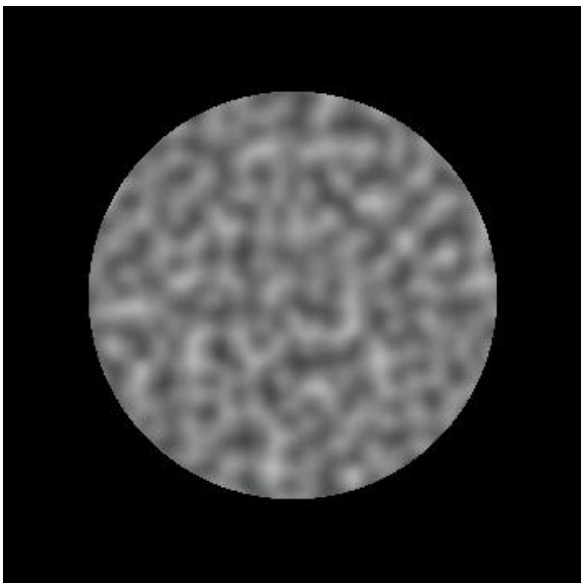
Minecraft adalah video gim yang dikembangkan dan dipublikasikan oleh Mojang. Minecraft pertama kali dipublikasi pada bulan Mei tahun 2009, dengan nama penciptanya yaitu Markus Persson. Pada tahun 2011, Mojang digabungkan dengan Microsoft, perusahaan teknologi multinasional yang bekerja di bidang perangkat lunak komputer, dan elektronika. Gim ini berupa gim sandbox *survival* yang dunia gamenya terbuat dari balok-balok yang bisa dihancurkan atau dipasang sesuai keinginan pemainnya.

Minecraft memiliki beberapa jenis mode permainan, beberapa contohnya adalah mode pertahanan hidup (*survival*) dan mode kreatif. Mode survival adalah mode dimana pemain ditempatkan pada suatu dunia dan mereka harus bertahan hidup dari ancaman musuh-musuh yang ada di dunia, membangun suatu peradaban, dan menjelajah berbagai macam tempat yang ada di dunia tersebut. Mode kreatif adalah mode lain di Minecraft dimana pemain dapat menggunakan segala jenis balok (atau *block*) yang ada tanpa ada batasan apapun, selain kreativitas masing-masing pemain, mode digunakan untuk menyediakan media sepenuhnya pada gim sebagai tempat pemain mencurahkan segala kreativitasnya seperti membangun bangunan, dan melakukan eksperimen-eksperimen pribadi.

Minecraft adalah permainan yang bersifat *sandbox* yang artinya tidak ada tujuan yang ditentukan pada saat memainkan permainan tersebut. Tujuan yang dibuat adalah dari pemain-pemain sendiri, sehingga mendorong kreativitas dari pemain-pemain.

Gim ini berjalan dengan bahasa pemrograman *Java*, dan pada saat menjalankan dunia, Minecraft akan menjalankan suatu algoritma yang dapat menghasilkan dunia secara procedural tanpa harus menyimpan banyak data yang berisi desain level-level pada storage mesin komputer.

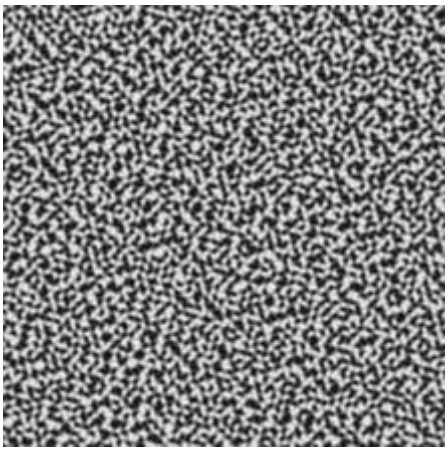
D. Perlin dan Simplex Noise



Gambar 2.4. Perlin Noise

Perlin Noise adalah algoritma yang didesain oleh Ken Perlin pada tahun 1983, pada saat ia sedang mengembangkan film *sci-fi Tron* pada tahun 1982. Ia menuliskan penemuannya pada paper yang ia buat pada tahun 1985 yang berjudul *An Image Synthesizer*.Pada saat itu ia tidak menetapkan paten apapun terhadap algoritma buatannya, tetapi pada tahun 2001 ia diberikan hak paten untuk implementasi algoritma *simplex noise* pada dimensi tiga dan ke atas.

Cara kerja algoritma tersebut secara garis besar adalah pada awalnya kita mendefinisikan suatu dimensi n dalam suatu petak-petak yang berisi vektor dengan gradien yang saling acak, lalu memerhitungkan dot product dari gradien tersebut dengan samping-sampingnya, serta memerhatikan interpolasi antara kedua value.



Gambar 2.5. Simplex Noise

Simplex noise juga merupakan fungsi yang menghasilkan noise mirip dengan Perlin noise, namun dengan data yang lebih efisien dan pada dimensi yang tinggi, memiliki kebutuhan komputasional yang lebih rendah. Dibandingkan dengan Perlin Noise yang memroses vektor-vektor menjadi suatu *hypercube* dengan n dimensi, Simplex Noise memroses vektor-vektor menjadi simplex-simplex dimana ia berdimensi n + 1 pada data berdimensi n. Kompleksitas dari Simplex Noise adalah $O(n^2)$ dibandingkan dengan Perlin Noise adalah $O(2^n)$. Namun, karena Simplex noise juga menyimpan data yang lebih sedikit, maka pada saat dibutuhkan data yang lebih banyak, data yang tersedia tidak cukup padat.

III. HASIL PENELITIAN

Pada suatu gim video, umumnya suatu level yang sudah didesain oleh pengembang gim akan disimpan dalam tempat penyimpanan lokal pada mesin komputer pemain. Hal ini tidak berlaku pada Minecraft, seperti yang dikatakan Henrik Kniberg, mantan desainer/developer di Mojang, pada pembaharuan gim Minecraft bulan Februari lalu, bahwa jika dibandingkan dengan planet Bumi yang berupa seperti bola dengan luas permukaan sekitar 0.5 milyar km^2, Minecraft adalah suatu persegi datar dengan luas sebesar 3.6 milyar km^2. Selain itu, setiap seorang pemain Minecraft hendak memulai suatu dunia, mereka akan selalu ditempatkan di dunia yang berbeda. Hal ini tidak dapat dilakukan jika setiap desain dunia disimpan dalam storage lokal pada mesin komputer. Dunia Minecraft secara praktis adalah suatu list atau matrix 3D dengan setiap komponennya bisa dianggap salah satu jenis balok yang ada di dalam gim, dan index matrix adalah posisi balok tersebut di dunia. Namun dengan besar 60 juta × 60 juta × 365 balok dalam satu dunia, bisa diperkirakan banyak memori yang dibutuhkan untuk menyimpan seluruh data satu dunia di Minecraft adalah 97.000 TB dan belum memperhitungkan data dari tiap dunia yang dibuat, namun ukuran dari seluruh program game Minecraft pada storage hanya sekitar 500 MB, hanya satu persekian persen dari 97000 TB.

Gambar 3.1. Dunia pada Minecraft yang dipotong

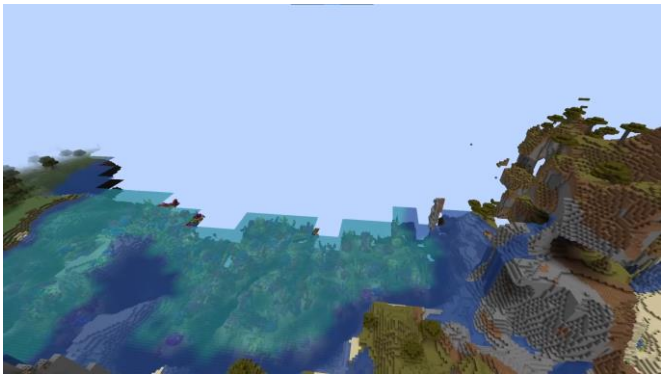
Hal ini karena Minecraft dapat menciptakan suatu dunia karena suatu algoritma. Pada setiap generasi suatu dunia, program akan membuat suatu nilai integer random, sebagai input awal dalam pembuatan dunia. Integer ini disebut dengan *Seed* dalam istilah yang awam. Seed dapat digenerasi oleh program gim, namun juga dapat diinput manual oleh pemain sendiri, dengan input berupa non-integer akan dikonversi menjadi integer untuk dimasukkan ke dalam proses selanjutnya.

Karena basis jalannya Minecraft yang terdapat pada Java, maka fungsi yang memberikan suatu integer acak sudah tersedia dari Java sendiri. Fungsi ini akan menghasilkan hasilnya dalam

kompleksitas O(1) karena tidak ada pengaruh dari besar inputan atau nilai inputan itu sendiri terhadap output fungsi.

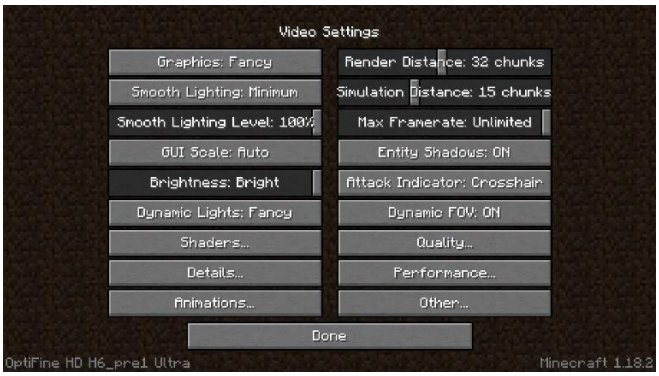
Selanjutnya, akan dipanggil fungsi noise Perlin yang akan menghasilkan suatu pola dimana pola tersebut akan menjadi data ketinggian pada dunia. Data itu juga diberi manipulasi dan *weight* tertentu dalam penghitungannya sebagai tanda pembedaan bioma-bioma dengan medan yang berbeda, seperti gunung dengan kecuraman-kecuraman tertentu, ataupun laut dengan kedalaman yang berbeda-beda.

Hal ini dilakukan dengan menambah amplitudo dari Perlin Noise yang ada dengan suatu konstanta agar memperbesar perbedaan ketinggian dari dunia yang dihasilkan. Generasi Perlin Noise akan menghasilkan kompleksitas $O(2^n)$.



Gambar 3.2. Proses generasi dunia secara procedural selagi pemain bergerak di dalam dunia

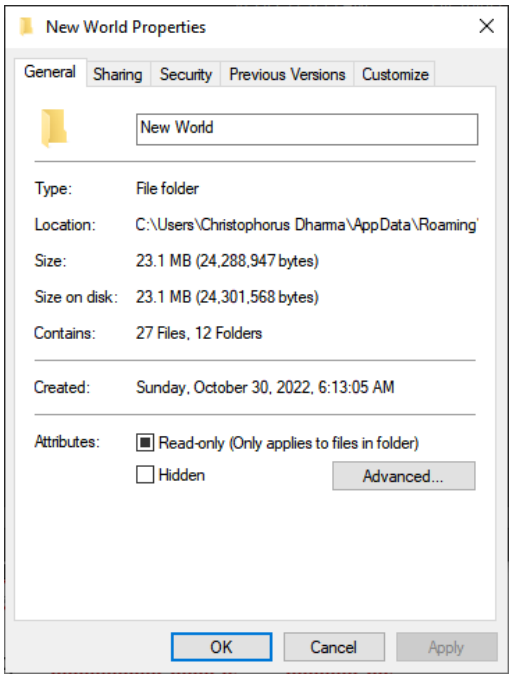
Secara notasi big O, kompleksitas eksponensial atau dengan notasi adalah $O(2^n)$, akan terlihat program tersebut tidak berjalan dengan efisien. Namun hal ini tidak sepenuhnya benar, karena setiap prosedur generasi dilakukan dalam *chunk* atau dalam ukuran block di Minecraft seukuran kurang lebih 16 × 16 × 365 balok, dan *chunk* ini akan diproses selagi pemain bergerak dalam dunia. Sehingga di saat yang bersamaan memberi ilusi bahwa semakin jauh pemain melihat ke cakrawala, dunianya semakin sulit dilihat.



Gambar 3.3. Pengaturan Game Minecraft

Efisiensi prosedur ini juga ditentukan dari pengaturan gim pada komputer pemain sesuai kapabilitas perangkat keras, pada *default* bisa ditetapkan sebagai 16 *chunk untuk* sekali proses, bisa ditambah dan bisa dikurangi. Pengaturan tersebut tertera pada pengaturan "Render Distance" yang ada pada gambar.

Pada saat program tidak dijalankan pun, tidak terlalu banyak data yang disimpan pada tempat penyimpanan di komputer pemain, sehingga menghasilkan kebutuhan ruang yang relatif kecil yaitu kurang lebih 20 MB jika dibandingkan dengan menyimpan seluruh data *chunk* pada satu dunia yang bisa mencapai ukuran 97000 TB.



Gambar 3.4. Ukuran satu dunia diakses dari penyimpanan lokal komputer

V. CONCLUSION

Minecraft adalah program gim dengan cara kerja yang berbeda dengan gim-gim pada umumnya. Ia menyimpan data dengan jauh lebih efisien dengan memroses data-data pada dunia secara prosedural yang akan memberikan kompleksitas $O(2^n)$ dengan kebutuhan ruang yang tidak terlalu banyak. Hal ini dibandingkan dengan program gim lain yang menyimpan data level langsung ke dalam penyimpanan di dalam *storage* yang mana dapat memakan banyak kebutuhan ruang. Minecraft juga memeliki opsi pengaturan sehingga frekuensi melakukan prosedur generasi dunia dapat dikurangi untuk mengurangi banyak kebutuhan pemrosesan pada komputer.

REFERENCES

[1] Wengrow, J. (2020). A Common-sense Guide to Data Structures and Algorithms: Level Up Your Core Programming Skills (2nd ed.). Pragmatic Bookshelf.

[2] Rowell, E., Pleple, Q., & Abed, M. (n.d.). Big-O Algorithm Complexity Cheat Sheet. Big-O Algorithm Complexity Cheat Sheet (Know Thy Complexities!) @ericdrowell. Diakses pada 11 Desember, 2022, dari https://www.bigocheatsheet.com/

[3] Mojang. (2009, May). What is Minecraft? Build, Discover Realms & More. Minecraft. Diakses pada 10 Desember, 2022, dari https://www.minecraft.net/en-us/about-minecraft

[4] breslin, s. (2009, July 16). The History and Theory of Sandbox Gameplay. Game Developer. Diakses pada 11 Desember, 2022, dari https://www.gamedeveloper.com/design/the-history-and-theory-of-sandbox-gameplay

[5] Kniberg, H. (2022, February 6). Minecraft terrain generation in a nutshell. YouTube. Diakses pada 11 Desember, 2022, dari https://www.youtube.com/watch?v=CSa5O6knuwI&ab_channel=HenrikKniberg

[6] Perlin, K. (1987, Juli 1). An image synthesizer. ACM SIGGRAPH Computer Graphics, 19(3), 287-. https://dl.acm.org/doi/10.1145/325165.325247

[7] Perlin, K. (n.d.). Making Noise. Wayback Machine. Diakses pada 11 Desember, 2022, dari https://web.archive.org/web/20071008165504/http://www.noisemachine.com/talk1/6.html

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, 12 Desember 2022

Christophorus Dharma Winata, 13521009