Implementasi Kompleksitas Algoritma dalam Dunia Permainan Game Online

Hidayatullah Wildan Ghaly Buchary - 13521015[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521015@std.stei.itb.ac.id

***Abstract*—Salah satu implementasi program yang paling banyak diminati di era sekarang ini adalah permainan game online. Ada berbagai macam game online yang menggunakan algoritma yang berbagai jenis. Setiap algoritma memiliki kompleksitas dan efisiensi yang berbeda-beda. Algoritma yang paling baik digunakan untuk game online tentunya adalah algoritma yang paling efisien. Algoritma yang dapat disebut algoritma efisien adalah algoritma yang tidak memakan banyak waktu pada saat pemrosesannya, juga tidak membutuhkan banyak memori dalam pemrosesannya. Oleh karena itu, kompleksitas algoritma perlu diperhatikan dalam pemrograman. Dalam makalah ini, akan dijelaskan beberapa algoritma yang mungkin sering dimanfaatkan untuk produksi berbagai game online sekaligus analisa efisiensinya, serta memberikan pemahaman tentang game yang dibuat dengan Kompleksitas algoritma seefisien mungkin dalam hal waktu dan ruang untuk memaksimalkan pengalaman pengguna.**

***Keywords*—Algoritma, Efisien, Game, Kompleksitas.**

I. PENDAHULUAN

Di era disrupsi ini, dimana pertukaran informasi dilakukan dengan sangat cepat dan akses platform atau media yang mudah, tidak bisa dipungkiri bahwa teknologi mendukung keduanya, atau Dalam hal ini, game sudah menjadi bagian dari kehidupan sehari-hari, terutama di kalangan milenial.

Game online merupakan salah satu bentuk perkembangan teknologi di era globalisasi saat ini. Hampir semua orang pernah menggunakan game online. Ini karena mengakses game online itu mudah. Yang Anda butuhkan hanyalah komputer (PC) atau smartphone yang terhubung ke Internet dan Kita dapat mulai menikmati game online. Internet telah mengubah teknologi komunikasi massa dimana siapapun dapat terhubung dimanapun di dunia selama memiliki koneksi internet.

Industri game sebagai pencipta game dan juga salah satu perusahaan hiburan komputer terbesar di dunia menyadari hal tersebut, industri game kelas dunia sedang mencari talenta-talenta muda aktif yang ingin mengembangkan kemampuannya. menjadi bagian dari perusahaan sebagai pengembang untuk game yang akan datang. Memang tidak sebanding jika kita bercermin beberapa tahun yang lalu, ketika bermain video game dianggap sebagai kegiatan yang sia-sia, selain membuang waktu, banyak pendapat bahwa bermain game lebih merugikan yang akhirnya menciptakan modelnya sendiri bahwa gaming adalah candu kehidupan.

Tapi itu dulu, sekarang dengan perkembangan teknologi yang semakin maju dan semua inovasi baru menciptakan ruang peluang baru bagi para gamer untuk lebih bebas hidup sesuai dengan preferensi mereka, contoh yang bisa kita lihat adalah *eSports*. *eSport* adalah istilah olahraga elektronik untuk kompetisi video game multipemain. *E-sports* seakan menjadi antitesis dari model negatif yang diciptakan untuk para gamers, berbagai argumentasi tentang gamers yang dulunya hanya pemalas bermain game sebagai sarana hiburan kini semakin populer.

Untuk itu, para programmer atau developer game terpaksa melipatgandakan usahanya untuk memenuhi tuntutan pasar yang semakin sulit, apalagi dengan kedatangan Next-Gen Gaming yang sudah lama diperbincangkan dan mungkin akan menjadi tren baru. standar untuk dunia game. Game generasi berikutnya adalah istilah yang digunakan untuk game dengan grafis mendekati dunia nyata, pengalaman gameplay baru, bahkan realitas virtual. Peran programmer di sini akan menjadi penting untuk menciptakan sinergi antara kebutuhan dan pengguna, walaupun komputer yang digunakan secara umum dapat mendukung judul-judul populer, namun kinerjanya rendah, hasilnya akan tetap muncul seiring perkembangan pemrosesan data dalam game.

Dengan hal-hal seperti grafik, komunikasi, multipemain, asisten suara, dan banyak lagi. Untuk mendukung hal tersebut, selain memiliki perangkat keras yang cukup, diperlukan arsitektur algoritmik yang efisien. Algoritma yang efisien akan menghasilkan konsumsi sumber daya ruang (memori) dan waktu yang minimal tanpa mengorbankan performa game dan akan memaksimalkan pengalaman bermain game bagi pengguna. Jika keduanya saling mendukung, maka akan tercipta arsitektur algoritmik yang efisien. Suatu algoritma efisien atau tidak dapat dilihat dari kompleksitas algoritma tersebut.

II. TEORI DASAR

A. Definisi Algoritma

Algoritma adalah serangkaian langkah logis yang sistematis dalam memecahkan suatu masalah. Algoritma tidak hanya harus benar, tetapi juga efisien. Untuk dapat mengetahui efisiensi suatu algoritma, maka digunakan teori kompleksitas algoritmik sebagai dasar analisisnya. Kompleksitas algoritma itu sendiri terdiri dari dua tingkat kompleksitas yaitu kompleksitas waktu

dan kompleksitas ruang. Secara teoritis, model pengukuran waktu/ruang abstrak harus independen dari pertimbangan kompiler dan mesin apa pun. Secara umum, definisi kompleksitas algoritmik dapat dibagi menjadi beberapa poin.

Algoritma tidak hanya harus benar, tetapi juga efisien. Algoritma yang baik adalah algoritma yang efisien. Efisiensi suatu algoritma dapat diukur dengan jumlah ruang dan waktu yang diperlukan untuk menjalankannya. Algoritma yang efisien adalah algoritma yang meminimalkan kebutuhan ruang dan waktu. Persyaratan waktu dan ruang dari algoritma bergantung pada ukuran input (n), yang mewakili jumlah data yang akan diproses. Efisiensi algoritma dapat digunakan untuk mengevaluasi algoritma mana yang lebih baik.

B. Kompleksitas Algoritma

Selain bergantung pada komputer, persyaratan waktu suatu program juga ditentukan oleh penyusun bahasa yang digunakan. Ada dua jenis kompleksitas algoritma, kompleksitas waktu dan kompleksitas ruang. Kompleksitas waktu, T(n), diukur dengan jumlah langkah komputasi yang diperlukan untuk menjalankan algoritme dengan ukuran input n. Kompleksitas ruang, S(n), diukur dalam memori yang digunakan oleh struktur data yang terdapat dalam algoritme sebagai fungsi dari ukuran masukan n. Dengan menggunakan besarnya kompleksitas waktu/ruang dari algoritme, kita dapat menentukan laju peningkatan waktu (ruang) yang diperlukan oleh algoritme saat ukuran input n bertambah.

Ada tiga jenis kompleksitas waktu yaitu $T_{max}(n)$, $T_{min}(n)$, dan $T_{avg}(n)$. $T_{max}(n)$ merupakan kompleksitas waktu kasus terburuk, $T_{min}(n)$merupakan kompleksitas waktu untuk kasus terbaik, sedangkan $T_{avg}(n)$ merupakan kompleksitas waktu untuk kasus rata-rata, membutuhkan waktu rata-rata. Berikut akan ditampilkan contoh-contoh algoritma sederhana serta cara menghitung kompleksitas dan tingkat efisiensinya dibandingkan dengan algoritma berbeda dengan tujuan yang sama.

```
procedure maksElemen(input a : array of integer,
                     output maks : integer)
KAMUS LOKAL
    i : integer
ALGORITMA
    maks ← a[0]
    i ← 1
    while (i ≤ n) do
        if (a[i] > maks) then
            maks ← a[i]
        i ← i + 1
    { k > n }
```

Gambar 2.1 contoh program dalam notasi algoritmik untuk mencari elemen terbesar
(sumber : Visual Studio Code)

Berdasarkan gambar 2.1, kompleksitas waktu algoritmanya bisa dihitung berdasarkan jumlah operasi perbandingan elemen larik $(a[i] > maks)$. Dengan begitu akan didapatkan kompleksitas waktu maksElemen adalah $T(n) = n - 1$.

```
procedure sequentialSearch (input a : array of integer,
                            output idx : integer)
KAMUS LOKAL
    i       : integer
    found   : boolean
ALGORITMA
    i ← 0
    found ← false
    while (k ≤ n) and (not found) do
        if (a[i] = x) then
            found ← true
        else
            k ← k + 1
    { k > n or found }
    if (found) then { x ditemukan }
        idx ← i
    else
        idx ← -1 { x tidak ditemukan }
```

Gambar 2.2 contoh program dalam notasi algoritmik untuk mencari elemen dalam array dengan sequential search
(sumber : Visual Studio Code)

Berdasarkan gambar 2.2, kompleksitas waktu algoritmanya bisa dihitung berdasarkan jumlah operasi perbandingan elemen larik $(a[i] = x)$. Dengan begitu akan didapatkan kompleksitas waktu maksimum sequentialSearch adalah $T(n) = n$.

```
procedure binarySearch (input a : array of integer,
                        output idx : integer)
KAMUS LOKAL
    i, j, mid : integer
    found     : boolean
ALGORITMA
    i ← 0
    j ← n
    found ← false
    while (not found) and ( i ≤ j) do
        mid ← (i+j) div 2
        if amid = x then
            found ← true
        else
            if amid < x then { cari di belahan kanan }
                i ← mid + 1
            else { cari di belahan kiri }
                j ← mid - 1;
    {found or i > j }
    if (found) then { x ditemukan }
        idx ← mid
    else
        idx ← -1 { x tidak ditemukan }
```

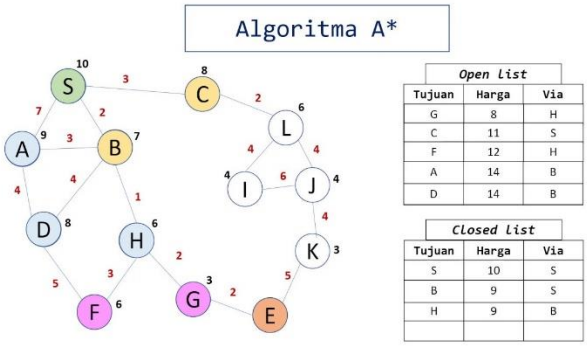
Gambar 2.3 contoh program dalam notasi algoritmik untuk mencari elemen dalam array dengan binnary search
(sumber : Visual Studio Code)

Berdasarkan gambar 2.3, kompleksitas waktu algoritmanya bisa dihitung dan nantinya akan didapatkan kompeksitas waktu maksimum algoritma binarySearch adalah $T(n) = {}^{2}log(n)$. Dibandingkan dengan algoritma sebelumnya, yang merupakan

sequential search, algoritma binary search merupakan algoritma yang lebih efisien berdasarkan kompleksitas waktunya.

C. Game yang Efisien

Saat memecahkan masalah yang berbeda, kita sering menghadapi banyak algoritma berbeda, tetapi kita akan selalu mendapatkan hasil yang sama. Meski begitu, saat merancang sebuah algoritma, kecepatan pembuatan algoritma harus selalu diperhatikan. Contohnya adalah algoritma quicksort dimana data diurutkan dengan sangat cepat tetapi juga memiliki ketergantungan pada input data, yang dalam kasus terburuk memiliki kompleksitas $O(n^2)$. Meskipun perbedaan kecepatan ini tidak terlihat di seluruh lingkungan dengan program kecil, pada program besar perbedaan kecepatan ini hanya akan terlihat. Sebagai perbandingan, kita dapat mengambil sebuah program yang kita gunakan setiap hari, yaitu sebuah game dimana 2 game identik dengan judul yang sama, menggunakan hardware dan software yang sama, tetapi menggunakan dua algoritma yang berbeda, akan memiliki performa dan kecepatan yang berbeda.



Gambar 2.4 Ilustrasi algoritma A-Star
(sumber : https://www.youtube.com/watch?v=opn-OSLzem4)

Game online atau game modern pada umumnya memiliki kompleksitas algoritma pemrosesan data yang besar. Algoritma A* adalah salah satu contoh algoritma yang paling populer dan sering digunakan dalam game. Algoritma A-Star (A*), pertama kali ditemukan oleh Peter Hart, Nils Nilsson dan Bertram Raphael pada tahun 1968, merupakan salah satu algoritma yang digunakan untuk pathfinding, yang merupakan algoritma kompleks untuk menemukan jalur terpendek dari satu titik ke titik lainnya dan tanpa bergerak di atas objek di peta.

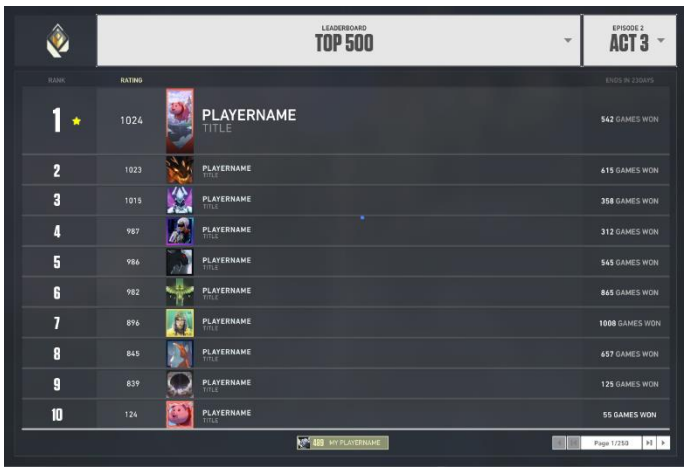
Gambar 2.5 salah satu game yang memerlukan algoritma path finding
(sumber : https://www.hoyolab.com/article/117729)

Di sisi lain, menggunakan algoritme seperti A* menghabiskan banyak sumber daya CPU. Belum lagi kemungkinan bahwa titik yang Anda cari sebenarnya tidak ada di peta, jadi Anda terpaksa memaksa komputer untuk mencari arah di seluruh peta, yang memakan waktu dan sebenarnya tidak perlu. Oleh karena itu, harus ada batasan untuk algoritme ini yang dapat diakali dengan beberapa cara seperti menggunakan peta yang lebih kecil, mengidentifikasi jalur dan jalan buntu, dll. Meminimalkan penggunaan algoritme semacam ini akan membuat permainan menjadi lebih baik. Dari apa yang telah kami rangkum di bagian ini, dapat disimpulkan bahwa penerapan berbagai jenis algoritma kompleks dalam sebuah game akan secara langsung memengaruhi kinerja game itu sendiri, bahkan saat menggunakan perangkat yang sama. Penulis memilih hanya sebagian kecil saja dari algoritma game, dan ternyata masih banyak algoritma yang berperan penting dalam pembuatan sebuah game.

III. Penerapan Algoritma Dalam Game Online

A. Leaderboard

Leaderboard dapat dipahami sebagai peringkat. Dengan sendirinya, sistem peringkat itu sendiri adalah sistem yang memeringkat pemain game online berdasarkan jumlah poin yang mereka kumpulkan.



Gambar 3.1 salah satu game online yang menggunakan algoritma sorting untuk menampilkan leaderboard
(sumber : https://playvalorant.com/)

Cara kerja papan peringkat adalah mengumpulkan poin dan mengurutkannya. Pemain dengan poin terbanyak mendapat peringkat pertama, dan seterusnya. Untuk mengurutkan peringkat sendiri tentunya diperlukan algoritma sorting yang paling efisien untuk game online tersebut.

B. Matchmaking

Game multipemain online dapat dimainkan dengan berbagai cara. Cara paling tradisional adalah mengumpulkan teman di lobi untuk bermain bersama, atau mencari server di daftar server

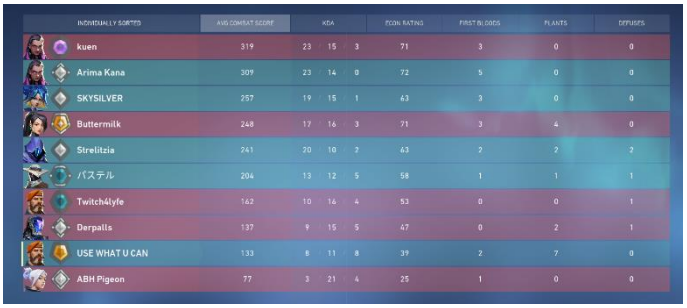
game. Cara ini terkadang dianggap tidak tepat dan valid di game mode rank. Oleh karena itu, pengembang game mengembangkan metode *matchmaking* berdasarkan berbagai parameter. Salah satu contoh *matchmaking* yang sering digunakan adalah SBMM (*skill based matchmaking*).

Gambar 3.2 salah satu game online yang menggunakan konsep SBMM untuk melakukan matchmaking
(sumber : https://esportsku.com/cara-kerja-sistem-match-making-mobile-legends-ml/)

SBMM adalah upaya *matchmaking* untuk mencocokkan orang-orang dengan keterampilan serupa ke dalam sebuah tim, melawan orang-orang dengan keterampilan yang sama dengan mereka. Meski terdengar sangat aman, SBMM sebenarnya memiliki berbagai kelemahan yang sistem matchmaking mana pun akan sangat sulit diatasi. Misalnya, dalam sebuah tim, ada satu orang dengan peringkat terbaik dalam suatu permainan (misalnya: Mythical Glory, Platinum, Radiant, Immortal, dan lain-lain). Sementara itu, mereka juga memiliki pemain lain dengan peringkat terendah (Warrior, Bronze, Herald, Iron). Gambar 3.2 merupakan salah satu bentuk *matchmaking* dalam suatu game yang memanfaatkan sistem SBMM.

Sebagian besar sistem *matchmaking* akan menggunakan salah satu dari tiga kemungkinan logika. Pertama, sistem *matchmaking* akan mencocokkan mereka dengan musuh peringkat tertinggi, peringkat terendah atau semua peringkat milik tim yang dijumlahkan dan dirata-rata.

Ketiga logika di atas masih harus dipadukan dengan parameter pemain bermain sendiri, berpasangan, tiga atau bahkan lima. Mengapa ini sangat penting? Karena komunikasi juga menjadi parameter penting dalam tim matchmaking. Lima master yang terkoordinasi dengan buruk dapat dengan mudah dikalahkan oleh lima pemain level menengah, yang berkomunikasi satu sama lain dan memberi tahu status mereka sepanjang waktu.

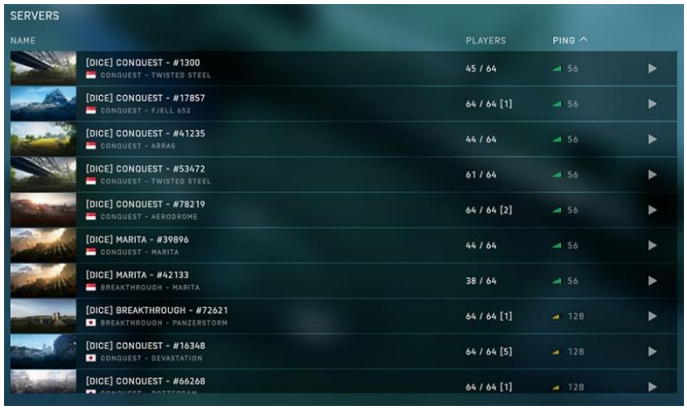


Gambar 3.3 salah satu game online yang menggunakan konsep SBMM tetapi terdapat rank gap yang jauh
(sumber : Valorant)

Hal ini menyebabkan SBMM sering menggabungkan banyak parameter lainnya. Misalnya, tim musuh juga harus memiliki pemain dalam grup tiga, grup dua, dan seterusnya. Anda tidak dapat bergabung dalam pertandingan ketika ada empat pertandingan (dengan meninggalkan slot) dan lain-lain. (dalam kasus Mobile Legends dan Valorant). Gambar 3.3 merupakan salah satu contoh kelemahan algoritma dalam game tersebut yaitu adanya gap yang signifikan pada permainan tersebut.

Biasanya, selain parameter pemeringkatan, SBMM juga mempertimbangkan banyak parameter lainnya. Mulai dari kill rate, headshot rate, assist, gol, dan lain-lain. Dengan demikian, mereka akan mengevaluasi parameter selain peringkat pemain untuk mencegah smurf (yang menggunakan pengenal baru atau karakter dengan peringkat lebih rendah dari yang seharusnya) merajalela.

Selain SBMM, salah algoritma *matchmaking* yang terkenal adalah CBMM (Connection base matchmaking). CBMM memiliki fungsi yang sedikit berkebalikan dengan SBMM. Sistem ini akan menempatkan orang-orang dalam pertandingan di rentang lokasi yang sama atau terkait dengan mereka. Ini memungkinkan setiap pertandingan berlangsung tanpa masalah koneksi. Namun efek sampingnya adalah dalam setiap pertandingan, satu atau dua orang akan mendominasi karena lebih unggul dalam skill, atau menjadi incaran penonton karena kalah.



Gambar 3.4 salah satu game online yang menggunakan konsep CBMM untuk melakukan matchmaking
(sumber : Battlefield V)

Faktanya, CBMM sering diimplementasikan dalam sistem yang sesuai dengan SBMM. Apalagi jika game tersebut dimainkan menggunakan peer-to-peer, bukan sistem server berbasis server untuk client. Namun parameter CBMM biasanya didahulukan saat melakukan quest pemain tanpa menunjukkan efek langsung. CBMM adalah salah satu metode *matchmaking* yang disukai oleh streamer video game. Karena CBMM akan memastikan bahwa pertandingan streaming selalu berjalan lancar dengan sedikit masalah. Selain itu, CBMM juga akan membuat mereka menghadapi lebih sedikit musuh yang "bertarung" sehingga bisa menang dalam pertempuran.

C. Searching

Ada berbagai cara game online melakukan implementasi terhadap pencarian terhadap sesuatu seperti pencarian teman, pencarian barang yang ingin dibuat, pencarian lokasi dalam game, ataupun pencarian lainnya. Biasanya, game online akan menampilkan berbagai pilihan untuk input yang telah dimasukkan oleh pengguna walaupun input tersebut tidak lengkap ataupun belum diisi sama sekali.

Gambar 3.5 searching item yang bisa dibuat di minecraft
(sumber : Minecraft)

Pada gambar 3.5, game menunjukkan satu barang yang mungkin dibuat dengan input *flint* karena hanya itulah barang yang bisa dibuat dengan nama yang mengandung *flint*.

Gambar 3.6 searching item yang bisa dibuat di minecraft dengan hanya menginput beberapa huruf
(sumber : Minecraft)

Pada gambar 3.6, game menampilkan banyak pilihan barang yang bisa dibuat dengan input pengguna yang hanya dua huruf. Hal ini terjadi karena ada banyak barang dalam game tersebut yang mengandung '*st*'. Apabila barang yang ada di dalam game tersebut hanyalah sedikit maka game tersebut masih bisa menggunakan algoritma searching satu persatu, algoritma ini berjalan dengan cara mengecek input pengguna lalu melakukan search dengan algoritma search manapun untuk melakukan validasi apakah ada barang yang mengandung input pengguna tersebut. Tetapi algoritma ini akan berjalan dengan kompleksitas paling lama $O(n^2)$. Untuk itu, setiap game akan memiliki algoritmanya masing-masing dengan penyesuaian terhadap apa yang dibutuhkan dalam game tersebut.

IV. KOMPLEKSITAS ALGORITMA DALAM GAME ONLINE

Setelah mengetahui garis besar dari algoritma yang sering digunakan untuk menjalankan game online, sekarang kita bisa mengetahui apa saja yang diperlukan developer atau para programmer untuk membuat suatu game tertentu dengan pengetahuan yang ada. Di dalam game online, ada sangat banyak algoritma yang diperlukan untuk diimplementasikan. Walaupun banyak, konsep yang paling sering digunakan dalam game online adalah leaderboard, matchmaking, dan searching.

```c
// C program for implementation of Bubble sort
#include <stdio.h>

void swap(int* xp, int* yp){
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n){
    int i, j;
    for (i = 0; i < n - 1; i++)

        // Last i elements are already in place
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(&arr[j], &arr[j + 1]);
}
```

Gambar 4.1 Algoritma bubble sort dalam bahasa C
(sumber : Visual Studio Code)

Dalam pemanfaatan konsep leaderboard, diperlukan algoritma sorting untuk mengurutkan para pemain berdasarkan rating yang dimiliki oleh player. Salah satu algoritma sorting yang paling terkenal adalah bubble sort. Pada gambar 4.1 telah ditampilkan algoritma bubble sort dalam bahasa C yang memiliki kompleksitas algoritma $O(n^2)$.

```python
# Python 3 program for Elo Rating
import math

# Function to calculate the Probability
def Probability(rating1, rating2):
    return 1.0 * 1.0 / (1 + 1.0 * math.pow(10, 1.0 * (rating1 - rating2) / 400))
# Function to calculate Elo rating
# K is a constant.
# d determines whether
# Player A wins or Player B.
def EloRating(Ra, Rb, K, d):
    # To calculate the Winning
    # Probability of Player B
    Pb = Probability(Ra, Rb)
    # To calculate the Winning
    # Probability of Player A
    Pa = Probability(Rb, Ra)
    # Case -1 When Player A wins
    # Updating the Elo Ratings
    if (d == 1) :
        Ra = Ra + K * (1 - Pa)
        Rb = Rb + K * (0 - Pb)
    # Case -2 When Player B wins
    # Updating the Elo Ratings
    else :
        Ra = Ra + K * (0 - Pa)
        Rb = Rb + K * (1 - Pb)
    print("Updated Ratings:-")
    print("Ra =", round(Ra, 6)," Rb =", round(Rb, 6))
```

Gambar 4.2 Algoritma Elo Rating dalam Bahasa Python
(sumber : Visual Studio Code)

Algoritma yang tidak kalah penting dengan leaderboard adalah matchmaking. Konsep matchmaking akan mengandalkan rating dari para pemain. Matchmaking akan memasangkan pemain tertentu dengan pemain lainnya berdasarkan rating yang mereka miliki. Gambar 4.2 merupakan salah satu cara game melakukan update terhadap rating para pemain yang menang maupun yang kalah sehingga dapat digunakan untuk melakukan matchmaking. Kompleksitasnya sendiri hanyalah O(1) sehingga tidak memerlukan waktu yang lama.

```python
# Binary tree node
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Function to print leaf
# nodes from left to right
def printLeafNodes(root: Node):
    if (not root):
        return
    # If node is leaf node,
    # print its data
    if (not root.left and
        not root.right):
        print(root.data,
              end = " ")
        return

    # If left child exists,
    # check for leaf recursively
    if root.left:
        printLeafNodes(root.left)

    # If right child exists,
    # check for leaf recursively
    if root.right:
        printLeafNodes(root.right)
```

Gambar 4.2 Algoritma untuk menampilkan semua daun pada tree dalam bahasa python
(sumber : Visual Studio Code)

Salah satu mekanisma dalam game online yang juga penting adalah mekanisme searching. Ada berbagai cara untuk melakukan searching di dalam game online. Salah satunya adalah dengan mengibaratkan barang yang ingin dicari di dalam sebuah bentuk pohon yang bisa diakses. Input pengguna akan menentukan bagian pohon mana yang diakses. Setelah pengguna memasukkan input dan algoritma telah mendapatkan node mana yang diakses, maka algoritma ini bisa menampilkan apa saja yang mungkin dicari dengan konsep algoritma pada gambar 4.2. Kompleksitasnya hanyalah membutuhkan waktu O(n) sehingga akan sangat efektif apabila barang yang dicari hanyalah sedikit.

V. Kesimpulan dan Saran

Berdasarkan referensi dan data yang diteliti, kita dapat menarik kesimpulan bahwa memang benar bahwa algoritma berdampak nyata pada eksekusi program yang kita gunakan sehari-hari, dalam hal ini adalah game online. Oleh karena itu, dapat dikatakan bahwa dalam pengembangan game online, kompleksitas algoritma diperlukan agar game online dapat berjalan dengan lancar dan efisien, pengembang game online di masa mendatang harus selalu mencari kemungkinan algoritma alternatif dengan kompleksitas yang lebih efisien untuk memproses data dalam jumlah besar dengan cepat.

VI. Ucapan Terima Kasih

Penulis mengucapkan banyak terima kasih untuk Tuhan Yang Maha Esa, karena rahmat dan hidayah-Nya penulis dapat menyelesaikan tugas makalah IF2120 Matematika Diskrit dengan lancar tanpa hambatan. Penulis juga menyampaikan banyak terima kasih kepada dosen Kelas 3 Matematika Diskrit, Dr. Ir. Rinaldi Munir, M.T. yang telah membawakan mata kuliah Matematika Diskrit selama satu semester ini. Selain itu, penulis juga mengucapkan terima kasih kepada keedua orang tua penulis yang telah mendukung penuliis dalam hal moral dan materi sehingga penulis dapat menuntut ilmu sampai saat ini. Tidak lupa penulis juga menyampaikan terima kasih kepada seluruh teman-teman penulis di Teknik Informatika Institut Teknologi Bandung 2021, khususnya yang berada di Kelas 3 IF2120 karena telah membantu penulis dalam menjalankan kegiatan perkuliahan selama satu semester perkuliahan ini.

References

[1] A. Harsh, “Print all leaf nodes of a binary tree from left to right,” GeeksforGeeks, 22-Nov-2022. [Online]. Available: https://www.geeksforgeeks.org/print-leaf-nodes-left-right-binary-tree/. [Accessed: 10-Dec-2022].

[2] B. Widmer, “How the google search algorithm works,” SEO Blog by Ahrefs, 27-Oct-2022. [Online]. Available: https://ahrefs.com/blog/google-search-algorithm/#:~:text=Google%27s%20official%20updates-,What%20is%20the%20Google%20Search%20Algorithm%3F,mentions%2C%20usability%2C%20and%20backlinks. [Accessed: 09-Dec-2022].

[3] E. P. Wibowo, “Penjelasan Berbagai Sistem matchmaking Dan Perbedaannya,” Metaco, 27-Sep-2022. [Online]. Available: https://metaco.gg/apex-legends/penjelasan-berbagai-sistem-matchmaking-dan-perbedaannya. [Accessed: 09-Dec-2022].

[4] K. Riswan, “Cara Kerja sistem matchmaking di Mobile Legends - Gamestation,” Matchmaking, 14-Jul-2020. [Online]. Available: https://esportsku.com/cara-kerja-sistem-match-making-mobile-legends-ml/. [Accessed: 09-Dec-2022]. T. S. Rahayu, “Leaderboard, Cara Baru Pantik Prestasi Dalam Literasi,” CIKGU TERE, 22-Feb-2021. [Online]. Available: https://www.cikgutere.com/2021/02/leaderboard-cara-baru-pantik-prestasi.html. [Accessed: 09-Dec-2022].

[5] M. Rinaldi, 2022, “Kompleksitas Algoritma (BAGIAN1) - institut teknologi bandung,” Kompleksitas Algoritma. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Kompleksitas-Algoritma-2020-Bagian1.pdf. [Accessed: 09-Dec-2022].

[6] T. S. Rahayu, “Leaderboard, Cara Baru Pantik Prestasi Dalam Literasi,” CIKGU TERE, 22-Feb-2021. [Online]. Available: https://www.cikgutere.com/2021/02/leaderboard-cara-baru-pantik-prestasi.html. [Accessed: 09-Dec-2022].

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, 10 Desember 2020

Hidayatullah Wildan Ghaly B.
13521015