ANALISIS PERBANDINGAN ALGORITMA ITERATIF DAN REKURSIF DALAM SIMULASI PERMAINAN "TANGGA KEBERUNTUNGAN"

LAPORAN TUGAS BESAR

ANALISIS KOMPLEKSITAS ALGORITMA

DISUSUN OLEH :

1. Keishin Naufa Alfaridzhi (103112400061)
2. Dafa Awal Wahyu Pambudi (103112400275)
3. Reza Sahrul Nuramdani (103112400265)

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

TAHUN 2025/2026

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Algoritma merupakan komponen fundamental dalam ilmu komputer yang berperan penting dalam penyelesaian berbagai permasalahan komputasi. Efisiensi suatu algoritma menjadi aspek yang krusial, terutama ketika algoritma tersebut dijalankan pada ukuran masukan yang besar. Oleh karena itu, analisis terhadap performa algoritma diperlukan untuk menentukan pendekatan yang paling efektif dan efisien dalam suatu permasalahan tertentu.

Dalam praktik pemrograman, suatu permasalahan sering kali dapat diselesaikan dengan lebih dari satu pendekatan algoritmik. Dua pendekatan yang umum digunakan adalah algoritma iteratif dan algoritma rekursif. Meskipun kedua pendekatan tersebut dapat menghasilkan keluaran yang sama, perbedaan struktur dan mekanisme eksekusinya dapat berdampak pada performa waktu dan penggunaan memori.

Permainan Tangga Keberuntungan merupakan salah satu contoh permasalahan yang dapat dimodelkan secara komputasional. Permainan ini bersifat stokastik dan melibatkan proses berulang hingga kondisi akhir tercapai, sehingga cocok digunakan sebagai studi kasus untuk membandingkan performa algoritma iteratif dan rekursif. Dengan memanfaatkan simulasi komputer, permainan ini dapat digunakan untuk mengamati perilaku algoritma secara teoretis maupun empiris.

Berdasarkan latar belakang tersebut, penelitian ini dilakukan untuk menganalisis dan membandingkan algoritma iteratif dan rekursif dalam simulasi permainan Tangga Keberuntungan. Analisis dilakukan melalui pendekatan teoretis menggunakan kompleksitas algoritma serta pendekatan empiris melalui pengukuran waktu eksekusi menggunakan simulasi berbasis web.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana pemodelan permainan Tangga Keberuntungan ke dalam bentuk algoritma iteratif dan rekursif?

2. Bagaimana kompleksitas waktu dan ruang dari algoritma iteratif dan rekursif dalam simulasi permainan Tangga Keberuntungan?
3. Bagaimana perbandingan performa algoritma iteratif dan rekursif berdasarkan hasil pengujian waktu eksekusi?
4. Algoritma manakah yang lebih efisien untuk digunakan dalam simulasi permainan Tangga Keberuntungan?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Memodelkan permainan Tangga Keberuntungan ke dalam bentuk algoritma iteratif dan rekursif.
2. Menganalisis kompleksitas waktu dan ruang dari kedua pendekatan algoritmik tersebut.
3. Mengukur dan membandingkan performa algoritma iteratif dan rekursif melalui pengujian empiris.
4. Menentukan pendekatan algoritmik yang lebih efisien untuk simulasi permainan Tangga Keberuntungan.

1.4 Batasan Masalah

Agar penelitian ini lebih terfokus dan terarah, maka ditetapkan beberapa batasan masalah sebagai berikut:

1. Simulasi permainan dilakukan untuk satu pemain.
2. Ukuran papan permainan ditetapkan sebesar 10 x 10 kotak sesuai dengan konfigurasi standar permainan Tangga Keberuntungan.
3. Pengujian performa algoritma difokuskan pada pengukuran waktu eksekusi, sedangkan analisis penggunaan memori dibahas secara teoretis.
4. Simulasi dan pengujian dilakukan menggunakan bahasa pemrograman JavaScript dalam lingkungan berbasis web.
5. Visualisasi permainan tidak disertakan dalam pengukuran waktu eksekusi.

1.5 Sistematika Penulisan

Sistematika penulisan laporan ini disusun sebagai berikut:

- BAB I Pendahuluan, berisi latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, dan sistematika penulisan.
- BAB II Studi Kasus Permasalahan, membahas deskripsi permainan Tangga Keberuntungan serta pemodelan masalah ke dalam bentuk algoritma.
- BAB III Deskripsi dan Metodologi Algoritma, menjelaskan algoritma iteratif dan rekursif yang digunakan serta metodologi pengujian.
- BAB IV Analisis Kompleksitas dan Hasil Pengujian, menyajikan analisis teoretis dan hasil pengujian empiris beserta pembahasannya.
- BAB V Kesimpulan dan Saran, memuat kesimpulan dari hasil penelitian dan saran untuk pengembangan selanjutnya.

BAB II STUDI KASUS PERMASALAHAN

2.1 Deskripsi Permainan Tangga Keberuntungan

Permainan Tangga Keberuntungan merupakan sebuah permainan papan (*board game*) yang memodelkan lintasan progresif pemain dari titik awal menuju titik akhir melalui papan yang terdiri dari sejumlah kotak berurutan. Pemain maju berdasarkan keluaran nilai dadu yang bersifat acak, dan pada beberapa kotak terdapat elemen khusus seperti "tangga" yang mempercepat pergerakan atau "rintangan" yang dapat menghambat atau mengembalikan posisi pemain. Permainan akan berakhir saat seorang pemain mencapai atau melewati kotak tujuan paling akhir. Sifat pergerakan yang bergantung pada hasil dadu menjadikan simulasi permainan ini bersifat stokastik, sehingga cocok untuk dimodelkan secara komputasional menggunakan algoritma untuk mendukung analisis performa suatu pendekatan komputasi.

2.2 Pemodelan Masalah ke Bentuk Algoritma

Dalam konteks simulasi komputer untuk permainan seperti Tangga Keberuntungan, diperlukan struktur algoritma yang mampu merepresentasikan rangkaian langkah pemain secara efisien. Algoritma iteratif menggunakan struktur perulangan (*loop*) untuk mengeksekusi instruksi berulang kali sampai kondisi tertentu terpenuhi. Pendekatan ini umumnya lebih hemat memori karena tidak memerlukan tumpukan panggilan fungsi (*call stack*), serta lebih mudah dioptimasi di banyak bahasa pemrograman imperatif. Sebaliknya, algoritma rekursif adalah pendekatan yang memecah masalah menjadi sub-masalah yang lebih kecil dengan cara fungsi memanggil dirinya sendiri hingga mencapai basis kasus. Pendekatan ini intuitif untuk beberapa struktur data dan permasalahan yang bisa diurai secara hierarkis.

2.3 Perbandingan Performa Iteratif vs. Rekursif

Dari hasil studi perbandingan algoritma iteratif dan rekursif yang dilakukan pada beberapa kasus, ditemukan bahwa kedua pendekatan memiliki karakteristik performa dan penggunaan sumber daya yang berbeda.

Pertama, algoritma iteratif cenderung lebih efektif dalam penggunaan memori karena tidak memerlukan penyimpanan kontekstual setiap panggilan fungsi seperti pada rekursi; hal ini membuat pendekatan iteratif sering kali lebih efisien untuk tugas linear sederhana yang berulang tanpa bercabang.

Kedua, meskipun rekursi seringkali lebih intuitif dalam mengungkapkan struktur permasalahan tertentu (misalnya *divide and conquer* atau struktur hierarkis), pendekatan tersebut rentan terhadap penggunaan tumpukan memori yang lebih besar dan risiko *stack overflow* ketika kedalaman rekursi sangat tinggi.

Ketiga, studi yang meninjau kasus empiris rekursif dan iteratif untuk berbagai permasalahan dasar menunjukkan bahwa meskipun kedua pendekatan dapat menghasilkan solusi yang setara secara logika, iterasi sering kali menawarkan performa yang lebih stabil dan hemat memori pada skenario berulang tanpa cabang kompleks.

2.4 Implikasi bagi Simulasi Permainan Tangga Keberuntungan

Dalam simulasi permainan Tangga Keberuntungan, pemilihan pendekatan algoritmik harus mempertimbangkan skenario penggunaan dan kebutuhan performa. Jika simulasi dilakukan dengan jumlah permainan yang besar dan berulang tanpa struktur bercabang kompleks, penggunaan pendekatan iteratif cenderung lebih efisien dalam hal waktu eksekusi dan penggunaan memori karena tidak memiliki *overhead* panggilan fungsi yang berulang. Namun, jika simulasi menyaratkan logika keputusan yang kompleks atau struktur data yang secara alami memanfaatkan pemanggilan fungsi berjenjang (misalnya dalam eksplorasi jalur keputusan tertentu), maka pendekatan rekursif dapat memberikan keuntungan dari segi kejelasan implementasi dan kemudahan pemodelan meskipun harus memperhatikan manajemen memori untuk mencegah kelebihan penggunaan *stack*.

-

BAB III DESKRIPSI DAN METODOLOGI ALGORITMA

3.1 Deskripsi Umum Algoritma yang Digunakan

Pada penelitian ini digunakan dua pendekatan algoritmik untuk mensimulasikan permainan Tangga Keberuntungan, yaitu algoritma iteratif dan algoritma rekursif. Kedua algoritma tersebut dirancang untuk menyelesaikan permasalahan yang sama, yaitu mensimulasikan pergerakan pemain dari posisi awal hingga mencapai posisi akhir berdasarkan aturan permainan yang telah ditentukan.

Pemilihan algoritma iteratif dan rekursif didasarkan pada karakteristik permasalahan simulasi yang bersifat berulang dan sekuensial. Menurut Cormen et al., permasalahan yang melibatkan pengulangan langkah dengan kondisi berhenti yang jelas dapat diimplementasikan menggunakan pendekatan iteratif maupun rekursif, dengan perbedaan utama terletak pada struktur kontrol dan penggunaan memori.

3.2 Algoritma Iteratif

Algoritma iteratif adalah algoritma yang menggunakan struktur perulangan (*loop*) untuk mengeksekusi serangkaian instruksi secara berulang hingga kondisi tertentu terpenuhi. Dalam simulasi permainan Tangga Keberuntungan, algoritma iteratif digunakan untuk memproses setiap giliran permainan secara berurutan hingga pemain mencapai kotak tujuan.

Pada setiap iterasi, algoritma akan menghasilkan nilai dadu secara acak, memperbarui posisi pemain berdasarkan nilai tersebut, serta melakukan penyesuaian posisi apabila pemain mendarat pada kotak yang memiliki tangga atau rintangan. Proses ini diulang terus-menerus hingga kondisi akhir permainan tercapai.

Pendekatan iteratif memiliki keunggulan dalam hal efisiensi penggunaan memori karena memerlukan penyimpanan konteks pemanggilan fungsi seperti pada algoritma rekursif. Hal ini menjadikan algoritma iteratif lebih stabil ketika digunakan untuk simulasi dengan jumlah langkah atau perulangan yang besar.

Tabel 3.1 Pseudocode - Simulasi Permainan Ular Tangga (Iteratif)

Program Algoritma Simulasi Permainan Ular Tangga (Iteratif)

ALGORITMA SimulasiIteratif
  WHILE position < boardSize DO
  dice ← Random(1, 6)
  position ← position + dice
  IF position terdapat dalam snakeLadderMap THEN
  position ← snakeLadderMap[position]
  ENDIF
  steps ← steps + 1
  ENDWHILE
  RETURN steps
ENDALGORITMA

3.3 Algoritma Rekursif

Algoritma rekursif adalah algoritma yang menyelesaikan suatu permasalahan dengan memanggil dirinya sendiri untuk menyelesaikan sub-permasalahan yang lebih kecil, hingga mencapai kondisi dasar (*base case*). Dalam simulasi permainan Tangga Keberuntungan, setiap pemanggilan rekursif merepresentasikan satu giliran permainan.

Pada pendekatan ini, fungsi rekursif akan menerima posisi pemain sebagai parameter. Jika posisi pemain belum mencapai kotak tujuan, maka fungsi akan menghasilkan nilai dadu, memperbarui posisi pemain, dan memanggil dirinya kembali dengan posisi yang baru. Proses rekursi akan berhenti ketika dasar terpenuhi, yaitu saat pemain mencapai atau melewati kotak akhir.

Meskipun algoritma rekursif sering kali lebih mudah dipahami secara konseptual, terutama untuk permasalahan yang memiliki struktur berulang pendekatan ini memiliki kelemahan dalam penggunaan memori karena setiap pemanggilan fungsi harus disimpan dalam *call stack*. Jika jumlah langkah permainan besar, penggunaan rekursi berpotensi menyebabkan *stack overflow*.

Tabel 3.2 Pseudocode - Simulasi Permainan Ular Tangga (Rekursif)

Program Algoritma Simulasi Permainan Ular Tangga (Rekursif)

ALGORITMA SimulasiRekursif(position)
IF position ≥ boardSize THEN
RETURN 0
ENDIF
dice ← Random(1, 6)
newPosition ← position + dice
IF newPosition terdapat dalam snakeLadderMap THEN
newPosition ← snakeLadderMap[newPosition]
ENDIF
RETURN 1 + SimulasiRekursif(newPosition)
ENDALGORITMA

3.4 Metodologi Pengujian Algoritma

Untuk membandingkan efisiensi algoritma iteratif dan rekursif, dilakukan pengujian empiris terhadap kedua algoritma dengan menjalankan situasi permainan dalam berbagai ukuran masukan. Ukuran masukan (**n**) didefinisikan sebagai jumlah simulasi permainan yang dijalankan secara berulang.

Pengujian dilakukan dengan menjalankan masing-masing algoritma untuk nilai **n** yang berbeda, misalnya 10, 100, 1.000, hingga 10.000 simulasi permainan. Waktu eksekusi dicatat menggunakan mekanisme pengukuran waktu dengan resolusi tinggi, kemudian diambil nilai rata-rata dari beberapa kali pengujian untuk mengurangi pengaruh fluktuasi sistem.

Pendekatan ini sejalan dengan metode analisis empiris algoritma yang digunakan untuk mengamati perilaku waktu eksekusi aktual dan membandingkannya dengan analisis kompleksitas teoretis. Dengan demikian, hasil pengujian dapat digunakan untuk memperkuat kesimpulan mengenai efisiensi algoritma iteratif dan rekursif dalam konteks simulasi permainan.

BAB IV ANALISIS KOMPLEKSITAS DAN EKSPERIMEN

4.1 Analisis Kompleksitas Waktu Teoretis

Analisis kompleksitas waktu dilakukan untuk mengetahui bagaimana pertumbuhan waktu eksekusi algoritma seiring dengan bertambahnya ukuran masukan (*input size*). Pada simulasi permainan Tangga Keberuntungan, setiap langkah permainan terdiri atas operasi konstan, yaitu pelemparan dadu, pembaruan posisi pemain, serta pengecekan kondisi tangga atau rintangan.

4.1.1 Algoritma Iteratif

Proses simulasi dijalankan menggunakan struktur perulangan hingga kondisi akhir permainan tercapai. Jika jumlah langkah yang diperlukan untuk menyelesaikan satu permainan dinyatakan sebagai T, maka jumlah operasi yang dieksekusi berbanding lurus dengan T, Dengan demikian, kompleksitas waktu algoritma iteratif untuk satu simulasi permainan dapat dinyatakan dalam notasi Big-O sebagai O(T).

4.1.2 Algoritma Rekursif

Setiap langkah permainan direpresentasikan sebagai satu pemanggilan fungsi rekursif. Relasi waktu eksekusi algoritma rekursif dapat dituliskan sebagai $T(n) = T(n-1) + c$, dengan c sebagai biaya operasi konstan pada setiap pemanggilan. Penyelesaian relasi tersebut menunjukkan bahwa kompleksitas waktu algoritma rekursif juga berada pada kelas O(T).

Analisis ini sesuai dengan teori dasar analisis algoritma yang menyatakan bahwa algoritma iteratif dan rekursif dapat memiliki kompleksitas waktu yang sama meskipun berbeda dalam struktur implementasinya.

4.2 Analisis Kompleksitas Ruang

Selain kompleksitas waktu, kompleksitas ruang juga menjadi faktor penting dalam menilai efisiensi algoritma. Pada algoritma iteratif, penggunaan memori relatif konstan karena tidak terdapat pemanggilan fungsi berulang yang memerlukan penyimpanan konteks eksekusi tambahan. Oleh karena itu, kompleksitas ruang algoritma iteratif dapat dinyatakan sebagai O(1).

Sebaliknya, algoritma rekursif memerlukan ruang tambahan pada *call stack* untuk setiap pemanggilan fungsi hingga kondisi dasar tercapai. Jika jumlah langkah permainan adalah T, maka diperlukan T *call stack* pemanggilan fungsi, sehingga kompleksitas ruang algoritma rekursif menjadi O(T). Kondisi ini menjadikan algoritma rekursif kurang efisien dari sisi penggunaan memori, terutama ketika simulasi yang dijalankan dalam jumlah langkah yang besar.

Perbedaan kompleksitas ruang ini telah banyak dibahas dalam literatur algoritma dan menjadi salah satu alasan utama mengapa pendekatan iteratif sering lebih disarankan untuk permasalahan yang bersifat linear dan berulang.

4.3 Pengujian Running Time

Pengujian running time dilakukan untuk mengevaluasi performa algoritma iteratif dan rekursif dalam simulasi permainan Tangga Keberuntungan. Ukuran masukan (n) didefinisikan sebagai jumlah simulasi permainan yang dijalankan secara berulang dalam satu kali pengujian.

Untuk setiap nilai n, masing-masing algoritma dijalankan sebanyak lima kali perulangan ($K = 5$). Waktu eksekusi diukur dengan mencatat selisih waktu sebelum dan sesudah proses simulasi menggunakan fungsi *performance.now()* yang menyediakan pengukuran waktu beresolusi tinggi. Nilai waktu eksekusi yang digunakan dalam analisis merupakan nilai rata-rata dari seluruh pengulangan untuk mengurangi pengaruh fluktuasi sistem dan ketidakpastian lingkungan eksekusi.

Pengujian dilakukan pada beberapa ukuran masukan, yaitu $n = 1,\ 10,\ 100,\ 1.000$, dan 10.000. Seluruh eksperimen dijalankan pada lingkungan web menggunakan bahasa pemrograman JavaScript dengan konfigurasi papan permainan yang sama untuk kedua algoritma. Pendekatan ini sejalan dengan praktik analisis empiris algoritma yang menekankan konsistensi lingkungan dan pengulangan pengujian untuk memperoleh hasil yang reliabel.

Seluruh pengujian dijalankan pada lingkungan berbasis web menggunakan bahasa pemrograman JavaScript pada browser Microsoft Edge (Chromium) di sistem operasi Windows. Pengujian dilakukan pada satu perangkat yang sama untuk menjaga konsistensi hasil.

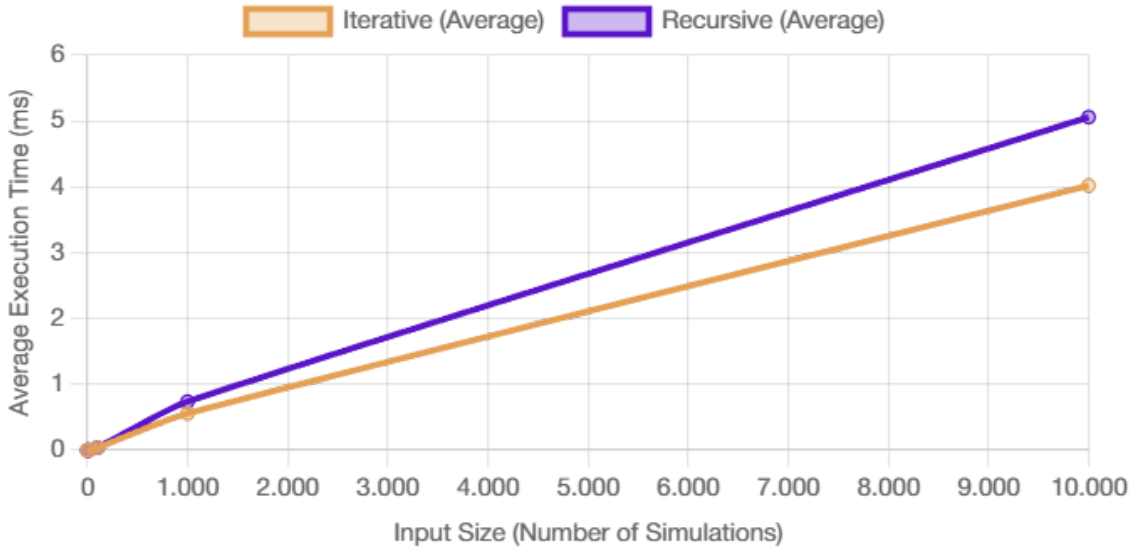
4.4 Hasil Pengujian dan Grafik Perbandingan

Hasil pengujian running time rata-rata untuk algoritma iteratif dan rekursif disajikan dalam bentuk tabel dan grafik. Tabel menunjukkan waktu eksekusi rata-rata (dalam milidetik) untuk setiap ukuran masukan yang diuji.

Tabel 4.1 *Records of Average Execution Time* (K = 5)

Input Size	Iterative (ms)	Recursive (ms)
1	0.00	0.00
10	0.00	0.00
100	0.04	0.04
1000	0.56	0.74
10000	4.02	5.06

Gambar 4.1 Line Chart - Perbandingan *Time Execution* Iteratif dan Rekursif



Berdasarkan tabel hasil pengujian, terlihat bahwa ukuran masukan (*input size*) kecil ($n = 1$ dan $n = 10$), waktu eksekusi kedua algoritma sangat kecil hingga mendekati nol. Hal ini disebabkan oleh jumlah operasi yang masih sangat sedikit sehingga pengaruh *overhead* sistem dan resolusi pengukuran waktu menjadi dominan.

Pada ukuran masukan yang lebih besar ($n \geq 100$), perbedaan waktu eksekusi mulai terlihat dengan jelas. Algoritma iteratif menunjukkan waktu eksekusi yang lebih

rendah dibandingkan algoritma rekursif. Misalnya, pada $n = 10.000$, algoritma iteratif memiliki waktu eksekusi rata-rata sebesar 4,02 ms, sedangkan algoritma rekursif mencapai 5,06 ms.

Grafik perbandingan menunjukkan bahwa waktu eksekusi kedua algoritma meningkat seiring bertambahnya ukuran masukan. Pola peningkatan yang terlihat bersifat hampir linear, yang mengindikasikan bahwa pertumbuhan waktu eksekusi sejalan dengan jumlah simulasi yang dijalankan.

4.5 Analisis Perbandingan dan Pembahasan

Berdasarkan hasil pengujian empiris, dapat diamati bahwa algoritma iteratif dan rekursif memiliki tren pertumbuhan waktu eksekusi yang serupa. Hal ini menunjukkan bahwa kedua algoritma berada dalam kelas kompleksitas waktu yang sama, yaitu linear terhadap jumlah simulasi permainan yang dijalankan.

Meskipun demikian, algoritma rekursif secara konsisten menunjukkan waktu eksekusi yang lebih tinggi dibandingkan algoritma iteratif pada ukuran masukan (*Input Size*) yang besar. Perbedaan ini disebabkan oleh adanya *overhead* pemanggilan fungsi dan penggunaan *call stack* pada algoritma rekursif, yang tidak terdapat pada algoritma iteratif. *Overhead* tersebut menjadi semakin signifikan seiring dengan bertambahnya jumlah simulasi dan langkah permainan.

Pada ukuran masukan kecil, perbedaan performa kedua algoritma tidak terlihat secara signifikan. Hal ini dapat dijelaskan oleh dominasi biaya tetap (*constant overhead*) dan keterbatasan resolusi pengukuran waktu pada lingkungan eksekusi. Namun, pada ukuran masukan yang lebih besar, pengaruh struktur algoritma menjadi lebih dominan sehingga perbedaan performa dapat diamati dengan jelas.

Hasil ini konsisten dengan analisis kompleksitas teoretis yang telah dibahas sebelumnya, serta sejalan dengan temuan pada penelitian terdahulu yang menyatakan bahwa algoritma iteratif dan rekursif dapat memiliki kompleksitas asimtotik yang sama, tetapi menunjukkan performa aktual yang berbeda akibat faktor implementasi dan penggunaan memori.

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis teoretis dan pengujian empiris yang telah dilakukan pada simulasi permainan Tangga Keberuntungan, dapat disimpulkan bahwa algoritma iteratif dan algoritma rekursif memiliki kelas kompleksitas waktu yang sama, yaitu linear terhadap jumlah simulasi permainan yang dijalankan. Hal ini ditunjukkan oleh pola pertumbuhan waktu eksekusi yang meningkat secara hampir linear seiring dengan bertambahnya ukuran masukan.

Meskipun memiliki kompleksitas asimtotik yang sama, hasil pengujian menunjukkan bahwa algoritma iteratif secara konsisten memiliki waktu eksekusi yang lebih rendah dibandingkan algoritma rekursif, terutama pada ukuran masukan yang besar. Pada pengujian dengan jumlah simulasi hingga 10.000, algoritma rekursif membutuhkan waktu eksekusi yang lebih tinggi akibat adanya *overhead* pemanggilan fungsi dan penggunaan *call stack* yang tidak ditemukan pada algoritma iteratif.

Pada ukuran masukan yang kecil, perbedaan waktu eksekusi antara kedua algoritma tidak terlihat secara signifikan. Hal ini disebabkan oleh dominasi biaya tetap (*constant overhead*) dan keterbatasan resolusi pengukuran waktu pada lingkungan eksekusi berbasis web. Namun, seiring meningkatnya ukuran masukan, struktur algoritma menjadi faktor yang lebih dominan sehingga perbedaan performa dapat diamati dengan jelas.

Dengan demikian, dapat disimpulkan bahwa meskipun algoritma rekursif menawarkan kejelasan struktur dan kemudahan pemodelan, algoritma iteratif lebih efisien dan lebih sesuai digunakan untuk simulasi permainan Tangga Keberuntungan yang bersifat berulang dan linear, terutama ketika dijalankan dalam jumlah simulasi yang besar.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa saran yang dapat dijadikan pertimbangan untuk pengembangan lebih lanjut. Pertama, penelitian selanjutnya dapat memperluas pengujian dengan variasi ukuran papan permainan atau

konfigurasi ular dan tangga yang berbeda untuk mengamati pengaruh kompleksitas permainan terhadap performa algoritma.

Kedua, pengujian performa dapat diperluas dengan mempertimbangkan aspek penggunaan memori secara lebih detail, khususnya untuk algoritma rekursif, sehingga analisis tidak hanya berfokus pada waktu eksekusi tetapi juga pada efisiensi sumber daya secara keseluruhan.

Ketiga, penelitian lanjutan dapat membandingkan implementasi algoritma iteratif dan rekursif pada bahasa pemrograman atau lingkungan eksekusi yang berbeda, seperti bahasa pemrograman berorientasi sistem atau lingkungan non-web, guna memperoleh gambaran performa yang lebih komprehensif.

Terakhir, pengembangan simulasi dapat dilengkapi dengan visualisasi yang lebih interaktif sebagai alat bantu pembelajaran, tanpa mengikutsertakan visualisasi tersebut dalam pengukuran performa, sehingga hasil analisis tetap objektif dan akurat.

DAFTAR PUSTAKA

Cormen, T. H., Leiserson, C. E., Rivest, R. L., dan Stein, C. (2009). *Introduction to Algorithms* (Edisi ke-3). Cambridge, MA: MIT Press.

Endres, M., Weimer, W., dan Kamil, A. (2021). An Analysis of Iterative and Recursive Problem Performance. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*, 205–211.

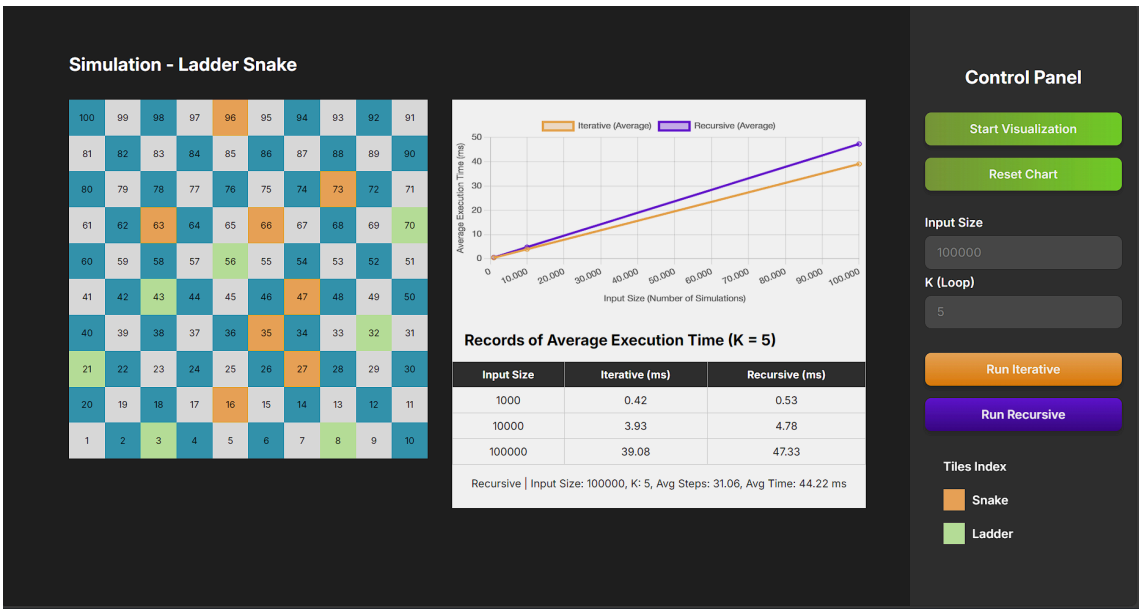
Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. New York: John Wiley & Sons.

Corrado, M. (2018). Using Board Games as Subject Matter for Developing System Modeling Skills. *NASA Technical Reports*.

Lutfina, E. (2022). Analisis Perbandingan Kinerja Metode Rekursif dan Metode Iteratif dalam Algoritma Linear Search. *Komputika: Jurnal Sistem Komputer*, 11(2), 143–150.

LAMPIRAN

```
function simulateIterative() {
    let position = 0;
    let steps = 0;

    while (position < BOARD_SIZE) {
        position += rollDice();

        if (snakeLadderMap[position]) {
            position = snakeLadderMap[position];
        }

        steps++;
    }

    return steps;
}

function simulateRecursive(position = 0) {
    if (position >= BOARD_SIZE) {
        return 0;
    }

    let newPosition = position + rollDice();

    if (snakeLadderMap[newPosition]) {
        newPosition = snakeLadderMap[newPosition];
    }

    return 1 + simulateRecursive(newPosition);
}
```


Simulation - Ladder Snake
100 99 98 97 96 95 94 93 92 91
81 82 83 84 85 86 87 88 89 90
80 79 78 77 76 75 74 73 72 71
61 62 63 64 65 66 67 68 69 70
60 59 58 57 56 55 54 53 52 51
41 42 43 44 45 46 47 48 49 50
40 39 38 37 36 35 34 33 32 31
21 22 23 24 25 26 27 28 29 30
20 19 18 17 16 15 14 13 12 11
1 2 3 4 5 6 7 8 9 10
Iterative (Average)
Recursive (Average)
Average Execution Time (ms)
0
10
20
30
40
50
10,000
20,000
30,000
40,000
50,000
60,000
70,000
80,000
90,000
100,000
Input Size (Number of Simulations)
Records of Average Execution Time (K = 5)
Input Size
Iterative (ms)
Recursive (ms)
1000
0.42
0.53
10000
3.93
4.78
100000
39.08
47.33
Recursive | Input Size: 100000, K: 5, Avg Steps: 31.06, Avg Time: 44.22 ms
Control Panel
Start Visualization
Reset Chart
Input Size
100000
K (Loop)
5
Run Iterative
Run Recursive
Tiles Index
Snake
Ladder