Laporan Tugas Benar Analisis Komplekasi Algoritma

Pengaruh Kualitas Tidur terhadap Kesehatan Mental Kalangan Remaja

Kelas IF-03-01

Dosen VRO

Dibuat Oleh:

Arina Qurrata Aini : 1203230053

Ahmad Assyifa Dzaky Rahman :1203230058

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOMSURABAYA
2024

A. Deskripsi *Study Case*

Penelitian ini berfokus pada hubungan antara kualitas tidur dan kesehatan mental di kalangan remaja. *Studi* ini akan mengkaji bagaimana kurang tidur atau gangguan tidur mempengaruhi tingkat kecemasan, depresi, serta kemampuan kognitif pada remaja. Data dari hasil analisis algoritma untuk melihat pola hubungan antara durasi dan kualitas tidur dengan kesejahteraan psikologis, serta memberikan rekomendasi untuk perbaikan pola tidur yang mendukung kesehatan mental.

Penelitian ini tidak hanya fokus pada analisis hubungan antara kualitas tidur dan kesehatan mental tetapi juga mengintegrasikan pendekatan algoritmik untuk memodelkan pola data. Studi ini menggunakan algoritma rekursif dan iteratif untuk mengolah data terkait durasi tidur dan kualitasnya terhadap tingkat kecemasan, depresi, dan kemampuan kognitif remaja. Analisis kompleksitas algoritmik menggunakan Big O akan diterapkan untuk mengevaluasi efisiensi pendekatan yang digunakan.

Pendekatan algoritmik akan dimanfaatkan untuk menganalisis korelasi antara variabel. Contohnya adalah perhitungan distribusi data (melalui fungsi rekursif dan iteratif) untuk mengidentifikasi pola tidur yang konsisten dan dampaknya pada kesejahteraan psikologis.

B. Deskripsi Algoritma

1. Algoritma Iteratif

Deskripsi:

- Algoritma iterative menggunakan *loop* untuk menghitung konsentrasi akhir berdasarkan jumlah hari dan kualitas tidur. Nilai konsentrasi diperbarui setiap hari berdasarkan kondisi kualitas tidur (kurang dari 6, antara 6-8, atau lebih dari 8 jam).
- Kondisi tambahan disertakan untuk memastikan bahwa nilai konsentrasi tidak turun di bawah ambang batas tertentu (yaitu, 10%).

Keuntungan:

- **Efesiensi Memori:** Pendekatan iterative lebih hemat memori karena tidak menggunakan stack rekursif.
- **Kinerja Stabil:** Iteratif langsung memproses data tampa ***overhead*** dari pemanggilan fungsi berulang.
- **Skalabilitas:** Cocok untuk data skala besar kompleksitas ruangnya tetap konstan (O(1)).

Kelemahan:

- **Kompleksitas Kode:** Untuk perhitungan yang komplek, pendekatan iterative bisa menjadi lebih sulit dibaca dibandingkan rekursif.
- **Terlalu Linear:** Tidak cocok untuk masalah yang membutuhkan pembagian atau struktur data bercabang.

Big-O Notation:

- **Kompleksitas Waktu: O(n),** karena *loop* berjalan sebanyak *n* kali.
- **Kompleksitas Ruang: O(1),** karena tidak ada penggunaan memori tambahan selain variable.

2. Algoritma Rekursif:

Deskripsi:

- Algoritma rekursif menyelesaikan masalah yang sama dengan memanggil dirinya sendiri untuk menghitung konsentrasi setiap hari sehingga mencapai batas tertentu atau konsentrasi minimum (100%).
- Pada setiap pemanggilan, konsentrasi diperbaharui berdasarkan kondisi kualitas tidur.

Keuntungan:

- Sederhana dan Elegan: Rekursif sering kali lebih mudah dipahami dalam permasalahan yang memiliki struktur berulang seperti perhitungan per hari.
- Fleksibilitas: Cocok untuk masalah yang dapat dipecah menjadi sub-masalah yang lebih kecil (*divide and conquer).*

Kelemahan:

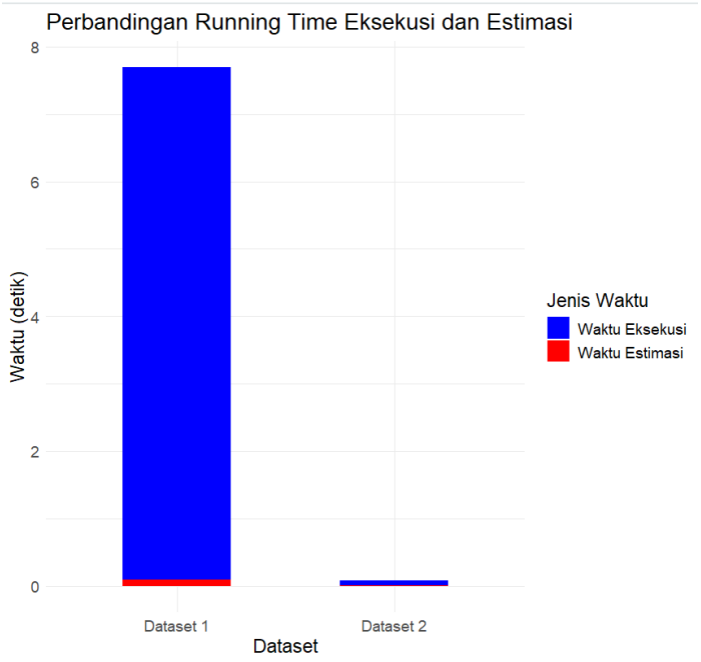
- **Penggunaan Memori:** Setiap pemanggilan fungsi memerlukan tambahan memori si stack, sehingga kurang efisien untuk data skala besar.
- **Resiko Stack Oaverflow:** Jika jumlah hari terlalu besar, sehingga kurang efisien untuk data skala besar.
- **Kinerja Kurang Optional:** Pemanggilan fungsi berulang memiliki *overhead* yang lebih tinggi dibandingkan *loop iterative*.

Big-O Notation:

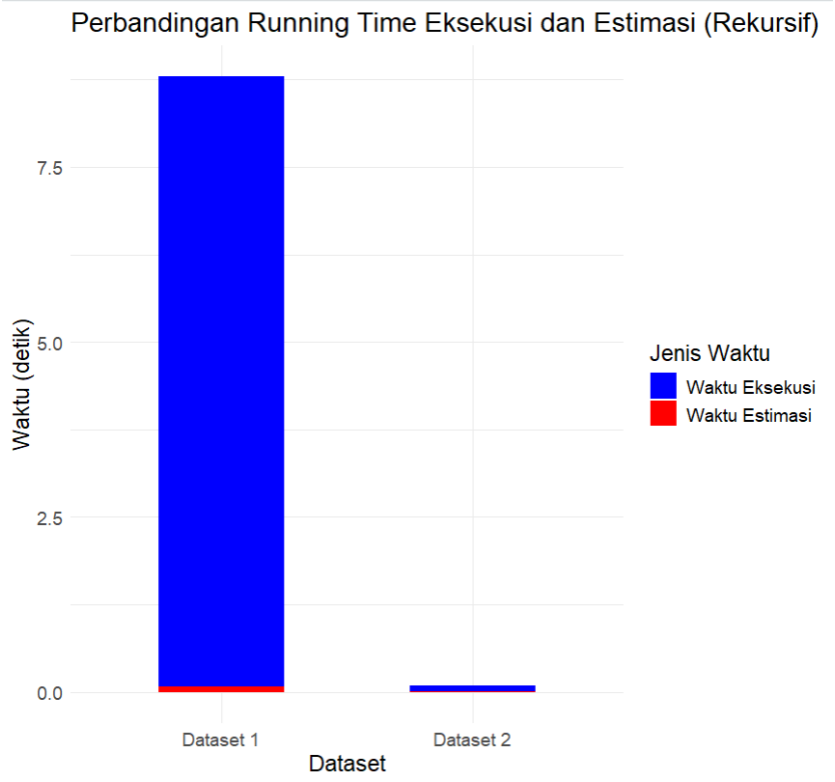
- **Kompleksitas Waktu:** O(n), karena fungsi dipanggil *n* kali untuk *n* hari.
- **Kompleksitas Ruang:** O(n), karena setiap pemanggilan fungsi membutuhkan tambahan memori di stack.

C. Grafik Perbandingan *running time*

1. Algoritma Iteratif



2. Algoritma Rekursif



D. Analisis perbandingan algoritma iteratif dan rekursif

1. Kompleksitas Algoritma
Iteratif :

- Kompleksitas Waktu: O(*n*) per-iterasi karena terdapat *loop* yang berjalan selama jumlah hari.
- Kompleksitas Ruang: O(1), karena pengguna memori hanya untuk variabel local.

Rekursif:

- Kompleksitas Waktu: O(n), dimana *n* adalah jumlah hari. Sama dengan *iterative* karena setiap hari diproses satu kali.
- Kompleksitas Ruang: O(n), karena setiap pemanggilan rekursif membutuhkan tambahan memori untuk *stack* sehingga *n* pemanggilan sebelum mencapai kondisi dasar (*base case)*.

2. Kecepatan Eksekusi

Iteratif

- Waktu eksekusi cenderung lebih cepat dibandingkan rekursif karena tidak ada *overhead* untuk memanggil fungsi berkali-kali.
- Tidak perlu menangani *overhead* daei pemanggilan fungsi yang terjadi pada algoritma rekursif.

Rekursif

- Lebih lambat dibandingkan *iterative* karena adanya *overhead* dari pemanggilan fungsi rekursif.
- Untuk jumlah iterasi yang besar (misalnnya $hari > 10^5$), resiko ***stack overflow*** meningkat karena *stack* akan penuh sebelum mencapai kondisi dasar.

Hasil Empiris dari Kode:

- Dataset 1:
    - *Iterative*: Lebih cepat dibanding rekursif.
    - Rekursif: Lebih lambat karena *overhead.*
- Dataset 2:
    - *Iterative*: Tetap lebih cepat karena lebih sederhana dalam eksekusi.
    - Rekursif: Lebih lambat karena *overhead* rekursif.

E. Kompeksitas Kode

Iteratif

- Lebih mudah dimengerti karena menggunakan sturuktur *loop* sederhana.
- Tidak ada konsep rekursif, sehingga lebih cocok untuk pemula.

Rekursif

- Lebih sulit dimengerti, terutama untuk pemula yang tidak familiar dengan konsep rekursif.
- Sulit dikelola jika ada banyak dependesi atau logika tambahan di dalam fungsi rekursif.

Kesimpulan

- Algoritma iterative lebih sederhana dan mudah di-debug.
- Rekursif lebih elegan utuk masalah yang secara alami rekursif (misalnya, pohon atau graf).

F. Keterbatasan Eksekusi

Iterative

- Tidak memiliki Batasan selain kemampuan system untuk menangani jumlah iterasi yang besar.
- Performa stabil untuk jumlsh iterasi besar.

Rekursif

- Terbatas pada kedalaman *stack*. Jika hari terlalu besar, resiko ***stack overhead*** menjadi signifikan.
- Harus lebih berhati-hati jika digunakan dalam system dengan kapasitas *stack* kecil.

G. Estimasi Big-O untuk dataset

Dataset 1

- Dataset 1 mencakup hingga 100 iterasi
    - *Iterative:* Estimasi waktu O(n) tetapi lebih rendah karena efensiensi ruang.
    - Rekursif: Waktu eksekusi lebih lama karena *overhead* rekursif.

- Dataset 2 mencakup 10 iterasi
    - *Iterative:* tetap lebih efisien dan cepat
    - Peforma menurun signifikan untuk *hari* besar (eksponensial pada jurnal iterasi).

H. Estetika dan Fleksibilitas

Iterative

- Memiliki fleksibilitas tinggi karena dapat diatur logikanya dengan lebih mudah
- Kode tidak terikat pada struktur panggilan fungsi.

Rekursif

- Lebih estetis dan cocok untuk masalah rekursif alami.
- Kurang fleksibel dalam menangani perbuhan besar pada logika.

I. *Source Code*

I. Code Iteratif

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <math.h>

double hitung_konsentrasi_iteratif(int hari, double konsentrasi_awal, int
kualitas_tidur) {
    double konsentrasi = konsentrasi_awal;

    for (int i = 0; i < hari; i++) {
        if (kualitas_tidur <= 0) {
            konsentrasi *= 0.50;
        } else if (kualitas_tidur < 6) {
            konsentrasi *= 0.90;
        } else if (kualitas_tidur >= 6 && kualitas_tidur <= 8) {
            konsentrasi *= 0.95;
        }
        if (konsentrasi < 10.0) {
            konsentrasi = 10.0;
            break;
        }
    }
    return konsentrasi;
}
void buat_nama_mahasiswa(char nama[], int id) {
    const char* daftar_nama[] = {"Andi", "Budi", "Citra", "Dewi", "Eka", "Fajar",
"Gita", "Hendra", "Indah", "Joko"};
    sprintf(nama, "%s", daftar_nama[id % 10]);
```

```c
}

void evaluasi_kesehatan_mental(const char* nama, double konsentrasi_akhir, int
kualitas_tidur) {
    printf("\n=== Evaluasi Kesehatan Mental untuk %s ===\n", nama);


    if (kualitas_tidur < 6) {
        printf("Kualitas tidur: Rendah. Risiko depresi tinggi.\n");
    } else if (kualitas_tidur <= 8) {
        printf("Kualitas tidur: Sedang. Risiko depresi sedang.\n");
    } else {
        printf("Kualitas tidur: Tinggi. Risiko depresi ringan.\n");
    }

    if (konsentrasi_akhir < 50.0) {
        printf("Konsentrasi: Rendah. Ada risiko gangguan serius.\n");
    } else if (konsentrasi_akhir <= 80.0) {
        printf("Konsentrasi: Sedang. Ada risiko gangguan kecil.\n");
    } else {
        printf("Konsentrasi: Tinggi. Kondisi mental baik.\n");
    }

    printf("Rekomendasi: ");
    if (kualitas_tidur < 6) {
        printf("Tingkatkan waktu tidur, hindari kafein, dan kurangi stres.\n");
    } else if (kualitas_tidur <= 8) {
        printf("Pertahankan pola tidur, hindari gadget sebelum tidur.\n");
    } else {
        printf("Pertahankan kualitas tidur dan gaya hidup sehat.\n");
    }
}

void cetak_hasil(const char* dataset, int no_urut, const char* nama, int hari,
double konsentrasi_awal, double konsentrasi_akhir, int kualitas_tidur) {
    printf("\n%s: Data %d\n", dataset, no_urut);
    printf("Nama: %s\n", nama);
    printf("Konsentrasi awal: %.2f%%\n", konsentrasi_awal);
    printf("Jumlah hari: %d\n", hari);
    printf("Kualitas tidur rata-rata: %d jam/hari\n", kualitas_tidur);
    printf("Konsentrasi akhir setelah %d hari: %.2f%%\n", hari,
konsentrasi_akhir);

    evaluasi_kesehatan_mental(nama, konsentrasi_akhir, kualitas_tidur);
}
```

```c
int main() {
    int hari, kualitas_tidur;
    double konsentrasi_awal;
    char nama[50];

    clock_t start, end;
    double waktu_dataset1, waktu_dataset2;

    printf("=== Program Pengaruh Kualitas Tidur terhadap Konsentrasi ===\n");

    // Dataset 1
    printf("\n--- Dataset 1: Iteratif ---\n");
    start = clock();

    for (int i = 1; i <= 100; i++) {
        buat_nama_mahasiswa(nama, i);
        hari = i;
        kualitas_tidur = (i % 3 == 0) ? 6 : ((i % 2 == 0) ? 8 : 7);
        konsentrasi_awal = 100.0;

        double konsentrasi_akhir = hitung_konsentrasi_iteratif(hari,
konsentrasi_awal, kualitas_tidur);
        cetak_hasil("Dataset 1", i, nama, hari, konsentrasi_awal,
konsentrasi_akhir, kualitas_tidur);
    }

    end = clock();
    waktu_dataset1 = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("\nWaktu eksekusi Dataset 1: %.2f detik\n", waktu_dataset1);

    double estimasi_waktu_dataset1 = pow(100, 2) * waktu_dataset1 / 100;
    printf("Estimasi waktu eksekusi O(n^2) untuk Dataset 1: %.2f detik\n",
estimasi_waktu_dataset1);

    // Dataset 2
    printf("\n--- Dataset 2: Iteratif ---\n");
    start = clock();

    for (int i = 1; i <= 10; i++) {
        buat_nama_mahasiswa(nama, i + 100);

        hari = i * 2;
        kualitas_tidur = (i % 2 == 0) ? 7 : 8;
        konsentrasi_awal = 90.0;
```

```
        double konsentrasi_akhir = hitung_konsentrasi_iteratif(hari,
konsentrasi_awal, kualitas_tidur);
        cetak_hasil("Dataset 2", i, nama, hari, konsentrasi_awal,
konsentrasi_akhir, kualitas_tidur);
    }

    end = clock();
    waktu_dataset2 = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("\nWaktu eksekusi Dataset 2: %.2f detik\n", waktu_dataset2);
    double estimasi_waktu_dataset2 = pow(10, 2) * waktu_dataset2
    printf("Estimasi waktu eksekusi O(n^2) untuk Dataset 2: %.2f detik\n",
estimasi_waktu_dataset2);

    printf("\nProgram selesai. Terima kasih!\n");
    return 0;
}
```

II. Code Rekursif

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <math.h>

double hitung_konsentrasi_rekursif(int hari, double konsentrasi, int
kualitas_tidur) {
    if (hari == 0 || konsentrasi < 10.0) {
        return konsentrasi < 10.0 ? 10.0 : konsentrasi;
    }

    if (kualitas_tidur <= 0) {
        konsentrasi *= 0.50;
    } else if (kualitas_tidur < 6) {
        konsentrasi *= 0.90;
    } else if (kualitas_tidur >= 6 && kualitas_tidur <= 8) {
        konsentrasi *= 0.95;
    }

    return hitung_konsentrasi_rekursif(hari - 1, konsentrasi, kualitas_tidur);
}

void buat_nama_mahasiswa(char nama[], int id) {
    sprintf(nama, "Mahasiswa_%d", id);
```

```c
}

void cetak_hasil(const char *dataset, int no_urut, const char *nama, int hari,
double konsentrasi_awal, double konsentrasi_akhir, int kualitas_tidur) {
    printf("%s: Data %d\n", dataset, no_urut);
    printf("Nama: %s\n", nama);
    printf("Konsentrasi awal: %.2f%%\n", konsentrasi_awal);
    printf("Jumlah hari: %d\n", hari);
    printf("Kualitas tidur rata-rata: %d jam/hari\n", kualitas_tidur);
    printf("Konsentrasi akhir setelah %d hari: %.2f%%\n", hari,
konsentrasi_akhir);
    printf("\n=== Evaluasi Kesehatan Mental untuk %s ===\n", nama);
    if (kualitas_tidur < 6) {
        printf("Kualitas tidur: Rendah. Risiko depresi tinggi.\n");
    } else if (kualitas_tidur >= 6 && kualitas_tidur <= 8) {
        printf("Kualitas tidur: Sedang. Risiko depresi sedang.\n");
    } else {
        printf("Kualitas tidur: Tinggi. Risiko depresi rendah.\n");
    }

    if (konsentrasi_akhir < 50.0) {
        printf("Konsentrasi: Rendah. Ada risiko gangguan berat.\n");
    } else if (konsentrasi_akhir >= 50.0 && konsentrasi_akhir < 80.0) {
        printf("Konsentrasi: Sedang. Ada risiko gangguan kecil.\n");
    } else {
        printf("Konsentrasi: Tinggi. Tidak ada risiko gangguan.\n");
    }

    printf("Rekomendasi: Tingkatkan kualitas tidur dengan mengatur rutinitas
tidur dan hindari gadget sebelum tidur.\n");
    printf("============================================================\n\n");
}

int main() {
    int hari, kualitas_tidur;
    double konsentrasi_awal;
    char nama[50];

    clock_t start, end;
    double waktu_dataset1, waktu_dataset2;

    printf("=== Program Pengaruh Kualitas Tidur terhadap Konsentrasi (Rekursif)
===\n");

    // Dataset 1
```

```
    printf("\n--- Dataset 1: Rekursif ---\n");
    start = clock();

    for (int i = 1; i <= 100; i++) {
        buat_nama_mahasiswa(nama, i);
        hari = i;
        kualitas_tidur = (i % 3 == 0) ? 6 : ((i % 2 == 0) ? 8 : 7);
        konsentrasi_awal = 100.0;

        double konsentrasi_akhir = hitung_konsentrasi_rekursif(hari,
konsentrasi_awal, kualitas_tidur);
        cetak_hasil("Dataset 1", i, nama, hari, konsentrasi_awal,
konsentrasi_akhir, kualitas_tidur);
    }

    end = clock();
    waktu_dataset1 = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Waktu eksekusi Dataset 1: %.2f detik\n", waktu_dataset1);
    double estimasi_waktu_dataset1 = pow(100, 2) * waktu_dataset1 / 100;
    printf("Estimasi waktu eksekusi O(n^2) untuk Dataset 1: %.2f detik\n",
estimasi_waktu_dataset1);

    // Dataset 2
    printf("\n--- Dataset 2: Rekursif ---\n");
    start = clock();

    for (int i = 1; i <= 10; i++) {
        buat_nama_mahasiswa(nama, i + 100);
        hari = i * 2;
        kualitas_tidur = (i % 2 == 0) ? 7 : 8;
        konsentrasi_awal = 90.0;

        double konsentrasi_akhir = hitung_konsentrasi_rekursif(hari,
konsentrasi_awal, kualitas_tidur);
        cetak_hasil("Dataset 2", i, nama, hari, konsentrasi_awal,
konsentrasi_akhir, kualitas_tidur);
    }

    end = clock();
    waktu_dataset2 = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Waktu eksekusi Dataset 2: %.2f detik\n", waktu_dataset2);


    double estimasi_waktu_dataset2 = pow(10, 2) * waktu_dataset2 / 10;
```

```
    printf("Estimasi waktu eksekusi O(n^2) untuk Dataset 2: %.2f detik\n",
estimasi_waktu_dataset2);

    printf("\nProgram selesai. Terima kasih!\n");
    return 0;
}
```

References

Meita Dhamayanti, F. E. (2019). Hubungan Kualitas Tidur dan Masalah Mental Emosional pada Remaja Sekolah Menengah. *Sari Pediatri*, 05.

Moeis RM, K. R. (2023). Internasional Journal of General Medicine. *Correlation Betwen Adolescent Mental Healt and Slepp Quality: A Study in Indonesian Rural Areas Durong the COVID-19 Pandemic*, 16 pages 3203-3210.