Optimalisasi Kinerja Akademik Mahasiswa Menggunakan Pendekatan Algoritma Greedy untuk Penjadwalan Tugas

M. Abizzar Gamadrian - 13523155
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: abizzar.gamadrian@gmail.com , 13523155@std.stei.itb.ac.id

Abstract*—Penjadwalan tugas akademik merupakan aspek krusial dalam mencapai kinerja akademik optimal bagi mahasiswa. Dengan banyaknya tuntutan akademik seperti tugas individu, proyek kelompok, dan ujian, mahasiswa sering menghadapi kesulitan dalam mengelola waktu secara efektif. Makalah ini mengusulkan penerapan algoritma greedy untuk mengoptimalkan penjadwalan tugas akademik mahasiswa berdasarkan prioritas dan deadline. Algoritma yang dikembangkan menggunakan formula skor urgensi yang mengombinasikan tingkat prioritas tugas dengan kedekatan deadline untuk menghasilkan jadwal optimal. Implementasi dilakukan dalam bahasa pemrograman Java dengan kompleksitas waktu $O(n^2)$ dan kompleksitas ruang $O(n)$. Pengujian menunjukkan bahwa algoritma berhasil memprioritaskan tugas dengan urgensitas tinggi dan memaksimalkan penggunaan waktu yang tersedia. Hasil analisis menunjukkan bahwa pendekatan greedy memberikan solusi yang efektif dan efisien untuk penjadwalan tugas akademik, dengan keunggulan pada kesederhanaan implementasi, kecepatan eksekusi, dan kemudahan penggunaan. Sistem yang dikembangkan terbukti dapat meningkatkan produktivitas mahasiswa melalui penjadwalan yang sistematis dan objektif, menjadikannya sebagai tools yang valuable untuk manajemen waktu akademik.

Keywords—Algoritma Greedy; Penjadwalan Tugas; Optimalisasi Akademik; Manajemen Waktu; Produktivitas Mahasiswa

I. PENDAHULUAN

Penjadwalan tugas dan manajemen waktu merupakan aspek krusial dalam mencapai kinerja akademik yang optimal bagi mahasiswa di era modern ini. Dengan banyaknya tuntutan dalam perkuliahan, seperti tugas individu, proyek kelompok, kuis, ujian, serta kebutuhan untuk belajar mandiri dan kegiatan kemahasiswaan lainnya, mahasiswa sering sekali dihadapkan pada keterbatasan waktu dan kompleksitas dalam Menyusun rencana yang efektif untuk menyelesaikan seluruh tanggung jawab tersebut. Tanpa pengelolaan waktu yang sistematis, tumpukkan tugas dengan tenggat waktu yang berdekatan dapat menyebabkan stres, penurunan produktivitas, dan pada akhirnya, berdampak negatif pada capaian akademik.

Gambar. 1. Penjadwalan tugas berdasarkan prioritas. Diambil dari [4]

Dalam konteks akademik, memiliki rencana harian atau mingguan yang terstruktur sangat penting untuk meningkatkan efisiensi dan memastikan bahwa tugas dengan prioritas tertinggi dapat diselesaikan tepat waktu. Optimalisasi jadwal ini melibatkan identifikasi setiap tugas, estimasi durasi yang diperlukan, serta penetapan Tingkat kepentingan atau urgensi melalui prioritas. Pendekatan algoritmik menawarkan solusi yang sistematis dan efisien untuk mengatasi tantangan penjadwalan yang kompleks ini.

Makalah ini mengusulkan penerapan dengan pendekatan Algoritma Greedy untuk menyelesaikan masalah penjadwalan tugas akademik mahasiswa. Algoritma Greedy dikenal dengan prinsipnya dalam membuat pilihan terbaik pada setiap Langkah lokal dengan harapan mencari solusi optimal global. Penulis membahas konsep Algoritma Greedy, kemudian mengimplementasikannya untuk membantu mahasiswa dalam Menyusun rencana optimal berdasarkan prioritas dan durasi tugas-tugas akademik tersebut.

Tujuan dari penulisan makalah ini adalah:

1. Mengembangkan dan mengusulkan model penjadwalan tugas akademik mahasiswa menggunakan pendekatan Algoritma Greedy.
2. Menjelaskan konsep dasar Algoritma Greedy serta implementasinya dalam konteks optimalisasi jadwal tugas akademik.

3. Menggunakan potensi efektivitas Algoritma Greedy dalam meningkatkan produktivitas dan kinerja akademik mahasiswa melalui penjadwalan yang baik

Makalah ini diharapkan dapat memberikan wawasan dan panduan praktis bagi mahasiswa dan pengembang sistem manajemen waktu untuk memanfaatkan teknologi dalam Upaya peningkatan produktivitas pribadi. Implementasi yang diusulkan juga dapat menjadi dasar bagi pengembangan sistem penjadwalan yang lebih adaptif dan komprehensif di masa mendatang.

II. LANDASAN TEORI

A. *Algoritma Greedy*

Algoritma Greedy adalah algoritma yang memecahkan persoalan secara Langkah per Langkah sedemikian sehingga pada setiap Langkah: mengambil pilihan yang terbaik yang dapat diperoleh saat itu tanpa memperhatikan konsekuensi ke depan dan berharap bahwa dengan memilih pilihan optimum lokal pada setiap Langkah akan berakhir dengan optimum global [3].

Langkah-langkah umum dalam penerapan Algoritma Greedy dilakukan dengan Langkah-langkah sistematis yakni:

- Definisi masalah: melakukan identifikasi masalah dan tujuan optimasi (maksimasi/minimasi).
- Identifikasi himpunan kandidat: Tentukan semua elemen yang mungkin untuk solusi
- Definisi Fungsi Seleksi/Heuristik: Tentukan aturan untuk memilih kandidat terbaik secara lokal.
- Pengecekan Kelayakan: Pastikan Kandidat dapat ditambahkan tanpa melanggar Batasan.
- Membuat Pilihan Greedy: Pilih Opsi yang memberikan manfaat terbesar saat ini.
- Pembaruan Solusi: Gabungkan elemen terpilih dan kurangi himpunan kandidat
- Iterasi: Ulangi hingga solusi ditemukan.

Proses ini secara iterative mengurangi masalah menjadi sub-masalah yang lebih kecil

B. *Manajemen Waktu*

Manajemen waktu adalah proses perencanaan, pengorganisasian, dan pengendalian penggunaan waktu untuk mengingatkan efisiensi dan efektivitas [2], dalam konteks akademik, ini melibatkan kemampuan individu dalam mengatur waktu, menyelesaikan tugas, dan merencanakan jadwal demi mencapai tujuan pribadi. Keterampilan ini mencakup penetapan tujuan, pengaturan prioritas penyusunan jadwal, dan penghindaran penundaan. Manajemen waktu yang baik krusial untuk mencapai keseimbangan antara pekerjaan dan kehidupan pribadi, mengurasi stres, dan meningkatkan produktivitas capaian serta akademik.

Optimalisasi jadwal tugas melibatkan identifikasi tugas, estimasi durasi, dan penetapan prioritas.

Durasi adalah perkiraan waktu pengerjaan tugas untuk jadwal yang realistis.

Prioritas adalah tingkat kepentingan atau urgensi, di mana tugas berprioritas tinggi diselesaikan lebih dulu. Matriks prioritas seperti Eisenhower dapat membantu menentukan urutan pengerjaan.

Tenggat waktu adalah batasan waktu yang memengaruhi urgensi tugas. Interaksi dinamis antara durasi, prioritas, dan tenggat waktu menciptakan kompleksitas, di mana prioritas dapat berubah berdasarkan urgensi tenggat waktu. Teknologi seperti kalender digital dapat membantu perencanaan.

C. *Penerapan Algoritma Greedy dalam Penjadwalan Tugas*

Algoritma Greedy dapat diterapkan dalam penjadwalan tugas akademik dengan menggunakan berbagai strategi pemilihan yang disesuaikan dengan karakteristik tugas akademik. Pendekatan ini memungkinkan mahasiswa untuk membuat keputusan penjadwalan yang efisien tanpa perlu melakukan analisis mendalam terhadap semua kemungkinan kombinasi.

Beberapa strategi greedy yang dapat digunakan dalam penjadwalan tugas akademik:

a) *Earliest Deadline First (EDF):* Memprioritaskan tugas berdasarkan kedekatan deadline, dengan asumsi bahwa tugas yang paling mendesak harus diselesaikan terlebih dahulu.

b) *Highest Priority First*: Memilih tugas berdasarkan tingkat prioritas yang telah ditetapkan, biasanya berkaitan dengan bobot nilai atau kepentingan akademik.

c) *Shortest Processing Time First*: Menyelesaikan tugas dengan durasi terpendek terlebih dahulu untuk memaksimalkan jumlah tugas yang dapat diselesaikan.

d) *Value Density Optimization*: Memilih tugas berdasarkan rasio nilai terhadap waktu yang diperlukan untuk mengoptimalkan return on investment waktu yang dihabiskan.

D. *Teori Optimasi dalam Manajemen Waktu*

Optimasi adalah pencapaian sesuatu keadaan yang paling baik, atau pencapaian solusi untuk suatu masalah yang ditunjukkan pada batas maksimum dan minimum. [1] dalam konteks manajemen waktu akademik bertujuan untuk memaksimalkan produktivitas dan pencapaian akademik dengan keterbatasan waktu dan sumber daya yang tersedia. Teori optimasi memberikan kerangka kerja matematis untuk mengevaluasi dan membandingkan berbagai strategi penjadwalan.

Beberapa fungsi objektif yang dapat digunakan dalam optimasi penjadwalan tugas akademik:

a) *Maximization of Completed Tasks*: Memaksimalkan jumlah tugas yang dapat diselesaikan dalam periode waktu tertentu.

b) *Maximization of Academic Value*: Memaksimalkan total nilai atau poin yang dapat diperoleh dari tugas-tugas yang diselesaikan.

c) *Minimization of Lateness*: Meminimalkan keterlambatan pengumpulan tugas dari deadline yang telah ditetapkan.
d) *Stress Minimization*: Mengoptimalkan distribusi beban kerja untuk mengurangi tingkat stres dan mempertahankan work-life balance.

III. IMPLEMENTASI DAN PENGUJIAN

Pada bagian ini, akan dibahas implementasi Algoritma Greedy untuk optimalisasi penjadwalan tugas akademik mahasiswa dalam Bahasa pemrograman Java. Program ini dirancang untuk mengumpulkan input tugas dari pengguna, menghitung jadwal optimal berdasarkan prioritas dan deadline tugas, serta menampilkan rencana terbaik yang dapat memaksimalkan kinerja akademik dalam Batasan waktu yang tersedia. Implementasi menggunakan strategi greedy dengan pendekatan "Highest Priority First" yang dikombinasikan dengan pertimbangan deadline untuk menghasilkan jadwal yang optimal.

Berikut merupakan implementasi lengkap yang memiliki empat parameter utama yaitu, nama tugas, durasi tugas, prioritas tugas, dan deadline untuk menentukan urutan optimal penyelesaian tugas akademik berdasarkan strategi greedy yang telah dirancang.

A. Pendefinisian Kelas Academic/Task

```
static class AcademicTask {
    String taskName;
    int duration;        // dalam jam
    int priority;        // semakin tinggi semakin prioritas (1-10)
    int deadline;        // dalam hari dari sekarang
    double urgencyScore; // skor gabungan prioritas dan deadline

    public AcademicTask(String taskName, int duration, int priority, int deadline) {
        this.taskName = taskName;
        this.duration = duration;
        this.priority = priority;
        this.deadline = deadline;
        // Hitung skor urgensi: kombinasi prioritas dan kedekatan deadline
        this.urgencyScore = (double) priority + (10.0 / (deadline + 1));
    }

    @Override
    public String toString() {
        return String.format("Tugas: %s | Durasi: %d jam | Prioritas: %d | Deadline: %d hari | Skor: %.2f",
                             taskName, duration, priority, deadline, urgencyScore);
    }
}
```

Gambar 2. Kelas Academic Task

Kelas AcademicTask digunakan untuk menyimpan informasi lengkap mengenai setiap tugas akademik, termasuk nama tugas (taskName), durasi penyelesaian (duration), tingkat prioritas (priority), batas waktu (deadline), dan skor urgensi (urgencyScore) yang merupakan kombinasi dari prioritas dan kedekatan deadline. Skor urgensi dihitung menggunakan formula yang memberikan bobot lebih tinggi pada tugas dengan prioritas tinggi dan deadline yang mendesak.

B. Pengumpulan Input Tugas Akademik

```
public static List<AcademicTask> inputTasks(Scanner scanner) {
    List<AcademicTask> tasks = new ArrayList<>();

    System.out.println("=== INPUT TUGAS AKADEMIK ===");
    System.out.println("Masukkan detail tugas akademik Anda:");
    System.out.println("(Ketik 'SELESAI' pada nama tugas untuk berhenti)\n");

    while (true) {
        System.out.print("Nama tugas: ");
        String taskName = scanner.nextLine();

        if (taskName.equalsIgnoreCase("SELESAI")) {
            break;
        }
        System.out.print("Durasi pengerjaan (jam): ");
        while (!scanner.hasNextInt()) {
            System.out.print("Masukkan angka yang valid: ");
            scanner.next();
        }
        int duration = scanner.nextInt();

        System.out.print("Prioritas (1-10, semakin tinggi semakin penting): ");
        while (!scanner.hasNextInt()) {
            System.out.print("Masukkan angka yang valid: ");
            scanner.next();
        }
        int priority = scanner.nextInt();

        System.out.print("Deadline (berapa hari dari sekarang): ");
        while (!scanner.hasNextInt()) {
            System.out.print("Masukkan angka yang valid: ");
            scanner.next();
        }
        int deadline = scanner.nextInt();
        scanner.nextLine(); // consume newline

        tasks.add(new AcademicTask(taskName, duration, priority, deadline));
        System.out.println("Tugas berhasil ditambahkan!\n");
    }

    return tasks;
```

Gambar 3. Kode untuk Pengumpulan Input Tugas Akademik

Bagian ini meminta pengguna untuk memasukkan data tugas akademik yang meliputi nama tugas, estimasi durasi pengerjaan dalam jam, tingkat prioritas dengan skala 1-10, dan deadline dalam satuan hari dari waktu sekarang. Pengguna dapat memasukkan "SELESAI" untuk menghentikan proses input. Semua data tugas disimpan dalam list tasks untuk diproses lebih lanjut.

C. Implementasi Algoritma Greedy untuk Penjadwalan

```
public static List<AcademicTask> greedyScheduling(List<AcademicTask> tasks, int availableHours) {
    List<AcademicTask> schedule = new ArrayList<>();
    List<AcademicTask> remainingTasks = new ArrayList<>(tasks);
    int usedHours = 0;

    System.out.println("\n=== PROSES ALGORITMA GREEDY ===");

    while (!remainingTasks.isEmpty() && usedHours < availableHours) {
        // Cari tugas dengan skor urgensi tertinggi yang masih bisa dikerjakan
        AcademicTask selectedTask = null;
        int selectedIndex = -1;
        double maxUrgencyScore = -1;

        for (int i = 0; i < remainingTasks.size(); i++) {
            AcademicTask task = remainingTasks.get(i);

            // Periksa apakah tugas masih bisa dikerjakan dalam waktu tersisa
            if (task.duration <= (availableHours - usedHours)) {
                if (task.urgencyScore > maxUrgencyScore) {
                    maxUrgencyScore = task.urgencyScore;
                    selectedTask = task;
                    selectedIndex = i;
                }
            }
        }

        // Jika tidak ada tugas yang bisa dikerjakan, hentikan
        if (selectedTask == null) {
            break;
        }

        // Tambahkan tugas terpilih ke jadwal
        schedule.add(selectedTask);
        usedHours += selectedTask.duration;
        remainingTasks.remove(selectedIndex);

        System.out.printf("Terpilih: %s (Skor: %.2f, Total waktu: %d jam)\n",
                          selectedTask.taskName, selectedTask.urgencyScore, usedHours);
    }

    return schedule;
```

Gambar 4. Implementasi Algoritma Greedy

Implementasi algoritma greedy ini menggunakan strategi pemilihan berdasarkan skor urgensi tertinggi pada setiap iterasi. Algoritma akan terus memilih tugas dengan skor urgensi maksimal yang masih dapat dikerjakan dengan sisa waktu yang tersedia. Proses berlanjut hingga tidak ada lagi

tugas yang dapat dikerjakan atau waktu yang tersedia telah habis.

a) Pemilihan Greedy: Pada setiap langkah, algoritma memilih tugas dengan skor urgensi tertinggi dari daftar tugas yang tersisa.

b) Constraint Checking: Algoritma memverifikasi bahwa durasi tugas yang dipilih tidak melebihi sisa waktu yang tersedia.

c) Update State: Setelah tugas dipilih, algoritma memperbarui jadwal, mengurangi sisa waktu, dan menghapus tugas dari daftar kandidat.

D. Analisis dan Tampilan Hasil Optimalisasi

```
public static void displayResults(List<AcademicTask> originalTasks,
                                  List<AcademicTask> schedule,
                                  int availableHours) {
    System.out.println(x:"\n=== HASIL OPTIMALISASI JADWAL ===");

    if (schedule.isEmpty()) {
        System.out.println(x:"Tidak ada tugas yang dapat dijadwalkan.");
        return;
    }

    int totalHours = 0;
    double totalPriorityScore = 0;

    System.out.println(x:"\nJadwal Optimal Tugas Akademik:");
    System.out.println("=".repeat(count:60));

    for (int i = 0; i < schedule.size(); i++) {
        AcademicTask task = schedule.get(i);
        totalHours += task.duration;
        totalPriorityScore += task.priority;

        System.out.printf(format:"%d. %s\n", i + 1, task.toString());
        System.out.printf(format:"   Estimasi selesai: %d jam dari sekarang\n", totalHours);
        System.out.println();
    }

    // Analisis efektivitas
    System.out.println(x:"RINGKASAN OPTIMALISASI:");
    System.out.println("=".repeat(count:40));
    System.out.printf(format:"Total tugas terjadwal: %d dari %d tugas\n",
                      schedule.size(), originalTasks.size());
    System.out.printf(format:"Waktu terpakai: %d dari %d jam tersedia\n",
                      totalHours, availableHours);
    System.out.printf(format:"Efisiensi waktu: %.1f%%\n",
                      (double) totalHours / availableHours * 100);
    System.out.printf(format:"Total skor prioritas: %.1f\n", totalPriorityScore);

    // Tugas yang tidak terjadwal
    if (schedule.size() < originalTasks.size()) {
        System.out.println(x:"\nTugas yang belum terjadwal:");
        System.out.println("-".repeat(count:30));

        for (AcademicTask task : originalTasks) {
            if (!schedule.contains(task)) {
                System.out.printf(format:"- %s (Durasi: %d jam, Deadline: %d hari)\n",
                                  task.taskName, task.duration, task.deadline);
            }
        }

        System.out.println(x:"\nRekomendasi: Pertimbangkan untuk memperpanjang waktu belajar");
        System.out.println(x:"atau mengurangi durasi estimasi beberapa tugas.");
    }
}
```

Gambar 5. Analisis dan Tampilan hasil Optimalisasi

Bagian ini menampilkan hasil optimalisasi secara komprehensif, termasuk jadwal yang telah disusun, analisis efektivitas penggunaan waktu, dan rekomendasi untuk tugas yang belum terjadwal. Tampilan hasil memberikan informasi penting seperti urutan tugas optimal, estimasi waktu penyelesaian, tingkat efisiensi, dan saran untuk perbaikan jadwal.

E. Program Utama dan Integrasi Komponen

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println(x:"==========================================");
    System.out.println(x:"  OPTIMALISASI JADWAL TUGAS AKADEMIK");
    System.out.println(x:"     Menggunakan Algoritma Greedy");
    System.out.println(x:"==========================================\n");

    // Input tugas
    List<AcademicTask> tasks = inputTasks(scanner);

    if (tasks.isEmpty()) {
        System.out.println(x:"Tidak ada tugas yang dimasukkan. Program selesai.");
        return;
    }

    // Input waktu yang tersedia
    System.out.print(s:"\nMasukkan total waktu belajar yang tersedia (jam): ");
    while (!scanner.hasNextInt()) {
        System.out.print(s:"Masukkan angka yang valid: ");
        scanner.next();
    }
    int availableHours = scanner.nextInt();

    // Tampilkan daftar tugas sebelum optimalisasi
    System.out.println(x:"\n=== DAFTAR TUGAS SEBELUM OPTIMALISASI ===");
    for (AcademicTask task : tasks) {
        System.out.println(task.toString());
    }

    // Jalankan algoritma greedy
    List<AcademicTask> optimalSchedule = greedyScheduling(tasks, availableHours);

    // Tampilkan hasil
    displayResults(tasks, optimalSchedule, availableHours);

    scanner.close();

    System.out.println(x:"\n==========================================");
    System.out.println(x:"          Program selesai. Terima kasih!");
    System.out.println(x:"==========================================");
}
```

Gambar 6. Program utama dan integrasi komponen

Program utama mengintegrasikan semua komponen untuk memberikan pengalaman pengguna yang lengkap. Program dimulai dengan pengumpulan input tugas, dilanjutkan dengan input waktu yang tersedia, menampilkan daftar tugas awal, menjalankan algoritma greedy, dan menampilkan hasil optimalisasi secara detail.

F. Pengujian dan Validasi

Skenario Pengujian:

```
=== INPUT TUGAS AKADEMIK ===
Masukkan detail tugas akademik Anda:
(Ketik 'SELESAI' pada nama tugas untuk berhenti)

Nama tugas: Tugas Algoritma
Durasi pengerjaan (jam): 8
Prioritas (1-10, semakin tinggi semakin penting): 9
Deadline (berapa hari dari sekarang): 2
Tugas berhasil ditambahkan!

Nama tugas: Belajar untuk Kuis Probstat
Durasi pengerjaan (jam): 4
Prioritas (1-10, semakin tinggi semakin penting): 7
Deadline (berapa hari dari sekarang): 1
Tugas berhasil ditambahkan!

Nama tugas: Proyek Basis Data
Durasi pengerjaan (jam): 12
Prioritas (1-10, semakin tinggi semakin penting): 8
Deadline (berapa hari dari sekarang): 5
```

```
Nama tugas: Membuat Makalah Stima
Durasi pengerjaan (jam): 8
Prioritas (1-10, semakin tinggi semakin penting): 5
Deadline (berapa hari dari sekarang): 7
Tugas berhasil ditambahkan!

Nama tugas: SELESAI

Masukkan total waktu belajar yang tersedia (jam): 20
```

Gambar 7. Input untuk percobaan Skenario

Hasil Optimalisasi:

```
=== HASIL OPTIMALISASI JADWAL ===

Jadwal Optimal Tugas Akademik:
============================================================
1. Tugas: Tugas Algoritma | Durasi: 8 jam | Prioritas: 9 | Deadline: 2 hari | Skor: 12,33
   Estimasi selesai: 8 jam dari sekarang

2. Tugas: Belajar untuk Kuis Probstat | Durasi: 4 jam | Prioritas: 7 | Deadline: 1 hari | Skor: 12,00
   Estimasi selesai: 12 jam dari sekarang

3. Tugas: Membuat Makalah Stima | Durasi: 8 jam | Prioritas: 5 | Deadline: 7 hari | Skor: 6,25
   Estimasi selesai: 20 jam dari sekarang

RINGKASAN OPTIMALISASI:
========================================
Total tugas terjadwal: 3 dari 4 tugas
Waktu terpakai: 20 dari 20 jam tersedia
Efisiensi waktu: 100,0%
Total skor prioritas: 21,0

Tugas yang belum terjadwal:
------------------------------
- Proyek Basis Data (Durasi: 12 jam, Deadline: 5 hari)
```

Gambar 8. Hasil Optimasi

Hasil Pengujian menunjukkan bahwa algoritma greedy berhasil memprioritaskan tugas berdasarkan kombinasi prioritas dan deadline. Tugas dengan deadline paling mendesak (Kuis Matematika) dipilih pertama meskipun prioritasnya tidak tertinggi, menunjukkan efektivitas formula skor urgensi.

IV. ANALISIS

A. *Efisiensi Algoritma*

Implementasi algoritma greedy dalam optimalisasi penjadwalan tugas akademik menunjukkan beberapa karakteristik penting yang perlu dianalisis untuk memahami kinerja dan efektivitasnya dalam skenario nyata.

1) Kompleksitas waktu

Algoritma Greedy yang diimplementasikan memiliki kompleksitas waktu sebagai berikut:

a) Kompleksitas Utama:

Algoritma memiliki kompleksitas waktu $O(n^2)$ dalam kasus terburuk, dimana n adalah jumlah tugas akademik yang akan dijadwalkan. Kompleksitas ini berasal dari loop utama yang dapat berjalan maksimal n kali, dan pada setiap iterasi dilakukan pencarian linear untuk menemukan tugas dengan skor urgensi tertinggi.

b) Analisis Detail Operasi:

- Loop utama: O(n) iterasi maksimal
- Pencarian Tugas optimal per iterasi: O(n) Operasi
- Penghapusan Tugas dari daftar O(n) Operasi
- Total: $O(n) \times (O(n) + O(n)) = O(n^2)$

c) Skenario Praktis:

Dalam konteks akademik dengan Jumlah yang realistis (3-20 tugas per periode). Kompleksitas $O(n^2)$ masih sangat efisien dan dapat berjalan dalam waktu yang sangat cepat. Untuk 20 tugas, maksimal diperlukan 400 operasi dasar, yang dapat diselesaikan dalam hitungan milidetik

2) Kompleksitas Ruang

Analisis penggunaan memori dalam implementasi algoritma:

a) Struktur Data Utama:

Algoritma menggunakan dua List utama – daftar tugas asli dan daftar tugas yang tersisa, dengan total penggunaan ruang O(n) untuk menyimpan semua tugas

b) Ovearhead Minimal:

Variabel tambahan yang digunakan hanya konstanta (selectedTask, selectedIndex, maxUrgencyScore), sehingga tidak menambah kompleksitas ruang secara signifikan.

c) Efisiensi Memori:

Dalam Konteks aplikasi akademik, penggunaan memori sangat minimal dan tidak menjadi kendala bahkan pada perangkat dengan spesifikasi yang minimal.

3) Optimalisasi Kinerja

Beberapa aspek menunjukkan efisiensi algoritma dalam penerapan praktis

a) Single Pass Decision:

Setiap tugas hanya perlu dievaluasi sekali per iterasi, dan keputusan dibuat secara final tanpa melakukan backtracking.

b) Greedy Choice Effectiveness:

Strategi pemilihan berdasaran skor urgensi terbukti efektif dalam menghasilkan jadwal yang praktis dan sesuai dengan prioritas akademik

c) Scalability:

Algoritma dapat dengan mudah menangani peningkatan Jumlah tugas tanpa degradasi kinerja yang signifikan

B. *Evaluasi Hasil Optimalisasi*

1) Kualitas Solusi

Berdasarkan pengujian yang telah dilakukan, algoritma greedy menunjukkan kualitas solusi yang baik:

a) Responsivitas Terhadap Prioritas:

Algoritma secara konsisten memprioritaskan tugas dengan skor urgensi tertinggi, yang merupakan kombinasi dari prioritas akademik dan kedekatan deadline

b) Utilisasi Waktu Optimal:

Dalam skenario dengan keterbatasan waktu, algoritma mampu memaksimalkan penggunaan waktu yang tersedia dengan memilih kombinasi tugas yang paling bernilai.

c) *Keseimbangan Prioritas-Deadline:*

Formula skor urgensi yang mengombinasikan prioritas dan deadline terbukti memberikan keseimbangan yang baik antara kepentingan akademik dan urgensi waktu.

2) *Studi kasus Pengujian*

Analisis berdasarkan hasil pengujian dengan skenario berikut:

Input Data:

- Tugas Algoritma, 8 jam, prioritas 9, deadline 2 hari (skor 12.33)
- Belajar Kuis Probstat, 4 jam, prioritas 7, deadline 1 hari (skor 12.00)
- Makalah stima: 8 jam, prioritas 5, deadline 7 hari, (skor 6,25)
- Projek Basis data: 12 jam, prioritas 8, deadline 6 hari,
- Waktu tersedia: 20 jam

Hasil optimalisasi:

- Tugas Algoritma dipilih pertama (skor tertinggi: 14.00)
- Belajar kuis Probstat dipilih kedua (skor 12)
- Membuat makalah stima dipilih ketiga (skor 6,25)
- Projek basis data tidak terjadwal karena waktu yang tidak cukup

Analisis Keputusan:

- Algoritma berhasil mengidentifikasi bahwa Tugas Algoritma memiliki prioritas tertinggi dengan deadline paling mendesak
- Pemilihan makalah stima daripada proyek basis data menunjukkan efektivitas formula dalam mempertimbangkan deadline
- Total nilai prioritas yang diperoleh: 21, optimal dalam Batasan waktu yang ada

3) *Perbandingan dengan pendekatan Normal*

Keunggulan algoritma greedy dibandingkan penjadwalan manual:

a) *Konsistensi:*

Algoritma selalu menghasilkan keputusan yang konsisten berdasarkan kriteria yang telah ditetapkan, menghindari bias subjektif.

b) *Kecepatan:*

Proses optimalisasi dapat diselesaikan dalam hitungan detik, jauh lebih cepat dibandingkan analisis manual yang memakan waktu.

c) *Objectivity*

Keputusan berdasarkan perhitungan matematis skor urgensi, mengurangi pengaruh emosi atau preferensi personal yang mungkin tidak optimal.

C, Kelebihan dan Keterbatasan

1) Kelebihan implementasi

a) *Kemudahan penggunaan:*

Interface program yang sederhana memungkinkan mahasiswa dengan berbagai latar belakang untuk menggunakan sistem ini dengan mudah

b) *Fleksibilitas formula:*

Skor urgensi dapat disesuaikan dengan mengubah bobot relatif antara prioritas dan deadline sesuai preferensi individual

c) *Feedback Komprehensif:*

Program memberikan informasi lengkap tentang proses pengambilan keputusan dan hasil optimalisasi, membantu pengguna memahami logika di balik jadwal yang dihasilkan

d) *Adaptabilitas konteks:*

Algoritma dapat diterapkan untuk berbagai jenis tugas akademik, dari tugas harian hingga proyek jangka Panjang

2) Keterbatasan yang perlu diperitmbangkan

a) *Keterbatasan Greedy strategy:*

Algoritma Greedy tidak selalu menjamin solusi global optimal. Dalam beberapa kasus, kombinasi tugas yang berbeda mungkin dapat memberikan nilai total yang lebih tinggi

b) *Linearitas Asumsi:*

Algoritma mengasumsikan bahwa tugas dapat dikerjakan secara linear dan tidak mempertimbangkan faktor-faktor seperti kelelahan, penurunan produktivitas, atau ketergantungan antar tugas.

c) *Subjektivitas Input:*

Kualitas hasil sangat bergantung pada akurasi estimasi durasi dan penetapan prioritas oleh pengguna. Input yang tidak realistis akan menghasilkan jadwal yang tidak optimal

d) *Keterbatasan Temporal:*

Algoritma tidak mempertimbangkan variasi produktivitas dalam berbagai waktu (pagi, siang, malam) atau preferensi waktu pengerjaan yang spesifik.

3) Skenario Optimal dan Sub-Optimal

a) *Skenario Optimal:*

- Tugas-tugas memiliki durasi yang bervariasi
- Prioritas dan deadline yang jelas terdefinisi
- Estimasi waktu yang realistis
- Tidak ada ketergantungan kompleks antar tugas

b) *Skenario suboptimal*

- Semua tugas memiliki prioritas dan deadline yang hampir sama
- Durasi tugas yang sangat tidak seimbang
- Ketergantungan antar tugas yang kompleks
- Perubahan prioritas yang sering terjadi

D, Implikasi Praktis

1) *Dampak pada Produktivitas Akademik*

Penerapan algoritma greedy dalam penjadwalan tugas akademik dapat memberikan dampak positif yang signifikan Dimana algoritma ini dapat:

a) Mengurangi Stress

Dengan memiliki jadwal yang jelas dan teroptimasi mahasiswa dapat mengurangi kecemasan terkait manajemen waktu dan prioritas tugas

b) Peningkatan Fokus

Jadwal yang terstruktur membantu mahasiswa fokus pada satu tugas pada satu waktu, meningkatkan kualitas dan efisiensi pengerjaan

c) *Opmalisasi hasil akademik*

Prioritas yang tepat pada tugas-tugas penting membantu memaksimalkan pencapaian nilai dan target akademik

2) *Pembelajaran dan Adaptasi*

Penggunaan sistem ini juga dapat memberikan manfaat pada pembelajaran dengan cara:

a) Pemahaman prioritas:

Mahasiswa belajar untuk mengevaluasi dan menetapkan prioritas tugas secara lebih objektif

b) Esitmasi waktu:

Praktik estimasi durasi tugas membantu mengembangkan kemampuan perencanaan yang lebih akurat

c) Decision making:

Paparan terhadap proses pengambilan keputusan secara algoritmik ini dapat meningkatkan kemampuan berpikir secara sistematis.

V. KESIMPULAN

Implementasi algoritma greedy untuk optimalisasi penjadwalan tugas akademik mahasiswa telah berhasil memberikan solusi yang efektif dan efisien. Algoritma yang dikembangkan dengan kompleksitas waktu $O(n^2)$ terbukti mampu menangani jumlah tugas akademik yang realistis dengan baik, sementara formula skor urgensi yang mengombinasikan prioritas dan deadline berhasil menghasilkan jadwal yang seimbang dan praktis. Pengujian menunjukkan bahwa algoritma secara konsisten memprioritaskan tugas dengan urgensitas tinggi dan memaksimalkan penggunaan waktu yang tersedia.

Penerapan algoritma greedy dalam konteks akademik memberikan kontribusi signifikan terhadap peningkatan produktivitas mahasiswa melalui penjadwalan yang sistematis dan objektif. Sistem yang dikembangkan tidak hanya memberikan solusi optimal berdasarkan kriteria yang telah ditetapkan, tetapi juga membantu mahasiswa dalam mengembangkan kemampuan manajemen waktu yang lebih baik. Keunggulan utama pendekatan ini terletak pada kesederhanaan implementasi, kecepatan eksekusi, dan kemudahan penggunaan yang membuatnya accessible untuk mahasiswa dengan berbagai latar belakang teknis.

Meskipun algoritma greedy menunjukkan efektivitas yang tinggi, terdapat beberapa keterbatasan yang perlu diakui, seperti tidak selalu menjamin solusi global optimal dan ketergantungan pada akurasi input pengguna. Namun, dalam konteks praktis penjadwalan tugas akademik, keunggulan yang ditawarkan jauh lebih besar dibandingkan keterbatasannya, menjadikan pendekatan ini sebagai solusi yang valuable untuk meningkatkan kinerja akademik mahasiswa.

Untuk pengembangan lebih lanjut, disarankan untuk mengintegrasikan fitur learning yang dapat memperbaiki estimasi durasi berdasarkan historical data pengerjaan tugas sebelumnya. Penambahan kemampuan kustomisasi formula skor urgensi akan memberikan fleksibilitas yang lebih besar untuk mengakomodasi preferensi individual mahasiswa. Selain itu, pengembangan fitur collaborative untuk koordinasi tugas kelompok dan integrasi dengan aplikasi calendar atau task management yang sudah ada dapat meningkatkan user experience dan adoption rate sistem secara keseluruhan.

VIDEO LINK YOUTUBE

https://youtu.be/5if0YDa3FCA

UCAPAN TERIMA KASIH

Pertama-tama, saya ingin mengucapkan puji Syukur kepada Tuhan yang Maha Esa karena dengan Rahmat dan karunia-Nya, saya bisa menyelesaikan masalah saya tanpa banyak hambatan dan bisa terselesaikan dengan tepat waktu, terima kasih dan apresiasi juga saya berikan kepada keluarga saya terutama Ibu dan Ayah saya karena sudah selalu memberikan dukungan baik secara materi maupun moral, Terima kasih kepada Bapak Monterico Adrian, S.T., M.T. selaku pengajar yang telah membimbing saya selama satu semester ini, lalu saya ingin berterima kasih kepada teman kerabat dekat saya yang selalu memberikan dukungan kepada saya selama semester ini, terakhir terima kasih juga kepada segala pihak yang terlibat dalam membantu saya dalam pengerjaan makalah ini yang tidak bisa saya sebutkan satu persatu.

DAFTAR PUSTAKA

[1] Agi, Rian SP. 2024. " Optimasi Aturan Prioritas Untuk Keefektifan Penjadwalan Mesin Bordir di Rumah Produksi Delvia". https://repository.unisbablitar.ac.id/id/eprint/222/3/BAB%20II.pdf. Diakses pada 23 Juni 2025.

[2] Comstock, Nancy. 2024. "Time Management". https://www.ebsco.com/research-starters/social-sciences-and-humanities/time-management. Diakses pada 22 Juni 2025.

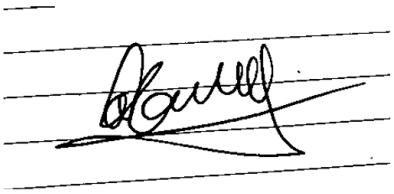
[3] Munir, rinaldi. 2025. "Algoritma Greedy (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf. diakses pada 22 Juni 2025

[4] Pratiwi, Asri Bekti. 2021. "Penjadwalan Permutation Flowshop Menggunakan Chaotic Maps. https://unair.ac.id/penyelesaian-permasalahan-penjadwalan-permutation-flowshop-menggunakan-chaotic-maps/. Diakses pada 22 Juni 2025

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, 24 Juni 2025



M. Abizzar Gamadrian - 13523155