

Analisis Pendekatan Model Knapsack untuk Menentukan Kombinasi Klip Video Pendek dengan Algoritma Greedy dan Program Dinamis

Reynard Nathanael - 13524103

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

E-mail: reynardnathanael11@gmail.com , 13524103@std.stei.itb.ac.id

Abstract— Popularitas konten video berdurasi pendek menuntut kreator untuk mengoptimalkan retensi penonton dalam batasan durasi maksimal 60 detik. Tantangan muncul ketika memilih kombinasi potongan klip video yang memiliki durasi bervariasi untuk memaksimalkan total skor *engagement score*. Penelitian ini memodelkan persoalan tersebut ke dalam *integer knapsack problem* dan membandingkan performa Algoritma Greedy berbasis densitas dengan Program Dinamis. Pengujian dilakukan secara eksperimental menggunakan data simulasi dengan variasi jumlah klip hingga 1000 data. Hasil eksperimen menunjukkan bahwa Program Dinamis selalu menghasilkan solusi optimal, terbukti pada pengujian 100 klip yang menghasilkan skor 1073, sedangkan Algoritma Greedy hanya mencapai skor 1058. Algoritma Greedy membutuhkan waktu yang lebih singkat dalam pemrosesannya, namun waktu eksekusi Program Dinamis pada pengujian 1000 klip hanya membutuhkan waktu 9.27 milidetik. Angka tersebut masih sangat wajar karena masih di bawah 10 milidetik, sehingga Program Dinamis disimpulkan sebagai strategi yang paling ideal untuk digunakan pada sistem otomatisasi pembuatan video pendek dengan menjamin kualitas video yang maksimal.

Keywords—Video Pendek, Knapsack Problem, Algoritma Greedy, Program Dinamis

I. PENDAHULUAN

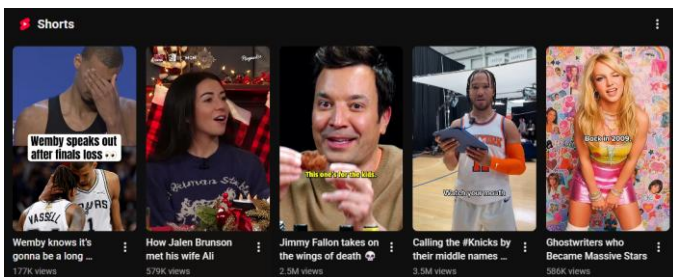


Fig. 1. Contoh Perkembangan Klip Video Pendek. [sumber](#)

Dalam beberapa tahun terakhir, perkembangan dunia teknologi beserta industri kreatif telah mengubah pola konsumsi atau bentuk *entertainment* media. Salah satu bentuk perubahan

ini adalah semakin meningkat atau bertambahnya jumlah konten video dengan durasi pendek (*short-form video*) yang diunggah di berbagai platform media sosial seperti *Youtube Shorts*, *Instagram Reels*, *TikTok*, maupun platform media sosial lainnya. Perubahan dalam pola konsumsi masyarakat maupun ekosistem perkembangan dunia *entertainment* atau media sosial ini diakibatkan karena dunia *content creation* yang semakin kompetitif dan platform media sosial biasanya menggunakan indikator keberhasilan (*key metrics*) berupa retensi penonton (*viewer retention*) dan *average watch time* untuk digunakan sebagai metrik utama apakah sebuah video yang diupload oleh seseorang layak untuk direkomendasikan oleh algoritma media sosial tersebut ke jumlah orang yang lebih banyak.

Dengan bentuk konten berupa video pendek, maka retensi penonton akan lebih berpotensi untuk terjaga sampai akhir video, sehingga memudahkan para kreator konten untuk menghasilkan video yang bisa mencakup lebih banyak orang karena terdorong oleh algoritma media sosialnya. Oleh karena itu, para kreator konten saat ini juga semakin memperhatikan bentuk narasi yang singkat, padat, memikat, jelas, dan bernilai tinggi dengan durasi terbatas yang umumnya berkisar maksimal 60 detik, meskipun beberapa *platform* media sosial memiliki durasi video pendek yang bisa bervariasi.

Namun, proses perubahan dengan bentuk video pendek ini tentunya memiliki tantangannya sendiri, yaitu dalam tahap *editing*. Para konten kreator tentu saja memiliki sejumlah video yang telah mereka persiapkan untuk diolah dan diproses menjadi video pendek, namun dalam tahap *editing* biasanya kreator kesulitan menentukan potongan klip video untuk dijadikan satu video pendek utuh. Seorang kreator biasanya memiliki berbagai potongan klip video dari satu video aslinya, dimana setiap klip video tersebut memiliki durasi yang bervariasi dengan estimasi *engagement score* yang berbeda-beda. Masalah bagi kreator tiba ketika harus memilih kombinasi potongan klip mana saja yang layak dan tepat untuk dijadikan satu video pendek final, sehingga *engagement score* tetap maksimal tanpa melewati batas durasi bentuk video pendek tersebut.

Persoalan memilih kombinasi klip ini merupakan salah satu bentuk nyata dari *0/1 Knapsack Problem*. Batas durasi video pendek secara keseluruhan akan dianggap sebagai kapasitas

maksimum dari ransel, sedangkan durasi dari masing-masing klip dan *engagement score*nya bertindak sebagai *weight* dan *value*. Oleh karena itu, makalah ini dibuat untuk mengimplementasikan dan melakukan analisis terhadap metode yang optimal untuk membantu mengoptimisasikan proses pengambilan keputusan ini sehingga menghasilkan hasil yang maksimal dengan menggunakan dan membandingkan antara Algoritma Greedy dan Program Dinamis (*Dynamic Programming*).

II. LANDASAN TEORI

A. Konten Video Pendek

Konten video berdurasi pendek (*short-form video*) didefinisikan sebagai bentuk video dengan format media visual dalam bentuk digital yang memiliki durasi yang umumnya berada diantara 15 hingga 60 detik. Dalam perkembangan dunia media sosial, platform seperti *Youtube Shorts*, *TikTok*, dan *Instagram Reels* telah menerapkan algoritma rekomendasi kepada orang lain yang sangat bergantung terhadap *viewer retention* dan *average watch time*.

Perhatian manusia (*attention span*) pada platform yang melakukan aksi *scrolling* bersifat sangat cepat dan terbatas. Apabila seorang kreator tidak mampu menghasilkan video pendek dengan kualitas yang baik untuk menangkap perhatian manusia tersebut, maka mereka tidak akan digemari oleh para penonton maupun algoritma. Oleh karena itu, setiap detik di dalam video harus memiliki nilai informasi atau hiburan yang tinggi.

Untuk memvalidasi kualitas dari setiap potongan klip yang dihasilkan dari satu video, maka digunakan istilah **Engagement Score**. Skor ini digunakan untuk merepresentasikan skala numerik dari 1 hingga 100 untuk mengevaluasi performa atau kualitas dari sebuah potongan klip video. Skor ini akan dihitung berdasarkan kombinasi beberapa variabel, yaitu:

- Persentase penonton yang menonton klip.
- Kepadatan aksi atau elemen visual yang dinamis.
- Informasi atau narasi pada klip.

B. 0/1 Knapsack Problem (Integer Knapsack Problem)

OBJECT	WEIGHT	VALUE	VALUE / WEIGHT
1	2	10	5
2	3	20	6.66
3	4	50	12.5
4	5	60	12

Sort

OBJECT	WEIGHT	VALUE	VALUE / WEIGHT
3	4	50	12.5
4	5	60	12
2	3	20	6.67
1	2	10	5

Fig. 2. Pemodelan 0/1 Knapsack Problem [sumber](#)

Persoalan *knapsack* merupakan sebuah persoalan optimasi. Masalah ini dimodelkan sebagai suatu himpunan barang, setiap barang dalam himpunan tersebut memiliki keuntungan dan bobot. Kemudian ditentukan kombinasi barang yang akan dimasukkan ke dalam *knapsack* dengan kapasitas daya tampung maksimal sebesar W . Setiap objek i memiliki bobot sebesar w_i dan nilai keuntungan sebesar v_i .

Pada persoalan 0/1 Knapsack, setiap objek hanya memiliki dua kemungkinan status, yaitu: diambil seluruhnya (dinyatakan dengan angka 1) atau ditinggalkan (dinyatakan dengan angka 0). Objek tidak dapat diambil sebagian atau dipotong. Secara matematis, persoalan ini bertujuan untuk memaksimalkan fungsi tujuan:

$$\text{maksimasi: } \sum_{i=1}^n V_i X_i$$

Dengan batasan fungsi:

$$\text{batasan: } \sum_{i=1}^n W_i X_i \leq W$$

- $X_i \in \{0,1\}$ menyatakan status pengambilan objek i .
- n merupakan jumlah total kandidat objek.

C. Algoritma Greedy

Algoritma *Greedy* merupakan salah satu strategi penyelesaian 0/1 *Knapsack Problem* yang menerapkan pendekatan heuristik dalam mencari solusi optimal pada setiap langkah. Pada setiap tahapan keputusan, akan selalu dipilih opsi yang paling menguntungkan pada saat itu tanpa memperdulikan konsekuensi jangka panjang atau dampak keputusan tersebut di masa depan (*local maximal*). Algoritma ini beranggapan bahwa dengan selalu menemukan nilai optimum lokal, maka gabungan semua pilihan lokal terbaik tersebut akan berakhir atau mengarahkan pada solusi optimal global (*global maximal*).

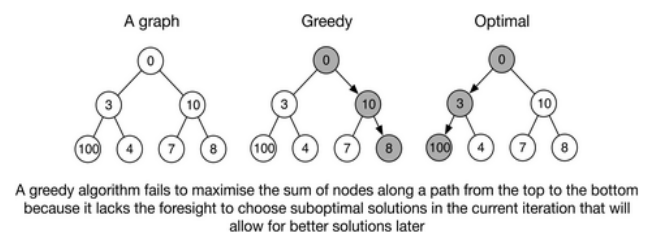


Fig. 3. Pemodelan Algoritma Greedy [sumber](#)

Dalam menyelesaikan 0/1 *Knapsack Problem*, terdapat tiga jenis strategi heuristik *Greedy* yang biasanya digunakan, yaitu:

- **Greedy by Value:** Algoritma selalu memilih objek yang memiliki nilai keuntungan V_i terbesar terlebih dahulu untuk dimasukkan ke dalam knapsack.
- **Greedy by Weight:** Algoritma selalu memilih objek yang memiliki bobot W_i paling ringan terlebih dahulu

agar knapsack dapat membawa objek sebanyak mungkin.

- **Greedy by Density:** Algoritma menghitung nilai density $\frac{V_i}{W_i}$ untuk setiap objek, kemudian memilih objek dengan nilai densitas tertinggi terlebih dahulu.

Meskipun algoritma *Greedy* memiliki kompleksitas waktu yang sangat efisien karena hanya membutuhkan proses *sorting*, algoritma *Greedy* tidak bisa menjamin bahwa hasil solusi yang dihasilkan berupa solusi optimal pada *0/1 Knapsack Problem*. Hal ini karena algoritma *Greedy* tidak dapat membatalkan pilihan yang telah diambil sebelumnya atau melakukan tindakan *backtracking* untuk memperbaiki pilihan yang telah dibuat.

D. Program Dinamis

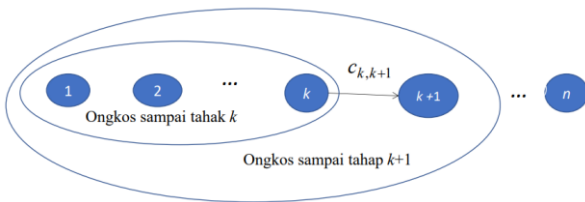


Fig. 4. Pemodelan Program Dinamis [sumber](#)

Program Dinamis (*Dynamic Programming*) adalah metode pemecahan masalah dengan menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*) atau sub-persoalan yang lebih kecil sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Penyelesaian suatu persoalan dengan program dinamis memiliki kemiripan dengan algoritma *greedy*, hanya saja pada algoritma *greedy* hanya terdapat satu solusi, sementara pada program dinamis dapat dihasilkan lebih dari satu kemungkinan solusi.

Hal ini juga berbeda dengan algoritma *Divide and Conquer* yang menyelesaikan sub-persoalan secara mandiri, Program Dinamis menyimpan hasil penyelesaian dari setiap sub-persoalan ke dalam sebuah tabel memori di dalam komputer melalui prinsip *memoization*. Hal ini memastikan bahwa sub-persoalan yang sama tidak perlu dihitung ulang.

Pada penyelesaian persoalan dengan metode ini:

- Terdapat sejumlah berhingga pilihan yang mungkin.
- Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya.
- Menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Pada Program Dinamis, rangkaian keputusan yang optimal harus memenuhi **Prinsip Optimalitas** (*Principle of Optimality* atau *Optimal Substructure*). Prinsip ini menyatakan bahwa sebuah solusi optimal global dapat diperoleh dengan menggabungkan solusi-solusi optimal dari sub-persoalannya, sehingga jika solusi total merupakan solusi optimal, maka bagian solusi sampai tahap ke-k juga optimal.

Dengan prinsip optimalitas, dapat dijamin bahwa pengambilan keputusan pada suatu sub-persoalan adalah keputusan yang benar untuk sub-persoalan selanjutnya. Oleh karena itu, pada program dinamis, tidak perlu kembali ke tahap awal karena digunakan hasil optimal dari tahap sebelumnya.

Pada *0/1 Knapsack Problem*, Program Dinamis membangun sebuah matriks dua dimensi C berukuran $(n+1) \times (W+1)$, di mana baris i adalah jumlah objek dan kolom w adalah kapasitas *knapsack* saat itu. Pengisian sel matriks dilakukan secara bertahap menggunakan rumus sebagai berikut:

$$C[i, w] = \begin{cases} 0, & \text{jika } i = 0 \text{ atau } w = 0 \\ C[i - 1, w], & \text{jika } w_i > w \\ \max(C[i - 1, w], C[i - 1, w - w_i] + v_i), & \text{jika } w_i \leq w \end{cases}$$

Melalui pengisian tabel terstruktur seperti ini, Program Dinamis dijamin akan menghasilkan pencarian solusi optimal dengan kompleksitas waktu $O(n \times W)$.

III. PERANCANGAN DAN IMPLEMENTASI ALGORITMA

A. Pemodelan dan Representasi ke *0/1 Knapsack*

Pemilihan klip video pendek yang akan digunakan harus ditransformasikan ke dalam variabel matematika sesuai keperluan *0/1 Knapsack Problem*. Proses pemetaan setiap parameternya didefinisikan sebagai berikut:

- Himpunan Objek (S): Daftar seluruh klip video yang tersedia, di mana $S = \{1, 2, 3, \dots, n\}$.
- Bobot Objek (W_i): Durasi klip video ke- i yang dinyatakan dalam satuan detik.
- Nilai Objek (V_i): Engagement Score dari klip video ke- i dengan skala dimulai dari 1 sampai 100.
- Kapasitas (W): Batas durasi total dari keseluruhan video pendek.
- Variabel Keputusan (X_i): Bernilai 1 jika klip video ke- i dipilih untuk digabungkan dengan klip lain pada video utama, dan bernilai 0 jika klip tidak dipilih untuk digabungkan.

Secara ringkas, visualisasi pemetaan setiap komponen dapat dilihat pada tabel di bawah ini:

TABLE I. PEMETAAN KOMPONEN VIDEO KE KNAPSACK PROBLEM

Komponen Masalah	Atribut Matematika Knapsack	Simbol Variabel	Satuan Data
Daftar Klip Video	Himpunan Objek	$S = \{1, 2, \dots, n\}$	Teks
Durasi Klip Video	Bobot (<i>Weight</i>)	W_i	Detik (int)

Engagement Score Klip	Value	V_i	Skala 1-100
Batas Durasi Video Final	Kapasitas (Capacity)	W	Detik (int)
Keputusan Penggabungan Klip	Status	X_i	Biner (boolean)

B. Perancangan Algoritma Greedy

Pada perancangan algoritma *Greedy*, Strategi heuristik yang dipilih dalam implementasi ini adalah **Greedy by Density**. Strategi ini dipilih karena secara teoritis memberikan pendekatan heuristik paling mendekati nilai optimal dibandingkan hanya melihat durasi atau skor potongan klip dari sebuah video.

1) Alur Logika Perancangan

- Program menerima input berupa array objek klip video dan kapasitas durasi maksimum W.
- Program kemudian menghitung nilai densitas rasio $r_i = v_i/w_i$ untuk setiap klip yang ada.
- Program kemudian melakukan pengurutan terhadap keseluruhan klip secara *descending* berdasarkan nilai rasio r_i .
- Program kemudian melakukan iterasi dari klip urutan pertama hingga terakhir untuk melakukan pemilihan sebagai berikut:
 - Jika durasi klip saat ini (w_i) masih lebih kecil atau sama dengan sisa kapasitas W, maka keputusannya adalah klip tersebut akan diambil ($x_i = 1$) dan sisa kapasitasnya akan dikurangi dengan w_i .
 - Jika ingin mengambil atau memilih klip tersebut, namun tidak muat, maka klip akan dilewati ($x_i = 0$) dan algoritma akan langsung melewati dan mengecek ke klip video berikutnya.

2) Analisis Kompleksitas

Karena algoritma *Greedy* yang dibuat didominasi oleh proses pengurutan data (*sorting*), maka kompleksitas dari algoritma *Greedy* yang dibuat ini adalah $O(n \log n)$, sedangkan proses pencarian atau iterasi (*looping*) hanya berupa waktu linear $O(N)$. Oleh karena itu, total kompleksitas waktu Algoritma *Greedy* adalah $O(N \log N)$.

C. Perancangan Algoritma Program Dinamis

Selanjutnya pada perancangan algoritma Program Dinamis, dirancang menggunakan skema pengisian tabel dari bawah ke atas (*bottom-up approach*) untuk menjamin bahwa hasil atau solusi yang diberikan merupakan solusi optimal global (*global maximal*).

1) Alur Logika Perancangan

- Program akan membuat sebuah matriks (tabel dua dimensi C) berukuran $(N+1) \times (W+1)$ dengan nilai inisialisasi awal seluruh sel dalam matriks atau tabel berupa nilai 0.
- Program kemudian melakukan *nested looping* untuk mengisi sel matriks secara bertahap, dengan ketentuan sebagai berikut:
 - Baris i bergerak dari 1 sampai N (hal ini mewakili indeks dari setiap klip video yang ada).
 - Kolom w bergerak dari 1 sampai W (hal ini mewakili durasi dari setiap klip yang sedang diiterasi).
- Pada setiap koordinat sel $C[i,w]$, program akan mengambil keputusan berdasarkan rumus rekurensi berikut:
 - Jika durasi dari klip yang sedang ditinjau atau diiterasi lebih besar dari kolom kapasitas ($w_i > w$), maka nilai sel tersebut akan disamakan dengan baris di atasnya, yaitu: $C[i, w] = C[i-1, w]$.
 - Jika durasi dari klip yang sedang ditinjau atau diiterasi muat ($w_i \leq w$), maka program akan mencari nilai maksimal antara mengambil klip tersebut atau tidak mengambil klip tersebut, dengan memperhatikan sisa ruangnya, yaitu: $C[i,w] = \max(C[i-1, w], C[i-1, w-w_i] + v_i)$.

2) Algoritma Backtracking

Setelah matriks C sudah terisi penuh dari kolom pertama hingga kolom terakhir, maka program akan melakukan penelusuran secara backtracking dari posisi $C[n,W]$ untuk mengidentifikasi klip video mana saja yang akan dipilih untuk disusun menjadi solusi optimal penyusunan video pendek tersebut. Jika nilai dari $C[i,w] \neq C[i-1, w]$, maka akan disimpulkan bahwa klip ke-i dipilih yang ditunjukkan dengan ($x_i = 1$) dan pencarian secara iterasi akan dilanjutkan ke koordinat $C[i-1, w-w_i]$.

3) Analisis Kompleksitas

Karena pada algoritma Program Dinamis pengisian tabel merupakan salah satu aspek penting, maka kompleksitas waktu untuk pengisian tabel ini bergantung pada ukuran matriks itu sendiri, yaitu sebesar $O(N \times W)$, sementara fase *backtracking* membutuhkan waktu $O(N)$. Total kompleksitas waktu dari keseluruhan metode ini adalah $O(N \times W)$.

D. Deklarasi Tipe Data

Untuk memudahkan proses pembuatan algoritma *Greedy* dan Program dinamis, maka dibuatkan deklarasi tipe data

seperti ini untuk mendefinisikan struktur data untuk merepresentasikan sebuah klip video.

Pseudocode Deklarasi Tipe Data

```
type VideoClip : record
  id : string
  duration : integer // bobot (weight)
  score : integer // skala 1-100
  density : real // rasio (score/duration)
end record
```

E. Implementasi Algoritma Greedy

Algoritma ini bekerja dengan mengurutkan seluruh kandidat klip video berdasarkan rasio (*engagement score*) per detik secara *descending*, kemudian mengambil klip tersebut selama durasi dari klipnya tidak melanggar batas durasi keseluruhan video pendek (*W*).

Pseudocode Algoritma Greedy

```
function SolveGreedy(clips : array of VideoClip, N : integer, W : integer) → list of VideoClip
// Input: clips (array of VideoClip), N (jumlah total klip), W (durasi video pendek).
// Output: daftar klip video yang dipilih untuk digabungkan menjadi satu video pendek.

// 1. Hitung nilai densitas untuk setiap klip video
for i ← 0 to N - 1 do
  clips[i].density ← clips[i].score / clips[i].duration
end for

// 2. Urutkan seluruh klip berdasarkan nilai density secara descending

// 3. Proses pemilihan klip (local maximum)
selectedClips ← list kosong of VideoClip
currentDuration ← 0

for i ← 0 to N - 1 do
  // Periksa apakah durasi klip ke-i masih muat di dalam sisa durasi W
  if (currentDuration + clips[i].duration) ≤ W then
    selectedClips.append(clips[i])
    currentDuration ← currentDuration + clips[i].duration
  end if
end for

return selectedClips
end function
```

F. Implementasi Algoritma Program Dinamis

Algoritma ini bekerja menggunakan matriks dua dimensi untuk menyimpan hasil optimasi sub-persoalan secara bertahap,

kemudian dilanjutkan dengan tahap *backtracking* untuk mengetahui klip mana saja yang akan dipilih menjadi satu video pendek yang utuh.

Pseudocode Algoritma Program Dinamis

```
function SolveDynamicProgramming(clips : array of VideoClip, N : integer, W : integer) → list of VideoClip
// Input: clips (array of VideoClip), N (jumlah total klip), W (durasi video pendek)
// Output: daftar klip video yang membentuk kombinasi optimal (global maximum).

// 1. Inisialisasi matriks C berukuran (N + 1) x (W + 1) dengan nilai 0
C : matriks [0 . . . N][0 . . . W] of integer

// Isi tabel matriks secara Bottom-Up
for i ← 0 to N do
  for w ← 0 to W do
    if i == 0 or w == 0 then
      C[i][w] ← 0
    else if clips[i - 1].duration > w then
      // Jika durasi klip ke-(i-1) melebihi kapasitas kolom w saat ini
      C[i][w] ← C[i - 1][w]
    else
      // Cari nilai maksimal
      takeValue ← C[i - 1][w - clips[i - 1].duration] + clips[i - 1].score
      leaveValue ← C[i - 1][w]

      C[i][w] ← max(takeValue, leaveValue)
    end if
  end for
end for

// 2. Backtracking untuk mencari susunan klip
selectedClips ← list kosong of VideoClip
currW ← W

for i ← N down to 1 do
  // Jika nilai sel berbeda dengan baris di atasnya, berarti klip tersebut diambil
  if C[i][currW] ≠ C[i - 1][currW] then
    selectedClips.append(clips[i - 1])
    currW ← currW - clips[i - 1].duration
  end if
end for

return selectedClips
end function
```

IV. HASIL DAN PEMBAHASAN

Pengujian dilakukan dengan menguji kedua algoritma pada lima variasi ukuran jumlah kandidat klip video (*N*), mulai dari skala kecil (10 klip) hingga skala besar (1.000 klip), dengan

kapasitas batas total durasi video pendek berada pada $W = 60$ detik. Hasil rekaman data skor total dan waktu komputasi disajikan pada gambar di bawah ini.

N Klip	Skor Greedy	Waktu Greedy (ms)	Skor DP	Waktu DP (ms)
10	431	0.0146	431	0.0909
50	819	0.0101	819	0.4258
100	1058	0.0176	1073	0.7981
500	1506	0.0869	1506	4.2248
1000	1798	0.1837	1798	9.2685

Fig. 5. Hasil Pengujian

A. Analisis Solusi

Berdasarkan data yang terlihat pada Fig. 5, pengujian pada $N = 10, 50, 500$, dan 1000 menunjukkan bahwa Algoritma Greedy berbasis densitas (*Greedy by Density*) mampu menghasilkan total skor yang sama persis dengan Program Dinamis. Hal ini mengindikasikan bahwa pada kondisi distribusi data tertentu, pendekatan heuristik pembagian rasio nilai per bobot secara kebetulan dapat menghasilkan hasil yang optimal.

Namun, terlihat kelemahan dari algoritma *Greedy* terlihat secara nyata pada pengujian dengan ukuran $N = 100$. Pada kasus uji ini, Algoritma *Greedy* menghasilkan total skor sebesar **1058**, sementara Program Dinamis berhasil menemukan kombinasi klip yang menghasilkan total skor lebih tinggi, yaitu **1073**.

Hal ini membuktikan bahwa teori Algoritma *Greedy* terjebak pada keputusan lokal (*local maximum*). Ketika mengambil klip dengan densitas tertinggi di awal, algoritma tersebut langsung menghabiskan sisa ruang kapasitas durasi, sehingga ruang bagi kombinasi klip-klip lain termabil padahal secara akumulatif justru menghasilkan solusi yang lebih optimal. Sebaliknya, Program Dinamis yang menelusuri seluruh ruang status secara terstruktur melalui tabel *memoization* menjamin bahwa hasil selalu berhasil menghasilkan solusi optimal (*global maximum*).

B. Analisis Efisiensi Running Time

Pada Fig. 5, terlihat bahwa terdapat perbedaan karakteristik waktu eksekusi antara kedua algoritma, dapat dilihat sebagai berikut:

- **Algoritma Greedy:** Pada pemrosesan 10 klip hingga 1.000 klip, waktu eksekusinya hanya berada pada 0.0146 ms hingga 0.1837 ms. Kinerja ini sesuai dengan teori kompleksitas *Greedy* yang didominasi oleh fungsi pengurutan (sorting) dengan performa sebesar $O(N \log N)$
- **Program Dinamis:** Mengalami pertumbuhan waktu yang berbanding lurus terhadap peningkatan jumlah klip (data). Waktu eksekusi meningkat dari 0.0909 ms pada $N = 10$ hingga 9.2685 ms pada $N = 1000$. Karakteristik ini mencerminkan kompleksitas waktu algoritma Program Dinamis yang berbasis pada ukuran dimensi matriks, yaitu $O(N \times W)$.

Meskipun Program Dinamis membutuhkan waktu eksekusi yang jauh lebih lama daripada algoritma *Greedy* pada data besar, durasi komputasi selama **9.2685 ms** pada 1.000 data masih

sangat aman untuk diterapkan sebagai sistem otomatisasi pemrosesan *editing* video secara nyata (*real-time processing*).

C. Batasan Program

Dalam proses pengujian yang telah dilakukan, terdapat beberapa batasan yang ditetapkan, yaitu sebagai berikut:

- Karena tujuan utama dari makalah ini adalah mengimplementasikan dan melakukan analisis terhadap metode yang optimal untuk membantu mengoptimisasi proses pengambilan keputusan klip video, maka proses analisis dan eksperimen menggunakan **data dummy** untuk mempercepat proses analisis dan penelusuran.
- Menggunakan potongan klip sebuah video dalam jumlah yang banyak (sampai 1000) membutuhkan kemampuan komponen komputer yang memadai beserta kemampuan untuk menggabungkan keseluruhan klip tersebut menjadi satu video pendek yang utuh, yang dimana hal tersebut tidak sesuai dengan tujuan makalah.
- Menghitung *engagement score* secara nyata untuk setiap klip video yang tersedia membutuhkan integrasi eksternal, seperti API data dari platform media sosial, dan penerapan model kecerdasan buatan untuk menganalisis dan mendeteksi aksi atau tindakan yang dilakukan pada klip tersebut, sehingga penghitungan score tersebut tidak dilakukan dan digantikan menggunakan data *dummy* untuk setiap klip.

Oleh karena itu, fokus penelitian dan pengujian pada makalah ini diarahkan sepenuhnya pada analisis komparatif performa logis, implementasi, dan efisiensi komputasi dari kedua strategi algoritma yang diuji.

V. KESIMPULAN

Program Dinamis terbukti selalu menghasilkan solusi yang optimal, seperti yang ditunjukkan pada pengujian dengan $N = 100$ di mana Program Dinamis mampu menghasilkan total skor kemenangan sebesar **1073**, yang lebih unggul dibandingkan Algoritma *Greedy* yang berada pada skor **1058** akibat sifat heuristik lokalnya.

Algoritma *Greedy* menunjukkan performa yang lebih baik, di mana pemrosesan 1.000 klip hanya membutuhkan waktu **0.1837 ms**, karena Program Dinamis mengalami pertumbuhan waktu yang linear terhadap dimensi ukuran matriks ($O(N \times W)$) dengan waktu **9.2685 ms** untuk jumlah data yang sama.

Meskipun terdapat perbedaan tersebut, waktu komputasi Program Dinamis yang masih berada di bawah 10 milidetik menunjukkan bahwa algoritma ini merupakan pilihan terbaik dan paling ideal untuk digunakan sebagai sistem otomatisasi video pendek, karena mampu menjamin retensi maksimal.

VI. APENDIKS

Github : <https://github.com/Hagon47/Knapsack-Video-Clipping>

Youtube : <https://youtu.be/la-zTPJWYMG>

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih terhadap Dr. Ir. Rinaldi Munir, M.T., atas semua ilmu dan dedikasi yang telah diberikan untuk mengajar dan membimbing keseluruhan mata kuliah IF2211 Strategi Algoritma. Penulis mengapresiasi dukungan semangat dari keluarga dan teman-teman selama proses penyusunan dan penulisan makalah ini. Penulis juga memanjatkan puji dan syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat yang telah diberikan dalam proses pembuatan makalah ini.

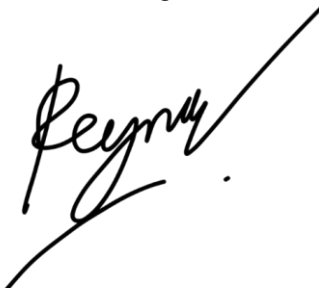
REFERENCES

- [1] R. Munir, "Algoritma Greedy (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/04-Algoritma-Greedy-\(2026\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/04-Algoritma-Greedy-(2026)-Bag1.pdf). [Accessed: June 16, 2026].
- [2] R. Munir, "Program Dinamis (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/25-Program-Dinamis-\(2026\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/25-Program-Dinamis-(2026)-Bagian1.pdf). [Accessed: June 16, 2026].
- [3] A. Lynch, "Decoding the Algorithm: The Mathematics Behind TikTok's Short-Form Content Success," Honors Scholar Theses, University of Connecticut, May 2025. [Online]. Available: https://digitalcommons.lib.uconn.edu/srhonors_theses/1102. [Accessed: June 16, 2026].
- [4] Z. Chen et al., "Shorter Is Different: Characterizing the Dynamics of Short-Form Video Platforms," arXiv preprint arXiv:2410.16058, Oct. 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2410.16058>. [Accessed: June 16, 2026].
- [5] U. Pferschy, "Dynamic Programming Revisited: Improving Knapsack Algorithms," Computing, vol. 63, no. 4, pp. 419–430, Dec. 1999. [Online]. Available: <https://doi.org/10.1007/s006070050042>. [Accessed: June 16, 2026].
- [6] T. Van Daele et al., "Making Short-Form Videos Accessible with Hierarchical Video Summaries," in Proceedings of the CHI Conference on Human Factors in Computing Systems, 2024, pp. 1–17. [Online]. Available: <https://doi.org/10.1145/3613904.3642839>. [Accessed: June 16, 2026].

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, 16 Juni 2026



Reynard Nathanael
13524103