Penerapan Pemrograman Dinamis dalam Mengoptimalkan Pemilihan Destinasi Wisata dengan Batasan Biaya dan Rating

Mesach Harmasendro – 13522117[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): mesachharmasendro@gmail.com

***Abstract*—Perencanaan perjalanan yang efisien dan efektif menjadi semakin penting bagi pelancong yang ingin memaksimalkan pengalaman mereka dengan anggaran terbatas. Penelitian ini mengembangkan model pemrograman dinamis untuk mengoptimalkan pemilihan destinasi wisata berdasarkan batasan biaya dan rating. Studi kasus dilakukan menggunakan 20 dan 50 data tempat wisata dengan tiga skenario maksimal biaya yang berbeda. Hasilnya menunjukkan bahwa pemrograman dinamis mampu menghasilkan kombinasi destinasi wisata dengan total rating tertinggi tanpa melebihi anggaran yang ditetapkan. Analisis menunjukkan bahwa waktu eksekusi program dipengaruhi oleh jumlah destinasi wisata dan batas maksimal biaya, tetapi tetap lebih efisien dibandingkan metode brute force. Kompleksitas waktu dan ruang dari algoritma ini adalah O(n·W)O(n \cdot W)O(n·W), yang meskipun memerlukan ruang memori besar, masih praktis untuk kasus nyata. Perbandingan dengan algoritma brute force menunjukkan bahwa pemrograman dinamis jauh lebih efisien dalam hal waktu eksekusi, meskipun brute force lebih sederhana dalam penggunaan memori. Implementasi ini menunjukkan bahwa pemrograman dinamis adalah metode yang efektif untuk menyelesaikan masalah optimasi dalam pemilihan destinasi wisata, memberikan kontribusi yang berharga dalam pengembangan alat bantu perencanaan perjalanan yang lebih cerdas dan adaptif.**

Keywords—Pemrograman Dinamis, Pemilihan Destinasi Wisata, Kompleksitas Algoritma, Optimasi, Bruteforce

I. PENDAHULUAN

Dalam dunia yang serba cepat dan penuh dengan pilihan, perencanaan perjalanan yang efisien dan efektif menjadi semakin penting bagi pelancong yang ingin memaksimalkan pengalaman mereka dengan anggaran terbatas. Pemilihan destinasi wisata yang optimal tidak hanya bergantung pada daya tarik visual atau popularitas tetapi juga pada pemenuhan kriteria spesifik seperti biaya perjalanan dan kepuasan atau rating yang diberikan oleh pengunjung sebelumnya. Keputusan ini menjadi semakin kompleks ketika kita mempertimbangkan berbagai faktor seperti ketersediaan waktu, preferensi pribadi, dan banyaknya informasi yang tersedia. Dalam konteks ini, teknik pemrograman dinamis menawarkan kerangka kerja matematis yang robust untuk mengatasi masalah optimasi multi-kriteria ini, memungkinkan pelancong dan perencana wisata untuk mencapai solusi yang optimal dengan mengingat keterbatasan sumber daya dan preferensi.

Pemrograman dinamis adalah metode yang dikenal luas dalam ilmu komputer dan optimasi, yang dirancang untuk menyelesaikan masalah kompleks dengan memecahnya menjadi sub-masalah yang lebih sederhana. Teknik ini sangat efektif untuk masalah yang dapat dipecah menjadi bagian-bagian yang lebih kecil dengan solusi yang saling bergantung. Dalam kasus perencanaan perjalanan, pemrograman dinamis memungkinkan kita untuk mempertimbangkan berbagai kombinasi destinasi wisata dan menghitung biaya serta rating yang sesuai untuk setiap kombinasi tersebut. Dengan demikian, kita dapat menemukan solusi optimal yang memberikan kepuasan maksimal bagi pelancong tanpa melebihi anggaran yang telah ditetapkan.

Penerapan metode pemrograman dinamis dalam konteks pariwisata, khususnya dalam memilih kombinasi destinasi wisata dengan pertimbangan biaya dan rating, belum banyak dieksplorasi secara mendalam dalam literatur akademik. Hal ini membuka peluang penelitian yang signifikan untuk mengembangkan model yang tidak hanya ekonomis tetapi juga efektif dalam memaksimalkan kepuasan pengguna. Sebagai contoh, banyak studi terdahulu yang fokus pada optimasi rute perjalanan atau jadwal penerbangan, namun jarang yang secara spesifik menggabungkan kriteria biaya dan rating destinasi wisata dalam satu model yang komprehensif. Oleh karena itu, penelitian ini berusaha untuk mengisi celah tersebut dengan mengembangkan model pemrograman dinamis yang mampu mengidentifikasi kombinasi destinasi wisata terbaik yang memenuhi kedua kriteria tersebut.

Studi ini diharapkan tidak hanya berkontribusi pada teori pengambilan keputusan dalam ilmu manajemen dan pariwisata tetapi juga memberikan manfaat praktis bagi pelancong individu dan agen perjalanan dalam merencanakan itinerari yang optimal. Dengan memanfaatkan teknik pemrograman dinamis, kita dapat mengembangkan alat bantu yang canggih dan mudah digunakan untuk membantu pelancong dalam merencanakan perjalanan mereka. Misalnya, sebuah aplikasi berbasis web atau mobile dapat dikembangkan untuk memungkinkan pengguna memasukkan anggaran dan preferensi mereka, dan kemudian menerima rekomendasi

destinasi wisata yang optimal berdasarkan model yang telah dikembangkan.

Dengan menggunakan pendekatan yang sistematis dan kuantitatif, makalah ini juga mendemonstrasikan bagaimana pemrograman dinamis dapat diadaptasi untuk memecahkan masalah nyata yang dihadapi dalam industri pariwisata, menawarkan solusi yang lebih adaptif dan responsif dibandingkan metode heuristik atau trial-and-error yang umum digunakan. Dalam dunia yang terus berubah dan dipengaruhi oleh tren global seperti digitalisasi dan personalisasi, kemampuan untuk merencanakan perjalanan yang optimal menjadi semakin penting. Oleh karena itu, penelitian ini tidak hanya relevan secara teoritis tetapi juga memiliki implikasi praktis yang signifikan.

Selain itu, penelitian ini diharapkan dapat membuka jalan bagi studi lebih lanjut di bidang ini, termasuk pengembangan model yang lebih kompleks yang dapat mempertimbangkan lebih banyak variabel, seperti musim, tren wisata, dan ketersediaan fasilitas. Dengan demikian, makalah ini berfungsi sebagai landasan untuk eksplorasi lebih lanjut dalam penerapan pemrograman dinamis untuk optimasi perjalanan wisata, memperluas cakupan penerapan teknik ini dalam berbagai konteks dan kebutuhan.

Dengan demikian, makalah ini bertujuan untuk tidak hanya mengembangkan dan menganalisis model pemrograman dinamis yang mampu mengidentifikasi kombinasi destinasi wisata terbaik yang memenuhi kriteria biaya dan rating, tetapi juga untuk menunjukkan bagaimana pendekatan ini dapat diterapkan dalam praktik, memberikan manfaat nyata bagi pengguna akhir. Melalui studi ini, kami berharap dapat memberikan kontribusi yang berharga bagi literatur akademik serta praktik industri pariwisata, membantu pelancong mencapai pengalaman perjalanan yang memuaskan dan berkesan dalam batasan anggaran yang mereka miliki.

II. DASAR TEORI

A. Pemrograman Dinamis

Pemrograman dinamis adalah salah satu metode penting dalam ilmu komputer yang digunakan untuk menyelesaikan masalah optimasi dan perhitungan dengan cara yang lebih efisien. Metode ini diperkenalkan oleh Richard Bellman pada tahun 1950-an dan telah menjadi salah satu teknik utama dalam algoritma komputasi. Prinsip dasar dari pemrograman dinamis adalah memecah masalah besar menjadi sub-masalah yang lebih kecil dan lebih mudah diselesaikan. Solusi dari sub-masalah ini kemudian disimpan untuk digunakan kembali (memoization), menghindari perhitungan ulang yang tidak perlu dan menghemat waktu komputasi.

Pendekatan ini sangat berguna untuk masalah yang memiliki struktur optimal subproblems. Artinya, solusi optimal dari masalah utama dapat diperoleh dengan menggabungkan solusi optimal dari sub-masalah yang lebih kecil. Dalam konteks ini, pemrograman dinamis berbeda dengan metode brute force yang mencoba semua kemungkinan solusi tanpa mengurangi redundansi dalam perhitungan.

Ada dua pendekatan utama dalam pemrograman dinamis:

- Top-Down Approach (Memoization): Pendekatan ini menggunakan rekursi untuk memecah masalah menjadi sub-masalah yang lebih kecil dan menyimpan hasil dari sub-masalah tersebut dalam tabel. Setiap kali sub-masalah yang sama ditemukan, hasil yang disimpan digunakan kembali tanpa perhitungan ulang. Ini mengurangi waktu komputasi secara signifikan.
- Bottom-Up Approach (Tabulation): Dalam pendekatan ini, sub-masalah dipecahkan terlebih dahulu dan hasilnya disimpan dalam tabel. Tabel ini kemudian digunakan untuk menyelesaikan masalah yang lebih besar secara iteratif. Pendekatan ini biasanya lebih efisien dalam penggunaan memori karena tidak memerlukan tumpukan panggilan rekursif.

B. Konsep Dasar Pemrograman Dinamis

Pemrograman dinamis biasanya diterapkan pada masalah optimasi, di mana tujuan utamanya adalah menemukan nilai terbaik di antara banyak kemungkinan. Ada beberapa konsep dasar yang penting dalam pemrograman dinamis:

- Optimal Substructure: Masalah dapat dipecah menjadi sub-masalah yang lebih kecil, dan solusi optimal dari masalah utama dapat diperoleh dengan menggabungkan solusi optimal dari sub-masalah tersebut.
- Overlapping Subproblems: Masalah sering kali melibatkan perhitungan ulang dari sub-masalah yang sama. Pemrograman dinamis menghindari perhitungan ulang ini dengan menyimpan hasil dari sub-masalah dalam tabel.
- State: Setiap masalah atau sub-masalah dapat direpresentasikan sebagai state yang unik. State ini mencakup semua informasi yang diperlukan untuk menyelesaikan sub-masalah tersebut.
- Transition: Merupakan aturan atau fungsi yang digunakan untuk berpindah dari satu state ke state lainnya. Fungsi ini menentukan bagaimana solusi dari satu sub-masalah dapat digunakan untuk menyelesaikan sub-masalah lainnya.

Pemrograman dinamis sering digunakan dalam berbagai algoritma seperti shortest path, knapsack problem, dan sequence alignment dalam bioinformatika. Implementasinya memerlukan pemahaman yang mendalam tentang masalah yang dihadapi dan cara memecahnya menjadi sub-masalah yang lebih kecil.

C. Algoritma Knapsack

Salah satu aplikasi klasik dari pemrograman dinamis adalah algoritma knapsack. Algoritma ini digunakan untuk menentukan item mana yang harus dimasukkan ke dalam knapsack (tas) dengan kapasitas tertentu untuk memaksimalkan nilai total item yang dimasukkan. Masalah knapsack sangat umum dalam optimasi dan memiliki beberapa varian, termasuk:

- 0/1 Knapsack Problem: Setiap item hanya dapat dipilih atau tidak dipilih (0 atau 1). Algoritma ini biasanya

diselesaikan menggunakan pendekatan bottom-up dengan membuat tabel dua dimensi yang menyimpan nilai maksimum yang dapat dicapai untuk setiap kombinasi item dan kapasitas.

- Fractional Knapsack Problem: Setiap item dapat dipilih sebagian, dan biasanya diselesaikan menggunakan pendekatan greedy. Pendekatan ini tidak menggunakan pemrograman dinamis karena sifatnya yang memungkinkan pemecahan masalah secara langsung dengan memilih item berdasarkan rasio nilai terhadap berat.

Pada masalah 0/1 knapsack, setiap item memiliki berat dan nilai. Tujuannya adalah untuk memilih kombinasi item sehingga total beratnya tidak melebihi kapasitas knapsack dan nilai totalnya maksimal. Ini dilakukan dengan menggunakan tabel dua dimensi di mana baris mewakili item dan kolom mewakili kapasitas. Setiap entri dalam tabel menyimpan nilai maksimum yang dapat dicapai dengan kapasitas tersebut menggunakan item yang tersedia.

D. Penerapan Pemrograman Dinamis pada Pariwisata

Pemrograman dinamis dapat diterapkan dalam berbagai konteks, termasuk pariwisata. Dalam konteks ini, pemrograman dinamis dapat digunakan untuk memecahkan masalah optimasi seperti pemilihan destinasi wisata dengan keterbatasan anggaran dan rating yang diberikan oleh pengunjung. Masalah ini mirip dengan masalah knapsack di mana setiap destinasi memiliki "biaya" dan "nilai" (rating), dan tujuan utamanya adalah memaksimalkan total rating dengan batasan biaya.

Masalah pemilihan destinasi wisata dapat diformulasikan sebagai masalah optimasi di mana setiap destinasi wisata direpresentasikan oleh biaya dan ratingnya. Tujuannya adalah untuk menemukan kombinasi destinasi wisata yang memberikan rating tertinggi tanpa melebihi anggaran yang tersedia. Pendekatan pemrograman dinamis membantu mengatasi masalah ini dengan cara yang efisien, menghindari perhitungan ulang yang tidak perlu dan memastikan solusi optimal.

E. Pemodelan Masalah

Untuk memodelkan masalah pemilihan destinasi wisata menggunakan pemrograman dinamis, langkah-langkah berikut dapat diikuti:

- Definisi Variabel: Tentukan variabel yang akan digunakan dalam pemrograman dinamis. Misalnya, biayanya (c_i) dan rating (r_i) dari setiap destinasi wisata.
- Fungsi Tujuan: Tentukan fungsi tujuan yang ingin dioptimalkan. Dalam kasus ini, fungsi tujuan adalah memaksimalkan total rating dari destinasi wisata yang dipilih.
- Kondisi Batas: Tentukan kondisi batas untuk masalah, seperti total biaya yang tidak boleh melebihi anggaran yang ditetapkan.
- Rekursi: Tentukan relasi rekursif untuk menyelesaikan masalah dengan memecahnya menjadi sub-masalah yang lebih kecil. Relasi ini menentukan bagaimana solusi dari satu sub-masalah dapat digunakan untuk menyelesaikan sub-masalah lainnya.

Masalah ini dapat direpresentasikan dalam bentuk tabel dua dimensi di mana baris mewakili destinasi wisata dan kolom mewakili anggaran. Setiap entri dalam tabel menyimpan nilai maksimal yang dapat dicapai dengan kombinasi destinasi wisata yang tersedia untuk anggaran tersebut. Implementasi pemrograman dinamis dalam Python untuk masalah ini melibatkan pembuatan tabel dan pengisian tabel dengan cara iteratif berdasarkan relasi rekursif yang telah ditentukan.

Dengan pendekatan ini, penulis dapat dengan mudah menentukan kombinasi destinasi wisata yang memberikan rating tertinggi tanpa melebihi anggaran yang tersedia. Pendekatan ini tidak hanya efisien tetapi juga memastikan bahwa solusi yang diperoleh adalah optimal, mengingat semua kemungkinan kombinasi destinasi wisata dan biaya yang terkait.

III. PENERAPAN PEMROGRAMAN DINAMIS

Dalam implementasi ini, pemrograman dinamis digunakan untuk menyelesaikan masalah optimasi pemilihan destinasi wisata dengan mempertimbangkan batasan biaya dan rating. Pemrograman dinamis adalah pendekatan yang efektif untuk menyelesaikan masalah yang dapat dipecah menjadi sub-masalah yang lebih kecil, dan setiap sub-masalah tersebut diselesaikan secara optimal. Ini memungkinkan penulis untuk menemukan kombinasi destinasi wisata yang memberikan total rating tertinggi tanpa melebihi anggaran yang tersedia. Berikut adalah penjelasan rinci mengenai penerapan pemrograman dinamis dalam konteks ini.

A. Fungsi Rekursif

Untuk memodelkan masalah pemilihan destinasi wisata menggunakan pemrograman dinamis, penulis akan mendefinisikan beberapa elemen kunci, termasuk fungsi rekursif dan basisnya. Pada dasarnya permasalahn ini dapat diformulasikan sebagai masalah knapsack, di mana setiap destinasi wisata diwakili oleh biaya dan ratingnya. Tujuannya adalah untuk memaksimalkan total rating dengan batasan biaya yang diberikan.

Sebagai dasar fungsi rekursif penulis akan menggunakan notasi berikut:

- $f_k(y)$ adalah nilai maksimum (total rating) yang dapat dicapai dengan mempertimbangkan destinasi wisata dari 1 hingga kkk dengan anggaran yyy.
- c_i adalah biaya destinasi ke-i.
- r_i adalah rating destinasi ke-i.

Fungsi rekursif dan basis untuk masalah ini dapat dinyatakan sebagai berikut:

- Basis

  1. Jika tidak ada destinasi yang dipertimbangkan atau anggaran adalah 0, maka total rating adalah 0

  $$f_0(y) = 0, \qquad \forall y \geq 0$$

  2. Jika anggaran negatif, total rating adalah negatif tak hingga karena tidak mungkin terjadi.

  $$f_k(y) = -\infty, \qquad \forall y < 0$$

- Rekurens

  Untuk setiap destinasi *k* dan anggaran *y*, nilai maksimum (total rating) dapat diperoleh dengan memilih destinasi *k* atau tidak memilihnya:

  $$f_k(y) = \max\{f_{k-1}(y), r_k + f_{k-1}(y - c_k)\}, \qquad k = 1, 2, \dots, n$$

  Dalam notasi ini:

  1. $f_{k-1}(y)$ adalah nilai maksimum yang dapat dicapai tanpa memilih destinasi ke-*k*.
  2. $r_k + f_{k-1}(y - c_k)$ adalah nilai maksimum yang dapat dicapai dengan memilih destinasi ke-*k*, menambahkan rating destinasi ke-*k* ke nilai maksimum yang dapat dicapai dengan anggaran $r_k + f_{k-1}(y - c_k)$.

- Bentuk Matematis Lengkap

  $$f_0(y) = 0, \qquad \forall y \geq 0 \ (basis)$$

  $$f_k(y) = -\infty, \qquad \forall y < 0 \ (basis)$$

  $$f_k(y) = \max\{f_{k-1}(y), r_k + f_{k-1}(y - c_k)\}, \ k = 1, 2, \dots, n \ (\text{rekurens})$$

  Dengan fungsi rekursif ini, penulis dapat menentukan nilai maksimum (total rating) yang dapat dicapai untuk setiap kombinasi destinasi dan anggaran, sehingga penulis bisa memilih kombinasi destinasi wisata yang optimal dengan cara yang efisien.

B. Fungsi Rekursif dalam Bentuk Kode Program

Funsgi rekursif yang sudah didefinisikan di atas kemudian akan diubah ke dalam bentuk kode program untuk semakin mempermudah implementas. Dalam kasus permasalahan ini penulis akan menggunakan tabel dua dimensi dp[i][budget] di mana i mewakili jumlah destinasi yang dipertimbangkan dan budget mewakili anggaran yang tersedia. Setiap entri dalam tabel menyimpan tuple (total_rating, total_cost, list_tempat, detail_tempat) yang merepresentasikan rating total, biaya total, daftar destinasi yang dipilih, dan detail destinasi.

Penjelasan dari kode program fungsi rekursif yang akan digunakan akan dijelaskan pada bagian di bawah ini:

dp[i][budget] = max(dp[i-1][budget], dp[i-1][budget-cost] + rating)

Dimana:

- dp[i-1][budget] adalah nilai optimal yang diperoleh tanpa memasukkan destinasi ke-i.
- dp[i-1][budget-cost] + rating adalah nilai optimal yang diperoleh dengan memasukkan destinasi ke-i.

Dengan kata lain, program akan membandingkan antara dua pilihan:

- Tidak memasukkan destinasi ke-i dan mempertahankan rating optimal dari destinasi sebelumnya.
- Memasukkan destinasi ke-i jika biaya tidak melebihi anggaran yang tersedia dan menambahkan rating destinasi tersebut ke total rating sebelumnya.

Berdasarkan prinsip ini, tabel dp dapat diisi secara iteratif. Fungsi rekursif ini digunakan untuk menentukan nilai optimal untuk setiap kombinasi destinasi dan anggaran, dengan memanfaatkan solusi dari sub-masalah yang lebih kecil.

IV. IMPLEMENTASI KODE PROGRAM

Kode program yang dibuat akan dibuat dengan menggunakan bahasa pemrograman Python. Program yang dibuat akan berbasis CLI dan program nantinya akan menerima masukan berupa berupa biaya atau cost maksimum dari rencana liburan yang ingin dilakukan. Kemudian program akan mencari kombinasi tempat wisata yang sesuai berdasarkan data masukan tersebut. Data tempat wisata beserta dengan harga dan ratingnya disini dibuat secara random atau bisa dikatakan hanya menggunakan data dummy. Untuk kasus penerapan yang lebih nyata data ini bisa dimodifikasi sendiri mengikuti dengan data sebenarnya yang beredar di internet.

Berikut adalah kode program dari program yang dibuat:

Untuk foto yang lebih jelas bisa dilihat melalui link berikut:

https://drive.google.com/file/d/1Ka6keQYr6O6wtpQwQ1s5wo5O_TyA_B0S/view?usp=sharing

Kode di atas mengimplementasikan pemrograman dinamis untuk menemukan kombinasi destinasi wisata yang memberikan total rating tertinggi tanpa melebihi anggaran. Berikut adalah penjelasan dari setiap bagian kode:

- Inisialisasi Tabel: dp[i][budget] menyimpan tuple yang mencakup total rating, total biaya, daftar destinasi, dan detail destinasi untuk *i* destinasi dan anggaran budgetbudgetbudget.
- Iterasi Destinasi dan Anggaran: Dua loop digunakan untuk iterasi melalui setiap destinasi (*i*) dan setiap anggaran (*budget*).
- Pilih atau Tidak Memilih Destinasi: Jika destinasi ke-*i* tidak dipilih, maka nilai optimal adalah nilai dari destinasi sebelumnya dengan anggaran yang sama. Jika destinasi ke-*i* dipilih, maka rating destinasi tersebut dan biayanya ditambahkan ke dalam total sebelumnya.
- Memilih Nilai Optimal: Pemilihan antara tidak memilih destinasi atau memilih destinasi didasarkan pada total rating tertinggi atau biaya total terendah jika rating sama.
- Menghitung Hasil Optimal: Setelah iterasi selesai, kombinasi dengan rating tertinggi yang memenuhi batasan anggaran dicari.

Dengan implementasi ini, rencana perjalanan wisata yang optimal dapat ditentukan secara efisien, menggabungkan berbagai destinasi wisata sesuai dengan anggaran dan preferensi rating pengguna. Pendekatan ini juga dapat diperluas dan disesuaikan dengan penambahan variabel lain, seperti waktu perjalanan atau preferensi pribadi lainnya.

V. STUDI KASUS DAN ANALISIS

A. Studi Kasus

Pada bagian ini akan ditunjukkan hasil dari studi kasus dengan 3 masukan maksimum biaya yang berbeda. Seperti yang sudah dijelaskan sebelumnya data tempat wisata yang digunakan disini adalah data dummy bukan data yang sebenarnya. Berikut adalah data tempat wisata yang akan digunakan:

```python
# Data tempat wisata Indonesia
places = [
    ("Candi Borobudur", 500, 4.7), #(nama, harga, rating)
    ("Gunung Bromo", 350, 4.5),
    ("Raja Ampat", 1500, 4.9),
    ("Taman Laut Bunaken", 1200, 4.8),
    ("Pulau Komodo", 1300, 4.7),
    ("Taman Nasional Bali Barat", 800, 4.6),
    ("Danau Toba", 400, 4.3),
    ("Candi Prambanan", 500, 4.6),
    ("Gunung Rinjani", 1000, 4.8),
    ("Taman Nasional Way Kambas", 700, 4.4),
    ("Pantai Kuta", 650, 4.2),
    ("Pantai Sanur", 700, 4.1),
    ("Taman Safari Indonesia", 600, 4.6),
    ("Kepulauan Seribu", 750, 4.7),
    ("Labuan Bajo", 1400, 4.8),
    ("Gunung Merapi", 300, 4.4),
    ("Kawah Ijen", 350, 4.6),
    ("Ubud", 750, 4.8),
    ("Gili Trawangan", 800, 4.5),
    ("Nusa Penida", 850, 4.7)
]
```

Studi kasus pertama akan memasukkan data maksimum biaya 6000, lalu untuk studi kasus dua 5000, dan kemudian untuk studi kasus tiga 4300. Berikut adalah hasil dari ketiga studi kasus tersebut:

```
Masukkan maksimum biaya yang Anda inginkan: 6000
Total rating tertinggi yang bisa dicapai dengan biaya tidak lebih
 dari Rp6000 adalah 50.199999999999996 dengan biaya total 6000.
Destinasi terpilih: Candi Borobudur, Gunung Bromo, Taman Nasional
 Bali Barat, Danau Toba, Candi Prambanan, Taman Nasional Way Kamb
as, Taman Safari Indonesia, Kepulauan Seribu, Gunung Merapi, Kawa
h Ijen, Ubud
Rata-rata rating: 4.56
Destinasi: Candi Borobudur, Biaya: 500, Rating: 4.7
Destinasi: Gunung Bromo, Biaya: 350, Rating: 4.5
Destinasi: Taman Nasional Bali Barat, Biaya: 800, Rating: 4.6
Destinasi: Danau Toba, Biaya: 400, Rating: 4.3
Destinasi: Candi Prambanan, Biaya: 500, Rating: 4.6
Destinasi: Taman Nasional Way Kambas, Biaya: 700, Rating: 4.4
Destinasi: Taman Safari Indonesia, Biaya: 600, Rating: 4.6
Destinasi: Kepulauan Seribu, Biaya: 750, Rating: 4.7
Destinasi: Gunung Merapi, Biaya: 300, Rating: 4.4
Destinasi: Kawah Ijen, Biaya: 350, Rating: 4.6
Destinasi: Ubud, Biaya: 750, Rating: 4.8
Waktu eksekusi program: 0.2112 detik.
```

```
Masukkan maksimum biaya yang Anda inginkan: 5000
Total rating tertinggi yang bisa dicapai dengan biaya tidak lebih
 dari Rp5000 adalah 41.6 dengan biaya total 4950.
Destinasi terpilih: Candi Borobudur, Gunung Bromo, Candi Prambana
n, Taman Safari Indonesia, Kepulauan Seribu, Gunung Merapi, Kawah
 Ijen, Ubud, Nusa Penida
Rata-rata rating: 4.62
Destinasi: Candi Borobudur, Biaya: 500, Rating: 4.7
Destinasi: Gunung Bromo, Biaya: 350, Rating: 4.5
Destinasi: Candi Prambanan, Biaya: 500, Rating: 4.6
Destinasi: Taman Safari Indonesia, Biaya: 600, Rating: 4.6
Destinasi: Kepulauan Seribu, Biaya: 750, Rating: 4.7
Destinasi: Gunung Merapi, Biaya: 300, Rating: 4.4
Destinasi: Kawah Ijen, Biaya: 350, Rating: 4.6
Destinasi: Ubud, Biaya: 750, Rating: 4.8
Destinasi: Nusa Penida, Biaya: 850, Rating: 4.7
Waktu eksekusi program: 0.1279 detik.
```

```
Masukkan maksimum biaya yang Anda inginkan: 4300
Total rating tertinggi yang bisa dicapai dengan biaya tidak lebih
 dari Rp4300 adalah 36.9 dengan biaya total 4100.
Destinasi terpilih: Candi Borobudur, Gunung Bromo, Candi Prambana
n, Taman Safari Indonesia, Kepulauan Seribu, Gunung Merapi, Kawah
 Ijen, Ubud
Rata-rata rating: 4.61
Destinasi: Candi Borobudur, Biaya: 500, Rating: 4.7
Destinasi: Gunung Bromo, Biaya: 350, Rating: 4.5
Destinasi: Candi Prambanan, Biaya: 500, Rating: 4.6
Destinasi: Taman Safari Indonesia, Biaya: 600, Rating: 4.6
Destinasi: Kepulauan Seribu, Biaya: 750, Rating: 4.7
Destinasi: Gunung Merapi, Biaya: 300, Rating: 4.4
Destinasi: Kawah Ijen, Biaya: 350, Rating: 4.6
Destinasi: Ubud, Biaya: 750, Rating: 4.8
Waktu eksekusi program: 0.1075 detik.
```

Ketiga studi kasus diatas juga akan dicoba dengan data tempat wisata yang lebih banyak dengan tujuan untuk melihat perbedaan waktu eksekusinya. Berikut adalah data tempat wisata baru yang akan digunakan:

```python
# Data tempat wisata Indonesia
places = [
    ("Candi Borobudur", 500, 4.7),
    ("Gunung Bromo", 350, 4.5),
    ("Raja Ampat", 1500, 4.9),
    ("Taman Laut Bunaken", 1200, 4.8),
    ("Pulau Komodo", 1300, 4.7),
    ("Taman Nasional Bali Barat", 800, 4.6),
    ("Danau Toba", 400, 4.3),
    ("Candi Prambanan", 500, 4.6),
    ("Gunung Rinjani", 1000, 4.8),
    ("Taman Nasional Way Kambas", 700, 4.4),
    ("Pantai Kuta", 650, 4.2),
    ("Pantai Sanur", 700, 4.1),
    ("Taman Safari Indonesia", 600, 4.6),
    ("Kepulauan Seribu", 750, 4.7),
    ("Labuan Bajo", 1400, 4.8),
    ("Gunung Merapi", 300, 4.4),
    ("Kawah Ijen", 350, 4.6),
    ("Ubud", 750, 4.8),
    ("Gili Trawangan", 800, 4.5),
    ("Nusa Penida", 850, 4.7),
    ("Pulau Belitung", 900, 4.5),
    ("Taman Nasional Lorentz", 950, 4.8),
    ("Anambas", 1100, 4.9),
    ("Dieng Plateau", 400, 4.4),
    ("Pulau Weh", 950, 4.6),
    ("Bangka Island", 850, 4.3),
    ("Pulau Derawan", 1300, 4.8),
    ("Mount Krakatau", 1200, 4.7),
    ("Bunaken Marine Park", 1150, 4.9),
    ("Togean Islands", 1000, 4.6),
    ("Wakatobi National Park", 1400, 4.8),
    ("Lombok Island", 850, 4.5),
    ("Sumbawa Island", 800, 4.4),
    ("Flores Island", 1250, 4.7),
    ("Pulau Alor", 1350, 4.8),
    ("Pulau Bintan", 700, 4.2),
    ("Sulawesi Island", 950, 4.6),
    ("Kalimantan Island", 950, 4.5),
    ("Sumatra Island", 850, 4.3),
    ("Java Island", 900, 4.4),
    ("Moluccas Island", 1000, 4.5),
    ("Tana Toraja", 750, 4.7),
    ("Lake Sentani", 550, 4.4),
    ("Lake Kelimutu", 650, 4.8),
    ("Jayawijaya Peak", 1450, 4.9),
    ("Pulau Sumba", 1200, 4.6),
    ("Pulau Menjangan", 700, 4.7),
    ("Pulau Rote", 800, 4.5),
    ("Karimunjawa Islands", 900, 4.6),
    ("Pulau Seribu", 750, 4.5)
]
```

Berikut juga adalah hasil dari ketiga studi kasus sebelumnya dengan menggunakan data tempat wisata yang baru:

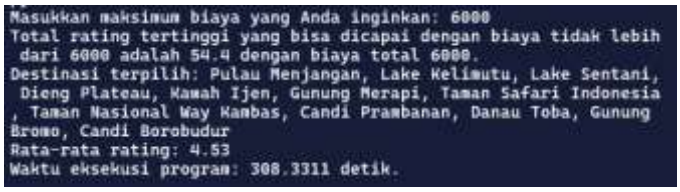
```
Masukkan maksimum biaya yang Anda inginkan: 6000
Total rating tertinggi yang bisa dicapai dengan biaya tidak lebih
 dari Rp6000 adalah 54.4 dengan biaya total 6000.
Destinasi terpilih: Candi Borobudur, Gunung Bromo, Danau Toba, Ca
ndi Prambanan, Taman Nasional Way Kambas, Taman Safari Indonesia,
 Gunung Merapi, Kawah Ijen, Dieng Plateau, Lake Sentani, Lake Kel
imutu, Pulau Menjangan
Rata-rata rating: 4.53
Destinasi: Candi Borobudur, Biaya: 500, Rating: 4.7
Destinasi: Gunung Bromo, Biaya: 350, Rating: 4.5
Destinasi: Danau Toba, Biaya: 400, Rating: 4.3
Destinasi: Candi Prambanan, Biaya: 500, Rating: 4.6
Destinasi: Taman Nasional Way Kambas, Biaya: 700, Rating: 4.4
Destinasi: Taman Safari Indonesia, Biaya: 600, Rating: 4.6
Destinasi: Gunung Merapi, Biaya: 300, Rating: 4.4
Destinasi: Kawah Ijen, Biaya: 350, Rating: 4.6
Destinasi: Dieng Plateau, Biaya: 400, Rating: 4.4
Destinasi: Lake Sentani, Biaya: 550, Rating: 4.4
Destinasi: Lake Kelimutu, Biaya: 650, Rating: 4.8
Destinasi: Pulau Menjangan, Biaya: 700, Rating: 4.7
Waktu eksekusi program: 0.4006 detik.
```

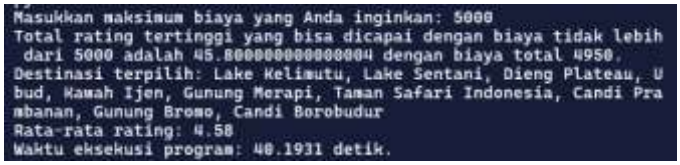
```
Masukkan maksimum biaya yang Anda inginkan: 5000
Total rating tertinggi yang bisa dicapai dengan biaya tidak lebih
 dari Rp5000 adalah 45.800000000000004 dengan biaya total 4900.
Destinasi terpilih: Candi Borobudur, Gunung Bromo, Danau Toba, Ca
ndi Prambanan, Gunung Merapi, Kawah Ijen, Ubud, Dieng Plateau, La
ke Kelimutu, Pulau Menjangan
Rata-rata rating: 4.58
Destinasi: Candi Borobudur, Biaya: 500, Rating: 4.7
Destinasi: Gunung Bromo, Biaya: 350, Rating: 4.5
Destinasi: Danau Toba, Biaya: 400, Rating: 4.3
Destinasi: Candi Prambanan, Biaya: 500, Rating: 4.6
Destinasi: Gunung Merapi, Biaya: 300, Rating: 4.4
Destinasi: Kawah Ijen, Biaya: 350, Rating: 4.6
Destinasi: Ubud, Biaya: 750, Rating: 4.8
Destinasi: Dieng Plateau, Biaya: 400, Rating: 4.4
Destinasi: Lake Kelimutu, Biaya: 650, Rating: 4.8
Destinasi: Pulau Menjangan, Biaya: 700, Rating: 4.7
Waktu eksekusi program: 0.3572 detik.
```

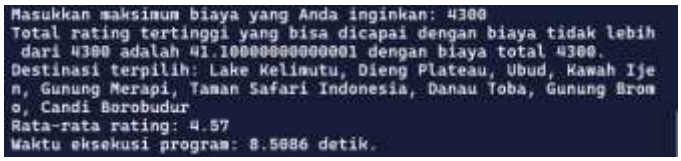
```
Total rating tertinggi yang bisa dicapai dengan biaya tidak lebih
 dari Rp4300 adalah 41.1 dengan biaya total 4200.
Destinasi terpilih: Candi Borobudur, Gunung Bromo, Danau Toba, Ca
ndi Prambanan, Gunung Merapi, Kawah Ijen, Ubud, Dieng Plateau, La
ke Kelimutu
Rata-rata rating: 4.57
Destinasi: Candi Borobudur, Biaya: 500, Rating: 4.7
Destinasi: Gunung Bromo, Biaya: 350, Rating: 4.5
Destinasi: Danau Toba, Biaya: 400, Rating: 4.3
Destinasi: Candi Prambanan, Biaya: 500, Rating: 4.6
Destinasi: Gunung Merapi, Biaya: 300, Rating: 4.4
Destinasi: Kawah Ijen, Biaya: 350, Rating: 4.6
Destinasi: Ubud, Biaya: 750, Rating: 4.8
Destinasi: Dieng Plateau, Biaya: 400, Rating: 4.4
Destinasi: Lake Kelimutu, Biaya: 650, Rating: 4.8
Waktu eksekusi program: 0.2184 detik.
```

Sebagai perbadingan juga akan diberikan hasil dari program serupa namun dengan algortima bruteforce biasa dengan pendekatan rekursif. Kode program yang akan digunakan adalah sebagai berikut: (https://drive.google.com/file/d/1b2LNO3IfmZUnsI3BN-kshtR_N0AEg2vW/view?usp=sharing)

Berikut adalah hasil dari program bruteforce dengan menggunakan 50 data tempat wisata (data kedua):







B. Analisis

Berdasarkan hasil studi kasus yang sudah dilakukan bisa dilihat bahwa program telah berhasil mengeluarkan hasil yang sesuai dengan harapan, dimana rata-rata rating dari tempat wisata yang dipilih cukup tinggi dengan total biaya yang tidak melebihi maksimal biaya yang dimasukkan. Melalui hasil studi kasus diatas juga bisa dilihat bahwa penggunaan pemrograman dinamis untuk permasalahan ini sangatlah sesuai karena sudah sangat efisien dan bisa memberikan waktu eksekusi yang singkat secara keseluruhan.

Waktu eksekusi dari program ini akan sangat bergantung dengan jumlah daftar tempat wisata yang digunakan semakin banyak jumlahnya maka akan semakin lama pula waktu eksekusinya. Hal tersebut bisa dilihat dari hasil studi kasus diatas dimana studi kasus dengan 50 data tempat wisata memiliki rata-rata waktu eksekusi yang lebih lama. Hal ini bisa terjadi karena iterasi yang dilakukan pada program ini dilakukan sejumlah dengan data tempat wisata tersebut.

Selain itu, batas maksimum dari biaya yang dimasukkan juga cukup berpengaruh terhadap waktu eksekusi, dimana semakin besar biaya maksimum maka akan semakin lama pula waktu eksekusinya. Hal ini juga bisa dilihat pada hasil studi kasus dimana ketika batas maksimum biayanya semakin besar maka waktu eksekusinya juga menjadi semakin lama. Hal ini bisa terjadi karena semakin besar maksimum biayanya maka akan semakin banyak pula kemungkinan yang mungkin untuk hasil kombinasi tempat wisata yang dipilih. Dengan semakin banyaknya kemungkinan maka akan semakin banyak pula operasi perbandingan dan operasi matematis yang dilakukan sehingga dapat memperlama waktu eksekusi. Hal ini secara sederhana juga bisa dilihat pada kode program dimana iterasi yang kedua (iterasi di dalam iterasi terluar) dilakukan berdsarkan dengan nilai besaran maksimum biaya yang dimasukkan.

Jika dibandingkan dengan program yang menggunakan bruteforce bisa dilihat bahwa waktu eksekusinya akan sangat jauh berbeda. Hal tersebut sudah menunjukkan bahwa pemrograman dinamis memang sangat efisien dan sangat sesuai untuk permasalahan ini. Ketika ukuran data yang digunakan semakin besar (maksimum biaya atau jumlah data tempat wisata) peningkatan waktu pada program dengan pemrograman dinamis juga tidak terlalu signifikan jika dibandingkan dengan program bruteforce yang memiliki perbedaan waktu yang cukup besar. Program yang memanfaatkan pemrograman dinamis terbukti akan lebih baik untuk digunakan di kebanyakan kasus yang ada karena mempunyai waktu eksekusi yang lebih cepat. Pemrograman dinamis bisa lebih cepat dibandingkan dengan algortima bruteforce karena pada pemrograman dinamis terdapat sebuah tabel 2 dimensi yang digunakan untuk menyimpan data sehingga program yang memanfaatkan pemrograman dinamis bisa mengurangi jumlah operasi berulang yang dilakukan.

Jika dilihat dari kode program maka kompleksitas waktu dari algoritma ini adalah *O(nW)* di mana *n* adalah jumlah destinasi wisata dan *W* adalah batas maksimal biaya. Hal ini disebabkan oleh dua loop bersarang yang mengiterasi melalui setiap destinasi dan setiap nilai anggaran hingga maksimal biaya. Untuk kompleksitas ruang sendiri program ini juga mempunyai nilai yang sama yaitu *O(nW)*, karena memerlukan tabel dua dimensi *dp* yang menyimpan hasil perhitungan untuk setiap kombinasi destinasi dan anggaran. Tabel ini memiliki ukuran *n+1* baris dan *W+1* kolom, di mana setiap entri menyimpan tuple yang mencakup total rating, total biaya, dan daftar destinasi.

Pada prakteknya sebenarnya pemanfaatan ruang memori yang besar ini dapat menjadi kendala, terutama untuk kasus dengan jumlah destinasi dan batas maksimal biaya yang sangat besar. Oleh karena itu, optimasi ruang dapat dipertimbangkan di masa depan, misalnya dengan menggunakan tabel satu dimensi dan memperbarui nilai secara langsung untuk mengurangi penggunaan memori.

Jika dilihat dari kompleksitas waktunya sebenarnya program ini sudah sangat baik dan efisien. Kompleksitas waktu dari program ini bisa dibilang adalah linear dan secara umum

akan aman untuk berbagai macam kasus. Program ini mungkin hanya akan terkendala ketika ukuran dari data tempat wisata dan maksimum biaya yang dimasukkan akan menjadi sangat besar namun hal terebut seharusnya tidak akan mungkin terjadi untuk kasus nyata.

Jika dibandingkan dengan algortima bruteforce dengan kompleksitas waktu $O(2^n)$ yang eksponensial maka sangat jelas bahwa algoritma pemrograman dinamis lebih unggul dari sisi waktu eksekusi dengan kompleksitas waktu yang linear. Namun jika dilihat dari kompleksitas ruangnya maka bisa dikatakan bahwa algoritma bruteforce lebih unggul dikarenakan tidak adanya data yang perlu disimpan oleh algoritma ini berbeda dengan algoritma pemrograman dinamis yang menyimpan semua data perhitungan pada setiap tahapnya. Meskipun algoritma pemrograman dinamis tidak efisien dari segi penggunaan memori, algoritma pemrograman dinamis tetap lebih sesuai untuk permasalahan ini dikarenakan pada kasus nyata data yang digunakan tidak akan sebanyak dan sebesar itu sehingga bisa membebani memori pada kmputer modern saat ini.

Secara keseluruhan program yang dibuat pada makalah ini telah berhasil menunjukkan pemanfaatan pemrograman dinamis pada contoh kasus permasalahan kehidupan sehari-hari seperti pemilihan destinasi wisata ini. Program yang dibuat juga sudah berjalan dengan baik dan efisien serta mampu mengeluarkan keluaran sesuai dengan kriteria yang diinginkan.

VI. KESIMPULAN

Berdasarkan analisis dan hasil studi kasus yang telah dilakukan, dapat disimpulkan bahwa implementasi pemrograman dinamis untuk pemilihan destinasi wisata dengan batasan biaya dan rating telah berhasil dengan baik. Program ini mampu menghasilkan kombinasi destinasi wisata yang optimal dengan total rating yang tinggi tanpa melebihi anggaran yang ditetapkan. Studi kasus yang melibatkan 20 dan 50 data tempat wisata menunjukkan bahwa algoritma ini berjalan efisien dan sesuai dengan ekspektasi, menghasilkan rata-rata rating destinasi wisata yang dipilih cukup tinggi dan memenuhi kriteria biaya yang diberikan.

Efisiensi waktu eksekusi program sangat baik, bahkan ketika jumlah data destinasi wisata dan batas maksimal biaya meningkat. Analisis menunjukkan bahwa waktu eksekusi program dipengaruhi oleh jumlah destinasi wisata dan batas maksimal biaya; semakin banyak jumlah destinasi atau semakin besar batas maksimal biaya, semakin lama waktu eksekusi. Peningkatan waktu eksekusi pada pemrograman dinamis tidak sebanding dengan metode brute force, yang memiliki kompleksitas waktu eksponensial $O(2^n)$. Sebaliknya, pemrograman dinamis memiliki kompleksitas waktu yang lebih rendah $O(nW)$ yang membuatnya lebih cepat dan efisien untuk digunakan dalam skenario nyata.

Kompleksitas waktu dan ruang dari algoritma ini adalah $O(nW)$, yang berarti penggunaan ruang memori yang besar menjadi salah satu kelemahan. Tabel dua dimensi dp yang digunakan untuk menyimpan hasil perhitungan setiap kombinasi destinasi dan anggaran memerlukan ruang memori yang signifikan. Namun, dalam praktik nyata, data yang digunakan biasanya tidak sebesar itu sehingga penggunaan memori yang besar tidak menjadi masalah besar pada komputer modern. Optimalisasi ruang dapat dilakukan di masa depan dengan menggunakan tabel satu dimensi dan memperbarui nilai secara langsung untuk mengurangi penggunaan memori.

Perbandingan antara pemrograman dinamis dan metode brute force menunjukkan bahwa pemrograman dinamis lebih unggul dalam hal efisiensi waktu, meskipun memerlukan lebih banyak memori. Metode brute force memerlukan waktu eksekusi yang jauh lebih lama dan menjadi tidak praktis untuk digunakan pada jumlah data yang besar, sedangkan pemrograman dinamis mampu memberikan solusi yang lebih cepat dan tetap optimal. Implementasi ini menunjukkan bahwa pemrograman dinamis adalah metode yang efektif untuk menyelesaikan masalah optimasi dalam pemilihan destinasi wisata.

Secara keseluruhan, program yang dibuat telah berhasil menunjukkan pemanfaatan pemrograman dinamis pada permasalahan kehidupan sehari-hari seperti pemilihan destinasi wisata. Program berjalan dengan baik dan efisien, serta mampu memberikan hasil yang sesuai dengan kriteria yang diinginkan. Penelitian ini diharapkan dapat memberikan kontribusi yang berharga dalam pengembangan alat bantu perencanaan perjalanan yang lebih cerdas dan adaptif, serta memberikan dasar yang kuat untuk penelitian lebih lanjut di bidang ini. Dengan pendekatan yang sistematis dan kuantitatif, makalah ini mendemonstrasikan bagaimana pemrograman dinamis dapat diadaptasi untuk memecahkan masalah nyata dalam industri pariwisata, menawarkan solusi yang lebih adaptif dan responsif dibandingkan metode heuristik atau trial-and-error yang umum digunakan.

REFERENCES

[1] Munir, Renaldi. 2024. Program Dinamis (Dynamic Programming) Bagian 1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf, diakses pada 11 Juni 2024.

[2] Munir, Renaldi. 2024. Program Dinamis (Dynamic Programming) Bagian 2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf, diakses pada 11 Juni 2024.

[3] 0-1 Knapsack Problem. 2024. GeeksforGeeks. https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/, diakses pada 11 Juni 2024.

[4] Introduction to Knapsack Problem, its Types and How to solve them. 2024. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-knapsack-problem-its-types-and-how-to-solve-them/, diakses pada 11 Juni 2024.

[5] Dynamic Programming or DP. 2024. GeeksforGeeks. https://www.geeksforgeeks.org/dynamic-programming/, diakses pada 11 Juni 2024.

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, 12 Juni 2024

Mesach Harmasendo 13522117