Penerapan *Markov Chain* dengan *Dynamic Programming* dan Optimasinya untuk Prediksi Cuaca di Dago

Nyoman Ganadipa Narayana - 13522066
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522066@std.stei.itb.ac.id

***Abstrak*—Prediksi cuaca di Dago, Bandung, memiliki peranan penting karena letak geografisnya yang unik sebagai kawasan dataran tinggi dengan perubahan cuaca yang cepat. Selain itu, Dago merupakan pusat pendidikan yang menjadi rumah bagi berbagai universitas terkemuka seperti Institut Teknologi Bandung, menarik ribuan mahasiswa setiap tahunnya. Mahasiswa dan staf akademis sangat bergantung pada prediksi cuaca yang akurat untuk perencanaan sehari-hari dan kegiatan luar ruangan. Penelitian ini bertujuan untuk menerapkan *Markov Chain* dengan *Dynamic Programming* serta mengoptimalkannya guna meningkatkan akurasi prediksi cuaca di Dago. Dengan memanfaatkan metode ini, diharapkan prediksi cuaca dapat lebih andal, membantu komunitas akademis dan lokal dalam mengantisipasi kondisi cuaca yang sering tidak terduga, sehingga mengurangi risiko dan meningkatkan kenyamanan serta keselamatan.**

***Kata Kunci*—Prediksi Cuaca, *Markov Chain*, *Dynamic Programming*, Optimasi, Dago.**

I. Pendahuluan

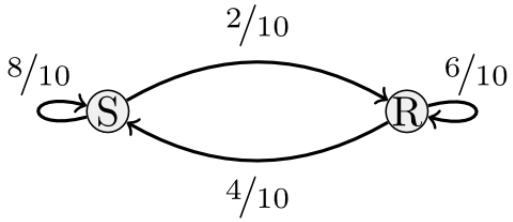
Prediksi cuaca di Dago, Bandung, memegang peranan penting tidak hanya karena letak geografisnya yang unik sebagai kawasan dataran tinggi yang sering mengalami perubahan cuaca cepat, tetapi juga karena daerah ini adalah pusat pendidikan dan kegiatan akademis. Dago dan sekitarnya adalah rumah bagi beberapa universitas dan institusi pendidikan tinggi terkemuka, seperti Institut Teknologi Bandung, yang menarik ribuan mahasiswa setiap tahun. Mahasiswa dan staf akademis ini bergantung pada prediksi cuaca yang akurat untuk perencanaan sehari-hari, perjalanan, dan kegiatan luar ruangan. Prediksi cuaca yang dapat diandalkan sangat penting untuk membantu mereka mengantisipasi dan mengadaptasi rencana mereka terhadap kondisi cuaca yang sering kali tidak terduga, mengurangi risiko dan meningkatkan kenyamanan serta keselamatan. Oleh karena itu, penelitian tentang penerapan *Markov Chain* dan *Dynamic Programming* untuk meningkatkan akurasi prediksi cuaca di Dago tidak hanya relevan secara akademis, tetapi juga sangat bermanfaat bagi komunitas lokal.

II. Landasan Teori

A. *Markov Chain*

Markov Chain adalah model statistik yang digunakan untuk menggambarkan sistem yang berubah dari satu keadaan (*state*) ke keadaan lain dalam ruang keadaan yang terbatas, dengan probabilitas yang hanya bergantung pada keadaan saat ini dan tidak pada bagaimana keadaan tersebut dicapai (sifat tanpa ingatan atau *memoryless*).

Dalam konteks prediksi cuaca, *Markov Chain* bisa digunakan untuk memodelkan transisi antara berbagai kondisi cuaca (misalnya cerah, berawan, hujan) sebagai proses stokastik. Gambar 1 adalah contoh dari model *Markov Chain* pada topik cuaca.



Gambar 1 Model *Markov Chain* yang berisi keadaan terang dan hujan. Diambil dari laman https://martin-thoma.com/python-markov-chain-packages/ yang diakses pada 10 Juni 2024.

Oleh karena sifat dari model *Markov Chain* yang seringkali direpresentasikan dengan graf, maka kita dapat mewakili model *Markov Chain* dengan matriks – lebih dikenal sebagai matriks transisi – sehingga suatu baris *i* dan kolom *j* menyatakan probabilitas dari keadaan *i* ke kolom *j*. Dengan menggunakan keadaan *sunny* (S) sebagai indeks 0 dan keadaan *rainy* (R) sebagai indeks 1, maka Gambar 1 dapat diwakili oleh matriks transisi pada persamaan 1.

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}$$

Persamaan 1 Matrik Transisi dari Gambar 1

B. *Graf berbobot*

Pada graf berbobot, setiap sisi memiliki nilai atau 'bobot' tertentu yang dapat mewakili jarak, biaya, atau parameter lainnya. Model *Markov Chain* adalah salah satu contoh dari graf berbobot di mana setiap sisi menunjukkan adanya transisi dari simpul satu ke suatu simpul lainnya dengan probabilitas tertentu.

C. *Dynamic Programming*

Dynamic Programming (DP) adalah metodologi yang digunakan dalam komputasi matematis untuk memecahkan masalah dengan memecahnya menjadi submasalah yang lebih sederhana, memastikan bahwa setiap submasalah dihitung sekali dan hasilnya disimpan untuk penggunaan di masa depan. Ini memungkinkan solusi yang efisien untuk masalah yang mungkin memiliki banyak solusi berulang jika didekati dengan metode yang naif. Prinsip utama DP adalah penggunaan tabelisasi atau *memoization* untuk menyimpan hasil dari perhitungan yang telah selesai, sehingga menghindari perhitungan ulang yang tidak perlu dan mempercepat proses keseluruhan.

D. *Divide and Conquer*

Divide and Conquer adalah sebuah paradigma algoritma yang memecah masalah menjadi beberapa sub-masalah yang lebih kecil, serupa, dan lebih mudah dikelola, kemudian menyelesaikan masing-masing sub-masalah tersebut secara independen, dan akhirnya menggabungkan solusi dari submasalah untuk mendapatkan solusi masalah awal. Pendekatan ini sangat efektif dalam menyelesaikan masalah kompleks karena ia secara efisien membagi masalah menjadi komponen yang dapat ditangani secara lebih mudah, seringkali melalui rekursi.

E. *API (Application Programming Interface)*

API (*Application Programming Interface*) adalah protokol dan *tools* yang memungkinkan berbagai aplikasi perangkat lunak untuk berinteraksi dan berkomunikasi satu sama lain. API memfasilitasi modularitas dan abstraksi, memungkinkan developer untuk mengintegrasikan fungsi-fungsi dari berbagai sistem tanpa perlu memahami detail implementasi yang mendalam. Dengan menyediakan antarmuka standar, API meningkatkan kemampuan perawatan, pembaruan, dan skalabilitas aplikasi, serta mendukung integrasi lintas platform yang efektif. Hal ini membuat API menjadi komponen penting dalam pengembangan aplikasi modern, memungkinkan ekosistem perangkat lunak yang lebih kooperatif dan fleksibel.

III. METODE

Metode merupakan bagian krusial dalam penelitian ini karena menyediakan landasan teoritis dan praktis untuk mencapai tujuan penelitian. Dalam upaya untuk meningkatkan akurasi serta menyederhanakan prediksi cuaca di Dago, Bandung, kita perlu metode yang terstruktur dan sistematis. Bab ini akan membahas secara rinci pendekatan yang digunakan dalam penerapan model *Markov Chain*, asumsi-asumsi yang diperlukan, modifikasi model untuk relevansi yang lebih tinggi, serta teknik pengambilan data yang mendukung model ini. Pendekatan metodologis yang jelas dan terperinci ini memastikan bahwa hasil yang diperoleh dapat dipertanggungjawabkan dan memiliki tingkat akurasi yang diharapkan.

A. *Representasi*

Dalam upaya memecahkan masalah konkret terkait prediksi cuaca di Dago, makalah ini mengadopsi model *Markov Chain* dengan keadaan (*state*) yang digunakan merupakan salah satu dari Terang, Berawan, Mendung, dan Hujan. Setiap transisi dari suatu keadaan ke keadaan yang lain merupakan transisi keadaan dalam waktu 6 jam.

B. *Asumsi*

Untuk menyederhanakan masalah konkret yang ada ke dalam terminologi model *Markov Chain*, dilakukan beberapa asumsi serta simplifikasi untuk mempermudah pengambilan dan pengolahan data. Beberapa asumsi yang dilakukan oleh penulis dinilai tidak akan berdampak besar terhadap hasil yang akan didapatkan, karena tujuan dari asumsi ini murni untuk penyederhanaan tanpa mengurangi makna dari kondisi sesungguhnya. Beberapa asumsi tersebut diterangkan pada berikut ini.

1. Dalam merepresentasikan masalah ini ke dalam model *Markov Chain* diperlukan beberapa asumsi terkait dengan *Markov Chain* itu sendiri, yaitu suatu keadaan di masa depan hanya bergantung pada keadaan sekarang dan matriks transisi yang dibuat.
2. Matriks transisi yang dibuat merupakan hasil observasi dari data yang diperoleh pada keadaaan konkretnya selama 14 hari, dengan menggunakan model yang mengakuisisi transisi setiap 6 jam, maka diperlukan 56 data transisi.
3. Probabilitas dari suatu keadaaan ke keadaan yang lain adalah banyak transisi keadaan tersebut dari total transisi yang ada.
4. Suatu transisi dari suatu keadaan ke keadaan tertentu murni hanya berdasarkan cuaca pada saat sekarang, tidak terpengaruh oleh kelembapan, kecepatan angin, suhu, dan lainnya.

C. *Modifikasi Model Markov Chain*

Untuk membuat prediksi yang lebih relevan terhadap masa kini, seperti musim yang pada keadaan konkretnya terbagi menjadi musim panas dan musim hujan, maka model *Markov Chain* yang dibuat merupakan hasil observasi dari data-data transisi keadaan 14 hari sebelum hari ini.

D. Pengambilan Data

Dari ketiga bagian yang telah disebutkan: Representasi, Asumsi, serta Modifikasi Model *Markov Chain*, maka kita dapat membuat model Markov Chain yang lebih sederhana.

Untuk setiap prediksi yang dilakukan, diperlukan data cuaca saat ini, dan data cuaca 14 hari ke belakang untuk mengkonstruksi matriks transisi yang mewakili model *Markov Chain.* Pengambilan data ini dapat dilakukan melalui data yang diberikan oleh API *weatherbit* pada laman https://www.weatherbit.io/api. Dengan menge-*set* API untuk mengambil data pada interval waktu tertentu dan pada lokasi tertentu, API tersebut akan memberikan kumpulan data yang memuat waktu, cuaca, kelembapan, kecepatan angin, dan lain-lain. Tetapi, *response* dari API tersebut memberikan cuaca dengan kategori yang cukup banyak, setiap response dari API tersebut dikategorisasi kembali oleh penulis sedemikian sehingga data yang didapat terkategorisasi menjadi hanya cuaca Terang, Berawan, Mendung, dan Hujan.

IV. Data dan Pengolahan

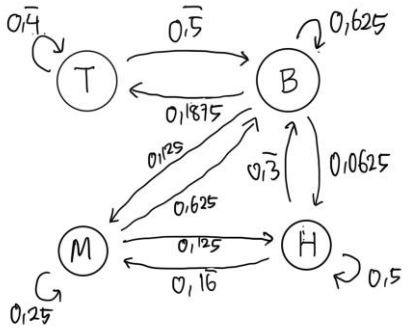
Dari bagian sebelumnya, telah dijabarkan bagaimana makalah ini akan mengambil data serta asumsi dibuat, sehingga telah disepakati bahwa data yang diambil adalah data riwayat 14 hari terakhir dihitung dari kapan prediksi tersebut dilakukan. Oleh karena itu, data yang ada pada makalah ini adalah data yang didapat oleh penulis saat makalah ini dibuat, yaitu pada tanggal 10 Juni 2024.

Pada 10 Juni 2024, menggunakan API dari https://www.weatherbit.io/api untuk mendapatkan cuaca-cuaca 14 hari terakhir, penulis mendapatkan data-data yang ditunjukkan oleh *Table I.*

TABLE I. Data 14 Hari Terakhir Dihitung Dari 10 Juni 2024

No	Tanggal	Pukul (0 – 23)	Cuaca
1	2024-05-27	18	Mendung
2	2024-05-28	00	Mendung
3	2024-05-28	06	Berawan
4	2024-05-28	12	Berawan
5	2024-05-28	18	Berawan
6	2024-05-29	00	Mendung
7	2024-05-29	06	Berawan
8	2024-05-29	12	Berawan
9	2024-05-29	18	Berawan
10	2024-05-30	00	Berawan
11	2024-05-30	06	Berawan
12	2024-05-30	12	Berawan
13	2024-05-30	18	Berawan
14	2024-05-31	00	Mendung
15	2024-05-31	06	Berawan
16	2024-05-31	12	Mendung
17	2024-05-31	18	Berawan
18	2024-06-01	00	Hujan
19	2024-06-01	06	Hujan
20	2024-06-01	12	Hujan
21	2024-06-01	18	Berawan
22	2024-06-02	00	Berawan
23	2024-06-02	06	Hujan
24	2024-06-02	12	Hujan
25	2024-06-02	18	Mendung
26	2024-06-03	00	Mendung
27	2024-06-03	06	Hujan
28	2024-06-03	12	Berawan
29	2024-06-03	18	Berawan
30	2024-06-04	00	Berawan
31	2024-06-04	06	Berawan
32	2024-06-04	12	Berawan
33	2024-06-04	18	Berawan
34	2024-06-05	00	Mendung
35	2024-06-05	06	Berawan
36	2024-06-05	12	Berawan
37	2024-06-05	18	Terang
38	2024-06-06	00	Terang
39	2024-06-06	06	Berawan
40	2024-06-06	12	Berawan
41	2024-06-06	18	Berawan
42	2024-06-07	00	Terang
43	2024-06-07	06	Berawan
44	2024-06-07	12	Berawan
45	2024-06-07	18	Berawan
46	2024-06-08	00	Terang
47	2024-06-08	06	Berawan
48	2024-06-08	12	Terang
49	2024-06-08	18	Terang
50	2024-06-09	00	Terang
51	2024-06-09	06	Berawan
52	2024-06-09	12	Berawan
53	2024-06-09	18	Terang
54	2024-06-10	00	Terang
55	2024-06-10	06	Berawan
56	2024-06-10	12	Terang

Dari pada *Table I* dapat dibuat model *Markov Chain* yang menggambarkan transisi dari suatu cuaca ke cuaca lainnya dalam kurun waktu 6 jam, Gambar 2 menggambarkan model *Markov Chain* data tersebut dengan keterangan simpul T untuk keadaan cuaca terang, simpul B untuk keadaan cuaca berawan, simpul M untuk keadaan cuaca mendung, dan simpul H untuk keadaan cuaca hujan, dengan sisi-sisi menyatakan suatu probabilitas terjadinya transisi tersebut dari simpul *source.*



Gambar 2 Model *Markov Chain* dari *Table 1*. Gambar merupakan dokumen penulis.

Model *Markov Chain* kemudian dapat diwakili oleh suatu matriks, yaitu matriks transisi yang direpresentasikan seperti *Table II*.

TABLE II. TABEL TRANSISI

		Tujuan			
		T	***B***	***M***	***H***
Awal	***T***	*0.44*	*0.56*	*0*	*0*
	B	*0.1875*	*0.625*	*0.125*	*0.0625*
	M	*0*	*0.625*	*0.25*	*0.125*
	H	*0*	*0.333*	*0.167*	*0.5*

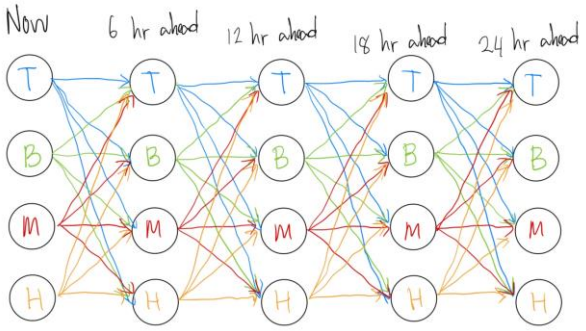
Table II kemudian dapat dikonversi menjadi matriks transisi yang digambarkan oleh Persamaan 2.

$$P = \begin{pmatrix} 0.44 & 0.56 & 0 & 0 \\ 0.1875 & 0.625 & 0.125 & 0.0625 \\ 0 & 0.625 & 0.25 & 0.125 \\ 0 & 0.333 & 0.167 & 0.5 \end{pmatrix}$$

Persamaan 2 Matriks Transisi

Setelah mendapatkan matriks transisi dari data yang relevan, yaitu matriks yang ditunjukkan oleh P pada Persamaan 2, sekarang hanya diperlukan keadaan cuaca pada prediksi tersebut dilakukan, data ini dapat diambil menggunakan API *weatherbit* pada laman https://www.weatherbit.io/api, pada saat makalah ini dibuat, 10 Juni 2024 pukul 18, cuacanya adalah terang.

Selesai mendapatkan matriks transisi dari 14 hari terakhir serta cuaca sekarang, maka data yang dibutuhkan sudah cukup untuk memprediksi cuaca berdasarkan model *Markov Chain*, dengan mencari setiap probabilitas cuaca yang akan terjadi pada waktu yang ingin diprediksi. Misalnya untuk mendapatkan prediksi cuaca pada 24 jam yang akan datang, maka kita perlu mencari cuaca dengan probabilitas tertinggi dalam 4 transisi dari keadaan sekarang – ingat bahwa 1 transisi pada makalah ini adalah 6 jam. Pencarian probabilitas tertinggi tersebut dapat dilakukan seperti skema pada Gambar 3.



Gambar 3 Skema pencarian masing-masing probabilitas cuaca. Gambar merupakan dokumen penulis.

Gambar 3 merupakan gambaran skema pencarian seluruh probabilitas cuaca pada 4 transisi yang akan datang. Yaitu dengan cara menelusuri semua rute dari simpul awal sampai simpul tujuan kemudian menjumlahan semua probabilitas dari rute yang memiliki simpul tujuan yang sama. Tetapi apabila algoritma ini dikerjakan dengan metode *brute force* maka jumlah eksekusi akan terlalu banyak sehingga waktu eksekusi pun sangat lama.

Perhatikan bahwa jika skema pada gambar 3 dilakukan dengan menggunakan metode *brute force* maka akan banyak terjadi perhitungan probabilitas yang tidak perlu karena sebetulnya sebelumnya sudah pernah dihitung. Salah satu solusi dari masalah ini adalah menggunakan *dynamic programming* yaitu dengan mengingat hasil dari kalkulasi yang sudah pernah dihitung, teknik ini dinamakan *memoization*. Berikut adalah *pseudocode* dari implementasi skema pada Gambar 3 dengan menggunakan *dynamic programming* untuk mendapatkan probabilitas dari seluruh cuaca.

```
function get_probabilities (transition_matrix : SquareMatrix of real, num_transition: integer, current_weather: integer) -> array of real

{ Mengembalikan probabilitas dari seluruh cuaca setelah transisi sejumlah num_transition kali }


ALGORITMA

    num_state <- length(transition_matrix)

    probabilities <- array [0..num_transition] of array [0..num_state] of real


    { initialize the first row with zero }

    i traversal [0..num_state - 1]

       probabilities[0][i] <- 0


    probabilities[0][current _weather] = 1


    i traversal [1..num_transition]

       j traversal [0..num_state - 1]

          k traversal [0..num_state - 1]

             probabilites[i][j] <-
    probabilities[i - 1][k] *
    transition_matrix[k][j]


    -> probabilities[num_transition]
```

Pseudocode tersebut mendapatkan seluruh probabilitas dari cuaca pada waktu tertentu dalam $O(nk^2)$ dengan n merupakan jumlah transisi dan k merupakan jumlah keadaan. Hal ini sudah sangat cepat jika dibandingkan dengan metode *brute force*. Lebih lanjut, dengan sedikit observasi lanjutan, perhatikan bahwa probabilitas dari setiap cuaca setelah jumlah transisi tertentu sama saja dengan P pada Persamaan 3, dengan N merupakan jumlah transisi.

$$P(N) = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0.44 & 0.56 & 0 & 0 \\ 0.1875 & 0.625 & 0.125 & 0.0625 \\ 0 & 0.625 & 0.25 & 0.125 \\ 0 & 0.333 & 0.167 & 0.5 \end{pmatrix}^N$$

Persamaan 3 Persamaan penentuan probabilitas setiap cuaca.

Persamaan 3 menyatakan P pada transisi ke N adalah matriks cuaca awal dikali dengan matriks transisi yang dipangkatkan dengan N. Dengan metode divide and conquer, kita dapat menyelesaikan Persamaan 3 dengan implementasi *pseudocode* sebagai berikut.

```
function multiplyMatrix(matrix1: Matrix, matrix2: Matrix) -> Matrix
{ this is implemented }


function matrixExponentitaion(m: squareMatrix of real, power: int) -> squareMatrix of real


ALGORITMA
   if (power = 1) then
      -> m
   else
      sqrt <- matrixExponentiation(m, power div 2)
      if (power % 2 = 0) then
         -> multiplyMatrix(sqrt, sqrt)
      else
         -> multiplyMatrix(matrix, multiplyMatrix(sqrt, sqrt))



function get_probabilities (transition_matrix : SquareMatrix of real, num_transition: integer, current_weather: integer) -> array of real
{ Mengembalikan probabilitas dari seluruh cuaca setelah transisi sejumlah num_transition kali }


ALGORITMA
   -> matrixExponentiation(transition_matrix, num_transition)[current _weather]
```

Pseudocode tersebut memberikan kompleksitas waktu yang jauh lebih cepat (untuk n >> k) dibandingkan dengan menggunakan *dynamic programming*, yaitu sebesar $O((\log n)k^3)$ dengan n adalah jumlah transisi dan k adalah jumlah keadaaan.

Sekarang kita telah mengetahui implementasinya, akan dilakukan uji coba prediksi cuaca untuk 6, 12, 18, dan 24 jam ke depan. Kemudian akan dicek oleh penulis seberapa besar persentase prediksi tersebut benar. Namun untuk melonggarkan prediksi pada uji coba, maka yang diambil sebagai prediksi cuaca adalah cuaca dengan probabilitas tertinggi dan kedua tertinggi. *Table III* menunjukkan hasil uji coba dan persentase kebenaran dari 4 prediksi yang dilakukan.

TABLE III. TABEL UJI COBA

	Prediksi Cuaca I	**Prediksi Cuaca II**	**Realita Cuaca**	**Prediksi Benar**
6 jam ke depan (2024-06-11:00)	Berawan	Terang	Terang	Benar
12 jam ke depan (2024-06-11:06)	Berawan	Terang	Terang	Benar
18 jam ke depan (2024-06-11:12)	Berawan	Terang	Berawan	Benar
24 jam ke depan (2024-06-11:18)	Berawan	Terang	Hujan	Salah
Persentase prediksi benar				75%

Hasil menunjukkan angka cukup baik, tetapi apabila sampelnya sedikit hasil performanya kurang dapat dipercaya, sehingga penulis melakukan uji coba 10000 riwayat cuaca yang sudah terjadi untuk menghitung berapa persentase bahwa suatu cuaca yang terjadi sudah terprediksi. *Table IV* Menunjukkan hasil yang diperoleh dari 10000 uji prediksi cuaca 12 jam ke depan.

TABLE IV. TABEL UJI 10000 DATA

	Persentase cuaca tersebut sudah diprediksi akan terjadi
Terang	55,6%
Berawan	99,8%
Mendung	4,03%
Hujan	83,2%

Hasil dari uji 10000 data menunjukkan: 55,6% keadaan terang; 99.8% keadaan berawan; 4,03% keadaan mendung; dan 83,2% keadaan hujan memang sudah diprediksi oleh program yang dibuat. Kemudian dilakukan juga uji coba 10000 prediksi untuk menghitung persentase prediksi tersebut benar. Ingat kembali bahwa program akan memprediksi 2 cuaca untuk kelonggaran uji coba, jika cuaca adalah salah satu dari 2 prediksi tersebut, maka dianggap benar. *Table V* menunjukkan hasil uji coba 10000 prediksi tersebut.

TABLE V. TABEL UJI 10000 PREDIKSI

	Persentase cuaca tersebut sudah diprediksi akan terjadi
Terang	**80,9%**
Berawan	**82,6%**
Mendung	**64,3%**
Hujan	**83.1%**

Hasil dari uji 10000 prediksi menunjukkan: 80,9% prediksi terang; 82,6% prediksi berawan; 64,3% prediksi mendung; dan 83,1% prediksi hujan memang akan terjadi.

V. KESIMPULAN

Hasil dari uji coba program pada *Table IV* menunjukkan bahwa hampir setiap keadaan cuaca berawan dan hujan memang sudah diprediksi, sementara hasil pada *Table V* menunjukkan sebagian besar prediksi memang akan terjadi. Mengingat bahwa *state* yang ditinjau hanya sedikit, dan juga tanpa melihat aspek kelembapan, kecepatan angin, temperatur, dan lain-lain, model *Markov Chain* untuk prediksi cuaca bekerja dengan cukup baik.

Pemanfaatan algoritma *dynamic programming* dalam implementasi program terbukti dapat melakukan pengurangan jumlah dan waktu eksekusi dibandingkan dengan metode *brute force*, selain itu, optimasinya dengan menggunakan *divide and conquer* terbukti mempercepat program untuk waktu prediksi yang cukup lama.

VI. CATATAN KENDALA DAN EVALUASI

Dalam proses pengambilan data untuk makalah ini, beberapa kali penulis kesulitan untuk mendapatkan API yang cocok. Selain itu proses *tuning* untuk menentukan berapa waktu yang cocok untuk satu transisi, berapa data yang cocok untuk digunakan dalam membentuk model *Markov Chain* juga cukup menghabiskan waktu penulis.

Meskipun telah melalui kendala tersebut, makalah ini masih jauh untuk dibilang sempurna, terdapat evaluasi yang dapat dijadikan pertimbangan dan koreksi dari proses penelitian ini, salah satunya adalah pengambilan asumsi, untuk pembuatan prediksi cuaca yang lebih baik seharusnya juga memperhatikan kelembapan, kecepatan angin, dan juga temperatur, sehingga sebaiknya mempertimbangkan untuk membuat model *Markov Chain* yang terdiri dari kombinasi dari cuaca dan atribut yang telah disebutkan.

PRANALA YOUTUBE

https://youtu.be/qWvkpc7qHBw?si=EThP6Hnjc_iI9C9G

PRANALA GITHUB

https://github.com/ganadipa/Weather-Prediction-for-Dago

UCAPAN TERIMA KASIH

Pada akhir makalah ini, saya ingin menyampaikan rasa syukur saya kepada Tuhan Yang Maha Kuasa atas kelancaran dan keberhasilan penyelesaian makalah IF2211 Strategi Algoritma ini. Saya juga ingin mengungkapkan rasa terima kasih saya yang sebesar-besarnya kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc., yang bimbingan, pengetahuannya, serta kesabarannya dalam setiap pemaparan materi di kelas sehingga saya dapat menuangkan materi tersebut ke dalam makalah ini. Penghargaan juga saya tujukan kepada Bapak Dr. Ir. Rinaldi Munir, M.T. atas penyediaan materi pembelajaran yang inovatif selama kuliah ini. Terima kasih tak terhingga juga saya ucapkan kepada keluarga, teman-teman, senior, dan seluruh anggota civitas akademika lainnya yang telah memberikan dukungan dan kontribusi dalam penyelesaian makalah ini. Saya berharap makalah ini memberikan manfaat nyata tidak hanya bagi mahasiswa ITB, tetapi juga bagi seluruh masyarakat yang sedang tinggal di Dago maupun sekitarnya.

REFERENSI

[1] A. N. Langville and C. D. Meyer, "Google's PageRank and Beyond: The Science of Search Engine Rankings," Princeton University Press, 2006.

[2] https://disi.unitn.it/~locigno/teaching-duties/spe/09_MarkovChains.pdf, diakses pada 12 Juni 2024.

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian1.pdf, diakses pada 12 Juni 2024

[4] https://martin-thoma.com/python-markov-chain-packages/, diakses pada 10 Juni 2024

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," 3rd ed., MIT Press, 2009.

[6] Laaksonen, Antti, “Competitive Programming Handbook.” 2018, pp. 126, 32.

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

Nyoman Ganadipa Narayana 13522066