# Pemanafaatan Algoritma Branch and Bound dalam Menyelesaikan TSP untuk Menentukan Rute Tur Destinasi Wisata di Bali

Annisa Rahim - 13518089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): annisarahim21@gmail.com

***Abstrak*—Salah satu cara untuk mempersingkat waktu untuk mengunjungi banyaknya tempat wisata di Bali dengan seefisien mungkin adalah dengan meminimalkan jarak tur yang ditempuh untuk mengunjungi seluruh destinasi, yang merupakan salah satu persoalan TSP, dan salah satu algoritma yang dapat dimanfaatkan untuk menyelesaikan TSP tersebut adalah dengan menggunakan algoritma Branch and Bound.**

***Kata kunci—destinasi wisata; bali; tsp; branch and bound***

## I. PENDAHULUAN

Bali merupakan salah satu tempat dari banyaknya destinasi wisata yang sangat diminati turis, baik dalam negeri maupun luar negeri. Pulau yang kerap disebut Pulau Dewata ini memiliki keindahan alam dan budaya yang luar biasa. Karena banyaknya destinasi wisata, kebanyakan turis datang ke Bali untuk mengunjungi lebih dari satu tujuan wisata sekaligus. Destinasi wisata yang dimiliki Bali terdapat di beberapa titik yang memiliki jarak satu sama lain. Oleh karena itu, dibutuhkan rute perjalanan terbaik untuk mengunjungi tempat-tempat tersebut, agar tur dapat berjalan dengan efisien, baik dalam aspek biaya maupun waktu, dan semua tempat sempat dikunjungi sebelum kembali ke penginapan.

Gambar 1 Bali
(Sumber: https://www.portonews.com/2017/keuangan-dan-portfolio/pariwisata/bali-permata-pariwisata-yang-terancam-kusam/diakses pada tanggal 2 Mei 2020 pukul 18.02 WIB)

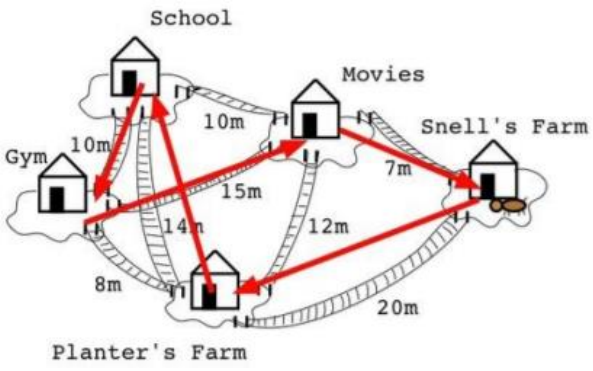
Persoalan ini mirip dengan permasalahan *Traveling Salesman Problem*, yaitu mencari jalur terpendek untuk megunjungi semua titik yang ada di sebuah graf lalu kembali ke tempat semula. Dalam kasus ini, titik merepresentasikan destinasi wisata, dan tempat penginapan menjadi titik awal. Ada banyak cara untuk menyelesaikan persoalan ini, salah satunya adalah dengan menggunakan algoritma Branch and Bound. Pemanfaatan algoritma ini juga memiliki beragam teknik, namun teknik yang kali ini akan dibahas adalah dengan menggunakan *reduced-cost matrix*.

## II. DASAR TEORI

### A. *Travelling Salesman Problem (TSP)*

TSP atau *Travelling Salesman Problem* adalah suatu masalah yang sering diasosiasikan dengan algoritma-algoritma seperti brute force, exhaustive search, dan lain-lain. Masalah ini adalah sebagai berikut: Diberikan *n* buah kota, diketahui juga jarak antar setiap kota satu sama lain, yang harus dicari adalah perjalanan (*tour*) terpendek yang melalui setiap kota lainnya hanya sekali dan kembali lagi ke kota asal keberangkatan.

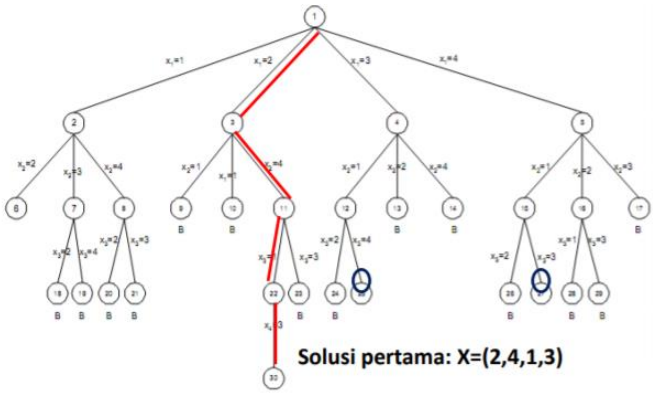
Persoalan ini dimodelkan sebagai graf lengka dengan *n* buah simpul, dan bobot pada setiap sisi menyatakan jarak antara dua buah kota yang bertetangga.




Gambar 2 Ilustrasi *Travelling Salesman Problem*
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-(2016).pdf diakses pada tanggal 3 Mei 2020 pukul 21.05 WIB)

Cara menemukan persoalan ini sebeneranya adalah dengan menemukan sirkuit Hamilton pada graf, namun selain memperhatikan sirkuit, yang harus dipertimbangkan adalah pencarian sirkuit Hamilton dengan bobot yang seminimum mungkin

*B. Algoritma Branch and Bound (BnB)*




Gambar 3 Ilustrasi pohon pencarian dalam algoritma *Branch and Bound* (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf diakses pada tanggal 3 Mei 2020 pukul 21.05 WIB)

Branch and Bound adalah salah satu algoritma traversal graf, yaitu algoritma untuk mengunjungi simpul secara sistematik. Pencarian graf yang merepresentasikan persoalan memiliki tujuan untuk mencari sebuah solusi. Algoritma ini termasuk kedalam pencarian dengan informasi (informed search).

Algoritma ini mencari solusi dengan membentuk pohon ruang status, mengekspansi simpul-simpul dan membunuh simpul hidup yang tidak mengarah ke solusi. Terdapat *bound* yang akan menentukan suatu simpul masih layak atau harus dibunuh. Simpul yang dipilih untuk diekspan adalah estimasi solusi terbaik sejauh ini.

Algoritma Branch and Bound mirip seperti Breadth First Searc(BFS), namun urutan simpul hidupnya bergantung pada nilai cost dari simpul-simpul tersebut.

Breadth First Search adalah salah satu algoritma traversal graf. Algoritma ini memiliki simpul awal, lalu membangkitkan simpul-simpul lainnya dalam pencarian dengan urutan melebar, yaitu dengan membangkitkan semua anak pada satu simpul, dilanjut dengan membangkitkan anak-anak dari simpul tersebut. Hal ini terus dilanjutkan hingga mencapai simpul yang dicari.

Pada algoritma Branch and Bound, cara pencariannya sama, namun setiap simpul diberi sebuah nilai cost, misalnya c(i), yang merupakan nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i. Simpul berikutnya yang akan di-expand adalah yang memiliki *cost* paling kecil pada kasus minimasi, atau paling besar pada kasus maksimasi.

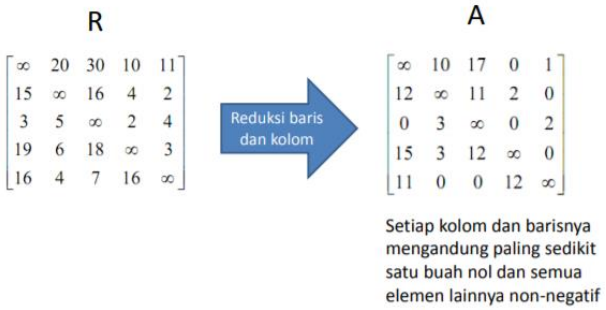
Sebuah simpul akan dipangkas ketika: nilai simpul tidak lebih baik dari nilai terbaik, simpul tidak merepresentasikan solusi yang layak karena melanggar batas, atau solusi yang mungkin pada simpul tersebut hanya terdiri atas satu titik (tidak ada pilihan lain, mengambil solusi terbaik).

Dalam permasalahan yang membutuhkan optimisasi, yaitu meminimalkan atau memaksimalkan suatu fungsi objektif, tanpa melanggar batasan dari persoalan tersebut, menggunakan algoritma Branch and Bound adalah salah satu pilihan yang tepat.

*C. TSP menggunakan Branch and Bound*

Terdapat dua jenis cara untuk memecahkan TSP menggunakan algoritma *Branch and Bound*: Matriks ongkos tereduksi, atau bobot minimum tur legkap. Pada makalah ini, cara yang digunakan adalah menggunakan matriks ongkos tereduksi.

Sebuah matriks dikatakan tereduksi jika setiap kolom dan barisnya mengandung paling sedikit satu buah nol, dan semua elemen lainnya non-negatif.




Gambar 4 Contoh reduksi baris dan kolom (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf, diakses pada tanggal 2 Mei 2020 pukul 20.53 WIB)

Algoritma ini menggunakan matriks keterhubungan untuk menentukan jarak semua titik satu sama lain. Langkah-langkahnya adalah sebagai berikut:

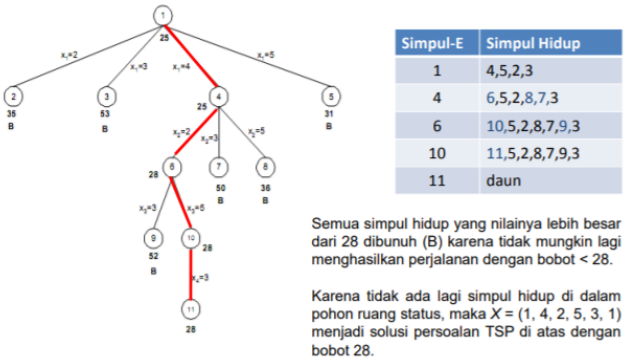
1. Menentukan matriks jarak
2. Mereduksi matriks dan menyimpan semua nilai pengurang baris dan kolom. Jika sudah menjadi matriks tereduksi, matriks tersebut akan menjadi simpul akar dari pencarian, dan jumlah total semua pengurang akan menjadi cost akar.
3. Membangkitkan semua anak yang mungkin dimiliki akar, yaitu pergerakan dari titik sekarang ke titik-titik yang belum dikunjungi. Misalnya matriks jarak adalah matriks A dan pergerakan dilakukan dari titik i ke titik j, maka langkah-langkahnya adalah sebagai berikut:
    a. Mengubah semua nilai pada baris I dan kolom j menjadi ∞. Hal ini dilakukan untuk mencegah agar tidak ada lintasan yang keluar dari simpul i atau masuk pada simpul j
    b. Mengubah A(j,1) menjadi ∞, untuk mencegah j pergi ke titik awal.
    c. Mereduksi kembali semua baris dan kolom pada matriks A, kecuali elemen yang bernilai ∞.

d. Nilai cost untuk simpul yang bersangkutan adalah:

$$\hat{c}(S) = \hat{c}(R) + A(i, j) + r$$

c(s) : cost simpul terkait,
c(R) : cost simpul sebelumnya,
A(i,j) : elemen ke (i,j) pada matriks, dan
r : total semua pengurang pada proses reduksi.

4. Simpul selanjutnya yang akan dibangkitkan bergantung pada nilai cost, yaitu nilai cost terkecil akan dibangkitkan duluan. Ulangi dari langkah nomor 3.
5. Jika sudah mencapai daun, bandingkan cost terbaik sekarang dengan semua cost simpul hidup. Jika ada simpul yang cost nya lebih besar, **simpul tersebut harus dibunuh.**
6. Jika masih ada simpul hidup yang costnya tidak lebih besar dengan cost terbaik sekarang, periksa kembali simpul tersebut sampai mencapai daun. Hal ini terus terulang hingga mendapatkan simpul dengan cost terbaik, yaitu ketika semua simpul lain sudah dibunuh.



| Simpul-E | Simpul Hidup |
|---|---|
| 1 | 4,5,2,3 |
| 4 | 6,5,2,8,7,3 |
| 6 | 10,5,2,8,7,9,3 |
| 10 | 11,5,2,8,7,9,3 |
| 11 | daun |

Semua simpul hidup yang nilainya lebih besar dari 28 dibunuh (B) karena tidak mungkin lagi menghasilkan perjalanan dengan bobot < 28.

Karena tidak ada lagi simpul hidup di dalam pohon ruang status, maka $X$ = (1, 4, 2, 5, 3, 1) menjadi solusi persoalan TSP di atas dengan bobot 28.

Gambar 5 Contoh pohon ruang status branch and bound
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf, diakses pada tanggal 2 Mei 2020 pukul 20.55 WIB)

## D. *Destinasi Wisata Terpopuler di Bali*

Menurut salah satu artikel pada sebuah situs yang membahas tentang objek wisata dan tujuan turis-turis internasional, planetware, terdapat 15 destinasi wisata terpopuler di Bali, yaitu:

1. Tanah Lot
2. Gunung Batur
3. Uluwatu
4. Ubud Monkey Forest
5. Ubud Art & Culture
6. Sawah Tegallalang & Jatiluwih
7. Pura Ulun Danu Bratan
8. Seminyak
9. Pantai Nusa Dua
10. Kepulauan Nusa
11. Kuta
12. Lembah Sidemen
13. Air Terjun Sekumpul
14. Candi Tirta Empul
15. Waterboom Bali

Namun, tentunya dalam satu kali perjalanan tidak mungkin mengunjungi semua tempat wisata. Oleh karena itu, pada makalah ini tempat wisata yang akan dijadikan data uji tidak sampai 15 buah.

# III. Langkah-langkah Penentuan Rute Tur

## A. *Persiapan dan Pengambilan Data*

Mengingat keterbatasan waktu untuk mengunjungi semua destinasi hanya dalam satu hari, destinasi yang akan digunakan pada makalah ini adalah:

1. Kuta (titik awal)
2. Denpasar
3. Uluwatu
4. Tanah Lot
5. Ubud
6. Gunung Batur

Kuta dipilih sebagai titik awal dari tur karena di daerah tersebut terdapat bandar udara internasional ngurah rai. Selain itu, di kawasan ini juga terdapat banyak penginapan.

Denpasar dipilih menjadi salah satu tujuan karena kota tersebut merupakan ikon dan ibukota dari Provinsi Bali.

Selanjutnya, kota-kota tersebut dicari titik lokasinya menggunakan scribblemaps.


Gambar 6 Letak titik-titik destinasi wisata di Bali
(Sumber: milik sendiri, dibuat dengan menggunakan aplikasi online scribblemaps (https://www.scribblemaps.com/) pada tanggal 2 Mei 2020 pukul 21.02 WIB)

Keterangan gambar:

0. Kuta (titik awal)
1. Denpasar
2. Uluwatu
3. Tanah Lot
4. Ubud

5. Gunung Batur

$$A = \begin{bmatrix} - & 8.5 & 23 & 22 & 34 & 72 \\ 8.6 & - & 29 & 20 & 24 & 60 \\ 21 & 31 & - & 43 & 53 & 93 \\ 22 & 20 & 42 & - & 32 & 69 \\ 36 & 23 & 53 & 33 & - & 37 \\ 77 & 65 & 92 & 71 & 36 & - \end{bmatrix}$$

Gambar 7 Matriks keterhubungan jarak dengan urutan titik: Kuta, Denpasar, Uluwatu, Tanah Lot, Ubud, dan Gunung Batur.
(Sumber: milik sendiri, dibuat dengan menggunakan aplikasi online pada tanggal 3 Mei 2020 pukul 21.02 WIB)

Dengan urutan titik dari atas ke bawah dan dari kiri ke kanan sebagai berikut: Kuta (0), Denpasar (1), Uluwatu (2), Tanah Lot (3), Ubud (4), dan Gunung Batur (5) Data jarak yang ada pada matriks dicari menggunakan aplikasi google map (jarak antara dua tempat wisata di Bali).

## *B. Eksperimen dalam Bahasa Python*

Berikut hasil uji matriks diatas dengan menggunakan program python:

```
C:\Users\Annisa Rahim\Desktop>bali.py
matriks awal:
inf  8.5  23   22   34   72
8.6  inf  29   20   24   60
21   31   inf  43   53   93
22   20   42   inf  32   69
36   23   53   33   inf  37
77   65   92   71   36   inf
last place: 0
visited: [0]
cost: 0
```

Gambar 8 Matriks keterhubungan jarak dalam uji coba implementasi program.
(Sumber: milik penulis)

Selanjutnya, matriks dimasukkan kedalam program python yang mengimplementasikan penyelesaian persoalan TSP. Untuk menjadi simpul pertama/akar, matriks tersebut akan diubah menjadi matriks tereduksi. Caranya adalah dengan mengurangi setiap baris dan kolom yang belum 0 dengan nilai minimal pada baris/kolom tersebut. Berikut implementasi program dalam bahasa Python:

```
def reduce(self):
    Mhasil=self.CopyMatrix()
    minval=0
    for i in range (6):
        if (Mhasil.no0inrow(i)):
            min=Mhasil.minrow(i)
            Mhasil.reducerow(i,min)
            minval+=min
    for i in range (6):
        if (Mhasil.no0incol(i)):
            min=Mhasil.mincol(i)
            Mhasil.reducecol(i,min)
            minval+=min
    Mhasil.cost+=minval
    return Mhasil
```

Gambar 9 Implementasi reduksi baris dan kolom pada matriks
(sumber: milik penulis)

Simpul akar pada data uji memiliki cost sebesar 155.6. berikut *screenshot* hasil pengujian:

```
simpul akar:
inf  0.0  0.0  3.5  25.5 49.5
0.0  inf  5.9  1.4  15.4 37.4
0    10   inf  12   32   58
2    0    7.5  inf  12   35
13   0    15.5 0    inf  0
41   29   41.5 25   0    inf
last place: 0
visited: [0]
cost: 155.6
```

Gambar 10 Simpul akar pada data uji
(Sumber: milik penulis)

Pertama, simpul akar digunakan untuk membangkitkan anak-anak simpul, yaitu destinasi tujuan selanjutnya setelah melalui simpul akar.

```
def BangkitAnak(mtrx,list):
    list.pop(0)
    for i in range (1,6):
        mat=mtrx.CopyMatrix()
        if i not in mtrx.visited:
            mat=mtrx.CopybutMove(mtrx.lastplace,i)
            insertPrio(mat,list)
```

Gambar 11 Algoritma pembangkitan anak dari suatu simpul
(sumber: milik penulis)

Simpul-simpul hidup disimpan dalam sebuah *priority queue* yang terurut menaik berdasarkan nilai cost. Simpul yang dibangkitkan adalah simpul yang belum pernah dikunjungi sebelumnya.

Selanjutnya, pencarian akan terus berlangsung hingga *priority queue* kosong, atau semua titik telah dikunjungi. Saat mencapai daun, pengecekan kembali dilakukan untuk menentukan apakah ada simpul hidup lain yang costnya lebih kecil daripada solusi sekarang.

```
# pengecekan simpul
while (len(queue)!=0) and (daun==False):
    if (len(queue[0].visited)==6):
        solution=queue[0]
        costmin=queue[0].cost
        daun=True
    else:
        costmin=queue[0].cost
        BangkitAnak(queue[0],queue)

# pengecekan ulang setelah mencapai goal
for i in queue:
    if i.cost>costmin:
        queue.remove(i)

while (len(queue)!=0):
    if (len(queue[0].visited)==6):
        if (costmin>queue[0].cost):
            solution=queue[0]
            costmin=queue[0].cost
        else:
            queue.pop(0)
    else:
        BangkitAnak(queue[0],queue)
```

Gambar 12 Algoritma Branch and Bound
(sumber: milik penulis)

Ketika sudah menemukan solusi, program akan mencetak hasil rute ke layar

```
---HASIL---
200.0
inf  inf  inf  inf  inf  inf
inf  inf  inf  inf  inf  inf
inf  inf  inf  inf  inf  inf
inf  inf  inf  inf  inf  inf
inf  inf  inf  inf  inf  inf
inf  inf  inf  inf  inf  inf
last place: 0
visited: [0, 3, 5, 4, 1, 2, 0]
cost: 200.0
```

Gambar 13 Hasil akhir pengujian
(sumber: milik penulis)

Menurut hasil uji, rute terbaik untuk melakukan tur adalah melalui titik 0, 3, 5, 4, 1, 2, dan kembali ke 0. Jika diubah menjadi nama destinasi wisata, rute perjalanannya adalah: Kuta, Tanah Lot, Gunung Batur, Ubud, Uluwatu, lalu kembali ke Kuta, dengan cost sebesar 200.

Jika melihat matriks jarak di awal, kita juga dapat menghitung total jarak dengan rute tersebut, yaitu 22+69+36+23+29+21 = 200 kilometer.

Berikut ilustrasi rute yang terbentuk sesuai dengan hasil akhir pengujian menggunakan program python:


Gambar 14 ilustrasi rute menurut hasil program
(Sumber: milik penulis, dibuat dengan menggunakan aplikasi online scribblemaps (https://www.scribblemaps.com/) pada tanggal 2 Mei 2020 pukul 21.02 WIB)

### *C. Pembahasan dan Perbandingan dengan Algoritma A**

Algoritma Branch and Bound yang digunakan pada makalah ini bekerja dengan mencari rute terdekat dengan membandingkan nilai-nilai jarak pada matriks, lalu mengambil jarak terdekat yang bisa diambil agar perjalanan dapat berlalu secara efisien.

a. Jumlah Pengurang
   Dengan mengurangi matriks sehingga menjadi matriks tereduksi, dapat dilihat bahwa nilai taksiran dari jarak terdekat adalah dengan melihat total jumlah pengurang, dengan harapan jumlah pengurang terkecil hingga matriks menjadi tereduksi adalah gambaran kecilnya angka-angka pada matriks. Semakin kecil angka pada matriks, semakin sedikit pula total jarak yang mungkin ditempuh dengan melihat matriks tersebut.
b. Cost sebelumnya
   nilai cost sebelumnya adalah total jarak dari titik awal hingga simpul sebelumnya. Maka, nilai itu tidak lain adalah pertimbangan jarak yang sudah diambil pada rute tersebut.
c. Nilai A(i,j)
   A(i,j) merepresentasikan jarak dari titik i ke titik j. Nilai ini, ditambah dengan nilai cost sebelumnya, merupakan total jarak yang akan ditempuh jika memilih simpul yang bersangkutan

Jika dibandingkan dengan algoritma A* yang memiliki fungsi evaluasi sebagai berikut:

f(n) = g(n) + h(n)

Dengan g(n) sebagai nilai cost dari awal sampai simpul, dan h(n) sebagai nilai heuristik, algoritma Branch and Bound pada makalah ini tidak lain adalah algoritma A* yang menggunakan cost sebagai f(n). Nilai cost sebelumnya ditambah dengan nilai A(i,j) adalah nilai g(n), dan total nilai pengurangan matriks berlaku sebagai fungsi heuristik.

## IV. KESIMPULAN

Algoritma Branch and Bound yang menggunakan *reduced-cost matrix* dalam menghitung cost setiap simpul untuk menyelesaikan persoalan *Travelling Salesman Problem* atau TSP dapat dimanfaatkan untuk mencari rute tur destinasi wisata yang efisien di Bali.

Meskipun pada makalah ini kasus tur yang diambil adalah di Pulau Bali, tidak menutup kemungkinan algoritma ini untuk dapat menyelesaikan persoalan serupa lainnya di tempat lain dengan menggunakan cara dan langkah-langkah yang sama.

Algoritma ini cukup optimal untuk menyelesaikan persoalan TSP, selama graf yang akan dicari solusinya memiliki jumlah tempat yang terbatas.

## VIDEO LINK AT YOUTUBE

Video berkaitan dapat dilihat pada link berikut: https://youtu.be/XRR5sz4ycS8


## UCAPAN TERIMA KASIH

Pertama-tama, saya mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena telah melancarkan urusan saya sehingga makalah ini dapat selesai pada waktunya. Selain itu, tidak lupa juga saya mengucapkan terima kasih kepada dosen mata kuliah Strategi Algoritma, Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc., atas dukungannya dan kegiatan pengajaran selama semester ini. Terakhir, saya mengucapkan terima kasih kepada seluruh pihak yang terlah menunjang selesainya makalah ini.


## REFERENCES

[1] Levitin, Anany. 2012. *Introduction to The Design and Analysis of Algorithms.* New Jersey: Pearson Education, Inc.

[2] Munir, Rinaldi. 2007. *Diktat Kuliah Strategi Algoritma.* Bandung: Prodi Informatika STEI ITB.

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-(2016).pdf

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/BFS-dan-DFS-(2020).pdf

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf,

[6] https://www.portonews.com/2017/keuangan-dan-portfolio/pariwisata/bali-permata-pariwisata-yang-terancam-kusam/diakses pada tanggal 2 Mei 2020 pukul 18.02 WIB

[7] https://www.planetware.com/tourist-attractions-/bali-ina-b-b.htm diakses pada tanggal 2 Mei 2020 pukul 20.21 WIB

## 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, 4 Mei 2020

Annisa Rahim 13518089