FINAL REPORT TUGAS BESAR
STRATEGI ALGORITMA

Agil Ditya Rafiazmi (1301223044)

Muhammad Farhan Editya (1301223077)

Rafianto Tri Aushaf (1301223274)

PROGRAM STUDI S1 INFORMATIKA
TELKOM UNIVERSITY
2023/2024

Abstraksi

Penerapan algoritma backtracking dan brute force dalam penerapan mencari rute terpendek antar kota dalam agenda “Summer Tour” atau tour antar kota, dimana algoritma backtracking dan brute force akan kami gunakan untuk menentukan rute terpendek dan akan memberikan perkiraan konsumsi bahan bakar yang digunakan.

Pendekatan algoritma brute force yang dimana akan mengunjungi semua jalur dan akan memberikan jalur terpendek sedangkan algoritma backtracking akan mengeksplorasi jalur dari kota awal ke kota tujuan,dan memberikan jalur terpendek, keduanya memiliki tujuan yang sama namun memiliki cara pencarian yang berbeda, dimana disini kami akan membandingkan algoritma mana yang memiliki efisiensi terbaik dalam menentukan jalur terpendek dalam summer tour.

Pendahuluan

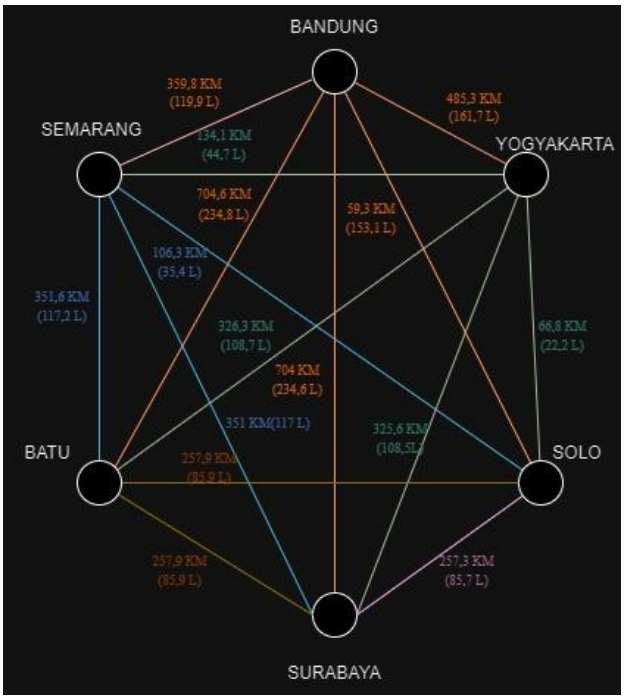
Masalah yang akan diselidiki adalah efisiensi dan efektivitas algoritma brute force dan backtracking dalam menyelesaikan masalah pencarian jalur (pathfinding) di mana sebuah agensi memiliki agenda “Summer Tour” atau tour antar kota. Sekaligus menghitung jarak minimum yang dapat diperoleh namun tetap memperhatikan bahan bakar yang dikonsumsi.

Untuk mengatasi masalah jarak minimum dan memperoleh penggunaan bahan bakar yang minimum, kami menggunakan algoritma brute force dan backtracking dimana algoritma ini dapat memberikan solusi yang paling optimal dan memberikan hasil yang terbaik dari semua rute yang ada.

Pekerjaan yang berhubungan

Ada 2 makalah dan jurnal yang kami temukan dan sebagai referensi dari tugas akhir mata kuliah Strategi Algoritma kami, yang pertama adalah penerapan algoritma backtracking untuk menyelesaikan TSP untuk menentukan rute wisata di daerah bandung, dan penerapan algoritma brute force dalam pencarian rute optimal untuk survei kost di bandung dengan Google Maps API. Dimana letak perbedaanya adalah kami menggunakan algoritma brute force dan backtracking untuk kasus menentukan rute tour terbaik untuk melakukan tour antar kota yang diberi nama “Summer Tour”. Yang nantinya 2 algoritma ini akan memberikan rute terbaik dan estimasi konsumsi bahan bakar yang akan digunakan selama “Summer Tour” ini dilaksanakan.

Data



Gambar Graf sempurna dari data yang kami buat.

Bandung - Yogyakarta = 485,3 KM (161,7 L)
Bandung - Semarang = 359,8 KM(119,9 L)
Bandung - Batu = 704,6 KM(234,8 L)
Bandung - Solo = 459,3 KM(153,1 L)
Bandung - Surabaya = 704 KM(234,6 L)
Yogyakarta - Semarang = 134,1 KM(44,7 L)
Yogyakarta - Batu = 326,3 KM(108,7 L)
Yogyakarta - Solo = 66,8 KM(22,2 L)
Yogyakarta - Surabaya = 325,6 KM(108,5L)
Semarang - Batu = 351,6 KM(117,2 L)
Semarang - Solo = 106,3 KM(35,4 L)
Semarang - Surabaya = 351 KM(117 L)
Batu - Solo = 257,9 KM(85,9 L)
Batu - Surabaya = 104.3 KM(34,7 L)
Solo - Surabaya = 257,3 KM(85,7 L)

Dengan data yang sudah ada algoritma brute force dan backtracking dapat memberikan hasil yang terbaik dan sebagai perbandingan untuk menguji algoritma mana yang lebih efisien dalam menyelesaikan masalah yang kami angkat. Data kami disini merupakan jarak antara kota dan perkiraan konsumsi bahan bakar untuk menempuh jarak antar kota.

Metode

Metode yang akan kami gunakan disini adalah algoritma brute force dan backtracking.

- Brute Force: Mengimplementasikan pendekatan brute force untuk menghasilkan semua jalur yang mungkin dari kota awal ke kota tujuan dan memilih jalur dengan jarak tempuh terpendek dengan perkiraan konsumsi bahan bakar yang akan digunakan

- Backtracking: Mengimplementasikan pendekatan backtracking untuk mengeksplorasi jalur dari kota awal ke kota tujuan, dan akan memberikan hasil jarak tempuh yang terpendek dan memberikan perkiraan konsumsi bahan bakar yang digunakan.

Hasil dan Analisis

Dalam eksperimen kali ini kami membuat 2 algoritma yaitu algoritma brute force dan algoritma backtracking, dimana 2 algoritma memiliki 1 tujuan yaitu untuk mencari rute terpendek untuk melakukan “Summer Tour” dimana pengguna dapat memasuki kota awal dan kota akhir (kota tujuan) dan 2 algoritma ini akan menampilkan kota terpendek untuk melakukan tour beserta total jarak dan total bahan bakar yang diperlukan.

Sebagai Contoh, kami mengambil kota awal yaitu Bandung dan kota akhir yaitu Surabaya. Dengan menggunakan Algoritma Brute Force dan Backtracking, diperoleh hasil sebagai berikut

1. Brute Force

Sebagai contoh, kami mengambil jarak dari kota Bandung ke Surabaya. Dengan mempertimbangkan semua kemungkinan , memperoleh hasil sebagai berikut

Semua kemungkinan rute yang dilalui:
Rute : ['Bandung', 'Yogyakarta', 'Semarang', 'Batu', 'Solo', 'Surabaya']
Jarak : 1486.2 km
Bensin : 495.2 liter
Rute : ['Bandung', 'Yogyakarta', 'Semarang', 'Solo', 'Batu', 'Surabaya']
Jarak : 1087.8999999999999 km
Bensin : 362.4 liter
Rute : ['Bandung', 'Yogyakarta', 'Batu', 'Semarang', 'Solo', 'Surabaya']
Jarak : 1526.8 km
Bensin : 508.69999999999993 liter
Rute : ['Bandung', 'Yogyakarta', 'Batu', 'Solo', 'Semarang', 'Surabaya']
Jarak : 1526.8 km
Bensin : 508.69999999999993 liter
Rute : ['Bandung', 'Yogyakarta', 'Solo', 'Semarang', 'Batu', 'Surabaya']
Jarak : 1114.3 km
Bensin : 371.2 liter
Rute : ['Bandung', 'Yogyakarta', 'Solo', 'Batu', 'Semarang', 'Surabaya']

Jarak : 1512.6 km
Bensin : 503.99999999999994 liter
Rute   : ['Bandung', 'Semarang', 'Yogyakarta', 'Batu', 'Solo', 'Surabaya']
Jarak : 1335.3999999999999 km
Bensin : 444.90000000000003 liter
Rute   : ['Bandung', 'Semarang', 'Yogyakarta', 'Solo', 'Batu', 'Surabaya']
Jarak : 922.8999999999999 km
Bensin : 307.40000000000003 liter
Rute   : ['Bandung', 'Semarang', 'Batu', 'Yogyakarta', 'Solo', 'Surabaya']
Jarak : 1361.8 km
Bensin : 453.7 liter
Rute   : ['Bandung', 'Semarang', 'Batu', 'Solo', 'Yogyakarta', 'Surabaya']
Jarak : 1361.7000000000003 km
Bensin : 453.7 liter
Rute   : ['Bandung', 'Semarang', 'Solo', 'Yogyakarta', 'Batu', 'Surabaya']
Jarak : 963.5 km
Bensin : 320.9 liter
Rute   : ['Bandung', 'Semarang', 'Solo', 'Batu', 'Yogyakarta', 'Surabaya']
Jarak : 1375.9 km
Bensin : 458.40000000000003 liter
Rute   : ['Bandung', 'Batu', 'Yogyakarta', 'Semarang', 'Solo', 'Surabaya']
Jarak : 1528.6 km
Bensin : 509.29999999999995 liter
Rute   : ['Bandung', 'Batu', 'Yogyakarta', 'Solo', 'Semarang', 'Surabaya']
Jarak : 1555.0 km
Bensin : 518.0999999999999 liter
Rute   : ['Bandung', 'Batu', 'Semarang', 'Yogyakarta', 'Solo', 'Surabaya']
Jarak : 1514.3999999999999 km
Bensin : 504.59999999999997 liter
Rute   : ['Bandung', 'Batu', 'Semarang', 'Solo', 'Yogyakarta', 'Surabaya']
Jarak : 1554.9 km
Bensin : 518.0999999999999 liter
Rute   : ['Bandung', 'Batu', 'Solo', 'Yogyakarta', 'Semarang', 'Surabaya']
Jarak : 1514.3999999999999 km
Bensin : 504.6 liter
Rute   : ['Bandung', 'Batu', 'Solo', 'Semarang', 'Yogyakarta', 'Surabaya']
Jarak : 1528.5 km
Bensin : 509.3 liter
Rute   : ['Bandung', 'Solo', 'Yogyakarta', 'Semarang', 'Batu', 'Surabaya']
Jarak : 1116.1000000000001 km
Bensin : 371.9 liter
Rute   : ['Bandung', 'Solo', 'Yogyakarta', 'Batu', 'Semarang', 'Surabaya']
Jarak : 1555.0 km
Bensin : 518.2 liter

Rute : ['Bandung', 'Solo', 'Semarang', 'Yogyakarta', 'Batu', 'Surabaya']
Jarak : 1130.3 km
Bensin : 376.59999999999997 liter
Rute : ['Bandung', 'Solo', 'Semarang', 'Batu', 'Yogyakarta', 'Surabaya']
Jarak : 1569.1 km
Bensin : 522.9 liter
Rute : ['Bandung', 'Solo', 'Batu', 'Yogyakarta', 'Semarang', 'Surabaya']
Jarak : 1528.6 km
Bensin : 509.4 liter
Rute : ['Bandung', 'Solo', 'Batu', 'Semarang', 'Yogyakarta', 'Surabaya']
Jarak : 1528.5 km
Bensin : 509.4 liter

Berikut adalah algoritma yang diterapkan

```python
def bruteforce(start_city, end_city, distances_and_fuel):
    cities = list(distances_and_fuel.keys())
    cities.remove(start_city)
    cities.remove(end_city)

    best_route = None
    best_distance = float('inf')
    best_fuel = float('inf')

    all_possible_routes = []

    for perm in permutations(cities):
        route = [start_city] + list(perm) + [end_city]
        total_distance, total_fuel = calculate_route_distance_and_fuel(route, distances_and_fuel)
        all_possible_routes.append((route, total_distance, total_fuel))
        if total_distance < best_distance or (total_distance == best_distance and total_fuel < best_fuel):
            best_route = route
            best_distance = total_distance
            best_fuel = total_fuel

    return all_possible_routes, best_route, best_distance, best_fuel
```

Gambar algoritma brute force.

```
Algoritma Brute Force

Kota Awal         : Bandung
Kota Tujuan       : Surabaya
Rute terbaik      : ['Bandung', 'Semarang', 'Yogyakarta', 'Solo', 'Batu', 'Surabaya']
Total Jarak       : 922.8999999999999 km
Total Bensin      : 307.40000000000003 liter
Execution time    : 2.384185791015625e-07 seconds
```

Gambar contoh output algoritma Brute Force

Dari hasil percobaan diatas ,menghasilkan rute terbaik sebagai berikut
Kota Awal : Bandung
Kota Tujuan : Surabaya
Rute terbaik : ['Bandung', 'Semarang', 'Yogyakarta', 'Solo', 'Batu', 'Surabaya']
Total Jarak : 922.8999999999999 km

Total Bensin : 307.40000000000003 liter
Execution time : 2.384185791015625e-07 seconds

Sehingga didapatkan Kompleksitas pada algoritma Brute Force diatas yaitu O(n!)

2. Backtracking

Setelah menggunakan algoritma Brute force, kami akan menggunakan algoritma Backtracking untuk mencari rute terbaik dari Bandung ke Surabaya, Berikut adalah algoritma yang kami gunakan

```python
def backtrack(current_city, end_city, visited, current_route):
    global best_route, best_cost, best_fuel

    if len(visited) == len(distances_and_fuel):
        if current_city == end_city:
            total_distance, total_fuel = calculate_route_distance_and_fuel(current_route, distances_and_fuel)
            if total_distance < best_cost:
                best_cost = total_distance
                best_fuel = total_fuel
                best_route = list(current_route)
        return

    for next_city in distances_and_fuel[current_city]:
        if next_city not in visited:
            visited.add(next_city)
            current_route.append(next_city)
            backtrack(next_city, end_city, visited, current_route)
            current_route.pop()
            visited.remove(next_city)
    return
```

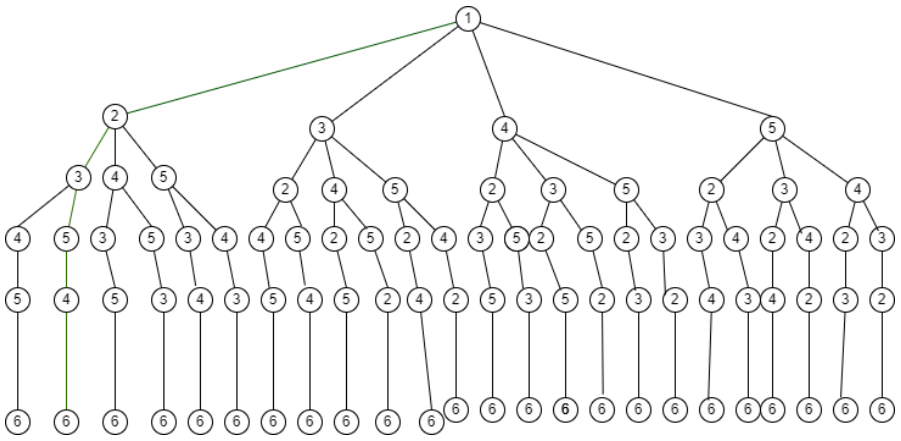
Gambar algoritma backtracking.

```
Algoritma Backtracking

Kota Awal         : Bandung
Kota Tujuan       : Surabaya
Rute Terbaik      : ['Bandung', 'Semarang', 'Yogyakarta', 'Solo', 'Batu', 'Surabaya']
Total jarak       : 922.8999999999999 KM
Total bahan bakar: 307.40000000000003 L
Execution time    : 1.1920928955078125e-06 seconds
```

Gambar contoh output algoritma backtracking

Algoritma backtracking di atas digunakan untuk menemukan rute terbaik (dalam hal jarak dan konsumsi bahan bakar) dari kota awal ke kota tujuan dengan mengeksplorasi semua kemungkinan rute yang mungkin.



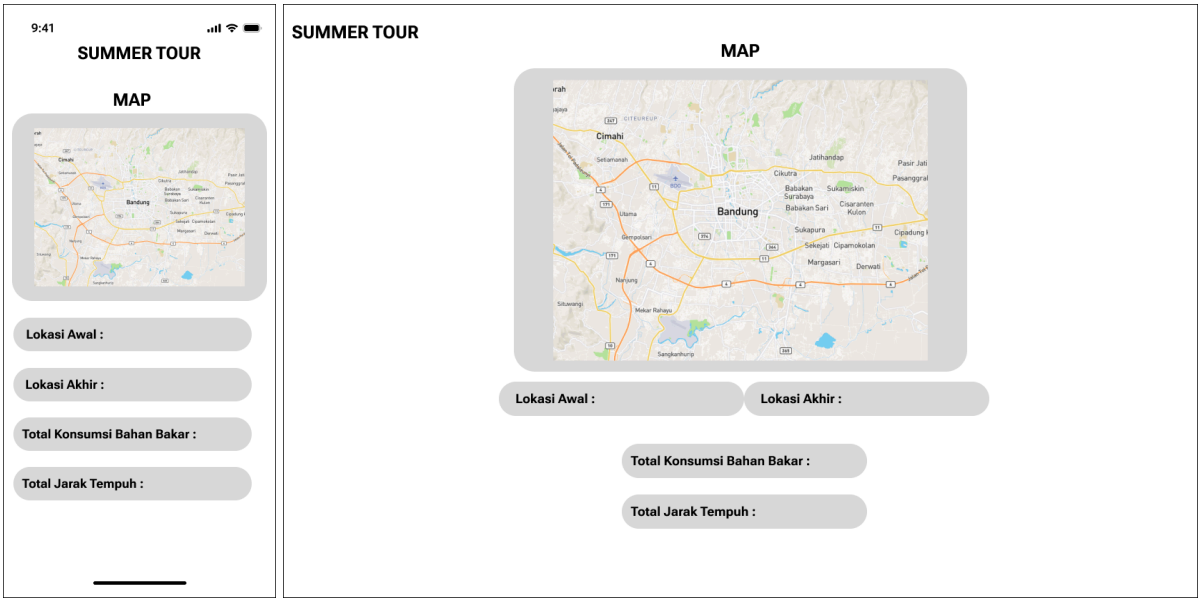
Gambar tree dari algoritma backtracking

Algoritma Backtracking

Kota Awal : Bandung
Kota Tujuan : Surabaya
Rute Terbaik : ['Bandung', 'Semarang', 'Yogyakarta', 'Solo', 'Batu', 'Surabaya']
Total jarak : 922.8999999999999 KM
Total bahan bakar: 307.40000000000003 L
Execution time : 1.1920928955078125e-06 seconds

Sehingga didapatkan Kompleksitas pada algoritma Brute Force diatas yaitu O(2^n)

3. Desain UI



Gambar Desain UI Website dan Mobile Phone.

Dalam desain UI kami, disini terdapat map yang menunjukan lokasi awal dari pengguna dan pengguna dapat memasukkan lokasi awal, dan pengguna dapat memasukan lokasi akhir kedalam kolom akhir yang dimana nanti map akan menunjukkan rute yang harus dilewati untuk melewati semua kota yang harus dilewati hingga menuju lokasi akhir. Pada

kolom total konsumsi bahan bakar setelah pengguna memasukan lokasi awal dan akhir, nantinya akan menunjukan estimasi konsumsi bahan bakar yang diperlukan, dan akan memberikan total jarak tempuh pada kolom jarak tempuh.

Kesimpulan

Kesimpulan yang kami dapatkan adalah kedua algoritma berhasil mendapatkan rute terpendek untuk melakukan tur “Summer Tour” yaitu ['Bandung', 'Semarang', 'Yogyakarta', 'Solo', 'Batu', 'Surabaya' dengan total jarak 922.8999999999999 KM dan Total bahan bakar 307.40000000000003 L. Dengan membandingkan kompleksitas algoritma, diperoleh bahwa Brute Force dengan kompleksitas O(n!) lebih besar daripada Backtracking dengan kompleksitas O(2^n). Sehingga waktu yang dibutuhkan algoritma Brute Force untuk mengeksekusi algoritma tersebut akan lebih lambat daripada algoritma Backtracking.

Referensi Bacaan

1. Pathfinding (martindevans.me)
2. https://www.routific.com/blog/travelling-salesman-problem
3. Wijaya, J., Frans, V., & Azmi, F. (2020). Aplikasi Traveling Salesman Problem Dengan GPS dan Metode Backtracking. Jurnal Ilmu Komputer dan Sistem Informasi (JIKOMSI), 3(2), 81-90.
4. Travelling Salesman Problem implementation using BackTracking - GeeksforGeeks
5. https://www.instagram.com/p/Cui83lkpEzQ/?igsh=MzV0ang3cHN1eGw0
6. https://otomotif.kompas.com/read/2022/09/04/124100415/konsumsi-solar-bus-akap-1-liter-solar-buat-3-km
7. Wiguna, Kristo Abdi (2021). Pengaplikasian Algoritma Backtracking untuk Menyelesaikan TSP untuk Menentukan Rute Wisata Kuliner di Daerah Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Makalah/Makalah-IF2211-Stima-2022-K1%20(21).pdf
8. Sianturi, Gerald Abraham(2022). Penerapan Algoritma Brute Force dalam Pencarian Rute Optimal untuk Survei Kos di Bandung dengan Google Maps API. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Makalah/Makalah-IF2211-Stima-2022-K3%20(45).pdf