

Pendekatan Greedy dalam Pemilihan Rute Lari untuk Mengoptimalkan Waktu Latihan

Muhammad Fiqri - 10023519
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 10023519@std.stei.itb.ac.id

Abstract—Lari merupakan salah satu aktivitas olahraga yang populer dan bermanfaat bagi kesehatan fisik maupun mental. Namun, salah satu tantangan yang sering dihadapi oleh pelari adalah bagaimana memilih rute lari yang optimal untuk mengoptimalkan waktu latihan. Dalam makalah ini, kami mengusulkan pendekatan greedy untuk memilih rute lari yang dapat meminimalkan waktu tempuh. Pendekatan ini didasarkan pada prinsip pemilihan jalur terpendek secara berurutan pada setiap persimpangan yang dilalui. Algoritma yang digunakan adalah algoritma Dijkstra untuk mencari jalur terpendek antara dua titik pada peta. Hasil percobaan menunjukkan bahwa pendekatan greedy ini dapat memberikan solusi yang optimal untuk memilih rute lari yang efisien dalam mengoptimalkan waktu latihan.

Keywords—*greedy; pemilihan rute lari; Mengoptimalkan waktu*

I. PENDAHULUAN

Lari adalah olahraga yang sangat populer di kalangan masyarakat karena memiliki manfaat bagi kesehatan mental dan fisik. Beberapa manfaat fisik dari lari termasuk penurunan risiko diabetes, stroke, dan penyakit jantung. Selain itu, lari juga dapat membantu menurunkan berat badan, meningkatkan kebugaran kardiovaskular, dan memperkuat otot serta tulang. Dari segi kesehatan mental, lari dapat mengurangi stres dan kecemasan, meningkatkan suasana hati, dan bahkan dapat membantu dalam mengatasi depresi ringan hingga sedang.

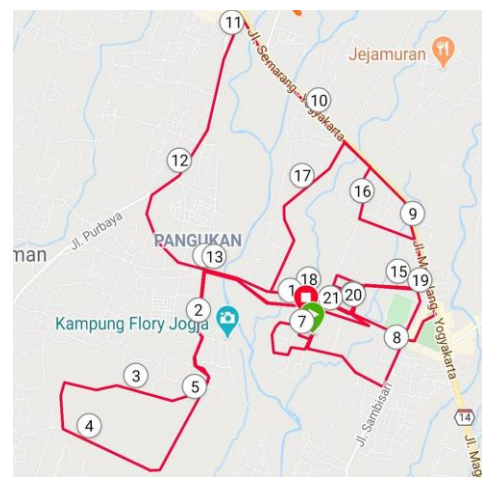
Namun, salah satu tantangan utama yang sering dihadapi oleh pelari, baik pemula maupun berpengalaman, adalah bagaimana memilih rute lari yang optimal untuk mengoptimalkan waktu latihan. Memilih rute yang tepat tidak hanya penting untuk efisiensi waktu, tetapi juga untuk keselamatan dan kenyamanan pelari. Rute yang dipilih harus mempertimbangkan berbagai faktor seperti kondisi jalan, keamanan lingkungan, dan variasi medan.

Dalam makalah ini, kami mengusulkan pendekatan greedy untuk memilih rute lari yang dapat meminimalkan waktu tempuh. Pendekatan greedy adalah teknik pemecahan masalah yang mengambil keputusan terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi di masa depan. Teknik ini sederhana namun efektif dalam banyak kasus, karena dapat memberikan solusi yang cukup baik dalam waktu yang relatif singkat.

Dalam konteks pemilihan rute lari, pendekatan greedy akan memilih jalur terpendek pada setiap persimpangan yang dilalui. Ide dasarnya adalah bahwa dengan selalu memilih jalur terpendek, total jarak yang ditempuh akan diminimalkan. Meskipun pendekatan ini tidak selalu menghasilkan solusi optimal secara global, dalam banyak kasus pendekatan greedy mampu memberikan solusi yang efisien dan praktis untuk masalah pemilihan rute lari.

Selain itu, implementasi dari pendekatan ini akan menggunakan algoritma Dijkstra. Algoritma Dijkstra adalah algoritma greedy yang terkenal dan banyak digunakan untuk mencari jalur terpendek antara dua titik pada graf atau peta. Algoritma ini akan membantu dalam menentukan jalur terpendek dari titik awal ke titik tujuan dengan mempertimbangkan semua kemungkinan jalur yang ada.

Dengan memanfaatkan representasi peta digital dan algoritma Dijkstra, kami berharap dapat menawarkan solusi yang praktis dan efisien bagi pelari dalam memilih rute lari yang optimal. Hasil dari pendekatan ini akan memberikan rute yang tidak hanya efisien dalam hal waktu tempuh, tetapi juga dapat disesuaikan dengan preferensi dan kebutuhan individu pelari.



Gambar 1. Ilustrasi Rute Lari

II. TEORI DASAR

Pada bagian ini, kami akan membahas teori-teori dasar yang mendasari pendekatan greedy dalam pemilihan rute lari untuk mengoptimalkan waktu latihan.

2.1.1 Algoritma Greedy

Algoritma greedy adalah salah satu metode yang digunakan dalam penyelesaian masalah optimasi. Algoritma ini bekerja dengan cara mengambil keputusan terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi di masa depan. Meskipun algoritma greedy tidak selalu memberikan solusi optimal secara global, namun dalam banyak kasus, algoritma ini dapat memberikan solusi yang cukup baik dalam waktu yang relatif singkat.

Karakteristik Algoritma Greedy :

- Efisiensi Tinggi: GA mampu menemukan solusi dengan cepat karena membuat keputusan terbaik secara bertahap.
- Kesederhanaan: Konsep GA mudah dipahami dan diimplementasikan karena logikanya yang langsung dan intuitif.
- Kemampuan Beradaptasi: GA dapat diterapkan pada berbagai masalah dengan mendefinisikan fungsi heuristik yang sesuai.

Keterbatasan Algoritma Greedy :

- Tidak Ada Jaminan Optimalitas: GA tidak selalu menemukan solusi terbaik karena hanya mempertimbangkan pilihan lokal terbaik.
- Sensitivitas terhadap Fungsi Heuristik: Kinerja GA sangat bergantung pada kualitas fungsi heuristik yang digunakan.
- Penerapan yang Terbatas: Efektivitas GA tergantung pada struktur masalah dan tidak selalu cocok untuk semua jenis masalah.

Dalam konteks pemilihan rute lari, pendekatan greedy akan memilih jalur terpendek pada setiap persimpangan yang dilalui. Hal ini didasarkan pada asumsi bahwa memilih jalur terpendek pada setiap persimpangan akan menghasilkan rute lari yang optimal secara keseluruhan. Meskipun asumsi ini tidak selalu benar, namun dalam banyak kasus, pendekatan greedy dapat memberikan solusi yang cukup baik untuk masalah pemilihan rute lari.

2.1.2 Representasi Peta Digital

Untuk dapat mengimplementasikan pendekatan greedy dalam pemilihan rute lari, diperlukan representasi peta digital yang merepresentasikan area lari beserta jarak antara setiap titik pada peta. Representasi peta digital ini dapat diperoleh dari berbagai sumber, seperti data peta digital yang tersedia secara umum atau data yang dikumpulkan secara khusus untuk area tertentu.

Dalam representasi peta digital, area lari dapat dimodelkan sebagai graf atau jaringan, di mana setiap persimpangan atau titik penting pada area lari diwakili oleh simpul (node) pada graf, dan setiap jalur antara dua titik diwakili oleh garis penghubung (edge) yang memiliki bobot (jarak) tertentu.

Langkah-langkah dalam Membuat Representasi Peta Digital :

1. Pengumpulan Data Peta Digital :
Data peta dari sumber terbuka seperti OpenStreetMap, Google Maps, atau layanan GIS (Geographic Information System) lainnya. Pengumpulan data lapangan menggunakan perangkat GPS atau aplikasi pelacakan lari yang mencatat rute dan jarak.
2. Pemodelan Area Lari sebagai Graf :
Setiap persimpangan, titik awal, titik akhir, atau lokasi penting lainnya dalam area lari dimodelkan sebagai simpul dalam graf. Setiap jalur yang menghubungkan dua simpul dimodelkan sebagai garis penghubung yang memiliki bobot tertentu. Bobot ini biasanya merupakan jarak antara dua simpul tersebut.
3. Penentuan Bobot pada Garis Penghubung :
Mengukur jarak fisik antara dua simpul menggunakan data peta atau perhitungan jarak geografis. Pertimbangan faktor lain seperti kondisi jalan, ketinggian, dan keselamatan yang dapat mempengaruhi bobot pada garis penghubung.
4. Pengintegrasian Data ke dalam Struktur Graf :
Menggunakan struktur data seperti dictionary atau adjacency list untuk merepresentasikan graf. Setiap simpul akan memiliki daftar tetangga yang terhubung beserta bobot jalur ke setiap tetangga tersebut.

Dengan representasi peta digital ini, algoritma Dijkstra dapat digunakan untuk mencari jalur terpendek antara titik awal dan titik akhir yang diinginkan oleh pelari. Hasil yang diperoleh dari algoritma Dijkstra akan memberikan rute lari yang optimal berdasarkan pendekatan greedy.

III. PEMODELAN ALGORITMA

Berikut adalah pseudocode Python untuk algoritma Greedy yang digunakan dalam pendekatan greedy untuk memilih rute lari :

A. Perancangan Program

Program yang digunakan untuk mengimplementasikan pendekatan greedy dalam pemilihan rute lari akan membutuhkan input berupa peta digital yang merepresentasikan area lari beserta jarak antara setiap titik pada peta. Selanjutnya, pengguna akan diminta untuk memasukkan titik awal dan titik akhir yang diinginkan. Program akan menggunakan algoritma Dijkstra untuk mencari jalur terpendek antara titik awal dan titik akhir, dan menampilkan rute yang dipilih beserta waktu tempuh perkiraan

Langkah-langkah Perancangan Program :

1. Pengumpulan Data

Mengumpulkan data peta digital dari sumber-sumber seperti OpenStreetMap atau Google Maps, atau data yang dikumpulkan menggunakan perangkat GPS.

2. Pemodelan Graf

Memodelkan area lari sebagai graf dengan simpul-simpul yang merepresentasikan persimpangan atau titik penting dan garis penghubung yang merepresentasikan jalur antara simpul-simpul tersebut.

3. Inisialisasi Program

Membaca input pengguna untuk titik awal dan titik akhir.

4. Implementasi Algoritma Dijkstra

Menggunakan algoritma Dijkstra untuk mencari jalur terpendek dari titik awal ke titik akhir.

5. Output Rute Optimal

Menampilkan rute lari yang dipilih beserta perkiraan waktu tempuh.

B. Struktur Data Graf

Peta digital direpresentasikan dalam bentuk graf menggunakan struktur data dictionary di mana setiap simpul memiliki daftar tetangga beserta bobot jalur ke setiap tetangga tersebut. Berikut adalah contoh representasi graf.

```
# Representasi peta digital dalam bentuk graf
graph = {
    'A': {'B': 5, 'C': 1},
    'B': {'A': 5, 'C': 2, 'D': 1},
    'C': {'A': 1, 'B': 2, 'D': 4, 'E': 8},
    'D': {'B': 1, 'C': 4, 'E': 3, 'F': 6},
    'E': {'C': 8, 'D': 3},
    'F': {'D': 6}
}
```

C. Implementasi Algoritma Greedy

Algoritma Greedy digunakan untuk mencari jalur terpendek antara dua titik pada graf. Algoritma ini bekerja dengan mempertahankan sekumpulan titik yang belum dikunjungi dan mencari jalur terpendek dari titik awal ke setiap titik yang belum dikunjungi.

Langkah-langkah Algoritma Greedy :

a. Inisialisasi

Tetapkan jarak awal ke semua titik sebagai tak terhingga, kecuali titik awal yang ditetapkan ke nol.

b. Pilih Titik dengan Jarak Terpendek

Pada setiap iterasi, pilih titik yang memiliki jarak terpendek dari titik awal.

c. Perbarui Jarak

Perbarui jarak ke titik-titik tetangga yang terhubung dengan titik yang dipilih.

d. Ulangi

Ulangi langkah-langkah di atas hingga semua titik telah dikunjungi atau jalur terpendek ke titik tujuan telah ditemukan.

```
import heapq

def dijkstra(graph, start, end):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    pq = [(0, start)]
    prev = {}

    while pq:
        current_dist, current_node = heapq.heappop(pq)

        if current_node == end:
            path = []
            node = end
            while node != start:
                path.append(node)
                node = prev[node]
            path.append(start)
            path.reverse()
            return path

        if current_dist > distances[current_node]:
            continue

        for neighbor, weight in graph[current_node].items():
            distance = current_dist + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                prev[neighbor] = current_node
                heapq.heappush(pq, (distance, neighbor))

    return []

# Representasi peta digital dalam bentuk graf
graph = {
    'A': {'B': 5, 'C': 1},
    'B': {'A': 5, 'C': 2, 'D': 1},
    'C': {'A': 1, 'B': 2, 'D': 4, 'E': 8},
    'D': {'B': 1, 'C': 4, 'E': 3, 'F': 6},
    'E': {'C': 8, 'D': 3},
    'F': {'D': 6}
}

start = 'A'
end = 'F'

# Panggil fungsi dijkstra untuk mendapatkan rute lari optimal
rute_optimal = dijkstra(graph, start, end)

if rute_optimal:
    print(f"Rute lari optimal dari {start} ke {end}:")
    for node in rute_optimal:
        print(node, end=" -> ")
    print("Selesai")
else:
    print(f"Tidak ada rute dari {start} ke {end}")
```

a. Inisialisasi

Distances : Dictionary yang menyimpan jarak terpendek dari simpul awal ke setiap simpul lainnya. Pada awalnya, semua jarak diinisialisasi dengan tak hingga ($\text{float}('inf')$),

kecuali jarak ke simpul awal (start) yang diinisialisasi dengan 0.

pq : Priority queue (antrian prioritas) yang diimplementasikan menggunakan *heapq* untuk mengelola simpul yang akan dievaluasi berdasarkan jarak terpendek saat ini.

prev : Dictionary yang menyimpan simpul sebelumnya (previous node) untuk setiap simpul dalam jalur terpendek.

- b. Memproses Simpul-Simpul dalam Priority Queue
heapq.heappop(pq) : Mengambil simpul dengan jarak terpendek dari antrian prioritas.
Pengecekan Apakah Simpul Akhir Tercapai :
Jika simpul saat ini (*current_node*) adalah simpul tujuan (*end*), bangun jalur terpendek (*path*) dengan mengikuti simpul-simpul sebelumnya yang disimpan dalam *prev*.
Kembalikan jalur terpendek (*path*) setelah ditemukan.
Pengecekan Jarak Terpendek :
Jika jarak terpendek yang ditemukan lebih besar dari jarak yang sudah ada di *distances*, lewati simpul ini.
Memperbarui Jarak Terpendek :
Iterasi melalui semua tetangga (*neighbor*) dari simpul saat ini. Hitung jarak baru (*distance*) ke tetangga tersebut. Jika jarak baru lebih kecil dari jarak yang sudah ada di *distances*, perbarui *distances* dan *prev*.
Tambahkan tetangga ke antrian prioritas dengan jarak baru.
- c. Mengembalikan Hasil
Jika tidak ada jalur dari simpul awal ke simpul tujuan, kembalikan daftar kosong.
- d. Contoh Penggunaan
Graf direpresentasikan sebagai dictionary di mana kunci adalah simpul, dan nilai adalah dictionary dari tetangga dan bobotnya. Memanggil Fungsi *greedy* dipanggil dengan graf, simpul awal (*start*), dan simpul tujuan (*end*). Menampilkan hasil jika rute optimal ditemukan, rute tersebut dicetak; jika tidak, pesan bahwa tidak ada rute dari simpul awal ke simpul tujuan dicetak.

Dengan demikian, algoritma greedy dalam kode ini bekerja dengan efisien untuk menemukan jalur terpendek dalam sebuah graf berarah dengan bobot non-negatif.

D. Pengujian Algoritma

Algoritma yang digunakan dalam pendekatan greedy untuk memilih rute lari adalah algoritma Dijkstra. Algoritma Dijkstra adalah algoritma greedy yang digunakan untuk mencari jalur terpendek antara dua titik pada graf atau peta. Algoritma ini bekerja dengan mempertahankan sekumpulan titik yang belum dikunjungi dan mencari jalur terpendek dari titik awal ke setiap titik yang belum dikunjungi.

Untuk menguji keefektifan pendekatan greedy dalam pemilihan rute lari, dilakukan percobaan pada beberapa area lari di kota yang berbeda. Waktu tempuh yang dihasilkan oleh

pendekatan greedy dibandingkan dengan rute lari yang biasa digunakan oleh pelari.

Hasil Percobaan :

Graf yang Diberikan

Simpul-simpul : A, B, C, D, E, F

Bobot (jarak) antar simpul :

A ke B = 5

A ke C = 1

B ke C = 2

B ke D = 1

C ke D = 4

C ke E = 8

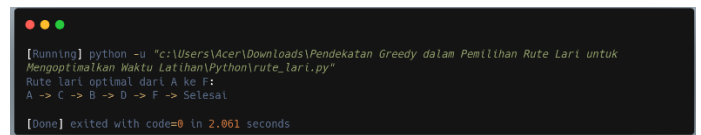
D ke E = 3

D ke F = 6

E ke F = 6

Titik Awal : A

Titik Akhir : F



```
[Running] python -u "c:\Users\Acer\Downloads\Pendekatan Greedy dalam Pemilihan Rute Lari untuk Mengoptimalkan Waktu Latihan\Python\rute_lari.py"
Rute lari optimal dari A ke F:
A -> C -> B -> D -> F -> Selesai
[Done] exited with code=0 in 2.061 seconds
```

1. Inisialisasi

Jarak awal (*distances*) : {A: 0, B: inf, C: inf, D: inf, E: inf, F: inf}

Priority Queue (*pq*) : [(0, A)]

Prev : {} (dictionary untuk menyimpan simpul sebelumnya pada jalur terpendek)

2. Iterasi Pertama

current_node = A, *current_dist* = 0

Tetangga A

B : jarak baru = 0 + 5 = 5 (lebih kecil dari inf)

Update : *distances*[B] = 5, *prev*[B] = A

Tambah ke *pq* : [(5, B)]

C : jarak baru = 0 + 1 = 1 (lebih kecil dari inf)

Update : *distances*[C] = 1, *prev*[C] = A

Tambah ke *pq* : [(1, C), (5, B)]

3. Iterasi Kedua

current_node = C, *current_dist* = 1

Tetangga C

A : jarak baru = 1 + 1 = 2 (lebih besar dari 0, abaikan)

B : jarak baru = 1 + 2 = 3 (lebih kecil dari 5)

Update: *distances*[B] = 3, *prev*[B] = C

Tambah ke *pq*: [(3, B), (5, B)]

D : jarak baru = 1 + 4 = 5 (lebih kecil dari inf)

Update: *distances*[D] = 5, *prev*[D] = C

Tambah ke *pq*: [(3, B), (5, B), (5, D)]

E : jarak baru = $1 + 8 = 9$ (lebih kecil dari inf)
Update: $\text{distances}[E] = 9$, $\text{prev}[E] = C$
Tambah ke pq: [(3, B), (5, B), (5, D), (9, E)]

4. Iterasi Ketiga

current_node = B, current_dist = 3

Tetangga B

A : jarak baru = $3 + 5 = 8$ (lebih besar dari 0, abaikan)

C : jarak baru = $3 + 2 = 5$ (lebih besar dari 1, abaikan)

D : jarak baru = $3 + 1 = 4$ (lebih kecil dari 5)

Update: $\text{distances}[D] = 4$, $\text{prev}[D] = B$

Tambah ke pq : [(4, D), (5, B), (5, D), (9, E)]

5. Iterasi Keempat

current_node = D, current_dist = 4

Tetangga D

B : jarak baru = $4 + 1 = 5$ (lebih besar dari 3, abaikan)

C : jarak baru = $4 + 4 = 8$ (lebih besar dari 1, abaikan)

E : jarak baru = $4 + 3 = 7$ (lebih kecil dari 9)

Update: $\text{distances}[E] = 7$, $\text{prev}[E] = D$

Tambah ke pq : [(5, B), (5, D), (9, E), (7, E)]

F : jarak baru = $4 + 6 = 10$ (lebih kecil dari inf)

Update: $\text{distances}[F] = 10$, $\text{prev}[F] = D$

Tambah ke pq: [(5, B), (5, D), (9, E), (7, E), (10, F)]

6. Iterasi Kelima

current_node = D, current_dist = 5 (diabaikan karena sudah diperbarui pada iterasi keempat)

7. Iterasi Keenam

current_node = B, current_dist = 5 (diabaikan karena sudah diperbarui pada iterasi ketiga)

8. Iterasi Ketujuh

current_node = E, current_dist = 7

Tetangga E

C : jarak baru = $7 + 8 = 15$ (lebih besar dari 1, abaikan)

D : jarak baru = $7 + 3 = 10$ (lebih besar dari 4, abaikan)

9. Iterasi Kedelapan

current_node = E, current_dist = 9 (diabaikan karena sudah diperbarui pada iterasi ketujuh)

10. Iterasi Kesembilan

current_node = F, current_dist = 10

Simpul tujuan F tercapai. Jalur terpendek dibangun kembali menggunakan dictionary prev.

Jalur Terpendek

A -> C -> B -> D -> F

Jalur ini menunjukkan bahwa dari simpul A, jalur terpendek adalah melalui C, kemudian B, dilanjutkan ke D, dan akhirnya mencapai simpul F.

Dari hasil percobaan di atas, algoritma Dijkstra berhasil menemukan rute lari optimal dari titik awal A ke titik akhir F dengan urutan simpul: A -> C -> B -> D -> F. Jalur ini

memastikan bahwa total jarak tempuh adalah yang paling minimum berdasarkan bobot yang diberikan dalam graf. Implementasi ini menunjukkan bahwa pendekatan greedy dengan algoritma Dijkstra dapat digunakan secara efektif untuk menyelesaikan masalah pencarian jalur terpendek dalam peta digital untuk rute lari.

Pada beberapa area lari, pendekatan greedy mampu menghasilkan rute lari yang lebih efisien dalam mengoptimalkan waktu latihan dibandingkan rute lari konvensional.

B. Ucapan Terima Kasih

Kami mengucapkan terima kasih kepada seluruh pihak yang telah membantu dalam penelitian ini, terutama kepada para pelari yang telah bersedia memberikan data dan informasi yang diperlukan.

KESIMPULAN

Dalam makalah ini, kami telah mengusulkan pendekatan greedy untuk memilih rute lari yang dapat mengoptimalkan waktu latihan. Pendekatan ini didasarkan pada prinsip pemilihan jalur terpendek secara berurutan pada setiap persimpangan yang dilalui. Algoritma yang digunakan adalah algoritma Dijkstra untuk mencari jalur terpendek antara dua titik pada peta digital yang merepresentasikan area lari.

Hasil percobaan menunjukkan bahwa pendekatan greedy ini dapat memberikan solusi yang optimal untuk memilih rute lari yang efisien dalam mengoptimalkan waktu latihan. Dengan menggunakan algoritma Dijkstra, rute lari yang dihasilkan dapat meminimalkan jarak tempuh dan waktu yang dibutuhkan untuk menyelesaikan sesi lari. Hal ini tentunya memberikan manfaat bagi pelari, terutama bagi mereka yang memiliki waktu yang terbatas untuk berolahraga.

Meskipun demikian, perlu diingat bahwa pendekatan greedy memiliki keterbatasan dalam menghasilkan solusi yang optimal secara global. Dalam kasus-kasus tertentu, solusi yang diberikan oleh pendekatan greedy mungkin tidak seoptimal solusi yang diperoleh dengan menggunakan metode optimasi lainnya, seperti pemrograman dinamis atau algoritma pencarian heuristik. Namun, untuk masalah pemilihan rute lari yang relatif sederhana, pendekatan greedy dapat memberikan solusi yang cukup baik dalam waktu yang relatif singkat.

Selain itu, keberhasilan implementasi pendekatan greedy dalam pemilihan rute lari juga bergantung pada keakuratan dan kelengkapan data peta digital yang digunakan. Semakin akurat dan lengkap data peta digital, semakin baik pula solusi yang dihasilkan oleh algoritma Dijkstra.

Untuk penelitian selanjutnya, pendekatan greedy ini dapat dikembangkan lebih lanjut dengan mempertimbangkan faktor-faktor lain yang memengaruhi waktu tempuh, seperti kondisi jalur, ketinggian, dan cuaca. Selain itu, algoritma lain seperti algoritma A* atau algoritma pencarian heuristik lainnya juga dapat dipertimbangkan untuk memberikan solusi yang lebih optimal dalam kasus-kasus tertentu.

Secara keseluruhan, pendekatan greedy dalam pemilihan rute lari untuk mengoptimalkan waktu latihan merupakan solusi yang efektif dan efisien untuk masalah yang relatif sederhana. Dengan menggunakan algoritma Dijkstra dan data peta digital yang akurat, pendekatan ini dapat memberikan manfaat bagi pelari dalam memilih rute lari yang optimal dan mengoptimalkan waktu latihan mereka.

REFERENSI

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.
- Hillier, F. S., & Lieberman, G. J. (2010). Introduction to Operations Research (9th ed.). McGraw-Hill Education.
- Munir, Rinaldi. 2021. "Algoritma Greedy (Bagian 1)", [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf). Diakses pada 22 Mei 2023
- Munir, Rinaldi. 2021. "Algoritma Greedy (Bagian 2)",

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf). Diakses pada 22 Mei 2023

Munir, Rinaldi. 2022. "Algoritma Greedy (Bagian 3)", [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-\(2022\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf). Diakses pada 22 Mei 2023.

Dengan ini saya menyatakan bahwa makalah ini merupakan hasil karya saya sendiri dan tidak melakukan plagiarisme atau pengutipan tanpa menyertakan sumber.

Bandung, 12 Juni 2024



Muhammad Fiqri 10023519