Analisis Lintasan Optimal pada Graf Keputusan Misi "*The Hostage*" untuk Memaksimalkan Peluang Keberhasilan Menggunakan Algoritma Dijkstra pada game *Detroit: Become Human*

Ramadhian Nabil Firdaus Gumay - 13524126
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: ramadhianx12@gmail.com , 13524126@std.stei.itb.ac.id

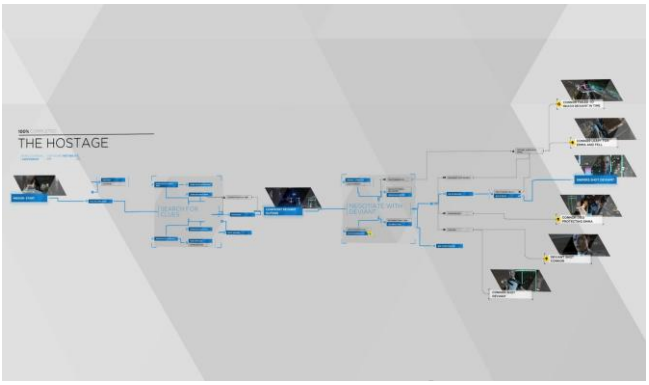
Abstrak*—Detroit: Become Human adalah permainan video yang berfokus pada narasi bercabang dan setiap pilihan pemain dapat mengubah alur cerita secara drastis. Misi pembuka, 'The Hostage,' menyajikan berbagai kemungkinan hasil berdasarkan serangkaian keputusan dalam waktu singkat. Makalah ini bertujuan untuk memodelkan pohon keputusan misi tersebut sebagai sebuah graf berarah dan berbobot. Dengan merepresentasikan pilihan sebagai edge dan keadaaan sepagai node, Algoritma Dijkstra diterapkkan untuk menemukan lintasan optimal. Hasil analisis akan menunjukkan satu lintasan spesifik yang secara matematis memberikan peluang tertinggi untuk mencapai akhir terbaik.

Kata Kunci*—Algoritma Djikstra, Detroit: Become Human, Graf Berbobot, Lintasan Optimal, Teori Graf, Pohon Keputusan.

I. PENDAHULUAN

Perkembangan industri permainan video telah melahirkan genre-genre baru yang lebih kompleks dan imersif. Salah satu genre yang menonjol adalah narasi interektif, yaitu ketika pemain tidak hanya menajdi konsumen cerita, tetapi juga menajdi agen aktif yang membentuk alur dan hasilnya. Detroit: Become Human, sebuah permainan yang dikembangkan oleh Quantic Dream dan dirilis pada tahun 2018, merupakan contoh paradigmatik dari genre ini. Permainan ini menyajikan sebuah dunia fiksi ilmiah ketika android dengan kecerdasan buatan setingkat dengan manusia hidup berdampingan dengan masyarakat, mengangkat tema-tema seperti kesadaran, kebebasan, dan kemanusiaan.

Pentingnya peran pemain dalam Detroit: Become Human divisualisasikan secara ekspilisit melalui fitur flowchart di akhir setiap babak. Flowchart ini memetakan semua kemungkinan jalur dan persimpangan keputusan yang telah atau dapat diambil oleh pemain, menunjukkan betapa luasya dampak pilihan mereka. Hal ini tidak hanya menjadikan setiap sesi permainan sebagi sebuah pengalaman unik, tetapi juga memunculkan sebuah pertanyaan mendasar dari perspektif analisis sistem, yaitu di antara banyak kemungkinan lintasan naratif, apakah ada sebuah lintasan yang dianggap "optimal".



Gambar 1. Flowchart misi "The Hostage"
Sumber: https://www.shacknews.com/article/105056/the-hostage---detroit-become-human, diakses pada 19/06/2025

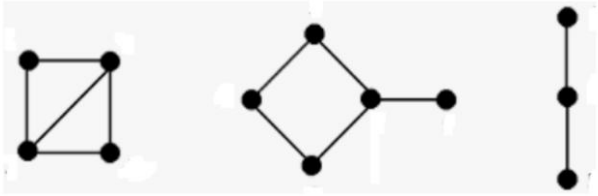
Untuk menjawab pertanyaan tersebut, makalah ini akan berfokus pada misi pembuka permainan, "*The Hostage*," sebagai sebuah studi kasus yang representatif. Misi ini dipilih karena merupakan misi yang sangat krusial, misi ini adalah perkenalan pertama pemain kepada mekanisme pengambilan keputusan dan konsekuensinya yang berisiko tinggi. Keragaman hasil dan banyaknya variabel keputusan juga menjadikan misi "*The Hostage*" sebagai model yang ideal untuk dianalisis sebagai sebuah sebuah graf keputusan.

II. LANDASAN TEORI

A. Teori Graf

A.1. Definisi Graf

Teori graf adalah cabang dari matematika diskrit yang menyediakan kerangka kerja abstrak dan kuat untuk memodelkan relasi dan jaringan. Sebuah graf memvisualisasikan sekumpulan objek yang disebut simpul dan koneksi di antara mereka yang disebut sisi.



Gambar 2. Graf
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf. diakses pada 19/06/2025

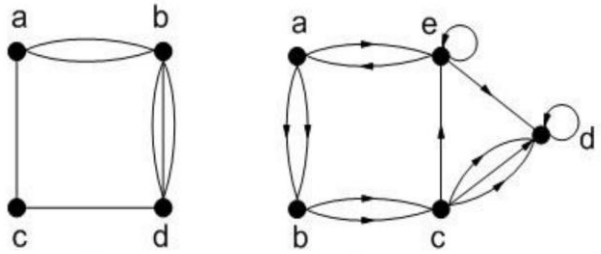
Secara formal, sebuah graf dinotasikan sebagai pasangan terurut $G = (V, E)$, yang teridiri dari dua himpunan fundamental. V adalah himpunan tak kosong dan terbatas dari elemen-elemen yang disebut simpul. Simpul merepresentasikan objek dalam sistem yang dimodelkan. Contohnya, dalam jaringan jalan, simpul dapat melambangkan persimpangan atau kota. E adalah himpunan sisi yang merupakan pasangan simpul dari V. Setiap sisi $e = (u, v)$ yang $u, v \in V$ menandakan adanya hubungan atau koneksi langsung antara simpul u dan v.

A.2. Klasifikasi Graf

Graf diklasifikasikan berdasarkan properti dan sisi-sisinya yang menentukan kesesuainnya untuk memodelkan berbagai jenis masalah. Untuk makalah ini, dua klasisfikas yang menjadi paling utama adalah graf berarah dan graf berbobot.

Dalam graf berarah, setiap sisi memiliki orientasi atau arah spesifik, direpresentasikan sebagai pasangan simpul terurut (u, v). Hal ini mengimplikasikan bahwa koneksi dardi u ke v berbedan dengan koneksi dari v ke u. Struktur ini sangat penting untuk memodelkan sistem yang prosesnya bersifat sekuensial dan berpotensi tidak dapat dibalik.

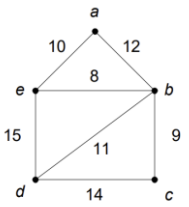
Dalam graf berarah, konsep derajat masuk (*in-degree*) dan derajat keluar (*out-degree*) menjadi relevan, yang masing-masing merepresentasikan jumlah sisi yang tiba di dan berangkat dari sebuah simpul.



a) *b)*

Gambar 3. Gambar Graf Tak-Berarah (a) dan Graf Berarah (b)
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, diakses pada 19/06/2025

Graf berbobot memberikan sebuah nilai numerik, atau bobot pada setiap sisinya. Bobot ini adalah bilangan riil yang memberikan kuantitas atribut fisik dari sebuah koneksi, seperti jarak, waktu, kapasitas, atau seperti dalam makalah ini, sebuah biaya yang diturunkan dari probabilitas. Pemberian bobot meguban peta tropologis sederhana menjadi model kuantitatif yang cocok untuk masalah optimisasi. Tujuannya sering kali adalah untuk menemukan lintasan yang meminimalkan atau memaksimalkan jumlah dari bobot-bobot ini.



Gambar 4. Graf Berbobot
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf, diakses pada 19/06/2025

A.3. Lintasan

Sebuah lintasan secara formal didefinisikan sebagai barisan $(v_0, v_1, \ldots, v_k)$ yang setiap pasangan simpul berdekatan (v_{i-1}, v_i) terhubung oleh sebuah sisi di E. Agar sebuah barisan disebut lintasan, semua simpul harus berbeda, Hal ini membedakannya dari *walk,* yang memperbolehkan pengulangan simpul dan sisi, serta *trail*, yang memperbolehkan pengulangan simpul tetapi tidak pada sisi. Perbedaan ini sangat penting untuk mencegah ambiguitas dalam konteks algoritmik karena algoritma seperti Dijkstra secara implisit mencari lintasan sederhana (lintasan tanpa siklus).

Panjang dari sebuah lintasan adalah jumlah sisi yang dikanduugnnya. Dalam graf berbobot, bobot atau biaya dari sebuah lintasan adalah jumlah dari bobot sisi-sisi penyusunnya.

B. Algortima Dijkstra

Algoritma Dijkstra, yang pertama kali diperkenalkan oleh Edsger W. Dijkstra dalam makalahnya yang berpengaruh pada tahun 1959, "*A Note on Two Problems in Connexion with Graphs*", merupakan algoritma fundamental untuk menyelesaikan masalah lintasan terpendek dari sumber tunggal dalam sebuah graf berbobot di mana semua bobot sisinya non-negatif.

Mekanisme kerja Algortima Dijkstra dimulai dengan inisialisasi, yaitu ketika jarak ke simpul awal diatur menjadi 0 dan ke semua simpul lain menjadi tak terhingga. Secara iteratif, algoritma ini akan memilih simpul "belum dikunjungi" yang memiliki jarak terpendek, lalu memeriksa setiap tetangganya. Jika sebuah rute yang lebih pendek ke tetangga tersebut ditemukan melalui simpul saat ini (sebuah proses yang disebut relaksasi), maka jaraknya akan diperbarui. Simpul saat ini kemudian ditandai sebagai "sudah dikunjungi", dan proses ini berlanjut hingga semua simpul yang dapat dijangkau telah dikunjungi.

Algorima Dijkstra secara alami merupakan algoritma minimasi, tetapi dapat diadaptasi untuk masalah maksimasi, seperti mencari jalur dengan probabilitas keberhasilan tinggi. Karena probabilitas total dihitung melalui perkalian, sedangkan Algoritma Dijkstra bekerja dengan penjumlahan, diperlukan

transformasi matematis. Bobot setiap sisi diubah menggunakakn probabilitas keberhasilannya ($P_{success}$) dengan rumus $w = -\log(P_{success})$. Dengan mengubah bobot menjadi logaritma negatif, masalah untuk memaksimalkan produk probabilitas diubah menjadi masalah minimasi jumlah bobot yang dapat diselesaikan oleh Algoritma Dijkstra, Transfornasi ini valid karena nilai $-\log(P_{success})$ selalu non-negatif untuk probabilitas $0 < P_{success} \leq 1$.

III. METODOLOGI

A. Persiapan Data

Tahap pertama dalam adalah persiapan data, yaitu proses penguraian alur cerita misi "*The Hostage*" menjadi komponen-komponen diskrit yang dapat dianalisis secara kuantitaif, seperti simpul, sisi, dan bobot.

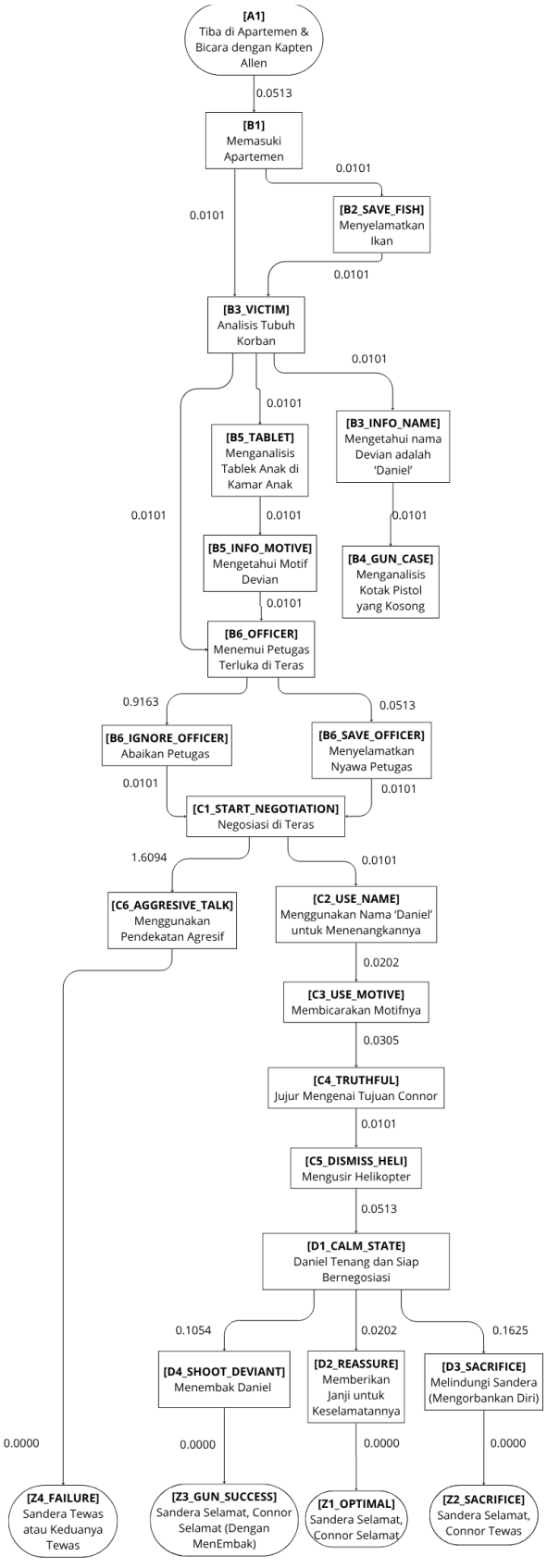
Penentuan simpul dilakukan dengan mengidentifikasi setiap keadaan atau titik keputusan krusial dalam misi. Setiap momen penting, mulai dari tiba di apartemen dan berbicara dengan Kapten Allen hingga berbagai kemungkinan hasil akhir, ditetapkan sebagai sebuah simpul di graf.

Selanjutnya, sisi merepresentasikan transisi antara dua simpul yang dipicu oleh sebuah aksi atau pilihan dialog yang diambil oleh pemain. Karena pilihan dalam permainan bersifat sekuensial dan tidak dapat dibalik, maka sisi yang digunakan bersifat terarah membentuk graf berarah.

Terakhir, setiap sisi diberi nilai numerik berupa bobot yang merepresentasikan biaya yang diturunkan dari probabilitas keberhasilan. Bobot ini dihitung menggunakan rumus $w = -\log(P_{success})$, yang mengubah tujuan untuk memaksmlakan keberhasilan menjadi masalah minimasi biaya total, agar bisa dipecahkan dengan Algoritma Dijkstra.

B. Pemodelan Graf

Setelah data simpul, sisi, dan bobot didefinisikan, langkah selanjutnya adalah memodelkan data tersebut ke dalam sebuah struktur graf berbobot. Model graf ini merepresentasikan seluruh alur keputusan yang mungkin dalam misi The Hostage.



Gambar 5. Graf Berbobot Misi "*The Hostage*"

Graf yang dibentuk adalah sebuah graf berarah berbobot yang dimulai dari simpul tuggal dan bercabang secara ekstensif sesuai dengan pilihan-pilihan pemain. Beberapa jalur investigasi awal, seperti memeriksa korban atau menganalisis tablet, dapat dianggap cabang paralel yang pada akhirnya harus menyatu sebelum fase negosiasi. Dari titik negosiasi, graf kembali bercabang berdasarkan pilihan dialog dan aksi, yang setiap cabang memiliki bobot yang berbeda. Lintasan-lintasan ini akan berujung pada beberapa simpul terminal yang merepresentasikan berbagai hasil akhir misi.

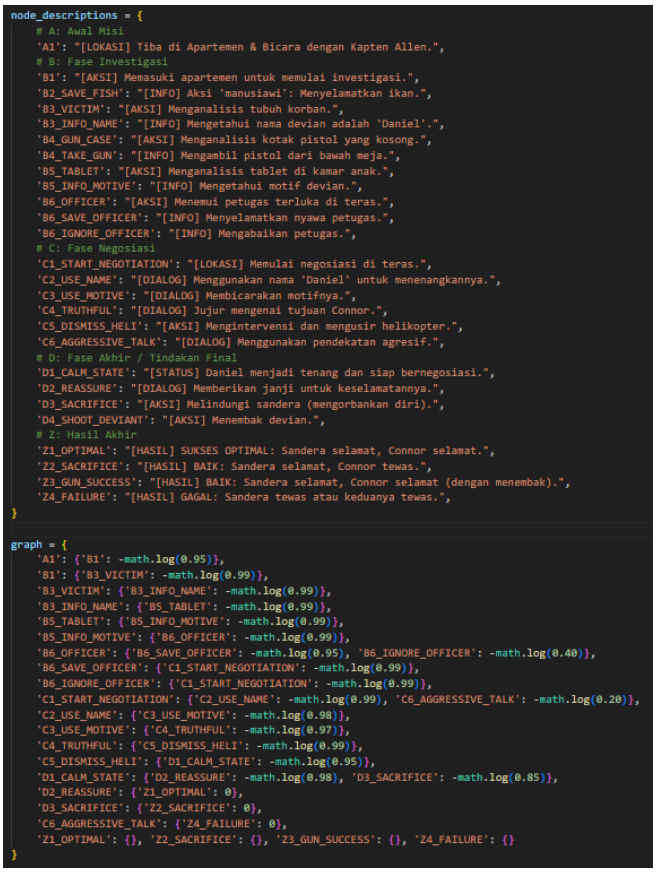
Tabel 1. Data sisi dan bobot pada graf misi The Hostage

Sisi	Bobot
[A1] - [B1]	0.0513
[B1] - [B2_SAVE_FISH]	0.010
[B2_SAVE_FISH] - [B3_VICTIM]	0.010
[B1] - [B3_VICTIM]	0.0101
[B3_VICTIM] - [B3_INFO_NAME]	0.0101
[B3_INFO_NAME] - [B4_GUN_CASE]	0.0101
[B1] - [B5_TABLET]	0.0101
[B5_TABLET] - [B5_INFO_MOTIVE]	0.0101
[B5_INFO_MOTIVE] - [B6_OFFICER]	0.0101
[B1] - [B6_OFFICER]	0.0101
[B6_OFFICER] - [B6_SAVE_OFFICER]	0.0513
[B6_SAVE_OFFICER] - [C1_START_NEGOTIATION]	0.0101
[C1_START_NEGOTIATION] - [C2_USE_NAME]	0.0101
[C2_USE_NAME] - [C3_USE_MOTIVE]	0.0202
[C3_USE_MOTIVE] - [C4_TRUTHFUL]	0.0305
[C4_TRUTHFUL] - [C5_DISMISS_HELI]	0.0101
[C5_DISMISS_HELI] - [D1_CALM_STATE]	0.0513
[D1_CALM_STATE] - [D2_REASSURE]	0.0202
[D2_REASSURE] - [Z1_OPTIMAL]	0.0000
[D1_CALM_STATE] - [D3_SACRIFICE]	0.1625
[D3_SACRIFICE] - [Z2_SACRIFICE]	0.0000
[C4_TRUTHFUL] - [D4_SHOOT_DEVIANT]	0.1054
[D4_SHOOT_DEVIANT] - [Z3_GUN_SUCCESS]	0.0000
[C1_START_NEGOTIATION] - [C6_AGGRESSIVE_TALK]	1.6094
[C6_AGGRESSIVE_TALK] - [Z4_FAILURE]	0.0000
[B6_OFFICER] - [B6_IGNORE_OFFICER]	0.9163
[B6_IGNORE_OFFICER] - [C1_START_NEGOTIATION]	0.0101

C. Implementasi Program

Program untuk analisis ini diimplementasikan menggunakan bahasa Python dengan memanfaatkan beberapa *library* standar untuk efisiensi. *Library* `math` digunakan untuk melakukan operasi logaritma dengan perhitungan bobot, sementara `heapq` digunakan untuk mengimplementasikan struktru data priority queue yang krusial untuk performa algortima Dijkstra.

Pada tahap pertama, graf yang telah dirancang diterjemahkan ke dalam struktur data Python. Dua buah *dictionary* utama dibuat: `node_description` yang berfungsi sebagai kamus yang memetakan ID simpul yang singkat ke deskripsi naratifnya yang lengkap dan `graph` yang merupakan implementasi dari adjacency list. Struktur `graph` ini menyimpan semua koneksi antar simpul beserta bobotnya, yang setiap bobot telah dihitung sebelumnya menggunakan $w = -\log(P_{sukses})$ untuk merepresentasikan "biaya" dari setiap pilihan.



```
node_descriptions = {
    # A: Awal Misi
    'A1': "[LOKASI] Tiba di Apartemen & Bicara dengan Kapten Allen.",
    # B: Fase Investigasi
    'B1': "[AKSI] Memasuki apartemen untuk memulai investigasi.",
    'B2_SAVE_FISH': "[INFO] Aksi 'manusiawi': Menyelamatkan ikan.",
    'B3_VICTIM': "[AKSI] Menganalisis tubuh korban.",
    'B3_INFO_NAME': "[INFO] Mengetahui nama devian adalah 'Daniel'.",
    'B4_GUN_CASE': "[AKSI] Menganalisis kotak pistol yang kosong.",
    'B4_TAKE_GUN': "[INFO] Mengambil pistol dari bawah meja.",
    'B5_TABLET': "[AKSI] Menganalisis tablet di kamar anak.",
    'B5_INFO_MOTIVE': "[INFO] Mengetahui motif devian.",
    'B6_OFFICER': "[AKSI] Menemui petugas terluka di teras.",
    'B6_SAVE_OFFICER': "[INFO] Menyelamatkan nyawa petugas.",
    'B6_IGNORE_OFFICER': "[INFO] Mengabaikan petugas.",
    # C: Fase Negosiasi
    'C1_START_NEGOTIATION': "[LOKASI] Memulai negosiasi di teras.",
    'C2_USE_NAME': "[DIALOG] Menggunakan nama 'Daniel' untuk menenangkannya.",
    'C3_USE_MOTIVE': "[DIALOG] Membicarakan motifnya.",
    'C4_TRUTHFUL': "[DIALOG] Jujur mengenai tujuan Connor.",
    'C5_DISMISS_HELI': "[AKSI] Mengintervensi dan mengusir helikopter.",
    'C6_AGGRESSIVE_TALK': "[DIALOG] Menggunakan pendekatan agresif.",
    # D: Fase Akhir / Tindakan Final
    'D1_CALM_STATE': "[STATUS] Daniel menjadi tenang dan siap bernegosiasi.",
    'D2_REASSURE': "[DIALOG] Memberikan janji untuk keselamatannya.",
    'D3_SACRIFICE': "[AKSI] Melindungi sandera (mengorbankan diri).",
    'D4_SHOOT_DEVIANT': "[AKSI] Menembak devian.",
    # Z: Hasil Akhir
    'Z1_OPTIMAL': "[HASIL] SUKSES OPTIMAL: Sandera selamat, Connor selamat.",
    'Z2_SACRIFICE': "[HASIL] BAIK: Sandera selamat, Connor tewas.",
    'Z3_GUN_SUCCESS': "[HASIL] BAIK: Sandera selamat, Connor selamat (dengan menembak).",
    'Z4_FAILURE': "[HASIL] GAGAL: Sandera tewas atau keduanya tewas.",
}

graph = {
    'A1': {'B1': -math.log(0.95)},
    'B1': {'B3_VICTIM': -math.log(0.99)},
    'B3_VICTIM': {'B3_INFO_NAME': -math.log(0.99)},
    'B3_INFO_NAME': {'B5_TABLET': -math.log(0.99)},
    'B5_TABLET': {'B5_INFO_MOTIVE': -math.log(0.99)},
    'B5_INFO_MOTIVE': {'B6_OFFICER': -math.log(0.99)},
    'B6_OFFICER': {'B6_SAVE_OFFICER': -math.log(0.95), 'B6_IGNORE_OFFICER': -math.log(0.40)},
    'B6_SAVE_OFFICER': {'C1_START_NEGOTIATION': -math.log(0.99)},
    'B6_IGNORE_OFFICER': {'C1_START_NEGOTIATION': -math.log(0.99)},
    'C1_START_NEGOTIATION': {'C2_USE_NAME': -math.log(0.99), 'C6_AGGRESSIVE_TALK': -math.log(0.20)},
    'C2_USE_NAME': {'C3_USE_MOTIVE': -math.log(0.98)},
    'C3_USE_MOTIVE': {'C4_TRUTHFUL': -math.log(0.97)},
    'C4_TRUTHFUL': {'C5_DISMISS_HELI': -math.log(0.99)},
    'C5_DISMISS_HELI': {'D1_CALM_STATE': -math.log(0.95)},
    'D1_CALM_STATE': {'D2_REASSURE': -math.log(0.98), 'D3_SACRIFICE': -math.log(0.85)},
    'D2_REASSURE': {'Z1_OPTIMAL': 0},
    'D3_SACRIFICE': {'Z2_SACRIFICE': 0},
    'C6_AGGRESSIVE_TALK': {'Z4_FAILURE': 0},
    'Z1_OPTIMAL': {}, 'Z2_SACRIFICE': {}, 'Z3_GUN_SUCCESS': {}, 'Z4_FAILURE': {}
}
```

Gambar 6. Pemodelan Data

Tahap kedua adalah implementasi inti dari Algoritma Dijkstra dalam sebuah fungsi bernama `dijkstra`. Fungsi ini secara sistematis mencari jalur dengan total bobot terendah dari simpul awal ke simpul tujuan.

```
def dijkstra(graph, start_node, end_node):
    distances = {node: float('inf') for node in graph}
    previous_nodes = {node: None for node in graph}
    distances[start_node] = 0
    priority_queue = [(0, start_node)]

    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)

        if current_distance > distances[current_node]:
            continue

        if current_node == end_node:
            break

        for neighbor, weight in graph.get(current_node, {}).items():
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous_nodes[neighbor] = current_node
                heapq.heappush(priority_queue, (distance, neighbor))

    path = []
    node = end_node
    if distances[node] == float('inf'):
        return None, float('inf')

    while node is not None:
        path.append(node)
        node = previous_nodes.get(node)
    path.reverse()

    if path and path[0] == start_node:
        return path, distances[end_node]
    else:
        return None, float('inf')
```

Gambar 7. Implementasi Algoritma Dijkstra

Tahap terakhir adalah blok eksekusi utama program. Di sini, simpul awal dan simpul tujuan untuk analisis didefiniskan secara eksplisit. Fungsi `diijkstra` kemudian dipanggil untuk melakukan perhitungan. Jika sebuah lintasan valid berhasil ditemukan, program akan memproses hasilnya. Hasil tersebut yang berupa daftar ID simpul, akan diterjemahkan kembali menjadi deskripsi naratif yang mudah dibaca. Selain menyajikan urutan langkah yang direkomendasikan, program juga akan menampilkan total biaya dari lintasan tersebut dan mengonversinya kembali menjadi estimasi probabilitas keberhasilan kumulatif agar hasilnya lebih intuitif.

```
if __name__ == "__main__":
    START_NODE = 'A1'
    OPTIMAL_END_NODE = 'Z1_OPTIMAL'

    print("~" * 60)
    print("Analisis Lintasan Optimal Misi 'The Hostage'")
    print("Menggunakan Algoritma Dijkstra")
    print("~" * 60)

    optimal_path, final_cost = dijkstra(graph, START_NODE, OPTIMAL_END_NODE)

    if optimal_path:
        print("\n[HASIL] LINTASAN OPTIMAL DITEMUKAN:")
        print("-" * 35)
        for i, node_id in enumerate(optimal_path):
            print(f"Langkah {i+1}: {node_descriptions.get(node_id, 'Aksi tidak diketahui')}")

        print("\n[ANALISIS BIAYA DAN PROBABILITAS]")
        print("-" * 35)
        print(f"Total Biaya (Cost) Lintasan: {final_cost:.4f}")

        if final_cost > 0:
            cumulative_probability = math.exp(-final_cost)
            print(f"Estimasi Probabilitas Keberhasilan Kumulatif: {cumulative_probability:.2%}")
        else:
            print("Estimasi Probabilitas Keberhasilan Kumulatif: 100.00%")

    else:
        print(f"\nTidak dapat menemukan lintasan dari '{node_descriptions[START_NODE]}' ke '{node_descriptions[OPTIMAL_END_NODE]}'.")

    print("~" * 60)
```

Gambar 8. Eksekusi dan Interpretasi Hasil

IV. HASIL DAN PEMBAHASAN

A. Lintasan Optimal Hasil Eksekusi Program

Setelah program dijalankan dengan simpul awal [A1] dan simpul tujuan [Z1_OPTIMAL], algoritma berhasil mengidentifikasi sebuah lintasan dengan total biaya terendah. Lintasan in merepresentasikan urutan tindakan yang paling efisien untuk memaksimalkan peluang keberhasilan misi. Hasil eksekusi program menunjukkan lintasan sebagai berikut:

Tabel 2. Lintasan Optimal Hasil Eksekusi Program

Langkah	ID Simpul	Biaya
1	A1	0.0000
2	B1	0.0513
3	B3_VICTIM	0.0614
4	B3_INFO_NAME	0.0715
5	B5_TABLET	0.0816
6	B5_INFO_MOTIVE	0.0917
7	B6_OFFICER	0.1018
8	B6_SAVE_OFFICER	0.1531
9	C1_START_NEGOTIATION	0.1632
10	C2_USE_NAME	0.1733
11	C3_USE_MOTIVE	0.1935
12	C4_TRUTHFUL	0.2240
13	C5_DISMISS_HELI	0.2341
14	D1_CALM_STATE	0.2854
15	D2_REASSURE	0.3056
16	Z1_OPTIMAL	0.3056

Total biaya terakumulasi dari lintasan ini adalah 0.3056. Dengan mengonversi nilai ini kembali menggunakan rumus invers $P_{total} = e^{-biaya\_total}$, didapatkan estimasi probabilitas keberhasilan kumulatif sebesar 73,70%.

B. Pembahasan

Lintasan Optimal menunjukkan bahwa keberhasilan dalam fase negosiasi sangat bergantung pada kelengkapan informasi yang dikumpulkan pada fase investigasi. Tindakan seperti menganalisis korban untuk mengetahui nama devian [B3_INFO_NAME] dan memeriksa tablet untuk memahami motifnya [B5_INFO_MOTIVE] bukanlah opsional, melainkan fondasi krusial. Kedua informasi ini membuka opsi [C2_USE_NAME] dan [C3_USE_MOTIVE] yang memiliki bobot sangat rendah, artinya sangat efektif dalam menurunkan tingkat stres devian dan membangun kepercayaan. Melewatkan fase ini akan menutup ke jalur dengan biaya terendah, sehingga memaksa pemain menempun lintasan yang lebih mahal dan berisiko.

Urutan dialog pada lintasan optimal ([C2] → [C3] → [C4]) menunjukkan bahwa pendekatan yang empatik, informatif, dan jujur adalah strategi yang paling efektif. Dimulai dengan memanggil nama "Daniel" untuk membangun hubungan personal, dilanjutkan dengan menunjukkan pemahaman atas motifnya, dan diakhiri dengan kejujuran mengenai peran Connor. Puncak dari strategi de-eskalasi ini adalah tindakan mengusir helikopter [C5_DISMISS_HELI)] yang secara drastis mengurangi ancaman eksternal dan membawa devian ke keadaan tenang [D1_CALM_STATE]. Pendekatan ini kontras dengan pilihan agresif [C6_AGGRESSIVE_TALK] yang memiliki bobot sangat tinggi (1.6094) yang hampir pasti akan langsung menuju kegagalan [Z4_FAILURE].

V. KESIMPULAN

Hasil analisis program secara tegas menunjukkan bahwa penerapan Algoritam Dijkstra berhasil mengidentifikasi sebuah lintasan optimal yang spesifik di antara banyak kemungkinan jalur. Lintasan ini, yang memiliki total biaya terendah, secara kuantitatif terbukti memberikan peluang keberhasilan tertinggi untuk mencapai hasil akhir terbaik, yaitu menyelamatkan sandera tanpa ada korban jiwa. Temuan utama dari lintasan optimal ini menekankan dua strategi kunci: pentingnya fase investigasi yang komprehensif untuk mengumpulkan semua informasi vital, dan penerapan pendekatan negosiasi yang bersifat de-eskalatif, empatik, dan jujur.

VI. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa,
2. orang tua penulis,
3. dosen pengampu mata kuliah Matematika Diskrit,
4. teman-teman penulis, dan
5. pihak-pihak lain

yang telah mendukung penulis selama proses pengerjaan makalah ini.

DAFTAR PUSTAKA

[1] Munir, R. (2024). *Graf – Bagian 1* [Materi Kuliah Matematika Diskrit]. Institut Teknologi Bandung. Diakses pada 21 Juni 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf.

[2] Aulia, R., Ofianto, & Saputra, E. P. (2024). PENERAPAN ALGORITMA DIJKSTRA DALAM MENENTUKAN RUTE TERPENDEK UNTUK JASA PENGIRIMAN BARANG DI PALANGKA RAYA. *Jurnal Sains Komputer & Informatika (J-SAKTI)*, pp3387-3388.

[3] Testa, A. (2018). *Choices Explained - Detroit: Become Human FAQ/Walkthrough*. GameFAQs. Diakses pada 21 Juni 2025, dari https://gamefaqs.gamespot.com/ps4/182637-detroit-become-human/faqs/75941/choices-explained.

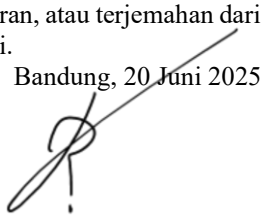
[4] GeeksforGeeks. (2024, 21 Mei). *Walks, Trails, Paths, Cycles and Circuits in Graph*. GeeksforGeeks. Diakses pada 20 Juni 2025, dari https://www.geeksforgeeks.org/walks-trails-paths-cycles-and-circuits-in-graph/.

[5] Dijkstra, E.W. (1959) A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, 1, 269-271. http://dx.doi.org/10.1007/BF01386390.

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, 20 Juni 2025



Ramadhian Nabil Firdaus Gumay - 13524126