Reduksi Polinomial *Traveling Salesperson Problem* (TSP) ke *Pac-Man Routing* sebagai Metode Pembuktian NP-*Completeness*

Narendra Dharma Wistara M.
NIM: 13524044[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13524044@std.stei.itb.ac.id*, [2]*andradaffamarpaung@gmail.com*

***Abstract*—Makalah ini memaparkan reduksi polinomial dari *Traveling Salesperson Problem* (TSP) ke *Pac-Man Routing* sebagai instrumen pembuktian NP-*Completeness*. Fokus utama penelitian terletak pada desain fungsi reduksi matematis f yang mentransformasi matriks ketetanggaan berbobot graf TSP menjadi matriks labirin Pac-Man yang mengandung distribusi titik *pellet* dan bobot jarak lorong. Pembuktian NP-*Completeness* Pac-Man Routing dilakukan melalui dua jalur: (1) menunjukkan keanggotaan Pac-Man Routing dalam kelas NP dengan mengonstruksi *verifier* polinomial, dan (2) membuktikan NP-*hardness* melalui reduksi polinomial TSP $\leq_p$ Pac-Man Routing. Kelayakan fungsi reduksi divalidasi secara empiris menggunakan program transformator yang diimplementasikan dalam C++. Metrik waktu eksekusi dicatat pada ukuran graf $N = 10$ hingga $N = 1.000$ untuk membuktikan bahwa transformasi masukan beroperasi dalam kompleksitas waktu polinomial $O(n^2)$. Hasil eksperimen mengonfirmasi bahwa waktu transformasi tumbuh secara kuadratik seiring pertambahan N, konsisten dengan klaim polinomial yang menjadi syarat keabsahan reduksi. Dengan demikian, makalah ini berkontribusi pada pengembangan bukti formal NP-*Completeness* berbasis permainan video klasik.**

***Index Terms*—NP-*Completeness*, Reduksi Polinomial, *Traveling Salesperson Problem*, Pac-Man Routing**

I. Pendahuluan

Teori kompleksitas komputasi menyediakan kerangka matematis untuk mengklasifikasikan masalah berdasarkan sumber daya komputasi—utamanya waktu dan memori—yang dibutuhkan untuk menyelesaikannya. Salah satu pencapaian terpenting dari teori ini adalah identifikasi kelas NP-*Complete* (himpunan masalah yang secara bersamaan berada dalam NP dan NP-*hard*). Pembuktian bahwa suatu masalah bersifat NP-*Complete* memiliki implikasi besar: jika sebuah algoritma polinomial ditemukan untuk salah satu masalah dalam kelas ini, maka seluruh masalah dalam NP dapat diselesaikan dalam waktu polinomial, yang berarti P = NP [1].

Traveling Salesperson Problem (TSP) adalah salah satu masalah NP-*Complete* paling terkenal dan telah dipelajari secara luas sejak dekade 1970-an [2]. Secara intuitif, TSP menanyakan apakah seorang salesman dapat mengunjungi seluruh kota tepat sekali dan kembali ke kota asal dengan total jarak tidak melebihi batas tertentu. Masalah ini muncul dalam berbagai aplikasi nyata, mulai dari perencanaan rute logistik hingga penjadwalan produksi.

Di sisi lain, permainan video Pac-Man yang diciptakan oleh Namco pada tahun 1980 menghadirkan tantangan rute yang secara struktural sangat menyerupai TSP: karakter Pac-Man harus mengumpulkan seluruh *pellet* di dalam labirin dengan rute seefisien mungkin sambil menghindari musuh. Kesamaan struktural ini membuka peluang untuk mengonstruksi reduksi polinomial dari TSP ke *Pac-Man Routing*, sebuah formalisasi masalah rute Pac-Man sebagai masalah keputusan.

II. Dasar Teori

A. Kompleksitas Algoritma

Kompleksitas algoritma adalah ukuran sumber daya komputasi—terutama waktu dan ruang memori—yang dibutuhkan oleh suatu algoritma sebagai fungsi dari ukuran masukan n. Kompleksitas waktu dinyatakan dalam notasi asimptotik: $O(f(n))$ mendeskripsikan batas atas pertumbuhan waktu, $\Omega(f(n))$ batas bawah, dan $\Theta(f(n))$ batas ketat [4].

Secara umum, algoritma dibedakan berdasarkan kelas kompleksitasnya menjadi algoritma polinomial yang membutuhkan waktu $O(n^k)$ untuk suatu konstanta k dan dianggap efisien serta traktabel secara komputasi, dan algoritma eksponensial yang membutuhkan waktu $O(2^n)$ atau lebih lambat sehingga dianggap tidak traktabel untuk masukan berukuran besar.

Garis batas antara dua kategori ini menjadi dasar klasifikasi masalah dalam teori kompleksitas komputasi. Untuk masalah keputusan (masalah dengan jawaban "Ya" atau "Tidak"), klasifikasi tersebut diformalisasi melalui kelas kompleksitas P, NP, dan NP-*Complete*.

B. Kelas P, NP, dan NP-Complete

Kelas P adalah himpunan masalah keputusan yang dapat *diselesaikan* oleh mesin Turing deterministik dalam waktu polinomial terhadap ukuran masukan. Contoh: pengurutan (*sorting*), pencarian jalur terpendek (Dijkstra), dan uji bilangan prima (AKS).

Kelas NP (*Nondeterministic Polynomial*) adalah himpunan masalah keputusan yang jawabannya—jika "Ya"—dapat *diverifikasi* dalam waktu polinomial menggunakan *certificate* yang diberikan [1]. Secara formal, masalah $L \in$ NP jika dan hanya jika terdapat *verifier* deterministik V dan polinom p sehingga:

$$x \in L \iff \exists\, c,\ |c| \le p(|x|) \text{ dan } V(x,c) = 1.$$

Jelas bahwa P $\subseteq$ NP; pertanyaan apakah P $=$ NP atau P $\neq$ NP masih merupakan masalah terbuka terpenting dalam ilmu komputer.

Kelas NP-*hard* adalah himpunan masalah yang setidaknya sama sulitnya dengan masalah tersulit dalam NP. Sebuah masalah B bersifat NP-*hard* jika setiap masalah $A \in$ NP dapat direduksi ke B dalam waktu polinomial ($A \le_p B$).

Kelas NP-*Complete* adalah irisan antara NP dan NP-*hard*. Masalah B bersifat NP-*Complete* jika: (1) $B \in$ NP, dan (2) terdapat masalah NP-*Complete* A dengan $A \le_p B$. Masalah NP-*Complete* pertama dibuktikan oleh Cook (1971) adalah *Boolean Satisfiability* (SAT) [3].

C. Traveling Salesperson Problem (TSP)

Traveling Salesperson Problem (TSP) dalam versi keputusan didefinisikan sebagai berikut. Diberikan graf berbobot lengkap $G = (V, E, w)$ dengan $n = |V|$ kota dan $w : E \to \mathbb{Z}^+$ fungsi bobot, serta bilangan bulat positif k; apakah terdapat siklus Hamiltonan pada G dengan total bobot $\le k$?

Representasi standar TSP menggunakan matriks ketetanggaan berbobot $W \in \mathbb{Z}^{n \times n}$:

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix}$$

di mana w_{ij} adalah biaya perjalanan dari kota i ke kota j, dan $w_{ii} = \infty$ (tidak ada *self-loop*). TSP dibuktikan NP-*Complete* oleh Karp (1972) melalui reduksi dari *Hamiltonian Cycle* [2].

D. Permainan Pac-Man

Pac-Man adalah permainan video *arcade* yang dikembangkan oleh Namco dan dirilis pada tahun 1980 [9]. Dalam permainan ini, pemain mengendalikan karakter Pac-Man di dalam sebuah labirin dua dimensi. Tujuan utama permainan adalah mengumpulkan seluruh titik (*pellet*) yang tersebar di sepanjang lorong labirin sembari menghindari empat musuh hantu (Blinky, Pinky, Inky, dan Clyde).

Labirin Pac-Man secara komputasi dapat direpresentasikan sebagai graf tak berarah berbobot $G_L = (V_L, E_L, w_L)$, di mana:

- V_L adalah himpunan sel lorong yang dapat dilalui,
- E_L adalah himpunan pasang sel yang terhubung langsung (berdampingan),
- $w_L(e)$ adalah biaya perpindahan melewati tepi $e \in E_L$.

Dari sudut pandang teori graf, Pac-Man harus menemukan suatu *walk* pada G_L yang melewati setiap *pellet* setidaknya satu kali dengan biaya total minimum—masalah yang dalam kasus *pellet* tersebar mengikuti struktur siklus Hamiltonan berbobot minimum.

E. Pac-Man Routing sebagai Masalah Keputusan

Pac-Man Routing diformulasikan sebagai masalah keputusan berikut. Diberikan:

- Matriks labirin $M \in \mathbb{Z}^{m \times m}$,
- Himpunan posisi *pellet* $P \subseteq \{(i,j) \mid M[i][j] = 1\}$,
- Posisi awal $s \in P$,
- Batas total bobot $k \in \mathbb{Z}^+$;

apakah terdapat rute bagi Pac-Man yang memulai dari s, mengunjungi setiap *pellet* dalam P tepat satu kali, dan kembali ke s, dengan total bobot perpindahan $\le k$?

Matriks M mengkodekan struktur labirin sebagai berikut:

$$M[i][j] = \begin{cases} 1 & \text{sel } \textit{pellet} \text{ (wajib dikunjungi)} \\ w > 0 & \text{bobot lorong antar dua sel} \\ -1 & \text{dinding (tidak dapat dilewati)} \end{cases}$$

Korespondensi antara TSP dan Pac-Man Routing dapat dilihat pada Tabel I.

Tabel I
PEMETAAN KOMPONEN TSP KE PAC-MAN ROUTING

Komponen TSP	Komponen Pac-Man Routing
Kota $v_i \in V$	Posisi *pellet* $p_i \in P$
Tepi $(v_i, v_j) \in E$	Lorong antara p_i dan p_j
Bobot tepi w_{ij}	Bobot lorong $M[2i][2j]$
Siklus Hamiltonan	Rute kunjungan semua *pellet*
Batas bobot k	Batas total bobot k'

F. Reduksi Polinomial

Reduksi polinomial (*polynomial-time many-one reduction*) $A \le_p B$ adalah suatu fungsi f yang dapat dihitung dalam waktu polinomial terhadap ukuran masukan, sehingga untuk setiap instans x:

$$x \in A \iff f(x) \in B.$$

Sifat ini menjamin: (1) jika B dapat diselesaikan dalam waktu polinomial, maka A juga dapat; dan (2) jika A diketahui NP-*hard*, maka B juga NP-*hard* [1]. Validitas sebuah reduksi bergantung pada dua syarat utama berikut.

- Syarat polinomialitas menyatakan bahwa fungsi reduksi f harus dapat dihitung dalam waktu polinomial $O(n^k)$ untuk suatu konstanta k.
- Syarat kebenaran (*correctness*) menyatakan bahwa jawaban "Ya" pada instans masalah A harus berkorespondensi secara tepat dengan jawaban "Ya" pada instans hasil pemetaan B, begitu pula sebaliknya.

III. HASIL DAN PEMBAHASAN

A. Penyusunan Fungsi Reduksi f

Inti dari pembuktian NP-*Completeness* Pac-Man Routing adalah konstruksi fungsi reduksi f yang memetakan setiap instans TSP ke instans Pac-Man Routing secara benar dan efisien. Diberikan instans TSP $\langle G, k \rangle$ dengan $G = (V, E, w)$ dan $|V| = n$, fungsi f menghasilkan instans Pac-Man Routing $\langle M, P, s, k' \rangle$ melalui langkah-langkah berikut.

1) Konstruksi Matriks Labirin: Matriks labirin M dibentuk berukuran $(2n-1) \times (2n-1)$. Ukuran ini dipilih agar setiap pasang *pellet* dapat dihubungkan oleh tepat satu sel lorong perantara tanpa tumpang-tindih. Sebagai contoh, untuk TSP dengan $n = 4$ kota, matriks M berukuran 7×7 dengan *pellet* pada posisi $(0,0)$, $(2,2)$, $(4,4)$, $(6,6)$.

Sel-sel pada posisi diagonal genap merepresentasikan simpul (kota) TSP sebagai titik *pellet*:

$$M[2i][2i] = 1 \quad \text{untuk } i = 0, 1, \ldots, n-1.$$

Sel-sel di luar diagonal merepresentasikan bobot tepi TSP sebagai bobot lorong:

$$M[2i][2j] = M[2j][2i] = w_{ij} \quad \text{untuk } i \neq j,\ w_{ij} \neq \infty.$$

Sel-sel yang tidak mewakili *pellet* maupun lorong diisi -1 (dinding). Secara ringkas, aturan pengisian M adalah:

$$M[r][c] = \begin{cases} 1 & \text{jika } r = c = 2i \text{ (pellet ke-}i\text{)} \\ w_{ij} & \text{jika } r = 2i,\ c = 2j,\ i \neq j,\ w_{ij} \neq \infty \\ -1 & \text{selainnya (dinding)} \end{cases}$$

Sebagai ilustrasi, misalkan diberikan instans TSP dengan $n = 3$ kota dan matriks bobot:

$$W = \begin{bmatrix} \infty & 5 & 8 \\ 5 & \infty & 3 \\ 8 & 3 & \infty \end{bmatrix}$$

Maka matriks labirin M berukuran 5×5 yang dihasilkan oleh f adalah:

$$M = \begin{bmatrix} 1 & -1 & 5 & -1 & 8 \\ -1 & -1 & -1 & -1 & -1 \\ 5 & -1 & 1 & -1 & 3 \\ -1 & -1 & -1 & -1 & -1 \\ 8 & -1 & 3 & -1 & 1 \end{bmatrix}$$

Pellet berada pada $M[0][0]$, $M[2][2]$, dan $M[4][4]$, sedangkan bobot lorong antar *pellet* i dan j dibaca dari sel $M[2i][2j]$.

2) Penentuan Parameter Reduksi: Setelah matriks M terbentuk, parameter instans Pac-Man Routing ditentukan secara sistematis. Himpunan *pellet* $P = \{(2i, 2i) \mid i = 0, \ldots, n-1\}$ didefinisikan agar berkorespondensi satu-satu dengan kota-kota pada TSP. Selanjutnya, posisi awal Pac-Man ditentukan pada koordinat $s = (0,0)$ untuk merepresentasikan kota awal yang dikunjungi oleh *salesman*. Terakhir, batas bobot perjalanan ditetapkan sebesar $k' = k$ karena total bobot rute perjalanan Pac-Man berkorespondensi langsung dengan akumulasi bobot dari siklus Hamiltonan pada TSP.

B. Analisis Kompleksitas Fungsi Reduksi

1) Kompleksitas Waktu: Fungsi reduksi f mengeksekusi tiga langkah utama secara berurutan. Pertama adalah langkah inisialisasi matriks dinding, di mana seluruh $(2n-1)^2$ sel diisi dengan nilai -1 untuk merepresentasikan dinding. Proses inisialisasi ini membutuhkan waktu sebesar:

$$T_1(n) = (2n-1)^2 = 4n^2 - 4n + 1 = O(n^2).$$

Kedua adalah langkah penempatan *pellet*, di mana sebanyak n posisi pada diagonal genap matriks diisi dengan nilai 1. Waktu yang dibutuhkan untuk menempatkan seluruh *pellet* ini adalah:

$$T_2(n) = n = O(n).$$

Ketiga adalah langkah penempatan bobot lorong, di mana sebanyak $n(n-1)$ pasang sel $(2i, 2j)$ dengan $i \neq j$ dan $w_{ij} \neq \infty$ diisi dengan nilai bobot w_{ij} dari matriks TSP. Karena pembacaan setiap entri matriks TSP dilakukan dalam waktu konstan $O(1)$, total waktu untuk langkah ini adalah:

$$T_3(n) = n(n-1) = O(n^2).$$

Dengan menjumlahkan kontribusi dari ketiga tahapan tersebut, total kompleksitas waktu dari fungsi reduksi f adalah:

$$T_f(n) = T_1(n) + T_2(n) + T_3(n) = O(n^2) + O(n) + O(n^2) = O(n^2).$$

Hasil ini membuktikan bahwa fungsi reduksi f memenuhi syarat polinomialitas yang diperlukan dalam pembuktian formal [1].

2) Kompleksitas Ruang: Matriks M berukuran $(2n-1) \times (2n-1)$ membutuhkan ruang:

$$S_f(n) = (2n-1)^2 \cdot O(1) = O(n^2).$$

Ruang ini juga polinomial terhadap n.

3) Perbandingan Kompleksitas TSP dan Pac-Man Routing: Tabel II merangkum kompleksitas beberapa operasi utama pada instans TSP berukuran n dan instans Pac-Man Routing hasil reduksi.

Tabel II
PERBANDINGAN KOMPLEKSITAS OPERASI

Operasi	TSP (n kota)	Pac-Man Routing
Ukuran representasi	$O(n^2)$	$O(n^2)$
Fungsi reduksi f	—	$O(n^2)$
Verifikasi solusi	$O(n)$	$O(n)$
Penghitungan bobot rute	$O(n)$	$O(n)$
Pengecekan kunjungan semua	$O(n)$	$O(n)$

Terlihat bahwa operasi-operasi utama pada Pac-Man Routing memiliki kompleksitas yang sebanding dengan TSP, dan fungsi reduksi f sendiri hanya menambah overhead $O(n^2)$—polinomial. Ini mengonfirmasi bahwa reduksi tidak mengubah kelas kompleksitas masalah yang terlibat.

C. Bukti Kebenaran Reduksi

1) TSP "Ya" $\Rightarrow$ Pac-Man Routing "Ya": Misalkan terdapat siklus Hamiltonan $\pi = (v_0, v_1, \ldots, v_{n-1}, v_0)$ pada G dengan total bobot $W(\pi) = \sum_{i=0}^{n-1} w_{v_i, v_{(i+1) \bmod n}} \leq k$. Rute Pac-Man dikonstruksi mengikuti urutan π: mulai dari sel $(2v_0, 2v_0)$, bergerak ke sel $(2v_1, 2v_1)$ melalui lorong berbobot $M[2v_0][2v_1] = w_{v_0, v_1}$, dan seterusnya hingga kembali ke $(2v_0, 2v_0)$. Total bobot rute Pac-Man:

$$W(\rho) = \sum_{i=0}^{n-1} M[2v_i][2v_{(i+1) \bmod n}] = \sum_{i=0}^{n-1} w_{v_i, v_{(i+1) \bmod n}} = W(\pi) \leq k =$$

Karena $W(\rho) \leq k'$, instans Pac-Man Routing menjawab "Ya".

2) Pac-Man Routing "Ya" ⇒ TSP "Ya": Misalkan terdapat rute Pac-Man $\rho = (p_0, p_1, \ldots, p_{n-1}, p_0)$ yang mengunjungi semua *pellet* dalam P dengan bobot total $W(\rho) \leq k'$. Setiap $p_i = (2u_i, 2u_i)$ berkorespondensi dengan kota $v_{u_i} \in V$ TSP. Karena:

$$M[2u_i][2u_{i+1}] = w_{u_i,u_{i+1}}$$

maka urutan $(v_{u_0}, v_{u_1}, \ldots, v_{u_{n-1}}, v_{u_0})$ membentuk siklus Hamiltonan pada G dengan total bobot:

$$W(\pi) = \sum_{i=0}^{n-1} w_{u_i,u_{(i+1) \bmod n}} = W(\rho) \leq k' = k.$$

Oleh karena itu, instans TSP menjawab "Ya". ■

D. Bukti Pac-Man Routing ∈ NP

Untuk membuktikan bahwa Pac-Man Routing berada dalam kelas NP, perlu ditunjukkan bahwa setiap solusi "Ya" dari masalah ini dapat *diverifikasi* dalam waktu polinomial menggunakan sebuah *certificate* berukuran polinomial [1]. Dengan kata lain, kita tidak perlu *menemukan* solusi—cukup diberikan solusi kandidat, kita bisa memverifikasi kebenarannya secara efisien.

1) Definisi Certificate: Untuk instans Pac-Man Routing $\langle M, P, s, k' \rangle$ dengan $|P| = n$, *certificate* yang digunakan adalah urutan terurut kunjungan *pellet*:

$$c = (p_0,\ p_1,\ p_2,\ \ldots,\ p_{n-1},\ p_0)$$

di mana setiap p_i adalah koordinat sebuah sel *pellet* dalam matriks M. Elemen pertama dan terakhir sama (p_0) untuk menegaskan bahwa rute membentuk siklus tertutup. Ukuran *certificate* adalah $|c| = n+1 = O(n)$, yang merupakan fungsi polinomial terhadap ukuran masukan.

2) Konstruksi Verifier V: Verifier deterministik V menerima pasangan masukan berupa instans $\langle M, P, s, k' \rangle$ dan *certificate* c. Untuk memastikan keabsahan *certificate* tersebut, verifikator melakukan pemeriksaan melalui poin-poin analisis berikut.

- Verifikasi posisi awal dilakukan dengan mencocokkan apakah elemen pertama rute p_0 sama dengan titik mula s yang ditentukan pada instans. Jika tidak cocok, verifikator langsung menolak solusi. Pemeriksaan ini membutuhkan waktu konstan $O(1)$.
- Verifikasi keanggotaan *pellet* dilakukan dengan memeriksa apakah seluruh posisi p_i dalam c merupakan bagian dari himpunan *pellet* P. Dengan menyimpan koordinat P ke dalam *hash set*, pengecekan untuk setiap elemen dapat berjalan dalam $O(1)$ sehingga pemeriksaan seluruh rute memakan waktu $O(n)$.
- Verifikasi kunjungan tepat satu kali bertujuan menjamin bahwa tidak ada *pellet* yang dikunjungi lebih dari sekali atau justru terlewat. Pengecekan ini memanfaatkan bantuan larik penanda `visited` bertipe boolean berukuran n. Setiap kali sebuah *pellet* dikunjungi, statusnya diubah menjadi `true`. Jika ditemukan *pellet* yang sudah ditandai sebelumnya, verifikator akan langsung menolak solusi. Langkah ini diselesaikan dalam waktu $O(n)$.
- Verifikasi batas total bobot dilakukan dengan menghitung jumlah bobot perpindahan Pac-Man sepanjang rute perjalanan. Jika verifikator mendeteksi adanya lintasan yang menabrak dinding (ditandai dengan nilai -1 pada matriks M) atau jika akumulasi bobot melebihi batas nilai k', solusi akan ditolak. Proses penjumlahan dan perbandingan akhir ini diselesaikan dalam waktu $O(n)$.

3) Analisis Kompleksitas Verifier: Total waktu V adalah jumlah waktu keempat langkah:

$$T_V(n) = O(1) + O(n) + O(n) + O(n) = O(n).$$

Karena $T_V(n) = O(n)$ dan ukuran *certificate* $|c| = O(n)$, maka *verifier* V adalah *verifier* polinomial yang valid.

4) Kebenaran Verifier: Kebenaran dari *verifier* V dibuktikan melalui dua kriteria utama berikut.

- Aspek kelengkapan (*completeness*) menjamin bahwa jika instans awal bernilai benar (jawaban "Ya"), maka terdapat minimal satu rute perjalanan Pac-Man ρ yang valid. Apabila rute ρ ini dimasukkan sebagai *certificate*, seluruh tahapan pemeriksaan pada verifikator dipastikan lolos, sehingga verifikator memberikan keputusan untuk menerima.
- Aspek kesahihan (*soundness*) menjamin bahwa jika verifikator memberikan keputusan menerima terhadap suatu *certificate* c, maka rute tersebut telah terverifikasi memenuhi semua batasan masalah. Rute tersebut terbukti dimulai dari posisi awal yang tepat, melintasi koordinat *pellet* yang valid tanpa menembus dinding, mengunjungi setiap *pellet* tepat satu kali, dan memiliki total bobot perjalanan tidak melebihi k'. Keberadaan rute semacam ini membuktikan bahwa instans Pac-Man Routing tersebut memang bernilai benar.

Karena *verifier* polinomial yang *sound* dan *complete* ada, dan *certificate* berukuran polinomial, maka:

Pac-Man Routing ∈ NP. ■

E. Kesimpulan Bukti NP-Completeness

Pembuktian formal NP-*Completeness* dari *Pac-Man Routing* disimpulkan dengan menggabungkan dua hasil utama berikut.

- Masalah *Pac-Man Routing* terbukti berada di dalam kelas NP karena memiliki verifikator deterministik berkompleksitas waktu polinomial $O(n)$ dengan bantuan *certificate* berukuran polinomial.
- Masalah TSP yang merupakan masalah NP-*Complete* klasik [2] terbukti dapat direduksi secara polinomial ke *Pac-Man Routing* melalui fungsi reduksi f dengan kompleksitas waktu $O(n^2)$.

Berdasarkan teorema dasar kompleksitas komputasi [1], jika suatu masalah NP-*Complete* dapat direduksi secara polinomial ke masalah baru yang terbukti berada dalam kelas NP, maka masalah baru tersebut juga merupakan NP-*Complete*. Berdasarkan relasi TSP $\leq_p$ Pac-Man Routing, disimpulkan bahwa:

Pac-Man Routing ∈ NP-Complete. ■

F. Program Transformator

1) Detail Algoritma dan Implementasi Program: Program implementasi reduksi dikembangkan untuk memvalidasi fungsi reduksi f secara langsung. Algoritma utama program bekerja dengan menerima representasi graf TSP dalam bentuk matriks ketetanggaan dan memetakan nilai-nilainya ke dalam matriks labirin Pac-Man berukuran $(2n-1) \times (2n-1)$.

Struktur data utama yang digunakan untuk merepresentasikan graf masukan dan labirin luaran didefinisikan sebagai tipe data larik dua dimensi dinamis dalam C++ (`std::vector<std::vector<int>>`). Algoritma reduksi diimplementasikan melalui fungsi `reduce()` yang ditunjukkan pada Potongan Kode 1.

```cpp
// Representasi matriks
using Matrix = std::vector<std::vector<int>>;

Matrix reduce(const Matrix& W) {
    int n = W.size();
    int m = 2 * n - 1; // Ukuran labirin

    // Tahap 1: Inisialisasi sel sebagai dinding (-1)
    Matrix M(m, std::vector<int>(m, -1));

    // Tahap 2: Tempatkan pellet pada diagonal genap
    for (int i = 0; i < n; i++) {
        M[2 * i][2 * i] = 1;
    }

    // Tahap 3: Tempatkan bobot lorong dari matriks TSP
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i != j && W[i][j] != -1) {
                M[2 * i][2 * j] = W[i][j];
            }
        }
    }
    return M;
}
```

Listing 1. Fungsi Reduksi TSP ke Pac-Man dalam C++

Implementasi lengkap dari kode program transformator ini, beserta berkas pengujian dan instruksi eksekusinya, telah diunggah dan disimpan pada Google Drive sebagai referensi algoritma tambahan yang dapat diakses publik melalui tautan: https://drive.google.com/drive/folders/1M08dI2LgVigM-WSWVtyRLiXXutqic4W1?usp=sharing.

2) Spesifikasi Program: Program transformator diimplementasikan dalam C++. Program membaca matriks ketetanggaan TSP berukuran $n \times n$ dari *file* teks, kemudian menghasilkan matriks labirin M berukuran $(2n-1) \times (2n-1)$ sesuai fungsi f. Pengukuran waktu dilakukan menggunakan `std::chrono::high_resolution_clock`. Setiap konfigurasi N diuji sebanyak lima kali dan diambil rata-rata untuk meredam *noise* lingkungan. Graf masukan dibangkitkan acak dengan bobot seragam $w_{ij} \in [1, 1000]$. Pengujian dilakukan pada mesin dengan prosesor Intel Core i5-1135G7, RAM 16 GB.

3) Hasil Pengukuran dan Analisis Eksperimen: Pengukuran waktu eksekusi program transformator dilakukan untuk memverifikasi kesesuaian antara estimasi teoritis kompleksitas waktu reduksi $O(n^2)$ dengan performa riil di lapangan. Hasil pengujian untuk berbagai nilai ukuran masukan N dicantumkan secara lengkap pada Tabel III.

Tabel III
WAKTU EKSEKUSI PROGRAM TRANSFORMATOR TSP → PAC-MAN

N (kota)	Ukuran matriks M	Waktu (ms)
10	19×19	0.001
50	99×99	0.010
100	199×199	0.055
200	399×399	0.789
300	599×599	1.630
500	999×999	4.298
750	1499×1499	10.259
1000	1999×1999	18.649

Tabel III menunjukkan pola pertumbuhan waktu eksekusi program transformator yang sejalan dengan prediksi teoritis $O(n^2)$. Guna melakukan verifikasi secara kuantitatif, dilakukan analisis terhadap dua metrik pengukuran berikut.

Analisis nilai rasio antara waktu eksekusi dengan kuadrat ukuran masukan $(T(N)/N^2)$ digunakan untuk membuktikan apakah pertumbuhan waktu bersifat kuadratis secara ketat. Apabila kompleksitas waktu berada pada kelas $\Theta(n^2)$, nilai rasio tersebut seharusnya konvergen menuju suatu nilai konstanta tertentu.

Tabel IV
VERIFIKASI KUADRATIK: RASIO $T(N)/N^2$

N	$T(N)$ (ms)	$T(N)/N^2$ ($\times 10^{-6}$)
10	0.001	11.944
50	0.010	4.090
100	0.055	5.501
200	0.789	19.715
300	1.630	18.111
500	4.298	17.192
750	10.259	18.239
1000	18.649	18.649

Berdasarkan Tabel IV, terlihat bahwa nilai rasio $T(N)/N^2$ sangat konsisten mendekati konstanta sekitar 18×10^{-6} ms untuk semua pengujian dengan ukuran $N \geq 200$. Hal ini memverifikasi secara empiris bahwa program transformator bekerja dengan kompleksitas $\Theta(n^2)$.

Analisis faktor pertumbuhan juga dilakukan dengan mengamati rasio kenaikan waktu ketika ukuran masukan dilipatgandakan. Sebagai contoh, saat nilai N dinaikkan sebesar dua kali lipat dari 500 menjadi 1000, waktu eksekusi program terpantau meningkat dari 4,298 ms menjadi 18,649 ms, yaitu naik sekitar 4,34 kali (mendekati 4 kali lipat). Karakteristik kenaikan empat kali lipat ini merupakan indikator khas dari fungsi kompleksitas kuadratis $O(n^2)$ karena relasi kuadratis $(2N)^2/N^2 = 4$.

Hasil validasi empiris ini secara keseluruhan konsisten dengan analisis teoritis yang telah dipaparkan sebelumnya dan memperkuat bukti bahwa reduksi TSP $\leq_p$ Pac-Man Routing dapat diselesaikan secara efisien dalam waktu polinomial. Perlu dicatat bahwa fungsi dari program transformator ini bukanlah memecahkan solusi optimal dari TSP atau Pac-Man Routing, melainkan hanya melakukan konstruksi pemetaan masukan dari satu masalah ke masalah lainnya.

IV. KESIMPULAN

Makalah ini telah memaparkan pembuktian formal NP-*Completeness* dari *Pac-Man Routing* melalui reduksi polinomial dari *Traveling Salesperson Problem* (TSP). Berikut adalah ringkasan temuan utama.

- Perancangan fungsi reduksi matematis f berhasil mentransformasikan matriks ketetanggaan berbobot TSP menjadi matriks labirin Pac-Man berukuran $(2n-1) \times (2n-1)$. Setiap simpul TSP dipetakan sebagai *pellet* pada sel diagonal matriks labirin, dan setiap bobot tepi w_{ij} dipetakan sebagai bobot lorong pada sel perantara $M[2i][2j]$, dengan batas bobot k dipertahankan sama di kedua instans.
- Kebenaran (*correctness*) fungsi reduksi dibuktikan secara dua arah, yaitu instans TSP dengan jawaban "Ya" mengimplikasikan instans Pac-Man Routing dengan jawaban "Ya", dan sebaliknya, sehingga membuktikan keabsahan reduksi secara logis.
- Keanggotaan Pac-Man Routing dalam kelas NP dibuktikan secara formal dengan mengonstruksi sebuah verifikator deterministik berkompleksitas waktu polinomial $O(n)$ untuk memvalidasi urutan kunjungan *pellet*.
- Kompleksitas waktu fungsi reduksi f terbukti secara teoritis adalah $O(n^2)$ karena didominasi oleh inisialisasi dan pengisian sel-sel matriks labirin. Analisis ini divalidasi secara empiris menggunakan program C++ untuk ukuran masukan $N = 10$ hingga 1.000, dengan nilai rasio $T(N)/N^2$ yang konstan dan mengonfirmasi pertumbuhan kuadratik.

Dari keempat poin di atas, karena TSP adalah NP-*Complete* [2], TSP $\leq_p$ Pac-Man Routing (melalui f polinomial), dan Pac-Man Routing $\in$ NP, dapat disimpulkan:

Pac-Man Routing $\in$ NP-Complete.

Penelitian ini membuka beberapa arah lanjutan. Dari sisi teori, konstruksi reduksi serupa dapat dikembangkan untuk varian TSP lain (asimetris, metrik) atau masalah rute pada permainan video lain. Dari sisi praktis, pemahaman NP-*Complete*ness Pac-Man Routing memotivasi pengembangan heuristik dan algoritma aproksimasi untuk permainan berbasis rute, di mana solusi optimal tidak dapat dijamin dalam waktu polinomial (dengan asumsi P $\neq$ NP).

REFERENCES

[1] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Boston, MA, USA: Cengage Learning, 2012.

[2] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York, NY, USA: Plenum Press, 1972, pp. 85–103. [Online]. Available: https://doi.org/10.1007/978-1-4684-2001-2_9.

[3] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd Annu. ACM Symp. Theory of Computing (STOC)*, New York, NY, USA, 1971, pp. 151–158. [Online]. Available: https://doi.org/10.1145/800157.805047.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[5] C. H. Papadimitriou, *Computational Complexity*. Reading, MA, USA: Addison-Wesley, 1994.

[6] G. Viglietta, "Gaming is a hard job, but someone has to do it!" *Theory of Computing Systems*, vol. 54, no. 4, pp. 595–621, 2014. [Online]. Available: https://doi.org/10.1007/s00224-013-9497-5.

[7] G. Kendall, A. Parkes, and K. Spoerer, "A survey of NP-complete puzzles," *ICGA Journal*, vol. 31, no. 1, pp. 13–34, 2008.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman, 1979.

[9] M. J. P. Wolf, Ed., *Before the Crash: Early Video Game History*. Detroit, MI, USA: Wayne State University Press, 2012.

[10] S. Arora, "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems," *J. ACM*, vol. 45, no. 5, pp. 753–782, Sep. 1998. [Online]. Available: https://doi.org/10.1145/290179.290180.

UCAPAN TERIMA KASIH DAN KATA HATI

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga makalah ini dapat diselesaikan. Ucapan terima kasih yang tulus penulis sampaikan kepada Bapak Ir. Rila Mandala, M.Eng., Ph.D. selaku dosen pengampu mata kuliah IF2221 Strategi Algoritma yang telah memberikan penjelasan mendalam tentang strategi algoritma, terutama materi kelas P, NP, dan NP-*Complete*.

Saya bersyukur setelah mengikuti mata kuliah ini saya menjadi lebih paham mengenai macam-macam algoritma yang dapat digunakan dalam menyelesaikan permasalahan-permasalahan di dunia. Beberapa kali dalam mata kuliah ini pun saya menjadi terdorong untuk tidak menggunakan kakas AI dalam menyelesaikan tugas-tugas di dalamnya. Mata kuliah ini menjadi pendorong bahwa mengerjakan segala sesuatu tanpa menggunakan AI terkadang lebih baik dalam proses pembelajaran.

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. Pada halaman selanjutnya juga saya lampirkan surat pernyataan penggunaan AI yang saya gunakan dalam mengerjakan tugas ini.

Bandung, 19 Juni 2026

Narendra Dharma Wistara M.
13524044

SURAT PERNYATAAN PENGGUNAAN AI

Dengan ini, saya:

Nama	:	Narendra Dharma Wistara M. / 13524044
Fakultas/Sekolah	:	Sekolah Teknik Elektro dan Informatika - Komputasi
Program Studi	:	Teknik Informatika
Kode Mata Kuliah	:	IF2221
Nama Mata Kuliah	:	Strategi Algoritma
Judul Tugas/Proposal	:	Reduksi Polinomial *Traveling Salesperson Problem* (TSP) ke *Pac-Man Routing* sebagai Metode Pembuktian NP-*Completeness*

Menyatakan bahwa saya menggunakan AI dalam pengerjaan maupun penyusunan luaran tugas pada mata kuliah yang tertulis di atas. Alat AI/kecerdasan buatan yang digunakan adalah:

Lingkup Pekerjaan	Digunakan?	Tingkat Penggunaan
Pemeriksaan Ejaan: menggunakan tools seperti Grammarly, Paperpal, Deepl, Quillbot, Grammarbot, LanguageTool, ProWritingAid, ChatGPT, Google Gemini, atau sejenisnya	Ya	Rendah
Pembuatan Teks: menggunakan tools seperti ChatGPT, Gemini, Paperpal, Copilot, GrammarlyGo, WordAI, WriteSonic, Jasper, Jenni AI, atau sejenisnya.	Ya	Sedang
Bantuan Diskusi dalam Penyusunan Konten: menggunakan tools seperti ChatGPT, Gemini, Copilot, Perplexity, Jenni AI, Zoom-Companion, atau sejenisnya.	Ya	Sedang
Pembuatan Gambar, Video, dan/atau Grafik: menggunakan tools seperti Craiyon, DALL-E, Midjourney, Stable Diffusion, Microsoft Designer, Gemini, Canva AI, atau sejenisnya.	Tidak	-
Pencarian Referensi: menggunakan tools, seperti Perplexity, Gemini, atau sejenisnya.	Ya	Sedang
Translasi Referensi Berbahasa Asing: menggunakan tools, seperti Perplexity, Gemini, atau sejenisnya.	Ya	Rendah

Saya juga menyatakan bahwa setiap penggunaan AI/kecerdasan buatan yang dilakukan pada mata kuliah tersebut telah dinyatakan di dalam surat ini.

Bandung, 19 Juni 2026
Penulis,

Narendra Dharma Wistara M.
13524044