Implementasi Algoritma *Best-First Search* (BeFS) Pada Penyelesaian *Traveling Salesman Problem* (TSP)
(Studi Kasus: Perjalanan Wisata di Kota Yogyakarta)

Muchammad Abrori dan Rike Nur Setiyani

Program Studi Matematika Fakultas Sains dan Teknologi, UIN Sunan Kalijaga, Jl. Marsda Adisucipto No. 1 Yogyakarta, Indonesia

Korespondensi; Muchammad Abrori; Email: muchammad.abrorii@uin-suka.ac.id

Abstrak

Yogyakarta menawarkan banyak tempat wisata, dari wisata alam, wisata kuliner sampai wisata budaya. Dengan begitu banyak tempat wisata yang ditawarkan oleh Yogyakarta, wisatawan sering menemukan kesulitan untuk mengatur jadwal perjalanan mereka (dari memilih mana tempat wisata yang akan dikunjungi sampai memilih rute yang turis perlukan untuk memaksimalkan waktu liburan mereka). Oleh karena itu, diperlukan untuk memiliki cara untuk menentukan rute terpendek sehingga turis dapat membuat tur mereka di Yogyakarta efektif. Masalah ini dapat dikategorikan sebagai kasus Traveling Salesman Problem (TSP). Ada banyak metode yang dapat digunakan untuk menemukan rute terpendek di kasus Travelling Salesman Masalah (TSP). Untuk memecahkan masalah, yaitu untuk menemukan rute terpendek di Yogyakarta, Algoritma Traveling Best-Pertama akan digunakan dalam penelitian ini. Pelaksanaan Algoritma Best-First Search untuk menemukan rute tur terpendek di Yogyakarta dapat digunakan untuk menghasilkan solusi bagi wisatawan untuk memilih paket tour terpendek dan memutuskan rute yang harus mereka ambil. Paket tour premium menghasilkan rute tur dari Bandara Adi Sucipto - kebun binatang Gembira Loka - Purawisata - N'dalem Gamelan Hotel - Yogyakarta Palace - Benteng Vredeburg Museum - Taman Pintar – Tamansari - Bandara Adi Sucipto dengan jarak yang ditempuh 20,297 meter. Paket tour tengah memproduksi rute tur dari Stasiun Tugu - Benteng Vredeburg - Taman Pintar - Yogyakarta Palace - Mawar Asri Hotel – Tamansari – Purawisata - Gembira Loka Zoo - Tugu dengan jarak yang ditempuh 11,772 meter. Paket tour ekonomi menghasilkan rute tur dari Giwangan bus stasiun - stasiun bus Gembira Loka Zoo – Purawisata - Yogyakarta Palace - Mitra Hotel - Benteng Vredeburg Museum - Taman Pintar – Tamansari - Giwangan dengan jarak yang ditempuh 14,037 meter.

Kata Kunci: Rute terpendek; Traveling Salesman Problem; Algoritma Best-First Search; City tour di Yogyakarta

Abstract

Yogyakarta offers many tourist attractions, from nature based tourism, culinary tourism until cultural tourism. With so many tourist attractions offered by Yogyakarta, tourist often finds it difficult to arrange their travel schedule (from choosing which tourist attractions to be visited until choosing which route tourist should takes to maximize their vacation time). Therefore, it's required to have a way to determine the shortest tour route so tourist can make their tour in the Yogyakarta effective. This problem can be categorized as Traveling Salesman Problem (TSP) case. There are a lot of methods can be used to find the shortest route in Travelling Salesman Problems (TSP) case. To solve the problem, which is to find the shortest tour route in Yogyakarta, Algorithm Best-First Travelling will be used in this undergraduate thesis. The implementation of Algorithm Best-First Search to find the shortest tour route in Yogyakarta can be used to produce a solution for tourist to choose the shortest tour package and decide which route they should take. The premium tour package produces tour route from Adi Sucipto Airport - Gembira Loka Zoo - Purawisata - N'dalem Gamelan Hotel - Yogyakarta Palace - Benteng Vredeburg Museum - Taman Pintar - Tamansari - Adi Sucipto Airport with distance covered 20.297 meter. The middle tour package produces tour route from Tugu railway station - Benteng Vredeburg Museum - Taman Pintar-Yogyakarta Palace-Mawar Asri Hotel-Tamansari – Purawisata - Gembira Loka Zoo - Tugu railway station with distance covered 11.772 meter. The economy tour package produces tour route from Giwangan bus station- Gembira Loka zoo – Purawisata - Yogyakarta Palace - Mitra Hotel - Benteng Vredeburg Museum - Taman Pintar – Tamansari - Giwangan bus station with distance covered 14.037 meter.

Keywords: Shortest route; Travelling Salesman Problem; Best-First Search Algorithm; City tour in Yogyakarta

Pendahuluan

Yogyakarta dikenal sebagai salah satu destinasi wisata favorit di Indonesia. Berbagai tempat wisata ditawarkan di Yogyakarta, baik wisata alam maupun wisata budaya. Hal ini menarik banyak minat wisatawan baik wisatawan domestik maupun wisatawan asing untuk berkunjung ke tempat wisata di Kota Yogyakarta, baik menggunakan kendaraan pribadi maupun kendaraan umum. Umumnya wisatawan tersebut ingin mengunjungi salah satu atau beberapa tempat wisata sekaligus dalam waktu singkat. Akan tetapi banyak dari wisatawan tersebut belum mengetahui rute tempat wisata sehingga menghabiskan banyak waktu di perjalanan dan cenderung tidak efektif.

Oleh karena itu dibutuhkan suatu solusi bagaimana mengetahui rute terpendek untuk mencapai suatu tujuan. Pencarian suatu jalur perjalanan yang efisien merupakan salah satu hal penting yang harus ada, karena dengan adanya perencanaan jalur perjalanan akan memberikan kemudahan dalam menentukan jalur yang akan ditempuh dengan jarak terpendek sehingga dapat mengefisienkan waktu, tenaga, dan biaya.

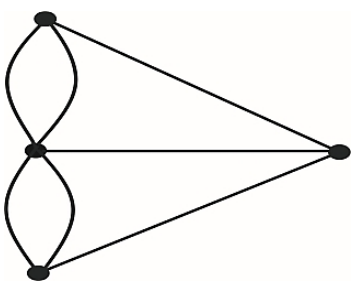
Pencarian rute terpendek merupakan masalah yang rumit. Salah satu masalah pencarian rute terpendek adalah mencari rute terpendek dari sejumlah objek wisata dan jarak antar objek wisata yang harus dilalui oleh wisatawan yang berangkat dari titik A dan menyinggahi setiap tempat objek wisata tepat satu kali dan kembali lagi ke titik A. Secara teoritis, apabila ada n objek wisata maka terdapat $n!$ (n faktorial) rute yang harus dicari. Misalkan terdapat n = 6 maka yang harus dicari sebanyak 720 rute dan apabila jumlah n = 30 maka rute yang harus dicari lebih dari 4 x 1030, oleh karena akan membutuhkan waktu yang sangat lama untuk pencarian rute tersebut.

Terdapat banyak algoritma yang digunakan untuk menentukan jalur terpendek, diantaranya algoritma Dijkstra, algoritma Best-First Search, dll. Adapun di tiap-tiap algoritma tersebut memiliki cara kerja yang berbeda-beda dalam menemukan solusi yang paling optimal. Algoritma Best-First Search bekerja menggunakan fungsi perkiraan heuristic yaitu dengan memprioritaskan pemeriksaan node-node yang berurut dan berada pada arah yang benar, karena hanya menggunakan fungsi heuristic tanpa memperhitungkan biaya untuk menuju suatu node, sehingga jalur yang ditemukan dengan algoritma ini kemungkinan adalah jalur terpendek, tetapi belum tentu jalur tersebut memiliki biaya terkecil (Anonymous, 2000). Dengan memanfaatkan algoritma Best-First Search akan diketahui rute terpendek beberapa objek wisata yang akan dikunjungi di Yogyakarta sehingga dapat membantu wisatawan dalam memilih tempat wisata.

Landasan Teori

Graf

Menurut catatan sejarah, masalah jembatan Konigsberg adalah masalah yang pertama kali menggunakan graf (tahun 1736). Di kota Konigsberg (sebelah timur Prussia, yang sekarang Jerman), sekarang bernama kota Kaliningrad, terdapat sungai Pregal yang mengitari pulau Kneiphof kemudian bercabang menjadi dua buah anak sungai. Ada tujuh jembatan yang menghubungkan daratan yang dibelah oleh sungai tersebut. Masalah jembatan Konigsberg yaitu apakah mungkin melalui ketujuh buah jembatan kota itu masing-masing tepat satu kali, dan kembali ke tempat semula? Tahun 1736, seorang matematikawan Swiss Leonhard Euler adalah orang pertama yang berhasil menemukan jawaban masalah itu dengan pembuktian sederhana. Ia memodelkan masalah ini dengan graf (Munir, 2010: 354).



Gambar 1 Model Graf Jembatan Konigsberg.

1. Definisi Graf
   Graf dapat didefinisian sebagai kumpulan simpul-simpul yang dihubungkan dengan garis. Simpul biasa dinyatakan dengan istilah *vertex* atau *node* atau titik dan garis biasa dinyatakan dengan istilah *edge* atau sisi atau busur.
   Suatu graf *G* terdiri dari dua himpunan yang berhingga, yaitu himpunan titik-titik tidak kosong ditulis dengan notasi *V(G)* dan himpunan garis-garis dinotasikan *E(G)* (Siang, 2009: 218).

2. Properti Graf
   Properti yang berkaitan dengan graf antara lain:
   a. Derajat (*degree*)
      Misalkan v adalah titik dalam suatu graf G. Derajat titik v yang dinotasikan $d(v)$ adalah jumlah garis yang berhubungan dengan titik v dan garis suatu *loop* dihitung dua kali. *Loop* adalah sisi ganda yang menghubungkan sebuah simpul dengan simpul itu sendiri. Derajat total G adalah jumlah derajat semua titik dalam G (Siang, 2009:236).
   b. Lintasan (*Path*)
      Lintasan dalam suatu graf *G* terdiri dari barisan *vertex* dan sisi yang silih berganti dalam bentuk $v_0, e_1, v_1, e_2, v_2, \cdots, e_{n-1}, v_{n-1}, e_n, v_n$ dimana setiap sisi e_i mengandung *vertex* v_{i-1} dan v_i. Jumlah n dari sisi-sisinya disebut panjang dari lintasan, dituliskan sebagai $(v_0, v_1, \cdots, v_n)$. Lintasan tersebut dikatakan tertutup jika $v_0 = v_n$ (Seymour dan Lipson, 2007:138). Suatu lintasan sederhana (*simple path*) adalah suatu lintasan dimana titiknya berbeda (setiap sisi yang dilalui hanya satu kali).
   c. Sirkuit (*circuit*) atau Siklus (*cycle*)
      Sirkuit dalam suatu graf adalah lintasan yang berbentuk $v_0 e_1 v_1 e_2 v_2 \cdots v_{n-1} e_n v_n$ (Siang, 2009: 242). Misalkan v ada dalam graf *G*, sebuah sirkuit adalah sebuah lintasan *(path*) dengan panjang bukan nol dari v ke v tanpa ada sisi yang berulang (Limbong, Prijono, 2006: 168). Suatu siklus (*cycle*) adalah suatu lintasan tertutup dengan panjang 3 atau lebih dimana setiap titiknya berbeda kecuali $v_0 = v_n$.
      Sirkuit sederhana merupakan sirkuit yang semua titiknya berbeda. Sirkuit sederhana berbentuk $v_0 e_1 v_1 e_2 v_2 \cdots v_{n-1} e_n v_n$ dengan $e_i \neq e_i$ untuk $i \neq j$ dan $v_k \neq v_m$ untuk $k \neq m$, kecuali $v_0 = v_n$ (Siang, 2006: 242).

3. Graf Tak Sederhana (*Multiple Graph*)
   Graf tak sederhana adalah graf yang mengandung *multiple edges* yang terhubung dengan *vertek* yg sama (Rosen, 1998: 590). Terdapat dua macam graf tak sederhana, yaitu Graf Ganda dan Graf Semu.
   Graf ganda adalah graf yang hanya mengandung sisi ganda. Sisi ganda adalah dua buah simpul atau lebih yang terhubung oleh dua atau lebih sisi. Sedangkan graf semu adalah graf yang mengandung *loop*.

4. Graf Tak Berarah (*Undirected Graph*)
   Graf tak berarah adalah graf yang sisinya tidak memiliki orientasi arah. Urutan pasangan simpul pada graf tak berarah yang dihubungkan oleh sisi tidak diperhatikan. Jadi sisi (v_1, v_2) sama dengan sisi (v_2, v_1).

5. Graf Terhubung (*Connected Graph*)
   Dua buah titik v_1 dan v_2 disebut terhubung jika terdapat sisi dari v_1 ke v_2. Graf G disebut graf terhubung (*connected graph*) jika untuk setiap pasang titik v_i dan v_j dalam himpunan V terdapat lintasan dari v_i ke v_j.

6. Graf Lengkap
   Adalah graf sederhana dengan n titik, dimana 2 titik berbeda dihubungkan dengan suatu garis (Siang, 2009: 227). Graf lengkap dengan n simpul dinotasikan dengan K_n. Setiap simpul pada K_n berderajat $n-1$. Banyaknya sisi dalam suatu graf lengkap dengan n titik adalah $\frac{n(n-1)}{2}$ buah.

7. Graf Berbobot (*Weighted Graph*)
   Adalah suatu graf dimana setiap sisinya memiliki nilai atau bobot berupa bilangan non negatif. Bobot pada tiap sisi dapat berbeda-beda bergantung pada masalah yang dimodelkan dengan graf. Bobot dapat menyatakan jarak antar dua buah kota atau tempat, biaya perjalanan antara dua buat kota, dan sebagianya.

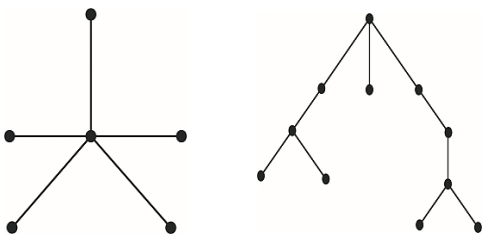
8. Sirkuit Hamilton
   Suatu graf terhubung G disebut *sirkuit* Hamilton apabila ada *sirkuit* yang mengunjungi setiap simpulnya tepat satu kali, kecuali simpul awal yang sama dengan simpul akhirnya (Siang, 2009: 251).

9. Lintasan Terpendek (*Shortest Path*)
   Persoalan mencari lintasan terpendek di dalam graf merupakan salah satu persoalan optimasi. Graf yang digunakan dalam pencarian lintasan terpendek adalah graf berbobot (*weighted graph*), yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Bobot pada sisi graf dapat dinyatakan sebagai jarak antar kota atau tempat, waktu pengiriman pesan atau barang dan lain-lain. Asumsi yang digunakan adalah bahwa semua bobot bernilai positif (Munir, 2010: 412).

   Dengan kata lain lintasan terpendek merupakan suatu jaringan atau kerangka jalur petunjuk perjalanan dari suatu simpul atau titik ke simpul lainnya atau yang menjadi tujuan perjalanan dengan beberapa pilihan jalur yang mungkin untuk dijalani. Tujuan dari permasalahan rute terpendek yaitu mencari rute yang memiliki jarak terdekat antara titik asal dan titik tujuan. Apabila jarak belum diketahui, jarak dapat dihitung berdasarkan koordinat tempat kemudian menghitung jarak terpendek yang dapat dilalui.

Pohon

1. Definisi Pohon
   Pohon (*tree*) adalah sebuah graf terhubung yang tidak mengandung *cycle* (*acyclic*). (Bondy dan Murty, 1976: 25) Definisi pohon dapat diilustrasikan pada gambar 3.1 sebagai berikut:



Gambar 2 Pohon.

2. Pohon Berakar
   Adalah suatu pohon dimana ada satu simpul yang diperlakukan sebagai akar dan sisi-sisinya diberi arah menjauh dari akar. Akar adalah simpul yang paling atas di dalam pohon. Sembarang pohon tak-berakar dapat diubah menjadi pohon berakar dengan memilih sebuah simpul sebagai akar.

3. Terminologi pada Pohon Berakar
   Terminologi yang berkaitan dengan pohon antara lain (Siang, 2009: 281):
   a. Tingkat (level) suatu titik adalah banyaknya baris antara titik-titik tersebut dengan akar.
   b. Anak (*child*), Orang tua (*parent*) dan Saudara (*sibling*)
      Anak dari simpul v adalah semua titik yang berhubungan langsung dengan v, tetapi memiliki tingkat yang lebih tinggi dari v. Jika w adalah anak dari v, maka v disebut orang tua dari w. Dua titik yang memiliki orang sama disebut saudara.
   c. Lintasan (*path*)
      Lintasan dari simpul v_1 ke simpul v_k adalah runtutan simpul-simpul $v_1, v_2, \cdots, v_k$ sedemikian sehingga v_i adalah orang tua dari v_{i+1} untuk $1 \leq i \leq k$.
   d. Daun (*leaf*)
      Daun adalah simpul yang berderajat nol (tidak memiliki anak).

Traveling Salesman Problem (TSP)

Traveling Salesman Problem (TSP) dapat didefinisikan sebagai pencarian urutan semua lokasi (misalnya kota) yang harus dikunjungi, mulai dari suatu kota tertentu dan kembali ke kota tersebut, dengan meminimalkan biaya. Setiap kota hanya diperbolehkan untuk dikunjungi satu kali (Suyanto, 2010:32). *Traveling Salesman Problem* (TSP) termasuk salah satu persoalan yang sulit dalam masalah optimasi. TSP dikemukakan pada tahun 1800 oleh William Rowan Hamilton dan Thomas Penyngton K, sedangkan bentuk umum TSP pertama dipelajari oleh matematikawan pada tahun 1930.

Persoalan *Traveling Salesman Problem* (TSP) termasuk ke dalam persoalan yang populer dalam teori graf. Persoalan ini terinspirasi oleh masalah seorang *salesman* yang akan mengunjungi n kota, perjalanan dimulai dan diakhiri pada satu kota dan harus mengunjungi $n-1$ kota yang lain dengan alternatif rute terpendek. Jaringan transportasi yang menghubungkan ke n kota tersebut adalah *completely connected,* artinya dari tiap kota terdapat jalur transportasi langsung ke $n-1$ kota lainnya tanpa melalui kota perantara lainnya. Dengan kata lain, permasalahan TSP merupakan permasalah di mana seorang *salesman* harus mengunjungi semua kota di mana tiap kota hanya dikunjungi sekali dan harus dimulai dan kembali ke kota asal. Tujuannya adalah menentukan rute dengan jarak total atau biaya yang paling minimum.

Traveling Salesman Problem (TSP) dapat diselesaikan dengan mencari semua sirkuit Hamilton yang mungkin kemudian dipilih sirkuit Hamilton dengan total jarak terpendek. Sirkuit Hamilton yang dapat dibentuk dari sebuah graf dengan simpul *n* adalah $(n-1)!$. Sebuah sirkuit Hamilton dapat dilewati dengan arah sebaliknya maka jumlah sirkuit Hamilton yang dibutuhkan adalah $\frac{(n-1)!}{2}$ sirkuit Hamilton (Rosen, 2007: 654).

Metode Pencarian Heuristik

Kata *heuristic* berasal dari sebuah kata kerja bahasa Yunani, yaitu *heuriskein*, yang berarti mencari atau menemukan. Metode heuristik adalah suatu metode yang menggunakan sistem pendekatan dalam melakukan pencarian optimasi. Dalam metode pencarian, kata heuristik diartikan sebagai suatu fungsi yang menghitung biaya perkiraan (estimasi) dari titik awal menuju titik tujuan (Suyanto, 2007: 22).

Heuristik mempunyai informasi tentang *cost*/ biaya untuk mencapai *goal state* dari *current state.* Dengan informasi tersebut, pencarian heuristik dapat melakukan pertimbangan untuk mengembangkan atau memeriksa *node-node* yang mengarah ke *goal state*. Misalnya pencarian rute pada suatu peta, bila kita berangkat dari kota *A* ke kota tujuan *B* yang letaknya di utara kota *A*, dengan pencarian heuristik, pencarian akan lebih difokuskan ke arah utara (dengan informasi *cost* ke *goal*), sehingga secara umum pencarian heuristik lebih efisien.

Heuristik merupakan sebuah teknik yang mengembangkan efisiensi dalam proses pencarian. Untuk dapat menerapkan heuristik tersebut dengan baik dalam suatu domain tertentu, diperlukan fungsi heuristik (suatu fungsi untuk menghitung nilai atau biaya perkiraan dari suatu solusi permasalahan yang dicari) sebagai modal untuk melakukan iterasi menuju *goal state*. Fungsi heuristik berbeda dengan algoritma, dimana heuristik lebih merupakan perkiraan untuk membantu algoritma dan tidak harus valid setiap waktu. Meskipun begitu, semakin bagus fungsi heuristik yang dipakai, semakin cepat dan akurat pula solusi yang diperoleh. Menentukan heuristik yang tepat untuk suatu kasus dan implementasi yang ada juga sangat berpengaruh terhadap kinerja algoritma pencarian.

Jenis-jenis pencarian heuristik yaitu: *Generate and Test*, *Hill Climbing*, *Best-First Search*, dan lain-lain, namun dalam penelitian ini penulis hanya akan membahas jenis pencarian *Best-First Search*.

Algoritma

1. Pengertian Algoritma

   Algoritma adalah urutan logis langkah-langkah penyelesaian masalah yang disusun secara sistematis (Munir, 2009: 176). Menurut *Kamus Besar Bahasa Indonesia* terbitan Widya Karya 2009, definisi kata algoritma (*algoritma*) adalah prosedur sistematis untuk memecahkan masalah matematis dalam langkah-langkah terbatas. Sebagai pembanding, algoritma adalah urutan langkah-langkah untuk memecahkan suatu masalah (Munir, 2011: 4). Dari ketiga definisi di atas dapat disimpulkan bahwa algoritma adalah urutan logis untuk menyelesaikan masalah yang disusun secara sistematis.

2. Algoritma *Best-First Search* (BeFS)
   Algoritma *Best-First Search* ini merupakan kombinasi dari algoritma *Depth-First Search* dan algoritma *Breadth-First Search* dengan mengambil kelebihan dari kedua algoritma tersebut. *Best-First Search* merupakan salah satu bagian dari tipe *informed search*.

3. Operasi pada Algoritma *Best-First Search* (BeFS)
   Algoritma BeFS menggunakan nilai heuristik pada tiap simpul-simpulnya. Nilai heuristik dalam penelitian ini merupakan nilai estimasi dari jarak antar dua titik. Simpul dengan nilai heuristik terbaik akan dibuka atau dikerjakan lebih dahulu. Nilai heuristik dikatakan terbaik artinya apabila nilai tersebut mendekati nilai sebenarnya. Diasumsikan bahwa dalam permasalahan pencarian rute terpendek, nilai yang dianggap baik apabila nilai tersebut memberikan hasil yang lebih kecil dari nilai yang lainnya karena konteks yang bibahas adalah jarak. Setelah simpul dengan nilai terbaik diperoleh, jika *goal state* belum ditemukan maka akan dilakukan pemeriksaan pada simpul berikutnya dengan nilai heuristik terbaik pada kedalaman yang sama. Simpul tersebut kemudian dibuka dan diperiksa apakah terdapat *goal state* pada cabang-cabangnya. Bila *goal state* belum ditemukan, akan dilakukan proses yang sama pada simpul berikutnya.

   Penentuan nilai terbaik dapat dilakukan menggunakan suatu operasi fungsi yang disebut fungsi heuristik yang akan mengestimasi seberapa baik dibangkitkannya setiap *node*. Fungsi heuristik dikatakan baik atau dapat diterima apabila nilai perkiraan yang dihasilkan tidak melebihi nilai sebenarnya. Semakin mendekati nilai sebenarnya, fungsi heuristik akam semakin optimal (Suyanto, 2007: 59). Heuristik yang dipakai dalam penelitian ini, yaitu heuristik jarak garis lurus yang dinotasikan h_{SLD} (Russel, 2010: 94). Fungsi heuristik yang digunakan dalam algoritma ini hanya memperhitungkan jarak untuk mencapai tujuan dan mengabaikan jarak jalan sebenarnya yang sudah ditempuh untuk sampai ke titik tujuan, sehingga sering disebut sebagai *Greedy Best-First Search*.

   Secara harfiah, *greedy* artinya rakus atau tamak (sifat yang berkonotasi negatif). Sesuai dengan arti tersebut, prinsip *greedy* adalah mengambil keputusan yang dianggap terbaik untuk saat itu saja yang diharapkan dapat memberikan solusi terbaik secara keseluruhan. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.

   Greedy Best-First Search akan mencoba untuk memperluas *node* yang terdekat dengan tujuan, dengan alasan bahwa akan menemukan kemungkinan solusi tercepat. Sehingga fungsi evaluasi *node* yang digunakan hanya fungsi heuristik, yang dinyatakan dengan (Russel, 2010: 94):

$$f(n) = h(n)$$

   dimana:
   $f(n)$ = fungsi evaluasi
   $h(n)$ = fungsi heuristi atau estimasi jarak terpendek dari *vertex* (*node*) n ke *vertex* (*node*) tujuan (heuristik).

4. Prosedur Algoritma *Greedy Best-First Search*
   Pada masing-masing langkah dari proses *Best-First Search*, algoritma menyeleksi *node* yang lebih layak untuk dibangkitkan atau dimunculkan sebagai kandidat solusi lebih jauh, hal ini dilakukan dengan menerapkan fungsi heuristik yang tepat untuk masing-masing permasalahan. Algoritma kemudian memperluas pemilihan *node* dengan menggunakan *rule* untuk membangkitkan penggantinya. Bila satu darinya adalah solusi yang diharapkan, maka algoritma keluar, bila tidak maka semua *node* yang baru ditambahkan pada himpunan *node* yang dibangkitkan lebih jauh, kemudian dilakukan pemilihan *node* yang lebih layak untuk dijadikan sebagai kandidat solusi dan dilakukan proses selanjutnya. Pada kondisi seperti ini titik yang lebih layak yang sebelumnya dihindari akan dieksploitasikan tetapi cabang yang sebelumnya akan disimpan dalam himpunan *node* yang dibangkitkan akan tetapi merupakan *node* yang tidak dieksploitasi. Pencarian dapat kembali pada *node* kapan saja ketika semua *node* yang diperoleh tidak lebih layak dari sebelumnya.

   Berikut ini adalah prosedur algoritma *Greedy Best-First Search* (Kusumadewi, 2003:42):
   a. Tempatkan *node* awal pada antrian *OPEN*.

b. Kerjakan langkah-langkah berikut hingga tujuan ditemukan atau sampai antrian *OPEN* sudah kosong:
   i. Ambil *node* terbaik dari *OPEN*.
   ii. Bangkitkan semua *successornya.*
   iii. Untuk tiap-tiap *successornya* kerjakan:
      - Jika *node* tersebut belum pernah dibangkitkan, evaluasi *node* tersebut dan masukkan ke *OPEN*.
      - Jika *node* tersebut sudah pernah dibangkitkan sebelumnya, ubah *parent* jika lintasan baru lebih menjajikan. Hapus *node* tersebut dari antrian *OPEN*.

Pembahasan

Pencarian Rute Perjalanan Wisata Menggunakan Algoritma *Best-First Search* (BeFS)

Berikut disajikan tabel untuk *starting point* dan hotel yang dapat dipilih oleh wisatawan dalam penentuan rute terpendek perjalanan wisata di kota Yogyakarta.

Table 1 Starting Point dan Hotel.

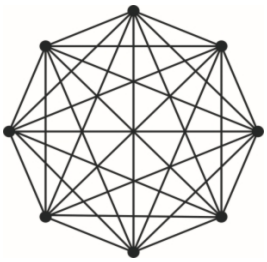
No	Paket Wisata	Titik start	Hotel
1	Premium	Bandara Internasional Adi Sucipto	N'dalem Gamelan
2	*Middle*	Stasiun Kereta Api Tugu	Mawar Asri
3	Ekonomis	Terminal Giwangan	Mitra

Tabel 2 Asumsi titik (Simpul) pada Graf.

Simpul	Nama Objek Wisata
A	Bandara Adisucipto Yogyakarta
A′	Stasiun Tugu Yogyakarta
A′′	Terminal Giwangan Yogyakarta
B	Kebun Binatang Gembira Loka
C	Taman Pintar
D	Keraton Yogyakarta
E	Tamansari
F	Purawisata
G	Museum Benteng Vredeburg
H	Ndalem Gamelan Hotel
H′	Hotel Mawar Asri
H′′	Hotel Mitra

Permasalahan rute terpendek perjalanan objek wisata di Kota Yogyakarta secara garis besar merupakan kasus TSP yang dapat direpresentasikan ke dalam suatu bentuk graf lengkap. Sebelum melakukan perhitungan akan ditentukan terlebih dahulu titik (simpul) lokasi yang dituju menggunakan aplikasi *Google Earth*. Berdasarkan data yang diperoleh dari *Google Earth* pada masing-masing simpul berikut disajikan tabel koordinat simpul lokasi yang akan dituju:

Kemudian titik-titik tersebut akan direpresentasikan ke dalam bentuk graf lengkap dengan 8 simpul yang mewakili 1 titik awal sekaligus titik akhir, 6 lokasi wisata dan 1 hotel, sedangkan sisi merupakan representasi dari jalan yang menghubungkan keduannya. Berdasarkan uraian tersebut diperoleh graf lengkap dengan 8 simpul sebagai berikut:



Gambar 2 Graf lengkap K_8.

1. Paket Wisata Premium
   Permasalahan rute terpendek dapat direpresentasikan ke dalam suatu graf lengkap dengan jumlah simpul sebanyak 8 yang merupakan representasi dari 1 titik awal sekaligus titik akhir yaitu Bandara Internasional Adisucipto Yogyakarta, 6 lokasi objek wisata dan Ndalem Gamelan hotel, sedangkan sisi merupakan representasi dari jalan yang menghubungkan keduannya.

   Dimisalkan simpul awal kedatangan adalah Bandara Internasional Adi Sucipto Yogyakarta, terdapat 7 kandidat lokasi wisata yang akan dikunjungi terlebih dahulu. Simpul (*node*) awal sekaligus akhir tujuan adalah *A*, maka nilai dari $A = (x; y) = (0,0)$ adalah *goal state*nya. Berikut nilai $h(n)$ atau estimasi jarak antara dua titik koordinat yang diperoleh dari penarikan garis lurus dalam *Google Earth*.

Table 3 Nilai $h(n)$ Paket Wisata Premium dalam satuan meter.

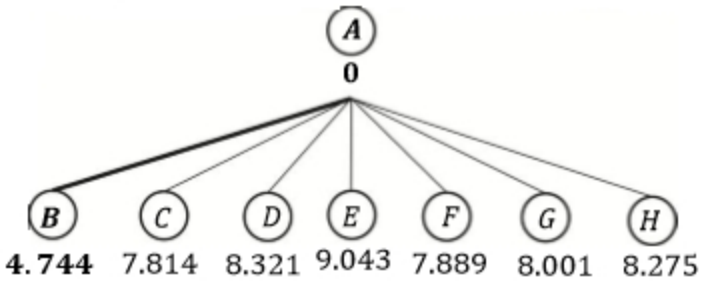
Simpul	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
A	0	4.744	7.814	8.321	9.043	7.889	8.001	8.275
B	4.744	0	3.323	3.719	4.375	3.198	3.530	3.569
C	7.814	3,323	0	634	1.424	853	211	1.130
D	8.321	3,719	634	0	795	661	579	677
E	9.043	4,375	1.424	795	0	1.173	1.333	854
F	7.889	3,198	853	661	1.173	0	964	421
G	8.001	3,530	211	579	1.333	964	0	1.175
H	8.275	3,569	1.130	677	854	421	1.175	0

Berdasarkan Tabel 3, berikut ini langkah-langkah untuk menyelesaikan permasalahan pencarian rute terpendek perjalanan objek wisata di Kota Yogyakarta dari simpul (*node*) *A* ke simpul tujuan dengan menggunakan algoritma *Best-First Search*:

Iterasi 1

Gambar 3 Simpul awal pencarian untuk Rute Wisata Premium.

State awal adalah simpul (*node*) *A* yang selanjutnya menjadi *OPEN*, karena di *OPEN* hanya terdapat satu simpul (*node*) maka *A* terpilih sebagai *bestnode* dan dipindahkan menjadi *CLOSED*. Kemudian semua kandidat (*successor*) A dibangkitkan, yaitu *B, C, D, E, F, G* dan *H*. Karena ketujuh *successor* tidak ada di *OPEN* maupun *CLOSED*, maka semuanya dimasukkan ke antrian *OPEN*. Langkah ini menghasilkan antrian *OPEN* = [*B, C, D, E, F, G*, H] dan *CLOSED* = [*A*].

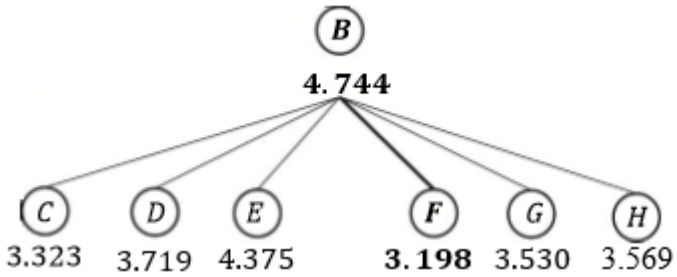


Gambar 4 Pohon pencarian iterasi 1 Rute Wisata Premium.

Simpul yang menempati urutan antrian terbaik adalah *B* (mempunyai nilai heuristik terkecil), yaitu 4.744 sehingga dipilih sebagai *bestnode*. Selanjutnya *B* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul *A* - simpul *B*. Langkah ini menghasilkan *CLOSED* = [*A, B*] dan antrian *OPEN* = [*C, D, E, F, G*, H].

Iterasi 2

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *B*, kemudian semua successornya dibangkitkan, yaitu *C, D, E, F, G*, H. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

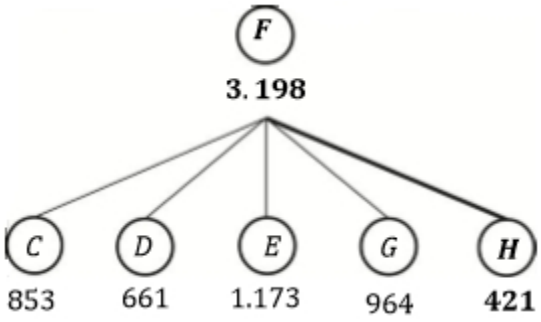


Gambar 5 Pohon pencarian iterasi 2 Rute Wisata Premium.

Simpul yang menempati urutan antrian terbaik adalah *F* (mempunyai nilai heuristik terkecil), yaitu 3.198 sehingga dipilih sebagai *bestnode*. Selanjutnya *F* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul *A* simpul *B* simpul *F*. Langkah ini menghasilkan *CLOSED* = [*A, B, F*] dan antrian *OPEN* = [*C, D, E, G*, H].

Iterasi 3

Diperoleh *CLOSED* baru, yaitu simpul (*node F,* kemudian semua successornya dibangkitkan, yaitu *C, D, E, G*, H. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.



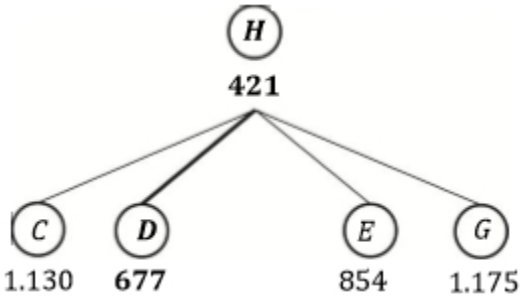
Gambar 6 Pohon pencarian iterasi 3 Rute Wisata Premium.

Simpul yang menempati urutan antrian terbaik adalah *H* (mempunyai nilai heuristik terkecil), yaitu 421 sehingga dipilih sebagai *bestnode*. Selanjutnya *H* dipindahkan ke *CLOSED* sehingga

didapatkan lintasan pohon pencarian yaitu simpul *A* - simpul *B* - simpul *F* - simpul *H*. Langkah ini menghasilkan *CLOSED* = [*A, B, F,* H] dan antrian *OPEN* = [*C, D, E, G*].

Iterasi 4

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *H*, kemudian semua successornya dibangkitkan, yaitu *C, D, E, G*. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

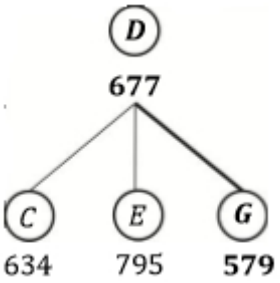


Gambar 7 Pohon pencarian iterasi 4 Rute Wisata Premium.

Simpul yang menempati urutan antrian terbaik adalah *D* (mempunyai nilai heuristik terkecil), yaitu 677 sehingga dipilih sebagai *bestnode*. Selanjutnya *D* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul *A* - simpul *B* - simpul *F* - simpul *H* - simpul *D*. Langkah ini menghasilkan *CLOSED* = [*A, B, F, H, D*] dan antrian *OPEN* = [*C, E, G*].

Iterasi 5

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *D*, kemudian semua successornya dibangkitkan, yaitu C, E, G. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

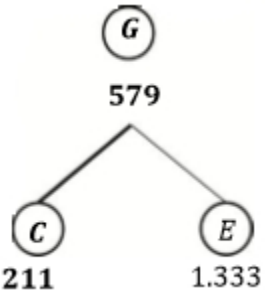


Gambar 8 Pohon pencarian iterasi 5 Rute Wisata Premium.

Simpul yang menempati urutan antrian terbaik adalah *G* (mempunyai nilai heuristik terkecil), yaitu 579 sehingga dipilih sebagai *bestnode*. Selanjutnya *G* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul *A* - simpul *B* - simpul *F* - simpul *H* - simpul *D* simpul *G*. Langkah ini menghasilkan *CLOSED* = [*A, B, F, H, D, G*] dan antrian *OPEN* = [*C, E*].

Iterasi 6

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *G* kemudian semua successornya dibangkitkan, yaitu *C, E*. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.



Gambar 8 Pohon Pencarian iterasi 6 Rute Wisata Premium.

Simpul yang menempati urutan antrian terbaik adalah *C* (mempunyai nilai heuristik terkecil), yaitu 211 sehingga dipilih sebagai *bestnode*. Selanjutnya *C* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul *A* - simpul *B* - simpul *F* - simpul *H* - simpul *D* - simpul *G* - simpul C. Langkah ini menghasilkan *CLOSED* = [*A, B, F, H, D, G, C*] dan antrian *OPEN* = [*E*].

Iterasi 7

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *C*, kemudian semua successornya dibangkitkan, yaitu *E*.

Gambar 9 Pohon pencarian iterasi 7 Rute Wisata Premium.

Karena *E* satu-satunya antrian *OPEN*, maka *E* dipilih sebagai node terbaik dengan nilai heuristik terkecil, yaitu 1.333. Langkah ini menghasilkan *OPEN* = [ ] dan *CLOSED* = [*A, B, F, H, D, G, C, E*].

Iterasi berhenti ketika seluruh simpul telah dikunjungi dan antrian *OPEN* sudah kosong, sehingga diperoleh solusi rute terbaik dari permasalahan perjalanan objek wisata di Kota Yogyakarta. Lintasan pohon pencarian yang dilalui untuk memperoleh solusi terbaik adalah simpul *A* - simpul *B* - simpul *F* - simpul *H* - simpul *D* - simpul *G* - simpul *C* - simpul E, atau dari Bandara Adi Sucipto Yogyakarta - Kebun Binatang Gembira Loka - Purawisata - Ndalem Gamelan Hotel - Keraton Yogyakarta - Museum Benteng Vredeburg - Taman Pintar - Tamansari - Bandara Adi Sucipto Yogyakarta. Jadi total jarak tempuh dari permasalahan pencarian rute wisata di kota Yogyakarta adalah 4.744 + 3.198 + 421 + 677 + 579 + 211 + 1.424 + 9.043 = 20.297 meter.

2. Paket Wisata *Middle*

Permasalahan rute terpendek dapat direpresentasikan ke dalam suatu graf lengkap dengan jumlah simpul sebanyak 8 yang merupakan representasi dari 1 titik awal sekaligus titik akhir yaitu Stasiun Tugu Yogyakarta, 6 lokasi objek wisata dan hotel Mawar Asri, sedangkan sisi merupakan representasi dari jalan yang menghubungkan keduannya.

Dimisalkan simpul awal kedatangan adalah Stasiun Tugu Yogyakarta, terdapat 7 kandidat lokasi wisata yang akan dikunjungi terlebih dahulu. Simpul (*node*) awal sekaligus akhir tujuan adalah A', maka nilai dari $A' = (x; y) = (0,0)$ adalah *goal state*nya. Berikut nilai $h(n)$ atau estimasi jarak antara dua titik koordinat yang diperoleh dari penarikan garis lurus dalam *Google Earth*.

Table 4 Nilai $h(n)$ Paket Wisata *Middle* dalam satuan meter.

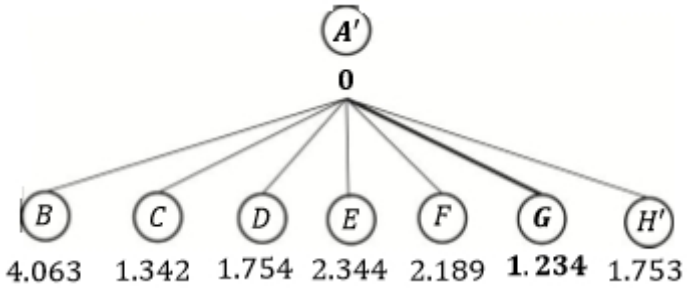
Simpul	*A′*	*B*	*C*	*D*	*E*	*F*	*G*	*H′*
A′	0	4.063	1.342	1.754	2.344	2.189	1.234	1.753
B	4.063	0	3.323	3.719	4.375	3.198	3.530	4.334
C	1.342	3.323	0	634	1.424	853	211	1.100
D	1.754	3.719	634	0	795	661	579	636
E	2.344	4.375	1.424	795	0	1.173	1.333	623
F	2.189	3.198	853	661	1.173	0	964	1.268
G	1.234	3.530	211	579	1.333	964	0	938
H′	1.753	4.334	1.100	636	623	1.268	938	0

Berdasarkan Tabel 4, berikut ini langkah-langkah untuk menyelesaikan permasalahan pencarian rute terpendek perjalanan objek wisata di Kota Yogyakarta dari simpul (*node*) A' ke simpul tujuan dengan menggunakan algoritma *Best-First Search*:

Iterasi 1

Gambar 10 Simpul awal pencarian untuk Rute Wisata *Middle*.

State awal adalah simpul (*node*) *A'* yang selanjutnya menjadi *OPEN*, karena di *OPEN* hanya terdapat satu simpul (*node*) maka *A'* terpilih sebagai *bestnode* dan dipindahkan menjadi *CLOSED*. Kemudian semua *successor A'* dibangkitkan, yaitu B, C, D, E, F, G dan H'. Karena ketujuh *successor* tidak ada di *OPEN* maupun *CLOSED*, maka semuanya dimasukkan ke antrian *OPEN*. Langkah ini menghasilkan antrian *OPEN* = $[B, C, D, E, F, G, H']$ dan *CLOSED* = $[A']$.

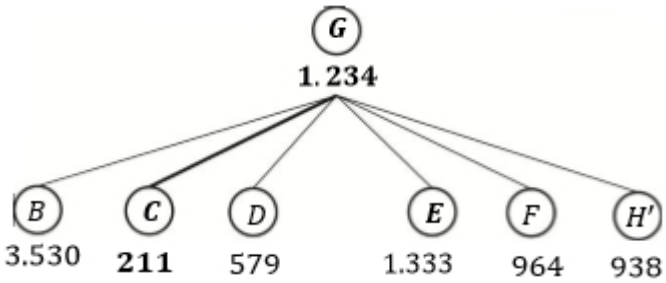


Gambar 11 Pohon pencarian iterasi 1 Rute Wisata *Middle*.

Simpul yang menempati urutan antrian terbaik adalah *G* (mempunyai nilai heuristik terkecil), yaitu 1.234 sehingga dipilih sebagai *bestnode*. Selanjutnya *G* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A' - simpul *G*. Langkah ini menghasilkan *CLOSED* = $[A', G]$ dan antrian *OPEN* = $[B, C, D, E, F, H']$.

Iterasi 2

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *G*, kemudian semua successornya dibangkitkan, yaitu B, C, D, E, F, H'. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

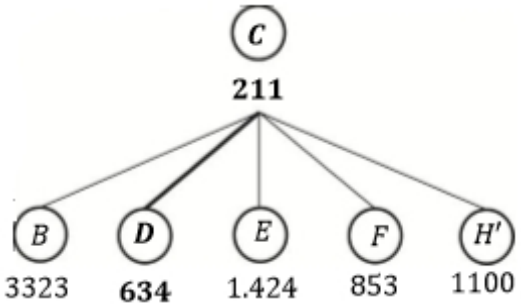


Gambar 12 Pohon pencarian iterasi 2 Rute Wisata *Middle*.

Simpul yang menempati urutan antrian terbaik adalah *C* (mempunyai nilai heuristik terkecil), yaitu 211 sehingga dipilih sebagai *bestnode*. Selanjutnya *C* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A' - simpul G - simpul *C*. Langkah ini menghasilkan *CLOSED* = $[A', G, C]$ dan antrian *OPEN* = $[B, D, E, F, H']$.

Iterasi 3

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *C,* kemudian semua successornya dibangkitkan, yaitu B, D, E, F, H'. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

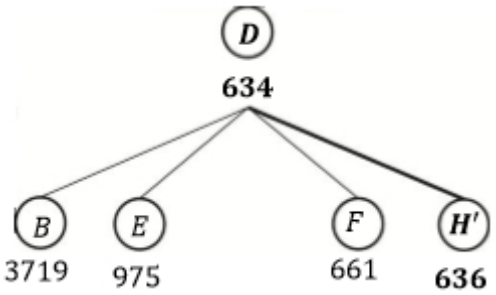


Gambar 13 Pohon pencarian iterasi 3 Rute Wisata *Middle*.

Simpul yang menempati urutan antrian terbaik adalah *D,* (mempunyai nilai heuristik terkecil), yaitu 634 sehingga dipilih sebagai *bestnode*. Selanjutnya D dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A' - simpul G - simpul C - simpul *D.* Langkah ini menghasilkan *CLOSED* = $[A', G, C, D]$ dan antrian *OPEN* = $[B, E, F, H']$.

Iterasi 4

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *D,* kemudian semua successornya dibangkitkan, yaitu B, E, F, H'. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

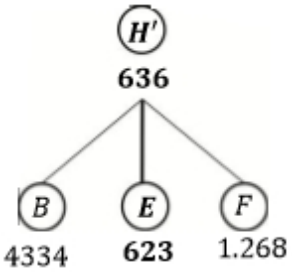


Gambar 14 Pohon pencarian iterasi 4 Rute Wisata *Middle*.

Simpul yang menempati urutan antrian terbaik adalah H' (mempunyai nilai heuristik terkecil), yaitu 636 sehingga dipilih sebagai *bestnode*. Selanjutnya H' dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A' - simpul G - simpul C - simpul D - simpul H'. Langkah ini menghasilkan *CLOSED* = $[A', G, C, D, H']$ dan antrian *OPEN* = $[B, E, F]$.

Iterasi 5

Diperoleh *CLOSED* baru, yaitu simpul (*node*) H', kemudian semua successornya dibangkitkan, yaitu B, E, F. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

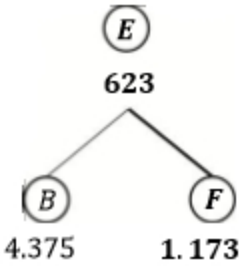


Gambar 15 Pohon pencarian iterasi 5 Rute Wisata *Middle*.

Simpul yang menempati urutan antrian terbaik adalah E (mempunyai nilai heuristik terkecil), yaitu 623 sehingga dipilih sebagai *bestnode*. Selanjutnya E dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A' - simpul G - simpul C - simpul D - simpul H' - simpul E. Langkah ini menghasilkan *CLOSED* = $[A', G, C, D, H', E]$ dan antrian *OPEN* = $[B, F]$.

Iterasi 6

Diperoleh *CLOSED* baru, yaitu simpul (*node*) E, kemudian semua successornya dibangkitkan, yaitu B, F. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.



Gambar 16 Pohon pencarian iterasi 6 Rute Wisata *Middle*.

Simpul yang menempati urutan antrian terbaik adalah *F* (mempunyai nilai heuristik terkecil), yaitu 1.173 sehingga dipilih sebagai *bestnode*. Selanjutnya *F* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A' - simpul G - simpul C - simpul D - simpul H' - simpul *E* - simpul *F.* Langkah ini menghasilkan *CLOSED* = $[A', G, C, D, H', E, F]$ dan antrian *OPEN* = $[B]$.

Iterasi 7

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *F,* kemudian successornya dibangkitkan, yaitu *B.*

Gambar 17 Pohon pencarian iterasi 7 Rute Wisata *Middle*.

Karena *B* satu-satunya antrian *OPEN*, maka *B* dipilih sebagai node terbaik dengan nilai heuristik terkecil, yaitu 3.198. Langkah ini menghasilkan *OPEN* = [ ] dan *CLOSED* = $[A', G, C, D, H', E, F, B]$

Iterasi berhenti ketika seluruh simpul telah dikunjungi dan antrian *OPEN* sudah kosong, sehingga diperoleh solusi rute terbaik dari permasalahan perjalanan objek wisata di Kota Yogyakarta. Lintasan pohon pencarian yang dilalui untuk memperoleh solusi terbaik adalah simpul A' simpul G simpul C - simpul D - simpul H' - simpul E - simpul F - simpul B atau dari Stasiun Tugu Yogyakarta Museum Benteng Vredeburg Taman Pintar Keraton Yogyakarta Hotel Mawar Asri Tamansari Purawisata Kebun Binatang Gembira Loka Stasiun Tugu Yogyakarta. Jadi total jarak tempuh dari permasalahan pencarian rute wisata di kota Yogyakarta adalah 1.234 + 211 + 634 + 636 + 623 + 1.173 + 3.198 + 4.063 = 11.772 meter.

3. Paket Wisata Ekonomis

Permasalahan rute terpendek dapat direpresentasikan ke dalam suatu graf lengkap dengan jumlah simpul sebanyak 8 yang merupakan representasi dari 1 titik awal sekaligus titik akhir yaitu Terminal Giwangan Yogyakarta, 6 lokasi objek wisata dan Hotel Mitra, sedangkan sisi merupakan representasi dari jalan yang menghubungkan keduannya.

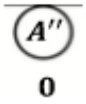
Dimisalkan simpul awal kedatangan adalah Terminal Giwangan Yogyakarta, terdapat 7 kandidat lokasi wisata yang akan dikunjungi terlebih dahulu. Simpul (*node*) awal sekaligus akhir tujuan adalah A'', maka nilai dari $A'' = (x; y) = (0,0)$ adalah *goal state*nya. Berikut nilai $h(n)$ atau estimasi jarak antara dua titik koordinat yang diperoleh dari penarikan garis lurus dalam *Google Earth*.

Table 5 Nilai h(n) Paket Wisata Ekonomis dalam satuan meter.

Simpul	A''	B	C	D	E	F	G	H''
A''	0	3.539	4.541	4.406	4.489	3.780	4.641	4.760
B	3.539	0	3.323	3.719	4.375	3.198	3.530	3.766
C	4.541	3.323	0	634	1.424	853	211	459
D	4.406	3.719	634	0	795	661	579	417
E	4.489	4.375	1.424	795	0	1.173	1.333	1.086
F	3.780	3.198	853	661	1.173	0	964	979
G	4.641	3.530	211	579	1.333	964	0	280
H''	4.760	3.766	459	417	1.086	979	280	0

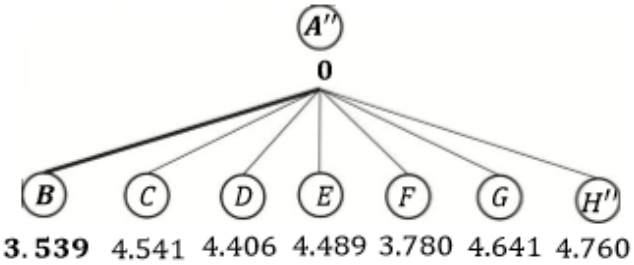
Berdasarkan Tabel 5, berikut ini langkah-langkah untuk menyelesaikan permasalahan pencarian rute terpendek perjalanan objek wisata di Kota Yogyakarta dari simpul (*node*) A'' ke simpul tujuan dengan menggunakan algoritma *Best-First Search*:

Iterasi 1



Gambar 18 Simpul awal pencarian Rute Perjalanan Wisata Ekonomis.

State awal adalah simpul (*node*) A'' yang selanjutnya menjadi *OPEN*, karena di *OPEN* hanya terdapat satu simpul (*node*) maka A'' terpilih sebagai *bestnode* dan dipindahkan menjadi *CLOSED*. Kemudian semua *successor* A'' dibangkitkan, yaitu B, C, D, E, F, G dan H''. Karena ketujuh *successor* tidak ada di *OPEN* maupun *CLOSED*, maka semuanya dimasukkan ke antrian *OPEN*. Langkah ini menghasilkan antrian *OPEN* = $[B, C, D, E, F, G, H'']$ dan *CLOSED* = $[A'']$.



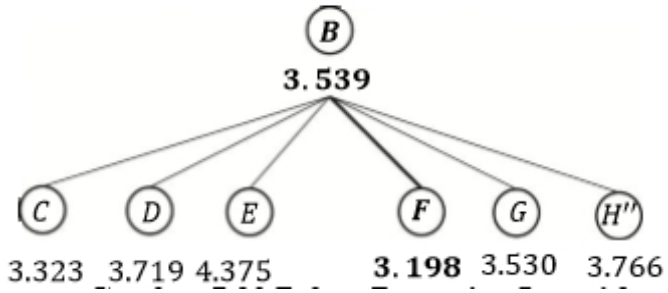
Gambar 19 Pohon pencarian iterasi 1 Rute Wisata Ekonomis.

Simpul yang menempati urutan antrian terbaik adalah B (mempunyai nilai heuristik terkecil), yaitu 3.539 sehingga dipilih sebagai *bestnode*. Selanjutnya B dipindahkan ke *CLOSED* sehingga

didapatkan lintasan pohon pencarian yaitu simpul A'' - simpul B. Langkah ini menghasilkan *CLOSED* = $[A'', B]$ dan antrian *OPEN* = $[C, D, E, F, G, H'']$.

Iterasi 2

Diperoleh *CLOSED* baru, yaitu simpul (*node*) B, kemudian semua successornya dibangkitkan, yaitu C, D, E, F, G, H''. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

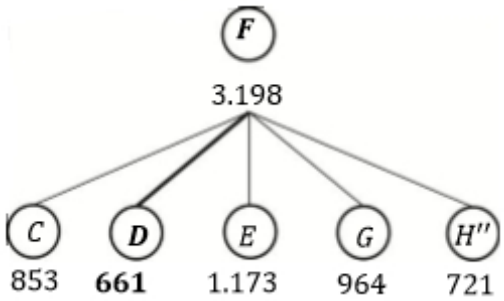


Gambar 20 Pohon pencarian Iterasi 2 Rute Wisata Ekonomis.

Simpul yang menempati urutan antrian terbaik adalah *F*, (mempunyai nilai heuristik terkecil), yaitu 3.198 sehingga dipilih sebagai *bestnode*. Selanjutnya F dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A'' - simpul B - simpul F. Langkah ini menghasilkan *CLOSED* = $[A'', B, F]$ dan antrian *OPEN* = $[C, D, E, G, H'']$.

Iterasi 3

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *F*, kemudian semua successornya dibangkitkan, yaitu C, D, E, G, H''. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

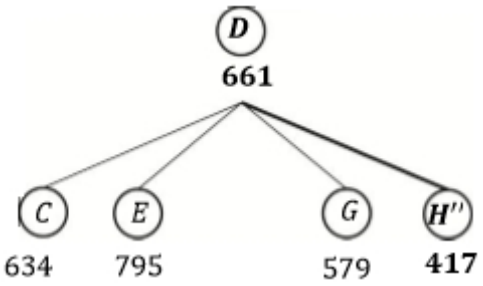


Gambar 21 Pohon pencarian iterasi 3 Rute Wisata Ekonomis.

Simpul yang menempati urutan antrian terbaik adalah *D* (mempunyai nilai heuristik terkecil), yaitu 661 sehingga dipilih sebagai *bestnode*. Selanjutnya *D* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A'' - simpul B - simpul F - simpul D. Langkah ini menghasilkan *CLOSED* = $[A'', B, F, D]$ dan antrian *OPEN* = $[C, E, G, H'']$.

Iterasi 4

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *D*, kemudian semua successornya dibangkitkan, yaitu C, E, G, H''. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

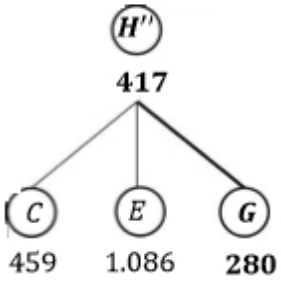


Gambar 22 Pohon pencarian iterasi 4 Rute Wisata Ekonomis.

Simpul yang menempati urutan antrian terbaik adalah H'' (mempunyai nilai heuristik terkecil), yaitu 417 sehingga dipilih sebagai *bestnode*. Selanjutnya C dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A - simpul B - simpul F - simpul D - simpul H''. Langkah ini menghasilkan *CLOSED* = $[A, B, F, D, H'']$ dan antrian *OPEN* = $[C, E, G]$.

Iterasi 5

Diperoleh *CLOSED* baru, yaitu simpul (*node*) H'', kemudian semua successornya dibangkitkan, yaitu C, E, G. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.

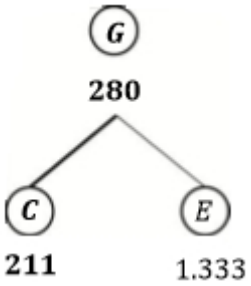


Gambar 23 Pohon pencarian iterasi 5 Rute Wisata Ekonomis.

Simpul yang menempati urutan antrian terbaik adalah G (mempunyai nilai heuristik terkecil), yaitu 280 sehingga dipilih sebagai *bestnode*. Selanjutnya G dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A'' - simpul B - simpul F - simpul D - simpul H'' - simpul G. Langkah ini menghasilkan *CLOSED* = $[A'', B, F, D, H'', G]$ dan antrian *OPEN* = $[C, E]$.

Iterasi 6

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *G*, kemudian semua successornya dibangkitkan, yaitu *C, E*. Karena semuanya sudah berada di antrian *OPEN*, maka akan dipilih simpul (*node*) dengan nilai heuristik terbaik.



Gambar 24 Pohon pencarian iterasi 6 Rute Wisata Ekonomis.

Simpul yang menempati urutan antrian terbaik adalah C (mempunyai nilai heuristik terkecil), yaitu 211 sehingga dipilih sebagai *bestnode*. Selanjutnya *C* dipindahkan ke *CLOSED* sehingga didapatkan lintasan pohon pencarian yaitu simpul A'' - simpul *B* - simpul *F* - simpul *D* simpul H'' - simpul *G* - simpul *C*. Langkah ini menghasilkan *CLOSED* = $[A'', B, F, D, H'', G, C]$ dan antrian *OPEN* = $[E]$.

Iterasi 7

Diperoleh *CLOSED* baru, yaitu simpul (*node*) *C*, kemudian successornya dibangkitkan, yaitu *E*.

Gambar 25 Pohon pencarian iterasi 7 Rute Wisata Ekonomis.

Karena *E* satu-satunya antrian *OPEN*, maka *E* dipilih sebagai node terbaik dengan nilai heuristik terkecil, yaitu 1.424. Langkah ini menghasilkan *OPEN* = [ ] dan *CLOSED* = $[A'', B, F, D, H'', G, C, E]$

Iterasi berhenti ketika seluruh simpul telah dikunjungi dan antrian *OPEN* sudah kosong, sehingga diperoleh solusi rute terbaik dari permasalahan perjalanan objek wisata di Kota Yogyakarta. Lintasan pohon pencarian yang dilalui untuk memperoleh solusi terbaik adalah simpul A'' - simpul *B* simpul *F* - simpul *D* - simpul H'' - simpul *G* simpul *C* simpul *E*. karena masalah pencarian rute objek wisata di Kota Yogyakarta menggunakan sirkuit Hamilton sehingga rute akan kembali ke titik awal keberangkatan. Oleh karena itu diperoleh rute optimum perjalanan objek wisata di Kota Yogyakarta, yaitu simpul A'' - simpul B - simpul F - simpul D - simpul H''- simpul G - simpul C - simpul E - simpul A'' atau dari Terminal Giwangan Yogyakarta - Kebun Binatang Gembira Loka - Purawisata Keraton Yogyakarta Hotel Mitra - Museum Benteng Vredeburg - Taman Pintar - Tamansari - Terminal Giwangan Yogyakarta. Jadi total jarak tempuh dari permasalahan pencarian rute wisata di Kota Yogyakarta adalah 3.539 + 3.198 + 661 + 417 + 280 + 211 + 1.242 + 4.489 = 14.037 meter.

Kesimpulan

Penelitian ini memberikan kesimpulan sebagai berikut:

Langkah untuk mencari rute terbaik perjalanan wisata di Kota Yogyakarta menggunakan Algoritma *Best-First Search* secara manual dimulai dengan merepresentasikan permasalahan ke dalam bentuk graf lengkap, kemudian dicari nilai $h(n)$ dari setiap simpul ke simpul lainnya. Selanjutnya perjalanan dimulai dari simpul awal hingga seluruh simpul terlewati dan kembali ke simpul awal. Pencarian rute wisata di Kota Yogyakarta dibuat menjadi tiga paket wisata dengan berbagai kelas sebagai berikut:

i. Paket Wisata Premium
   Berdasarkan langkah-langkah dari Algoritma *Best-First Search* diperoleh rute terbaik perjalanan wisata di Kota Yogyakarta dengan Paket wisata Premium dimulai dari Bandara Adi Sucipto Yogyakarta - Kebun Binatang Gembira Loka - Purawisata - Ndalem Gamelan Hotel - Keraton Yogyakarta - Museum Benteng Vredeburg - Taman Pintar - Tamansari - Bandara Adi Sucipto dengan total jarak yang ditempuh adalah 20.297 meter.
ii. Paket Wisata *Middle*
   Berdasarkan langkah-langkah dari Algoritma *Best-First Search* diperoleh rute terbaik perjalanan wisata di Kota Yogyakarta dengan Paket wisata *Middle* dimulai dari Stasiun Tugu Yogyakarta - Museum Benteng Vredeburg - Taman Pintar - Keraton Yogyakarta - Hotel Mawar Asri - Tamansari

- Purawisata - Kebun Binatang Gembira Loka - Stasiun Tugu Yogyakarta dengan total jarak tempuh adalah 11.772 meter.

iii. Paket Wisata Ekonomis

Berdasarkan langkah-langkah dari Algoritma *Best-First Search* diperoleh rute terbaik perjalanan wisata di Kota Yogyakarta dengan Paket wisata Ekonomis dimulai dari Terminal Giwangan Yogyakarta - Kebun Binatang Gembira Loka - Purawisata - Keraton Yogyakarta - Hotel Mitra - Museum Benteng Vredeburg - Taman Pintar - Tamansari - Terminal Giwangan Yogyakarta dengan total jarak tempuh adalah 14.037 meter.

Referensi

[1] **Bondy** and **Murty**. 1976. *Graph Theory with Aplications.* New York: North-Holland

[2] **Kusumadewi**, **Sri**. 2003. *Artifical Intelligence (Teknik dan Aplikasinya)*. Yogyakarta: Graha Ilmu.

[3] **Limbong** dan **Prijono**. 2006. *Matematika Diskrit*. Bandung: CV. UTOMO.

[4] **Lipschutz**, **Seymour** and **Lipson**, Marc. 2007. *Theory and Problems of Discrete Mathematics.* New York: McGraw-Hill

[5] **Munir**, **Rinaldi**. 2010. *Matematika Diskrit*. Bandung: Informatiaka Bandung.

[6] **Munir**, **Rinaldi**. 2011. *Algoritma dan Pemrograman*. Bandung: Informatiaka Bandung.

[7] **Rosen**, **Kenneth H**. 1988. *Discrete mathematics and its aplications*. New York: McGraw Hill.

[8] **Russell**, **Struat** and **Norvig**, **Peter**. 2010. *Artificial Intelligence A Modern Approach*. Boston: Pearson.

[9] **Siang**, **Jong Jek**. 2009. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer*. Yogyakarta: ANDI.

[10] **Suyanto**. 2007. *Artificial Intelligence: Searching, Reasoning, dan Learning*. Bandung: Informatika.

[11] **Wibisono**, **Samuel**. 2008. *Matematika Diskrit*. Yogyakarta: Graha Ilmu.