Penggunaan Algoritma A-Star untuk Menentukan Rute Tercepat di dalam Kampus Ganesha ITB

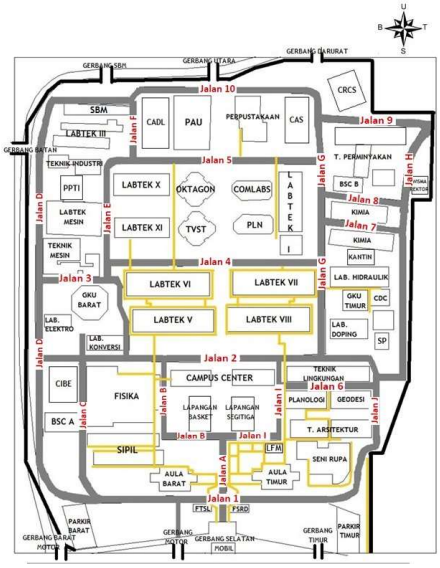
Fabian Savero Diaz Pranoto/13519140
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13519140@std.stei.itb.ac.id

***Abstract*—Persoalan yang sering muncul bagi mahasiswa baru ITB yang sedang berkuliah di dalam Kampus Ganesha adalah pencarian rute tercepat antara satu bangunan dengan bangunan lain untuk menghemat waktu dan tenaga. Persoalan ini bisa diselesaikan dengan menggunakan algoritma A-Star dengan peta setiap bangunan dan jalan di Kampus Ganesha sebagai sebuah simpul di dalam graf. Dengan menggunakan algoritma ini, rute paling efisien dapat diketahui.**

***Keywords*—Rute, Kampus, Graf, A-Star.**

I. PENDAHULUAN

Kampus Ganesha merupakan kampus utama Institut Teknologi Bandung yang berlokasi di Jl. Ganeca 10, Bandung, Jawa Barat, Indonesia. Kampus ini memiliki luas 286.830 m^2 atau sekitar 28 hektar. Setiap tahun, ITB menerima sekitar 6.500 mahasiswa baru. Mahasiswa baru yang memasuki Kampus Ganesha kerap kebingungan dalam mencari rute terpendek dan tercepat untuk mencapai tujuannya. Hal ini disebabkan oleh banyaknya bangunan di kampus yang relatif kecil serta banyaknya jalan yang memiliki tujuan akhir yang sama. Oleh karena itu, mahasiswa baru sering mengambil rute yang lebih panjang dari seharusnya dan menghabiskan waktu dan tenaga mereka untuk jalan kaki yang lebih lama. Ini tentu dapat berakibat negatif terhadap tingkat konsentrasi mahasiswa di dalam kelas yang tidak optimal karena kelelahan.



Gambar 1 : Peta Kampus Ganesha

Sumber : Institut Teknologi Bandung

Salah satu solusi terhadap permasalahan ini adalah membuat sebuah program yang dapat menentukan rute tercepat dari satu titik ke titik lain. Penulis membuat program sederhana berbasis Python yang menerapkan ilmu Graf yang menggunakan bangunan dan jalan-jalan yang ada di Kampus Ganesha sebagai simpul di dalam sebuah graf. Dengan menerapkan ilmu-ilmu graf, permasalahan ini dapat terselesaikan.

II. LANDASAN TEORI

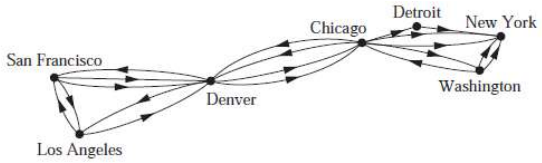
A. Definisi Graf

Graf adalah sebuah struktur yang terdiri dari simpul yang melambangkan objek yang dihubungkan oleh sisi yang menggambarkan relasi antar objek.

Definisi sebuah graf dari segi matematis sebagai berikut:

$$G = (V, E)$$

V = Himpunan tidak kosong berisi simpul $\{v_1, v_2, v_3, \ldots, v_n\}$
E = Himpunan tidak kosong berisi sisi $\{e_1, e_2, \ldots, e_n\}$



Gambar 2 : Contoh Graf
Sumber : Discrete Mathematics and Its Applications

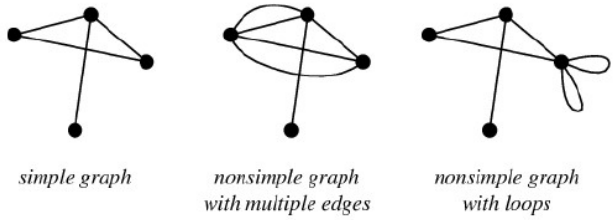
B. Jenis-Jenis Graf

Berdasarkan adanya sisi ganda pada graf:

1. Graf Sederhana
   Graf sederhana adalah sebuah graf yang setiap sisi menguhubungkan dua simpul dan tidak ada dua sisi yang berbeda yang menghubungkan sepasang simpul yang sama.

2. Graf Tak-Sederhana
   Graf Tak-Sederhana adalah graf yang mengandung sisi ganda dan/atau sisi gelang. Graf Tak-Sederhana dapat dibedakan menjadi:

a. Graf Ganda
   Graf yang mengandung sisi ganda.
b. Graf Semu
   Graf yang mengandung sisi gelang.

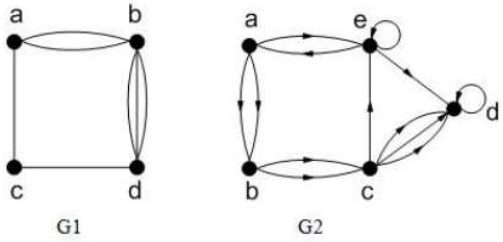


Gambar 3 : Graf Sederhana dan Tak-Sederhana
Sumber : Bahan Kuliah Matematika Diskrit

Berdasarkan orientasi arah sisi pada graf:

1. Graf Berarah
   Graf Berarah adalah graf yang sisi-sisinya menunjukkan arah.

2. Graf Tak-Berarah
   Graf Tak-Berarah adalah graf yang sisi-sisinya tidak menunjukkan arah.



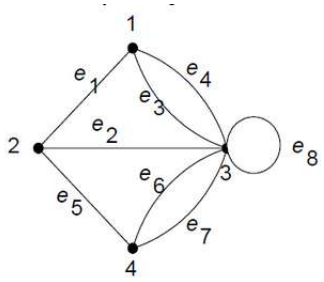
G1 : graf tak-berarah; G2 : Graf berarah

Gambar 4 : Graf Tak-Berarah dan Graf Berarah
Sumber : Bahan Kuliah Matematika Diskrit

C. Terminologi Graf

1. Ketetanggaan (*Adjacency)*
   Misalkan ada dua simpul *u* dan *v* di dalam sebuah graf. Kedua simpul tersebut dikatakan bertetangga jika sama-sama dihubungkan oleh sebuah sisi *e*.

2. Bersisian (*Incidency)*
   Sebuah sisi *e* dikatakan bersisian dengan dua simpul *u* dan *v* apabila kedua simpul tersebut dihubungkan oleh sisi *e*.



Gambar 5 : Ketetanggaan dan Bersisian
Sumber :
https://lmsspada.kemdikbud.go.id/mod/resource/view.php?id=47638

Dilihat dari gambar 5, simpul 1 dan 2 saling bertetangga karena dihubungkan oleh sisi e_1. Sisi e_1 bersisian dengan simpul 1 dan 2 karena menguhubungkan kedua simpul.

3. Simpul Terpencil (*Isolated Vertex)*
   Simpul Terpencil adalah sebuah simpul yang tidak

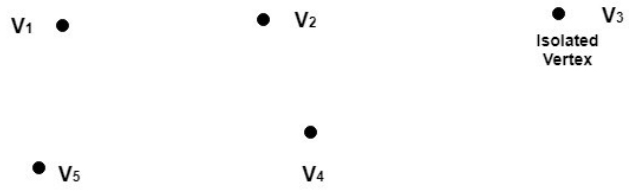


Fig:Null Graph

terhubung dengan simpul lain pada sebuah graf.

4. Graf Kosong (*Null Graph*)
   Graf Kosong adalah graf yang tidak memiliki sisi.

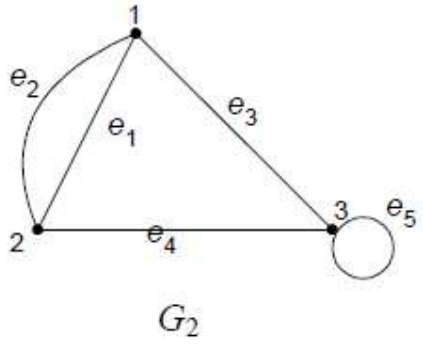
Gambar 6 : Graf Kosong dan Simpul Terpencil
Sumber : https://www.javatpoint.com/types-of-graphs

Graf di atas merupakan sebuah graf kosong karena tidak memiliki sisi dan setiap simpul merupakan sebuah simpul terpencil karena tidak terhubung dengan sisi lain.

5. Derajat
   Derajat sebuah simpul adalah merupakan jumlah sisi yang bersisian dengan simpul tersebut. Sebuah simpul gelang berkontribusi kepada derajat sebanyak 2. Pada gambar 4, simpul 1 memiliki derajat 3 karena bersisian dengan e_1, e_3, dan e_4.

6. Lintasan
   Lintasan merupakan barisan yang terdiri dari simpul dan sisi dengan bentuk v_0, e_1, v_1, e_2, v_2, ..., v_{n-1}, e_n, e_{n-}

$_1$. Ini menunjukkan jalan yang ditempuh dari sebuah simpul v_0 menuju simpul v_n.



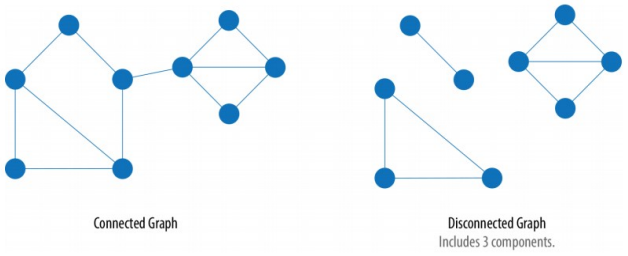
Gambar 7 : Graf G2
Sumber : Bahan Kuliah Matematika Diskrit

Pada gambar 7, lintasan 1,2,3 memiliki bentuk 1, e_1, 2, e_4, 3.

7. Sirkuit
Sirkuit merupakan lintasan yang memiliki simpul awal dan akhir yang sama. Pada gambar 5, lintasan 1, e_1, 2, e_4, 3, e_3, 1 merupakan sebuah sirkuit.

8. Keterhubungan
Sebuah simpul v_1 dan v_2 dikatakan terhubung apabila terdapat lintasan yang berawal dari v_1 dan berakhir di v_2. Sebuah graf dikatakan graf terhubung jika untuk setiap pasang simpul terdapat sebuah lintasan yang mengubungkan kedua simpul. Kebalikan dari graf terhubung adalah graf tak-terhubung.

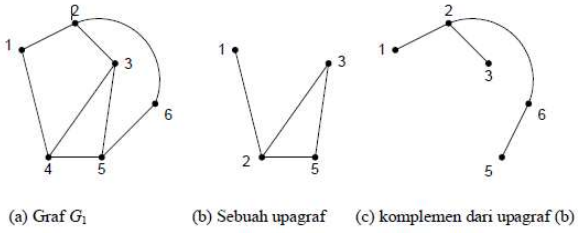
Pada sebuah graf berarah, terdapat istilah terhubung kuat dan terhubung lemah. Sepasang simpul u dan v pada graf berarah disebut terhubung kuat jika ada lintasan dari u ke v dan ada lintasan dari v ke u. Jika kedua simpul terhubung pada graf tak-berarahnya namun tidak terhubung kuat, kedua simpul dikatakan terhubung lemah. Sebuah graf berarah dikatakan graf terhubung kuat jika setiap pasang simpul terhubung kuat. Jika tidak memenuhi kondisi ini, dikatakan graf terhubung lemah.



Gambar 8 : Graf Terhubung dan Tak-Terhubung
Sumber : https://m.steveclarkapps.com/graphs/

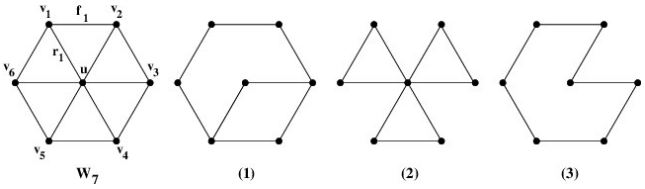
9. Upagraf (*Subgraph)*

Pada sebuah graf $G = V.E$, $G1 = V_1.E_1$ adalah upagraf dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$. Komplemen dari sebuah upgraf G1 adalah $G2 = V_2.E_2$, dimana $E_2 = E - E_1$.



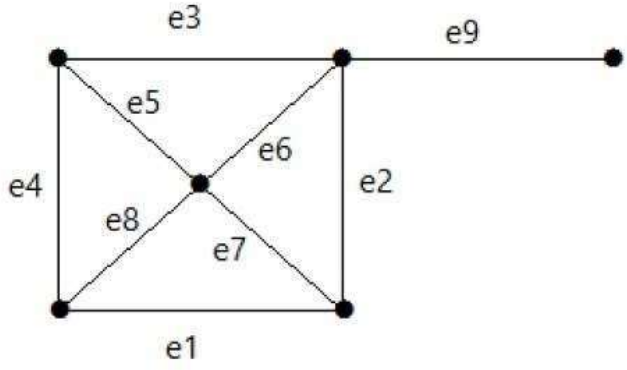
Gambar 9 : Upgraf
Sumber : Bahan Kuliah Matematika Diskrit

10. Upagraf Merentang (*Spanning Subgraph)*
Sebuah upgraf $G1 = V_1.E_1$ dapat dikatakan upgraf merentang dari graf $G = V.E$ jika $V_1 = V$. Jadi, upgraf merentang adalah upagraf yang mengandung seluruh simpul dari graf asalnya.



Gambar 10 : Upgraf Merentang
Sumber : https://www.researchgate.net

11. *Cut-Set*
Misalkan $G = V.E$ adalah sebuah graf terhubung, E' yang merupakan himpunan bagian dari E dikatakan *Cut-Set* apabila penghapusan E' dari G



menyebabkan graf menjadi tidak terhubung.
Gambar 11 : Cut-Set
Sumber : https://www.tutorialspoint.com/

Dari gambar 11, *Cut-Set* dari graf tersebut adalah {e3, e4, e5}. *Cut-Set* sebuah graf dapat lebih dari satu.

12. Graf Berbobot (*Weighted Graph)*
Graf berbobot adalah graf yang setiap sisinya diberi

sebuah nilai/bobot.

D. Algoritma A-Star

Algoritma A-Star (A*) merupakan salah satu algoritma untuk menentukan rute terpendek antar simpul di dalam sebuah graf. Algoritma A-Star (A*) mirip dengan algoritma Djikstra namun menggunakan sebuah fungsi aproksimasi di dalam perhitungannya yang disebut fungsi *heuristic*.

Fungsi *heuristic* akan memperkirakan jarak sebuah simpul n dengan tujuan akhir. Terdapat banyak cara yang dapat dipakai untuk memperkirakan jarak namun 3 cara yang paling sering digunakan sebagai berikut :

1. Jarak Manhattan
   Jarak Manhattan digunakan saat pergerakan dibatasi ke 4 arah saja. Rumus jarak Manhattan sebagai berikut:

   $$h = |awal.x - akhir.x| + |awal.y - akhir.y|$$

   Rumus ini menghitung jumlah dari perbedaan mutlak absis dan ordinat titik awal dan titik akhir.

2. Jarak Diagonal
   Jarak diagonal digunakan saat pergerakan dibatasi ke 8 arah saja. Rumus jarak diagonal sebagai berikut:

   $$h = \max(|awal.x - akhir.x|, |awal.y - akhir.y|)$$

   Rumus ini menghitung nilai maksimum dari perbedaan absis dan ordinat titik awal dan titik akhir.

3. Jarak Euclidean
   Jarak Euclidean digunakan saat pergerakan dapat ke semua arah. Rumus jarak Euclidean sebagai berikut:

   $$h = \sqrt{(awal.x - akhir.x)^2 + (awal.y - akhir.y)^2}$$

   Rumus ini menghitung jarak dari titik awal dan titik akhir.

Perhitungan rute terpendek pada algoritma A-Star menggunakan rumus sebagai berikut:

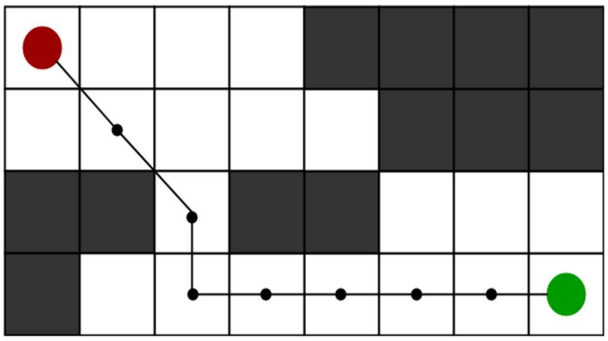
$$f(n) = g(n) + h(n)$$

g = Jarak yang dibutuhkan untuk pindah dari titik awal ke sebuah titik n pada sebuah peta.

h = Perkiraan jarak yang dibutuhkan untuk pindah dari titik n pada peta ke titik tujuan (*heuristic*).

Cara kerja algoritma A-Star sebagai berikut:

1. Inisialisasi dua list, satu untuk menyimpan simpul yang sudah dikunjungi (*visited*), satu untuk yang belum dikunjungi (*yet to visit*). Kedua list pada awalnya kosong.
2. Tambahkan titik awal ke list *yet to visit*.
3. Cari simpul dengan nilai f paling kecil dari list *yet to visit*. Simpul ini adalah simpul S.
4. Cek apakah simpul ini adalah titik tujuan.
5. Jika tidak, cek simpul anak lain yang dapat diakses melalui simpul S.
6. Jika simpul anak sudah ada di list *visited*, lompati.
7. Jika simpul anak sudah ada di list *yet to visit*, cek apakah simpul ini memiliki g yang lebih kecil dari simpul sekarang. Jika lebih kecil, jadikan simpul ini sebagai simpul S.
8. Jika simpul anak belum ada di list *yet to visit*, hitung f, g, dan h simpul tersebut dan masukkan ke dalam list *yet to visit*.
9. Ulangi langkah 3 – 8 sampai bertemu titik tujuan.

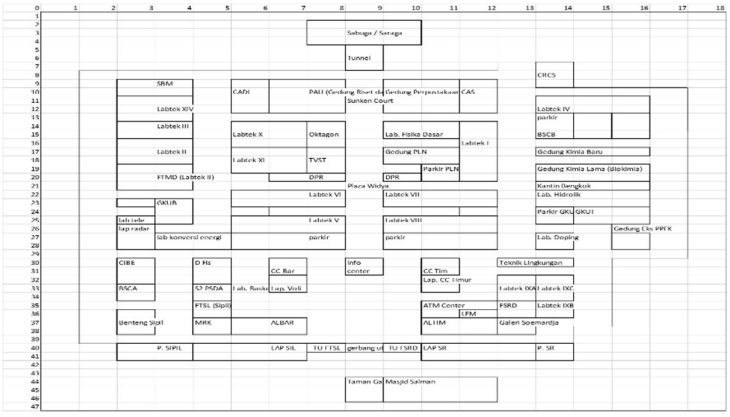


Gambar 12 : Contoh algoritma A-Star untuk menentukan rute terdekat
Sumber : https://www.geeksforgeeks.org/a-search-algorithm/

III. Aplikasi Algoritma A-Star Dalam Menentukan Rute Tercepat

A. Inisialisasi Peta

Dalam membuat peta untuk program ini, dibuat sebuah grid berukuran 42x19 untuk mensimulasikan peta Kampus Ganesha yang sebenarnya.



Gambar 13 : Grid Peta Kampus ITB

Dibuat fungsi untuk memasukkan data ITB dalam grid yang akan dibuat. Penempatan lokasi bangunan dan jalan seperti yang ada pada gambar 1.

```python
def Masukkan_data_ITB():
    global grid
    grid[11][9]="Sunken Court"
    grid[21][9]="Plaza Widya"
    grid[24][9]="Intel"
    grid[31][9]="Information Center"
    grid[25][8]="Labtek V"
    grid[23][8]="Labtek VI"
```

Gambar 14 : Cuplikan Inisialisasi Peta

B. Inisialisasi Simpul Graf

Dibuatkan sebuah class bernama Cell yang akan memuat atribut-atribut simpul graf. Atribut-atribut ini berupa absis, ordinat, nilai g, nilai h, nilai f, dan apakah Cell dapat ditembus atau tidak.

```python
class Cell(object):
    def __init__(self, x,y,reachable):
        self.reachable=reachable
        self.x=x
        self.y=y
        self.parent=None
        self.g=0
        self.h=0
        self.f=0
```

Gambar 15 : Class Simpul Graf

C. Fungsi Heuristic

Dalam perhitungan *heuristic*, digunakan metode perhitungan jarak Manhattan karena program ini memiliki ketentuan bahwa arah gerah hanya bisa ke atas, bawah, kiri, dan kanan.

```python
def get_heuristic(self,cell):
        return 20*(abs(cell.x-self.end.x)+abs(cell.y-self.end.y))
```

Gambar 16 : Fungsi *heuristic*

Perhitungan jarak Manhattan dikalikan dengan 20 karena diasumsikan bahwa satu simpul itu panjangnya 20.

D. Fungsi Pencarian Rute

Untuk mencari rute terpendek, dibuat sebuah fungsi bernama solve yang menerima parameter Object Astar yang memiliki atribut list *opened*, list *closed*, list *cells*, tinggi grid, dan lebar grid. Fungsi ini akan menggunakan library python yang bernama heapq. Library ini merupakan library algoritma heap queue atau priority queue yang akan menyimpan dan mengurutkan simpul mulai dari f terkecil dalam list *yet to visit*.

Pertama, dimasukkan titik awal ke dalam list bernama *opened* melalui pemanggilann fungsi heapq yaitu heappush. Setelah itu selama list *opened* memiliki isi, ambil titik dengan f terkecil di dalam list *opened* dengan memakai fungsi heappop. Masukkan titik tersebut ke dalam list *closed* yang berisi simpul-simpul yang sudah dikunjungi. Jika titik yang diambil merupakan titik tujuan, kembalikan lintasan yang ditempuh dengan menggunakan fungsi get_path yang telah dibuat sebelumnya.

Jika titik bukan merupakan titik tujuan, ambil semua simpul yang berada di atas, bawah, kiri, dan kanan titik dan masukkan ke dalam list *cells*. Setelah itu cek satu per satu elemen list *cells*. Jika simpul yang di dalam list dapat dicapai dan belum dilalui, cek apakah simpul memiliki nilai g yang lebih kecil daripada nilai g titik saat ini. Jika lebih kecil, masukkan ke dalam list *opened* menggunakan perintah heappush.

Perhitungan ini dilakukan terus menerus sampai ditemukan lintasan yang paling pendek.

```python
def solve(self):
  heapq.heappush(self.opened, (self.start.f,self.start))

  while len(self.opened):
    f,cell = heapq.heappop(self.opened)
    self.closed.add(cell)

    if cell is self.end:
        return self.get_path()

    adj_cells=self.get_adjacent_cells(cell)
    for adj_cell in adj_cells:
      if adj_cell.reachable and adj_cell not in self.closed:
        if (adj_cell.g>cell.g+10):
            self.update_cell(adj_cell, cell)
        else:
            self.update_cell(adj_cell, cell)
            heapq.heappush(self.opened, (adj_cell.f, adj_cell))
```

Gambar 17 : Fungsi Solve

E. Hasil Program

Untuk demonstrasi hasil program, akan dilakukan pencarian rute paling pendek menurut program dari Sarana Olahraga ITB ke GKU Barat. Hasil program sebagai berikut :

```
ASUS@FabianSavero MINGW64 /d/Projects/Pathfinder-ITB (master)
$ python -u "d:\Projects\Pathfinder-ITB\Alhamdulillah.py"
Masukkan Posisi Anda: Sarana Olahraga
Masukkan Tujuan Anda: GKU Barat
Pilihan Transportasi
1. Sepeda
2. Jalan Kaki
Tentukan Pilihan Transportasi: 1
Estimasi Waktu Kedatangan: 3.8333333333333335 menit
Rute jalan dari Sarana Olahraga ke GKU Barat
- Sarana Olahraga
- Tunnel
- Jalan 10
- Sunken Court
- Jalan 5
- Plaza Widya
- Jalan 4
- jalan 3
- GKU Barat
```

Gambar 18 : Hasil Program

Terlihat bahwa rute terpendek yang dapat diambil dari Sarana Olahraga ITB menuju GKU Barat adalah melalui Tunnel, Jalan 10, Sunken Court, Jalan 5, Plaza Widya, Jalan 4, dan Jalan 3.

Dihitung perkiraan waktu berdasarkan mode transportasi yang dipakai. Sepeda diasumsikan memiliki kecepatan 2 petak per detik dan jalan kaki memiliki kecepatan 1 petak per detik. Diasumsikan besar satu simpul adalah 20 petak. Rumus perhitungan waktu yang digunakan sebagai berikut:

$$waktu = \frac{jumlahpetak\ x\ 20}{kecepatan\ x\ 60}$$

IV. KESIMPULAN

Permasalahan yang dialami oleh mahasiswa di dalam Kampus Ganesha dapat diselesaikan dengan menggunakan Algoritma A-Star. Meskipun tidak seakurat algoritma Djikstra, algoritma ini dapat mencari rute terpendek dengan menggunakan waktu yang lebih sedikit.

Keakuratan algoritma ini sangat bergantung pada metode perhitungan *heuristic*. Perlu dicari keseimbangan antara kecepatan dan keakuratan perhitungan *heuristic*. Semakin akurat perhitungan, semakin akurat algoritma. Namun, sebaliknya kecepatan algoritma akan semakin lambat.

Penerapan Algoritma A-Star pada makalah ini tentu jauh dari sempurna. Ada banyak perbaikan yang dapat dilakukan terhadap program ini. Salah satu perbaikan yang dapat dilakukan adalah mengubah arah pergerakan sehingga dapat ke semua arah sehingga fungsi *heuristic* perlu diganti. Selain itu, peta juga dapat menambahkan simpul lain yang berupa jalan-jalan kecil yang banyak melalui bangunan-bangunan di Kampus Ganesha. Program juga dapat diperbagus dengan membuat GUI sehingga dapat dengan mudah dipakai oleh orang lain.

V. UCAPAN TERIMA KASIH

Puji syukur penulis haturkan kepada Tuhan Yang Maha Esa karena tanpa karunia-Nya, hampir tidak mungkin penulis menyelesaikan laporan penelitian yang berjudul Penggunaan Algoritma A-Star untuk Menentukan Rute Tercepat di dalam Kampus Ganesha ITB. Laporan penelitian ini merupakan salah satu tugas mata kuliah Matematika Diskrit pada Semester Ganjil Tahun Akademik 2020/2021 di Institut Teknologi Bandung.

Terselesaikannya penulisan karya tulis ini juga tidak terlepas dari bantuan maupun arahan beberapa pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Orang tua yang selalu memberikan doa serta dukungan kepada penulis dalam menyelesaikan karya tulis ilmiah ini.
2. Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc. selaku dosen mata kuliah Matematika Diskrit yang telah membimbing dan mengajari penulis selama satu semester sehingga penulis dapat memahami semua materi perkuliahan yang diajarkan.
3. Bapak Dr. Ir. Rinaldi Munir, MT. selaku pengurus website yang berisi slide materi kuliah yang sangat membantu penulis di pembelajaran semester ini serta dalam pembuatan karya tulis ini.
4. Teman-teman saya yang telah membantu dan memberi pendapat terhadap karya tulis penulis selama tahap penulisan.

REFERENSI

[1] Rosen, K. H. *Discrete Mathematics and Its Applications.* New York : McGraw – Hill, 2012.
[2] Riyadi, I. Anindya, *Aplikasi Algoritma Dijkstra dalam Menentukan Rute Tercepat,* Informatika Bandung, 2019.
[3] https://www.itb.ac.id/kampus-ganesha. Diakses pada 6 Desember 2020.
[4] https://www.geeksforgeeks.org/a-search-algorithm. Diakses pada 6 Desember 2020.
[5] https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb. Diakses pada 6 Desember 2020.
[6] http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html. Diakses pada 6 Desember 2020.
[7] https://www.raywenderlich.com/3016-introduction-to-a-pathfinding. Diakses pada 6 Desember 2020.
[8] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/matdis20-21.htm. Diakses pada 7 Desember 2020.

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, 7 Desember 2020

Fabian Savero Diaz Pranoto/13519140