Penerapan Algoritma UCS untuk Pencarian Rute Terpendek Pengiriman Surat di Kota Probolinggo

Benidictus Galih Mahar Putra 13519159
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519159@std.stei.itb.ac.id, benidictusgalih@gmail.com

***Abstraksi*—Kantor pos mengurus pengiriman barang yang memuat banyak alamat. Dalam pengiriman barang ini, tentu alamat yang dituju tidak selalu menghasilkan jarak yang optimal. Untuk meminimalkan jarak pengiriman, diperlukan rute terpendek untuk mencapai alamat yang dituju. Oleh karena itu, algoritma *Uniform Cost Search* dapat menemukan alamat yang banyak dengan rute terpendek.**

Kata Kunci—Rute; Terpendek; Pengiriman;

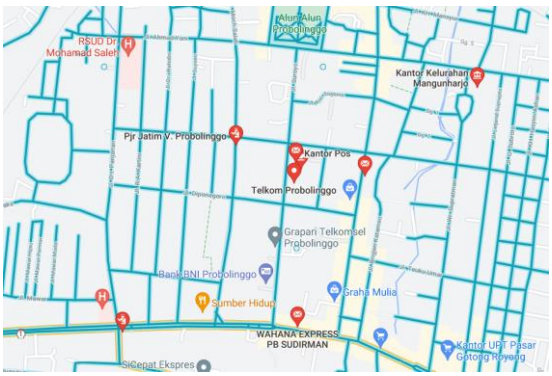
I. PENDAHULUAN

Kantor pos merupakan layanan masyarakat yang melayani bentuk pengiriman berupa barang dari sebuah alamat ke alamat lain. Pengiriman ini dilakukan dalam dua tipe, pengiriman antar kota dan pengiriman dalam kota/kabupaten. Nominal biaya kirim ini bervariasi dan tergantung dengan jarak alamat pengirim dengan alamat penerima. Layanan pengiriman juga melayani pengiriman ekspress dan pengiriman biasa/normal. Pengiriman ekspress akan dikirimkan dengan prioritas tertinggi dan pengiriman biasa/normal akan dikirimkan bersama dengan surat – surat lain dengan alamat yang dekat.

Dalam zaman modern ini, pengiriman surat sangat jarang dilakukan. Pengiriman banyak diisi dengan pengiriman paket berupa barang/*box* besar. Di Kota Probolinggo, beberapa pengiriman surat masih dilakukan. Pengiriman ini dikirim oleh masyarakat yang tidak belum beradaptasi dengan zaman modern untuk mengirimkan uang ataupun berkabar kepada anaknya yang di luar kota.

Dalam pengiriman surat/barang, tukang pos yang berpengalaman atau sudah lama tinggal di daerah tersebut lebih lancar mengirimkan surat – surat tersebut dengan efektif. Berbeda dengan tukang pos yang masih baru dan belum mengerti alamat – alamat ini. Tukang pos akan mengirimkan surat – surat ini tanpa memikirkan *cost* yang akan dibayar ketika mengirimkan surat. Oleh karena itu, dibutuhkan sebuah teknologi untuk menemukan rute terpendek yang dapat dicapai dari alamat – alamat surat yang dimiliki oleh tukang pos.

Dalam Gambar 1.1. Peta Kota Probolinggo, terdapat banyak jalan, persimpangan, dan jalan satu arah. Untuk melakukan implementasi, algoritma *Uniform Cost Search* tepat untuk digunakan karena setiap alamat tujuan akan dievaluasi jaraknya. Kemudian hasil dari evaluasi tersebut, dihasilkan sebuah rute dengan alamat tujuan yang paling dekat dengan lokasi tukang pos.

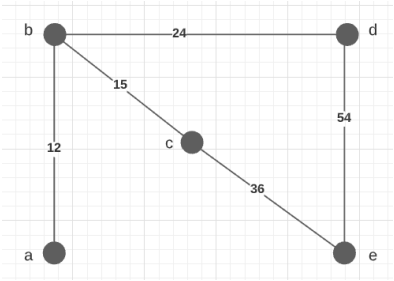


Gambar 1.1. Peta Kota Probolinggo

II. DASAR TEORI

A. *Graf Berbobot*

Graf terdiri dari beberapa simpul dan sisi. Dua Simpul dikatakan ketetanggaan, jika dua simpul ini dihubungkan oleh sisi secara langsung. Kemudian salah satu bentuk dari graf yaitu graf tidak berbobot dan graf berbobot. Graf berbobot merupakan graf yang setiap sisinya memiliki sebuah bobot berupa *state*/angka.

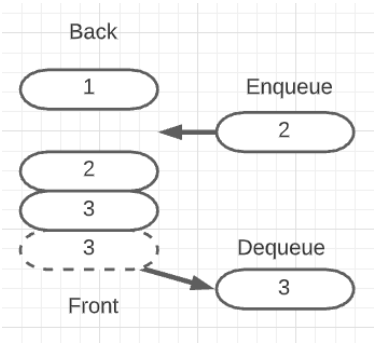


Gambar 2.1. Graf Berbobot

Pada Gambar 2.1. Graf Berbobot, simpul a saling ketetanggaan dengan simpul b. Akan tetapi, simpul a tidak ketetanggaan dengan simpul c.

B. *Priority Queue*

Priority Queue merupakan salah satu bentuk struktur data yang diilustrasikan sebagai antrian dengan prioritas tertentu seperti prioritas angka terbesar atau terkecil. Dalam *Priority Queue* memiliki ciri seperti setiap elemen dalam antrian memiliki prioritas yang dapat berupa integer atau lainnya. Kemudian setiap elemen dengan prioritas tertinggi akan diambil (*Dequeue)* terlebih dahulu sebelum prioritas lebih rendah. Jika dua elemen atau lebih memiliki prioritas yang sama, pengambilan elemen akan dilakukan sesuai dengan urutan dari antrian. Bentuk dari *Priority Queue* dapat berupa array atau list atau binary heap.



Gambar 2.2. *Priority Queue*

Dalam Gambar 2.2. *Priority Queue*, terdapat empat antrian dengan prioritas yang berbeda. Ketika dilakukan pengambilan (*Dequeue*) diambil antrian dengan prioritas tertinggi yaitu prioritas tiga. Kemudian saat kita memasukan (*Enqueue*) sebuah variabel dengan prioritas dua, variabel akan dimasukan ke dalam antrian dengan prioritas antara satu dan dua.

C. *Formula Haversine*

Haversine formula merupakan rumus yang digunakan untuk menghitung jarak dari dua titik di bidang bola dengan menggunakan posisi garis lintang (*latitude*) dan garis bujur (*longtitude*).

$$a = \sin^2((\varphi_2 - \varphi_1)/2) + \cos\varphi_1 * \cos\varphi_2 * \sin^2((\lambda_2 - \lambda_1)/2) \quad (1)$$

$$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{(1 - a)}) \quad (2)$$

$$d = R * c \quad (3)$$

Keterangan:

φ: Posisi pada garis lintang (*Latitude*)

λ: Posisi pada garis bujur (*Longtitude*)

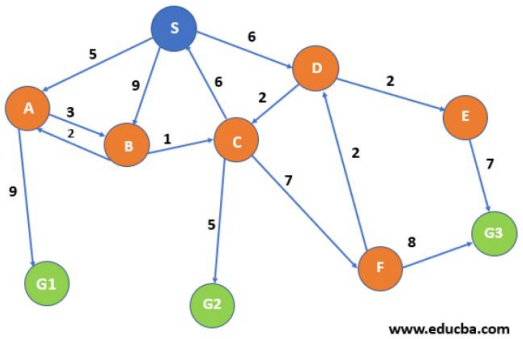
R: Radius bumi (6,371 km)

d: Jarak dua titik

D. *Algoritma Uniform Cost Search*

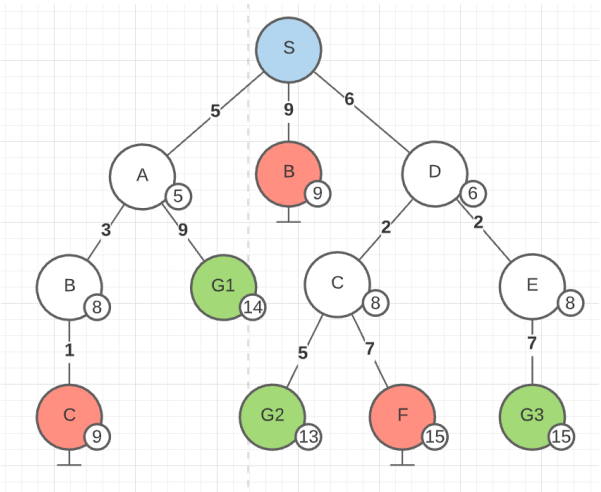
Algoritma *Uniform Cost Search* merupakan algoritma yang menelusuri setiap *node* dari *Tree* dan penelusuran berawal dari sebuah *node* awal dan berakhir dengan *node* yang memiliki *cost* paling rendah. Algoritma ini dipakai untuk mencari sebuah jalur dengan total *cost* yang paling rendah. Dalam implementasinya, algoritma ini memakai *priority queue* untuk menyimpan *node* dengan bobotnya masing – masing. Dalam pembangkitan simpul anak, nilai bobot dari simpul anak tersebut, bernilai bobot jarak dari simpul awal hingga simpul anak.

1. Memasukan node ke dalam antrian.
2. Kemudian node diperluas dengan memasukan anak – anak dari node prioritas tertinggi dari antrian, ke dalam antrian.
3. Mengeluarkan node prioritas tertinggi dari antrian.
4. Kita periksa jika node ini merupakan tujuan dari rute, kita hentikan pencarian dan keluarkan hasil path.
5. Jika bukan merupakan tujuan dari rute, diulangi langkah kedua.

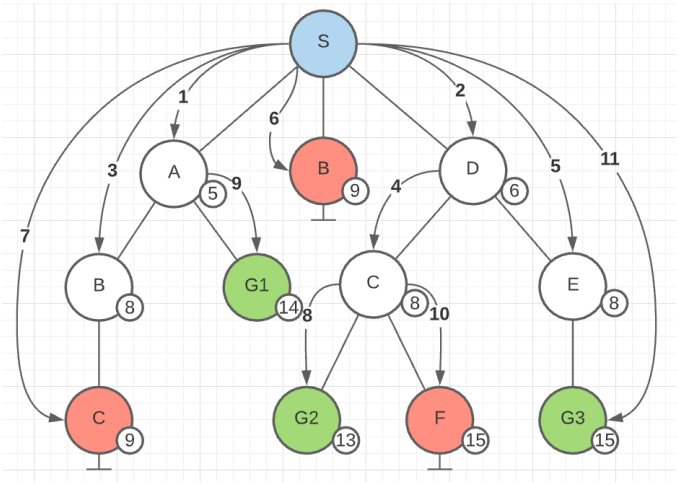


Gambar 2.3.1. Contoh Kasus Graf UCS

Sebagai contoh diberikan sebuah graf UCS yang ditampilkan pada Gambar 2.3.1. Contoh Kasus Graf UCS. Terdapat node tujuan berupa {G1, G2, G3} dan node penghubung {A, B, C, D, E, F}. Tujuan kasus ini adalah mencari node dengan bobot / *cost* terendah dari node awal *S* menuju salah satu dari node tujuan yaitu G1, G2, atau G3. Tampilan visualisasi dan langkah – langkah ditampilkan dalam Gambar 2.3.2. Hasil Visualisasi Graf Algoritma UCS dan Gambar 2.3.3. Flow Penelusuran Algoritma.



Gambar 2.3.2. Hasil Visualisasi Graf Algoritma UCS



Gambar 2.3.3. Flow Penelusuran Algoritma

No.	Visited	PriorityQueue
1.	S, A	D, B, B, G1
2.	S, A, D	B, C, E, B, G1
3.	S, A, D, B	C, E, B, C, G1
4.	S, A, D, B, C	E, B, C, G2, G1, F
5.	S, A, D, B, C, E	B, C, G2, G1, F, G3
6.	S, A, D, B, C, E	C, G2, G1, F, G3
7.	S, A, D, B, C, E	G2, G1, F, G3
8.	S, A, D, B, C, E	G1, F, G3 *G2 merupakan node tujuan sehingga penelusuran dihentikan

III. Penerapan Algoritma

Dalam penyusunan algoritma ini, penulis menyederhanakan beberapa masalah atau kasus variabel luar. Salah satunya, jalan yang dilalui oleh tukang pos beberapa dapat dilalui dengan dua arah dan satu arah (*one way traffic*). Setiap alamat yang digunakan sebagai simpul hanya untuk alamat surat yang akan dikirimkan. Kemudian pengiriman surat dikirimkan dengan pengiriman biasa/normal sehingga prioritas semua surat akan sama besarnya. Selain itu, diasumsikan juga, lebar jalan yang digunakan sebagai sisi cukup untuk dilalui oleh tukang pos.

Dalam penyusunan Tree, sebuah simpul dibangun oleh persimpangan atau lokasi alamat. Kemudian sebuah sisi mewakili jalan yang menghubungkan simpul – simpul tersebut.

A. *Struktur Data*

Priority Queue List of Array digunakan untuk menyimpan variabel bobot dan isi node. List ini berisi dua elemen yaitu Integer (Jarak antar simpul/jalan/persimpangan) dan String (Nama singkat jalan atau alamat)

Array of Boolean digunakan untuk memeriksa node yang telah dikunjungi.

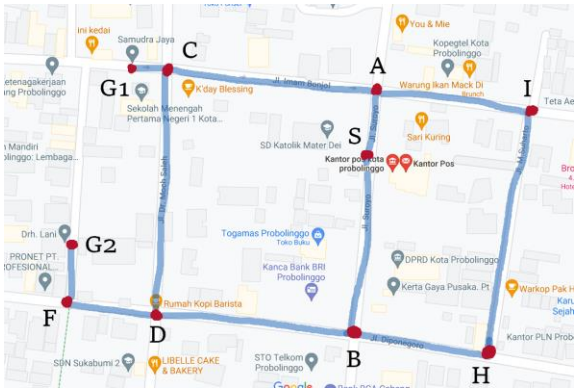
Matriks of Integer digunakan untuk memuat angka bobot antar simpul satu dengan simpul lainnya yang saling terhubung.

Matriks adjacency yang digunakan untuk menyimpan simpul – simpul yang ketetanggaan.

Dictionary yang digunakan untuk menyimpan indeks dari String simpul untuk digunakan kembali pada matriks of integer.

Tree digunakan untuk pencarian solusi terbaik yang dihidupkan melalui tiap – tiap node. Dalam node pohon, dimuat sebuah String (Nama) dan Integer (Total bobot dari simpul awal hingga simpul n).

B. *Data Input*



Gambar 3.1. Visualisasi Input Graf

Beberapa contoh input file untuk matriks ketetanggaan dari simpul adalah S-A, S-B, A-C, B-D, dll. Dalam Gambar 3.1. Visualisasi Input Graf, penelusuran diperlukan untuk menemukan simpul G1 atau G2 dengan berawal dari simpul S. Contoh input file untuk koordinat latitude dan longtitude dari simpul – simpul. Pencarian posisi *latitude* dan *longtitude* dapat dicari dengan menggunakan *google maps*.

Simpul	Latitude	Longtitude
S (Kantor Pos)	-7.748425745529926	113.21574099348207
G1 (Samudra Jaya)	-7.747571216095641	113.21362862279398
G2(Drh. Lani)	-7.748990438705093	113.21310290984366
A	-7.747879512046631	113.21549544020935
B	-7.749750544728555	113.21532914326112
I	-7.748044290830266	113.21667024774126
C	-7.747735994996102	113.21389147926912
H	-7.749904691935183	113.21635911150538
D	-7.749575135767116	113.21382174183694
F	-7.749490088972196	113.21312973193297

C. Algoritma Uniform Cost Search

Dari data input, diolah menjadi beberapa data struktur seperti penghitungan jarak dengan formula haversine yang ditampilkan pada tabel di bawah ini.

Simpul1	Simpul2	Jarak (meter)
S	A	66.491635691906
S	B	154.141805191348
A	I	130.730607421362
A	C	177.44346689807
B	H	114.768504925071
B	D	167.226109131493
H	I	209.688636297748
C	D	204.647707208503
C	G1	34.270807984206
D	F	76.829561018836
F	G2	55.637194774081

Dictionary simpul untuk Gambar 3.1. Visualisasi Input Graf.

```
index = {
    "S" = 0, "G1" = 1,
    "G2" = 2, "A" = 3,
    "B" = 4, "C" = 5,
    "D" = 6, "F" = 7,
    "H" = 8, "I" = 9
}
```

Implementasi untuk penghitungan formula Harversine

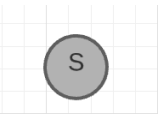
```
function calculateDistance(φ2: integer; λ2: integer;
φ1: integer; λ1: integer) → integer
    R = 6,371
    a ← (sin ((φ2 − φ1)/2))**(2) + cos φ1 * cos φ2 *
(sin ((λ2 − λ1)/2))**2
    c ← 2 * atan2((a)**(0.5), (1 - a)**(0.5))
    d ← R * c
    → d
```

Implementasi untuk pohon pencarian solusi. Dalam implementasi tsb, dapat dimungkinkan untuk mendapatkan nilai infinite loop ketika tidak ditemukan solusi saat penelusuran.

```
pqueue.enqueue(snode)

if(not visited[dictionary(snode)]) then
    i traversal
[0..length(madj[dictionary(snode)])]
```

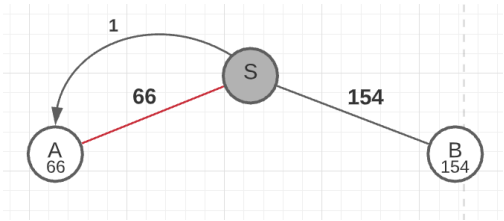
```
pqueue.enqueue(madj[dictionary(snode)][i])
    mbool[dictionary(snode)] <- True
    pqueue.dequeue()

    end <- False
    while (not end) do
        goal <- pqueue.dequeue()
        i traversal [0..length(arrgoal)]
            if(goal = arrgoal[i]) then
                end <- True
        if(not mbool[dictionary(goal)])
            i traversal
[0..length(madj[dictionary(goal)])]

pqueue.enqueue(madj[dictionary(goal)][i])
        mbool[dictionary(goal)] <- True
```

Node berawal dari simpul S sehingga simpul S dimasukkan ke dalam priority queue dengan bobot bernilai 0.



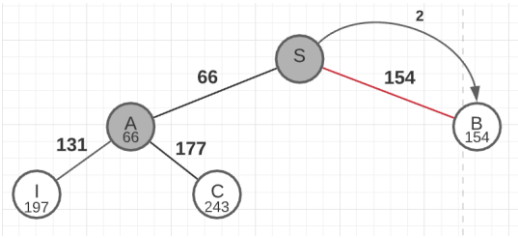
Visited	Priority Queue
-	S(0)

Kemudian, antrian dikeluarkan (*enqueue*) dan dibangkitkan dan dimasukan ke dalam antrian simpul – simpul anaknya serta menandai simpul S telah dikunjungi dengan nilai true.



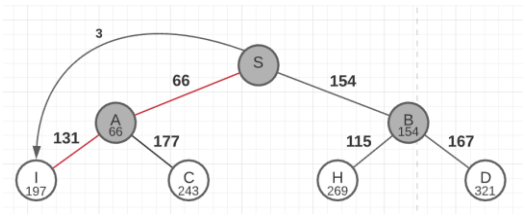
Visited	Priority Queue
S	A(66), B(154)

Kemudian sama seperti sebelumnya, kita mengambil elemen pertama dari queue dan dibangkitkan simpul anak – anaknya. Setelah itu, tandai simpul *dequeue* dengan nilai true.



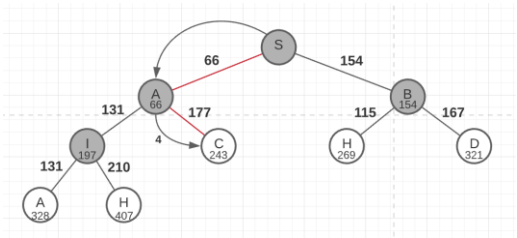
Visited	Priority Queue
S, A	B(154), I(197), C(243)

Pada saat penelusuran dari simpul anak – anak dari A, tidak ditemukan nilai minimum sehingga dilakukan penelusuran ulang dari S kemudian menuju simpul B.



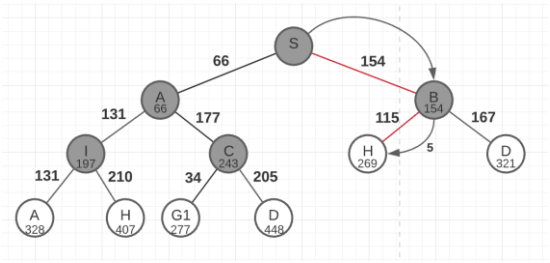
Visited	Priority Queue
S, A, B	I(197), C(243), H(269), D(321)

Setelah simpul – simpul anak dari B dibangkitkan, kita menelusuri kembali simpul dengan nilai minimum.



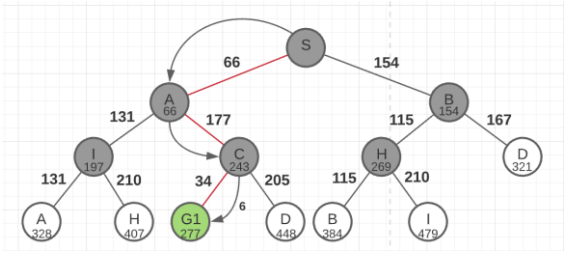
Visited	Priority Queue
S, A, B, I	C(243), H(269), D(321), A(328), H(407)

Kita melanjutkan penelusuran ke simpul C karena simpul ini masih belum dikunjungi dan bobot bernilai minimum.



Visited	Priority Queue
S, A, B, I, C	H(269), G1(277), D(321), A(328), H(407), D(448)

Kita menemukan pembangkitan simpul yang merupakan node tujuan. Akan tetapi kita perlu menelusuri lebih lanjut nilai minimum yang masih dimungkinkan.



Visited	Priority Queue
S, A, B, I, C, H	G1(277), D(321), A(328), B(384), H(407), D(448), I(479)
S, A, B, I, C, H, G1	D(321), A(328), B(384), H(407), D(448), I(479) *G1 merupakan simpul tujuan sehingga penelusuran dihentikan.

Setelah semua simpul dibangkitkan dan pengambilan elemen pertama dari *queue*, simpul yang diambil merupakan simpul tujuan sehingga simpul ini menjadi akhir dari penelusuran. Hasil yang dicapai dalam algoritma UCS dengan kasus di atas didapat rute dengan hasil terpendek yaitu simpul S-A-C-G1 dengan bobot 277. Kemudian simpul lain tidak dibangkitkan karena bobot dari simpul lain sudah melebihi dari bobot tujuan sehingga berapapun nilai minimal (tanpa nilai negatif atau non riil), jarak atau rute terpendek yang dapat dicapai hanya sampai rute yang didapat tadi.

IV. KESIMPULAN

Penelusuran ini dianggap efektif ketika tukang pos ingin mengirimkan surat dengan tanpa melihat jarak dari sebuah alamat tujuan dan memilih alamat terpendek dari pilihan alamat yang dimiliki oleh tukang pos. Implementasi dari algoritma UCS dalam menemukan rute terpendek dari pengiriman surat di Probolinggo tidak sepenuhnya dapat berjalan dengan baik. Sebagai contoh, ketika tidak ditemukan alamat pada graf yang dibangun, algoritma UCS beresiko menghasilkan *infinite loop* dalam pencariannya. Kemudian,

penyederhanaan yang dilakukan penulis masih belum sepenuhnya sempurna karena hasil keluaran dari algoritma yang disusun dalam laporan ini hanya mencari alamat terdekat yang dapat dicapai oleh Tukang Pos. Untuk ke depannya, bagi penulis lain sebaiknya, dilakukan pengurutan jarak awal dengan alamat tujuan secara dinamis saat program sedang berjalan.

VIDEO LINK AT YOUTUBE

https://youtu.be/a52RjFRhQjM

PENUTUP

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa. Atas berkat rahmat Tuhan, penulis dapat menyelesaikan makalah ini dengan baik. Penulis juga mengucapkan terima kasih kepada Bapak Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D. selaku Dosen Strategi Algoritma kelas 03 yang telah membimbing dan mengajar materi IF2211 kepada penulis selama proses mengajar semester II tahun 2020/2021. Penulis juga tidak lupa mengucapkan terima kasih kepada teman – teman dan kedua orang tua penulis yang senantiasa membantu doa dan memberi ide kepada penulis selama penyusunan makalah ini. Semoga makalah ini dapat dikembangkan lebih lanjut oleh penulis lain dan dimanfaatkan dengan baik untuk pengiriman surat di Kota Probolinggo dan kota lain.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf diakses pada tanggal 6 Mei 2021

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada tanggal 8 Mei 2021

[3] https://www.geeksforgeeks.org/priority-queue-set-1-introduction/#:~:text=Priority%20Queue%20is%20an%20extension,their%20order%20in%20the%20queue. diakses pada tanggal 9 Mei 2021

[4] https://community.esri.com/t5/coordinate-reference-systems/distance-on-a-sphere-the-haversine-formula/ba-p/902128#:~:text=For%20example%2C%20haversine(%CE%B8),longitude%20of%20the%20two%20points. diakses pada tanggal 10 Mei 2021

[5] https://www.educba.com/uniform-cost-search/ diakses pada tanggal 10 Mei 2021

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, 11 Mei 2021

Benidictus Galih Mahar Putra 13519159