PENGGUNAAN ALGORITMA FLOYD WARSHALL DALAM MASALAH JALUR TERPENDEK PADA PENENTUAN TATA LETAK PARKIR

Ni Ketut Dewi Ari Jayanti, M.Kom

STMIK STIKOM Bali
Jl. Raya Puputan No. 86 Renon Denpasar, telp. 0361 244445
e-mail: daj@stikom-bali.ac.id

Abstrak

Peningkatan yang pesat dalam jumlah kendaraan di kota-kota besar, memilikidampak terhadap kebutuhan parkir di tempat-tempat umum seperti dikantor, pusat perbelanjaan, sekolah, kampus, tempat rekreasi, dan tempat-tempatumum lainnya yang memiliki area parkir yang cukup luas.Diperlukan penataan areal parkir agar memiliki daya tampung yang maksimal tanpa mengesampingkan aspek kenyamanan untuk penggunanya sehingga penentuan tata letak dan waktu tempuh kendaraan dalam mencari lokasi parkir perlu untuk di perhatikan. Untuk menentukan jalur terpendek pada penentuan tata letak parkir dalam penelitian ini menggunakan metode *Floyd-Warshall* untuk melakukan perhitungan jalur terpendek. Floyd Warshall merupakan salah satu algoritma pencarian yang dapat digunakan dalam menghitung jalur terpendek, dan mampu membandingkan semua kemungkinan lintasan pada graph untuk setiap sisi dari semua simpul yang ada.Dalam penelitian ini telah berhasil dibentuk jalur terpendek pada tata letak parkir. *Algoritma Floyd-Warshall* dapat menyelesaikan permasalahan jalur terpendek pada tata letak parkir dengan menghitung jarak seluruh jalur/ lintasan yang ada antar blok parker dan hasil nya akan dapat membantu pengembang sistem dalam membangun sistem parkir serta memberikan solusi untuk mengoptimalkan tata letak parkir, sehingga tingkat kepuasan pengguna akan tercapai dan untuk memberikan solusi sistem manajemen parkir yang lebih baik.

Kata kunci : Parkir, Graf, Jalur Terpendek, *Floyd Warshall*

1. LATAR BELAKANG

Peningkatan yang pesat dalam jumlah kendaraan di kota-kota besar, memiliki dampak terhadap kebutuhan parkir di tempat-tempat umum seperti di kantor, pusat perbelanjaan, sekolah, kampus, tempat rekreasi, dan tempat-tempat umum lainnya yang memiliki area parkir yang cukup luas. Diperlukan penataan areal parkir agar memiliki daya tampung yang maksimal tanpa mengesampingkan aspek kenyamanan untuk penggunanya sehingga penentuan tata letak dan waktu tempuh kendaraan dalam mencari lokasi parkir perlu untuk di perhatikan.

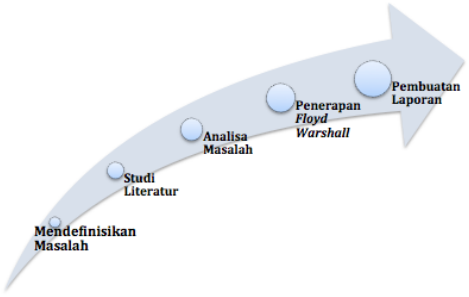
Perkembangan teknologi informasi, khususnya perangkat lunak saat ini sudah dapat diaplikasikan sebagai salah satu sistem yang sangat membantu dalam menghadapi masalah tersebut.Sudah cukup banyak sistem parkir konvesional yang dikelola oleh beberapa perusahaan yang tentunya memiliki kelebihan dan kekurangan dalam hal pengelolaannya.Kenyataan yang kita lihat dan rasakan sendiri adalah seringkali kita sebagai pengunjung harus bersusah payah untuk mencari tempat parkir yang kosong karena sistem parkir yang tersedia saat ini terbatas pada perhitungan berapa biaya parkir yang harus dibayar.Sistem parkir yang ada saat ini belum dilengkapi dengan fasilitas pencarian lokasi parkir yang masih tersedia.Tidak hanya itu saja, sistem parkir yang ada juga belum menyediakan fasilitas berupa letak parkir yang disarankan oleh sistem berdasar pada letak parkir dengan jalur terpendek yang dapat mengoptimalkan jarak tempuh pengunjung menuju pintu masuk gedung.

Masalah jalur terpendek adalah masalah menemukan suatu jalur antara dua simpul sedemikian sehingga jumlah bobot dari busur penyusunnya dapat seminimal mungkin (R. Kumar dan M. Kumar, 2010).Beberapa algoritma yang telah dikembangkan untuk menyelesaikan permasalahan jalur terpendek diantaranya algoritma *Djikstra*, algoritma *Floyd-Warshall* dan algoritma *Bellman-Ford*.Sementara metode yang paling efisien untuk permasalahan jalur terpendek dalam jaringan data adalah algoritma Djikstra. Namun, pada jaringan dinamis yang sangat besar, algoritma Djikstra menjadi tidak efisien karena simpul-simpul pada jaringan akan dikunjungi kembali sehingga banyak komputasi atau perhitungan-perhitungan yang diulang (R. Kumar dan M. Kumar, 2010).

Untuk menentukan jalur terpendek pada penentuan tata letak parkir dalam penelitian ini menggunakan metode *Floyd-Warshall* untuk melakukan perhitungan jalur terpendek.*Floyd Warshall* merupakan salah satu algoritma pencarin yang dapat digunakan dalam menghitung jalur terpendek, dan mampu membandingkan semua kemungkinan lintasan pada *graph* untuk setiap sisi dari semua simpul yang ada. Oleh sebab itu, algoritma ini cocok digunakan dalam menghadapi permasalahan perhitungan jalur terpendek khususnya dalam penerapannya pada tata letak parkir

2. METODE PENELITIAN

Pada metode penelitian membahas tentang tempat dan waktu penelitian, alur dari penelitian serta data yang digunakan.Adapun tempat penelitian dilakukan di STIKOM Bali dengan lama waktu penelitian 4 bulan. Alur penelitian di gambarkan sebagai berikut :



Gambar 1 Alur Penelitian

a. Mendefinisikan Masalah
Mendifinisikan masalah merupakan tahapan menentukan permasalahan yang ada berkaitan dengan penentuan jalur terpendek dalam penentuan tata letak parkir.

b. Studi Literatur
Tahap ini melakukan pengumpulan materi yang berasal dari tulisan-tulisan karya ilmiah, artikel populer, serta tanggapan dari praktisi dan profesional mengenai algoritma jalur terpendek.

c. Analisa Masalah
Melakukan proses analisa terhadap permasalahan yang dibahas yaitu sistem parkir yang ada saat ini belum dilengkapi dengan fasilitas pencarian lokasi parkir yang masih tersedia. Sistem parkir yang ada juga belum menyediakan fasilitas berupa letak parkir yang disarankan oleh sistem berdasar pada letak parkir dengan jalur terpendek yang dapat mengoptimalkan jarak tempuh pengunjung menuju pintu masuk gedung dan menentukan algoritma *Floyd-Warshall* sebagai algoritma dalam menyelesaikan permasalahan jalur terpendek pada tata letak parkir.

d. Penerapan *Floyd-Warshall*
Penerapan algoritma Floyd-Warshall dilakukan berdasarkan hasil yang diperoleh dari tahap analisa masalah.Algoritma Floyd-Warshall sangat efisien dari sudut pandang penyimpanan data karena dapat diimplementasikan dengan hanya pengubahan sebuah matriks jarak. Adapun mekanisme dari algoritma Floyd-Warshall ini terdiri dari beberapa langkah yang harus dilakukan, yaitu (Budiarsyah dan Dibi Khairurrazi , 2010) :
   - Langkah awal yang harus dilakukan untuk menentukan shortest path dengan menggunakan algoritma Floyd-Warshall adalah dengan merepresentasikan suatu graf sebagai suatu matriks berbobot. Format output berupa matriks n x n berjarak D = [dij] dimana dij merupakan jarak dari vertex i ke j.
   - Langkah kedua adalah melakukan dekomposisi *Floyd-Warshall*dengan urutan :
     - $d_{ij}^{(k)}$ merupakan panjang dari *shortest path* dari i ke j, sehingga semua *vertex intermediate* yang terdapat pada path (jika ada) terkumpul pada {1,2,....,k}
     - $d_{ij}^{(0)}$ dikumpulkan pada w_{ij}, yaitu tidak ada *vertex intermediate*.
     - $D^{(k)}$ menjadi matriks n x n $[d_{ij}^{(k)}]$
     - Tentukan $d_{ij}^{(n)}$ sebagai jarak dari i ke j kemudian hitung $D^{(n)}$
     - Hitung $D^{(k)}$ untuk k = 0,1,...., n
   - Langkah ketiga adalah menentukan struktur shortest path. Dalam hal ini, harus dilakukan dua pengamatan terlebih dahulu sebelum melangkah lebih jauh, yaitu :
     - Sebuah *shortest path* tidak memuat *vertex* yang sama sebanyak dua kali
     - Untuk sebuah shortest path dari i ke j dengan beberapa vertex intermediate pada path dipilih dari kumpulan {1, 2, ..., k}, dengan kemungkinan :
       a. k bukan merupakan *vertex* pada *path* (*path* terpendek memiliki panjang $d_{ij}^{(k-1)}$).
       b. k merupakan vertex pada path (path terpendek memiliki panjang dij(k-1) + dij(k-1)).
     - Setelah melakukan pengamatan diatas, kemudian dlakukan

penentuan shortest path dari i ke j yang memuat vertex k.

- Shortest path tersebut memuat sebuah subpath dari i ke k dan sebuah subpath dari k ke j.
- Setiap subpath hanya dapat memuat vertex intermediate pada {1, ..., k-1} dan sedapat mungkin memiliki nilai terpendek, kemudian beri nama panjangnya dik(k-1) dan dkj(k-1) sehingga path memiliki panjang dik(k-1) + dkj(k-1).

– Langkah terakhir adalah melakukan iterasi yang dimulai dari iterasi ke 0 sampai dengan n.Perhitungan yang dilakukan adalah :
  - Menentukan D(0) (iterasi ke 0) = [wij] merupakan matriks bobot.
  - Menentukan D(k) dengan menggunakan rumus , dij(k) = min {dij(k-1), dik(k-1) + dkj(k-1)}, untuk k = 1, ..., n dimana n adalah jumlah vertex.

Hasil akhir dari algoritma *Floyd-Warshall* adalah matriks untuk iterasi ke-n. Dari matriks ke-n ini, dapat dilihat *shortest path* untuk setiap *vertex* pada suatu *graph*.

e. Kesimpulan
Tahap akhir dari penelitian ini, untuk menghasilkan laporan penelitian serta penarikan kesimpulan.

Untuk perolehan data, penelitian ini menggunakan jenis data sekunder yang diperoleh melalui *state of the art review* penelitian lainnya yang sejenis serta melalui akses internet (jurnal, *ebook*).

3. ANALISA dan HASIL

Pada analisa dan hasil ini membahas tentang analisa permasalahan tata letak parkir dan hasil penerapan *Floyd-Warshall*.

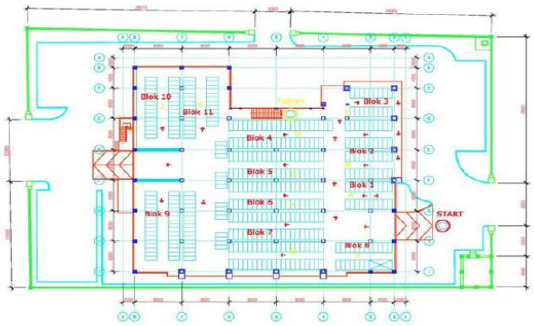
3.1 Analisa Masalah

Sudah cukup banyak sistem parkir konvesional yang dikelola oleh beberapa perusahaan yang tentunya memiliki kelebihan dan kekurangan dalam hal pengelolaannya.Kenyataan yang kita lihat dan rasakan sendiri adalah seringkali kita sebagai pengunjung harus bersusah payah untuk mencari tempat parkir yang kosong karena sistem parkir yang tersedia saat ini terbatas pada perhitungan berapa biaya parkir yang harus dibayar.Sistem parkir yang ada saat ini belum dilengkapi dengan fasilitas pencarian lokasi parkir yang masih tersedia.Tidak hanya itu saja, sistem parkir yang ada juga belum menyediakan fasilitas berupa letak parkir yang disarankan oleh sistem berdasar pada letak parkir dengan jalur terpendek yang dapat mengoptimalkan jarak tempuh pengunjung menuju pintu masuk gedung.

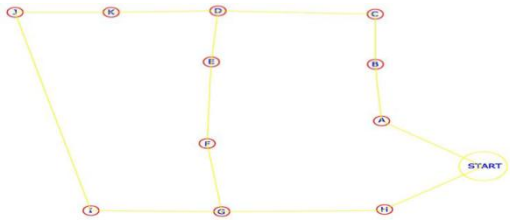
Masalah jalur terpendek adalah masalah menemukan suatu jalur antara dua simpul sedemikian sehingga jumlah bobot dari busur penyusunnya dapat seminimal mungkin.Untuk menentukan jalur terpendek pada penentuan tata letak parkir dalam penelitian ini menggunakan metode Floyd-Warshall untuk melakukan perhitungan jalur terpendek dari pintu masuk menuju lot parkir.

3.2 Hasil Penerapan *Floyd-Warshall*

Lot parkir yang diperoleh direpresentasikan ke dalam bentuk graf serta diberi bobot dari satu vertex kevertex yang lain.



Gambar 2. Ilustrasi Denah Parkir (Adnyana, 2011)



Gambar 3. Representasi Graf Denah parkir

Tabel 1. Jarak antar Blok (*Vertex*)

Jarak	Start (1)	A (2)	B (3)	C (4)	D (5)	E (6)	F (7)	G (8)	H (9)	I (10)	J (11)	K (12)
Start (1)	0	23.00	∞	∞	∞	∞	∞	∞	31.30	∞	∞	∞
A (2)	23.00	0	9.00	∞	∞	∞	∞	∞	∞	∞	∞	∞
B (3)	∞	9.00	0	7.81	∞	∞	∞	∞	∞	∞	∞	∞
C (4)	∞	∞	7.81	0	16.57	∞	∞	∞	∞	∞	∞	∞
D (5)	∞	∞	∞	16.57	0	6.59	∞	∞	∞	∞	∞	20.11
E (6)	∞	∞	∞	∞	6.79	0	6.59	∞	∞	∞	∞	∞
F (7)	∞	∞	∞	∞	∞	6.59	0	6.10	∞	∞	∞	∞
G (8)	∞	∞	∞	∞	∞	∞	6.10	0	6.57	15.72	∞	∞
H (9)	31.30	∞	∞	∞	∞	∞	∞	6.57	0	∞	∞	∞
I (10)	∞	∞	∞	∞	∞	∞	∞	15.72	∞	0	23.41	∞
J	∞	∞	∞	∞	∞	∞	∞	∞	∞	23.4	0	5.3
K (12)	∞	∞	∞	∞	20.11	∞	∞	∞	∞	∞	5.38	0

Selanjutnya dibuat suatu pemisalan untuk memudahkan proses perhitungan. Misalnya dibuat pemisalan sebagai berikut :
M = Matriks ; i = Kolom ; j = Baris ; n = Jumlah titik/ *vertex* ; k = Perulangan ke-n

Setelah melakukan proses pemisalan variabel yang dibutuhkan, dilanjutkan dengan melakukan perhitungan shortest path dengan menggunakan persamaan :M[i][j] = min(M[i][j], M[i][k] + M[k][j]). Jumlah titik/ vertex pada graf adalah 12 (n = 12) sehingga nilai k = 12.

Iterasi pertama untuk k = 1

Untuk i = 1, j = {1.....12}

M[1][1] = min (0, 0+0)→jarak= 0
M[1][2] = min (23, 0+23)→jarak= 23
M[1][3] = min (∞, 0+∞)→jarak=∞
M[1][4] = min (∞, 0+∞)→jarak=∞
M[1][5] = min (∞, 0+∞)→jarak=∞
M[1][6] = min (∞, 0+∞)→jarak=∞
M[1][7] = min (∞, 0+∞)→jarak=∞
M[1][8] = min (∞, 0+∞)→jarak=∞
M[1][9] = min (31.3, 0+31.3)→jarak= 31.3
M[1][10] = min (∞, 0+∞)→jarak=∞
M[1][11] = min (∞, 0+∞)→jarak=∞
M[1][12] = min (∞, 0+∞) →jarak = ∞

Untuk i = 2, j = {1.....12}

M[2][1] = min (23, 23+0) →jarak = 23
M[2][2] = min (0, 23+23) →jarak = 0
M[2][3] = min (9, 23+∞) →jarak = 9
M[2][4] = min (∞, 23+∞) →jarak = ∞
M[2][5] = min (∞, 23+∞) →jarak = ∞
M[2][6] = min (∞, 23+∞) →jarak = ∞
M[2][7] = min (∞, 23+∞) →jarak = ∞
M[2][8] = min (∞, 23+∞) →jarak = ∞
M[2][9] = min (∞, 23+31.3) →jarak = 54.30
M[2][10] = min (∞, 23+∞) →jarak = ∞
M[2][11] = min (∞, 23+∞) →jarak = ∞
M[2][12] = min (∞, 23+∞) →jarak = ∞

Untuk i = 3, j = {1.....12}

M[3][1] = min (∞,∞+0) →jarak = ∞
M[3][2] = min (9, ∞+23) →jarak = 9
M[3][3] = min (0, ∞+∞) →jarak = 0
M[3][4] = min (7.81, ∞+∞) →jarak = 7.81
M[3][5] = min (∞,∞+∞)→jarak = ∞
M[3][6] = min (∞,∞+∞)→jarak = ∞
M[3][7] = min (∞,∞+∞)→jarak = ∞
M[3][8] = min (∞,∞+∞)→jarak = ∞
M[3][9] = min (∞,∞+31.3) →jarak = ∞
M[3][10] = min (∞,∞+∞)→jarak = ∞
M[3][11] = min (∞,∞+∞)→jarak = ∞
M[3][12] = min (∞,∞+∞)→jarak = ∞

Untuk i = 4, j = {1.....12}

M[4][1] = min (∞,∞+0) →jarak = ∞
M[4][2] = min (∞, ∞+23) →jarak = ∞
M[4][3] = min (7.81, ∞+∞) →jarak = 7.81
M[4][4] = min (0, ∞+∞) →jarak = 0
M[4][5] = min (16.57,∞+∞) →jarak = 16.57
M[4][6] = min (∞,∞+∞)→jarak = ∞
M[4][7] = min (∞,∞+∞)→jarak = ∞
M[4][8] = min (∞,∞+∞)→jarak = ∞
M[4][9] = min (∞,∞+31.3) →jarak = ∞
M[4][10] = min (∞,∞+∞)→jarak = ∞
M[4][11] = min (∞,∞+∞)→jarak = ∞
M[4][12] = min (∞,∞+∞)→jarak = ∞

Untuk i = 5, j = {1.....12}

M[5][1] = min (∞,∞+0) →jarak = ∞
M[5][2] = min (∞, ∞+23) →jarak = ∞
M[5][3] = min (∞, ∞+∞) →jarak = ∞
M[5][4] = min (16.57, ∞+∞) →jarak = 16.57
M[5][5] = min (0,∞+∞) →jarak = 0
M[5][6] = min (6.79,∞+∞) →jarak = 6.79
M[5][7] = min (∞,∞+∞)→jarak = ∞
M[5][8] = min (∞,∞+∞)→jarak = ∞
M[5][9] = min (∞,∞+31.3) →jarak = ∞
M[5][10] = min (∞,∞+∞)→jarak = ∞
M[5][11] = min (∞,∞+∞) jarak = ∞
M[5][12] = min (20.11,∞+∞) →jarak = 20.11

Untuk i = 6, j = {1.....12}

M[6][1] = min (∞,∞+0) →jarak = ∞
M[6][2] = min (∞, ∞+23) →jarak = ∞
M[6][3] = min (∞, ∞+∞) →jarak = ∞
M[6][4] = min (∞, ∞+∞) →jarak = ∞
M[6][5] = min (6.59,∞+∞) →jarak = 6.59
M[6][6] = min (0,∞+∞) →jarak = 0
M[6][7] = min (6.59,∞+∞) →jarak = 6.59
M[6][8] = min (∞,∞+∞)→jarak = ∞
M[6][9] = min (∞,∞+31.3) →jarak = ∞
M[6][10] = min (∞,∞+∞)→jarak = ∞
M[6][11] = min (∞,∞+∞)→jarak = ∞
M[6][12] = min (20.11,∞+∞) →jarak = ∞

Untuk i = 7, j = {1.....12}

M[7][1] = min (∞,∞+0) →jarak = ∞
M[7][2] = min (∞, ∞+23) →jarak = ∞
M[7][3] = min (∞, ∞+∞) →jarak = ∞
M[7][4] = min (∞, ∞+∞) →jarak = ∞
M[7][5] = min (∞,∞+∞)→jarak = ∞
M[7][6] = min (6.59,∞+∞) →jarak

Untuk i = 8, j = {1.....12}

M[8][1] = min (∞,∞+0) →jarak = ∞
M[8][2] = min (∞, ∞+23) →jarak = ∞
M[8][3] = min (∞, ∞+∞) →jarak = ∞
M[8][4] = min (∞, ∞+∞) →jarak = ∞
M[8][5] = min (∞,∞+∞)→jarak = ∞
M[8][6] = min (∞,∞+∞)→jarak = ∞

= 6.59
M[7][7] = min (0,∞+∞) →jarak = 0
M[7][8] = min (6.10,∞+∞) →jarak = 6.10
M[7][9] = min (∞,∞+31.3) →jarak = ∞
M[7][10] = min (∞,∞+∞)→jarak = ∞
M[7][11] = min (∞,∞+∞)→jarak = ∞
M[7][12] = min (∞,∞+∞)→jarak = ∞

Untuk i = 9, j = {1.....12}

M[9][1] = min (31.30, 31.30+0) →jarak = 31.30
M[9][2] = min (∞,31.30+23) →jarak = 54.30
M[9][3] = min (∞,31.30+∞) →jarak = ∞
M[9][4] = min (∞,31.30+∞) →jarak = ∞
M[9][5] = min (∞,31.30+∞) →jarak = ∞
M[9][6] = min (∞,31.30+∞) →jarak = ∞
M[9][7] = min (∞,31.30+∞) →jarak = ∞
M[9][8] = min (6.57, 31.30+∞) →jarak = 6.57
M[9][9] = min (0, 31.30+31.3) →jarak = 0
M[9][10] = min (∞,31.30+∞) →jarak = ∞
M[9][11] = min (∞,31.30+∞) →jarak = ∞
M[9][12] = min (∞,31.30+∞) →jarak = ∞

M[8][7] = min (6.10,∞+∞) →jarak = 6.10
M[8][8] = min (0,∞+∞) □jarak = 0
M[8][9] = min (6.57,∞+31.3) →jarak = 6.57
M[8][10] = min (15.72,∞+∞) →jarak = 15.72
M[8][11] = min (∞,∞+∞)→jarak = ∞
M[8][12] = min (∞,∞+∞)→jarak = ∞

Untuk i = 10, j = {1.....12}

M[10][1] = min (∞,∞+0) →jarak = ∞
M[10][2] = min (∞, ∞+23) →jarak = ∞
M[10][3] = min (∞, ∞+∞) →jarak = ∞
M[10][4] = min (∞, ∞+∞) →jarak = ∞
M[10][5] = min (∞,∞+∞)→jarak = ∞
M[10][6] = min (∞,∞+∞)→jarak = ∞
M[10][7] = min (∞,∞+∞)→jarak = ∞
M[10][8] = min (15.72,∞+∞) →jarak = 15.72
M[10][9] = min (∞,∞+31.3) →jarak = ∞
M[10][10] = min (0,∞+∞) →jarak = 0
M[10][11] = min (23.49,∞+∞) →jarak = 23.49
M[10][12] = min (∞,∞+∞)→jarak = ∞

Untuk i = 11, j = {1.....12}

M[11][1] = min (∞,∞+0) →jarak = ∞
M[11][2] = min (∞, ∞+23) →jarak = ∞
M[11][3] = min (∞, ∞+∞) →jarak = ∞
M[11][4] = min (∞, ∞+∞) →jarak = ∞
M[11][5] = min (∞,∞+∞)→jarak = ∞
M[11][6] = min (∞,∞+∞)→jarak = ∞
M[11][7] = min (∞,∞+∞)→jarak = ∞
M[11][8] = min (∞,∞+∞)→jarak = ∞
M[11][9] = min (∞,∞+31.3) →jarak = ∞
M[11][10] = min (23.49,∞+∞) →jarak = 23.49
M[11][11] = min (0,∞+∞) →jarak = 0
M[11][12] = min (5.38,∞+∞) →jarak = 5.38

Untuk i = 12, j = {1.....12}

M[12][1] = min (∞,∞+0) →jarak = ∞
M[12][2] = min (∞, ∞+23) →jarak = ∞
M[12][3] = min (∞, ∞+∞) →jarak = ∞
M[12][4] = min (∞, ∞+∞) →jarak = ∞
M[12][5] = min (20.11,∞+∞) →jarak = 20.11
M[12][6] = min (∞,∞+∞)→jarak = ∞
M[12][7] = min (∞,∞+∞)→jarak = ∞
M[12][8] = min (∞,∞+∞)→jarak = ∞
M[12][9] = min (∞,∞+31.3) →jarak = ∞
M[12][10] = min (∞,∞+∞)→jarak = ∞
M[12][11] = min (5.38,∞+∞) □jarak = 5.38
M[12][12] = min (0,∞+∞) →jarak = 0

Iterasi dihitung hingga iterasi k = {2…12}.Berikut merupakan hasil iterasi pertama dan diperoleh bobot dalam matriks jarak seperti tabel berikut :

Tabel 2. Jarak antar *vertex* setelah iterasi pertama

Jarak	Start (1)	A (2)	B (3)	C (4)	D (5)	E (6)	F (7)	G (8)	H (9)	I (10)	J (11)	K (12)
Start(1)	0	23	∞	∞	∞	∞	∞	∞	31.3	∞	∞	∞
A (2)	23	0	9	∞	∞	∞	∞	∞	54.3	∞	∞	∞
B (3)	∞	9	0	7.81	∞	∞	∞	∞	∞	∞	∞	∞
C (4)	∞	∞	7.81	0	16.57	∞	∞	∞	∞	∞	∞	∞
D (5)	∞	∞	∞	16.57	0	6.59	∞	∞	∞	∞	∞	20.11
E (6)	∞	∞	∞	∞	6.79	0	6.59	∞	∞	∞	∞	∞
F (7)	∞	∞	∞	∞	∞	6.59	0	6.1	∞	∞	∞	∞
G (8)	∞	∞	∞	∞	∞	∞	6.1	0	6.57	15.72	∞	∞
H (9)	31.3	54.3	∞	∞	∞	∞	∞	6.57	0	∞	∞	∞
I (10)	∞	∞	∞	∞	∞	∞	∞	15.72	∞	0	23.49	∞
J (11)	∞	∞	∞	∞	∞	∞	∞	∞	∞	23.49	0	5.38
K (12)	∞	∞	∞	∞	20.11	∞	∞	∞	∞	∞	5.38	0

Tabel 3. Jarak antar *vertex* setelah iterasi keduabelas

Jarak	**Start (1)**	**A (2)**	**B (3)**	**C (4)**	**D (5)**	**E (6)**	**F (7)**	**G (8)**	**H (9)**	**I (10)**	**J (11)**	**K (12)**
Start (1)	0.00	23.00	32.00	39.81	56.38	50.56	43.97	37.87	31.30	53.59	77.08	76.49
A (2)	23.00	0.00	9.00	16.81	33.38	39.97	46.56	52.66	54.30	68.38	58.87	53.49
B (3)	32.00	9.00	0.00	7.81	24.38	30.97	37.56	43.66	50.23	59.38	49.87	44.49
C (4)	39.81	16.81	7.81	0.00	16.57	23.16	29.75	35.85	42.42	51.57	42.06	36.68
D (5)	56.38	33.38	24.38	16.57	0.00	6.59	13.18	19.28	25.85	35.00	25.49	20.11
E (6)	50.56	40.17	31.17	23.36	6.79	0.00	6.59	12.69	19.26	28.41	32.28	26.90
F (7)	43.97	46.76	37.76	29.95	13.38	6.59	0.00	6.10	12.67	21.82	38.87	33.49
G (8)	37.87	52.86	43.86	36.05	19.48	12.69	6.10	0.00	6.57	15.72	39.21	39.59
H (9)	31.30	54.30	50.43	42.62	26.05	19.26	12.67	6.57	0.00	22.29	45.78	46.16
I (10)	53.59	68.58	59.58	51.77	35.20	28.41	21.82	15.72	22.29	0.00	23.49	28.87
J (11)	77.08	58.87	49.87	42.06	25.49	32.08	38.67	39.21	45.78	23.49	0.00	5.38
K (12)	76.49	53.49	44.49	36.68	20.11	26.70	33.29	39.39	45.96	28.87	5.38	0.00

Perhitungan terus dilakukan hingga iterasi keduabelas. Tabel 3. merupakan tabel jarak antar vertex setelah iterasi keduabelas.

Berdasarkan tabel jarak vertex setelah iterasi keduabelas, dapat terlihat keseluruhan jarak dari masing-masing vertex setelah mengalami proses iterasi dan perhitungan dengan menggunakan algoritma Floyd-Warshall. Jalur terpendek untuk setiap blok dapat digambarkan dengan jelas. Sebagai contoh jalur terpendek dari start menuju blok F adalah **Start – H – G – F** dengan jarak **43.97.**

4. Kesimpulan

Kesimpulan yang dapat diambil dalam penelitian ini adalah :

1. Telah berhasil dibentuk jalur terpendek pada tata letak parkir.
2. Algoritma *Floyd-Warshall* dapat menyelesaikan permasalahan jalur terpendek pada tata letak parkir dengan menghitung jarak seluruh jalur/ lintasan yang ada antar blok parkir.

References

R. Kumar dn M. Kumar (2010). Exploring Genetic Algorithm for Shortest Path Optimization in Data Networks. Global Journal of Computer Science nd Technology. Vol 10.

Foulds (1992). Graph Theory Applications. Springer – Verlag, New York.

F. Saptono, I. Mutakhiroh, T. Hidayat, dan A. Fauziyah (2007). Perbandingan Performansi Algoritma Genetika dan Algoritma Semut untuk Penyelesaian Shortest Path Problem.Seminar Nasional Sistem dan Informatika. Bali. 16 November 2007.

Nugroho, Yohanes. Liem, Inggriani. 2003. Algoritma Dan Pemrograman : Bagian-7 Graph. Makalah IF6181-Bagian-7 Institut Teknologi Bandung.

Saputra, Ragil. 2011. Sistem Informasi Geografis Pencarian Rute Optimum Obyek Wisata Kota Yogyakarta Dengan Algoritma Floyd-Warshall. Program Studi Teknik Informatika FMIPA UNDIP. Jurnal Matematika Vol. 4, No. 1, Aprill 2011 : 19-24.

Aplikasi Pencarian Rute Optimal Menggunakan Metode Transitive Closure. 2008. Proceeding, Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2008). Auditorium Universitas Gunadarma, Depok, 20-21 Agustus 2008.

Diaz Novandi, Aprian. 2007. Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall dalam Penentuan Lintasan Terpendek (Single Pair Shortest Path). Makalah IF2251 Strategi Algoritmik Tahun 2007, Bandung.

Budiarsyah , Dibi Khairurrazi . 2010. Algoritma Djikstra, Bellman-Ford, Dan Floyd-Warshall Untuk Mencari Rute Terpendek Dari Suatu Graf. Makalah Strukdis 2010 , Bandung.

Kamayudi, Apri. 2006. Studi dan Implementasi Algoritma Djikstra, Bellman-Ford dan Floyd-Warshall dalam menangani masalah lintasan terpendek dalam Graf. Program Studi Teknik Informatika, Institut Teknologi Bandung.

Ajeng F.S., Tari T., Eka D. 2013. Algoritma Floyd Warshall Untuk Menentukan Jalur Terpendek Evakuasi Tsunami di Kelurahan Sanur. Jurusan Matematika FMIPA Universitas Udayana. E-Jurnal Matematika Vol. 2, No. 1, Januari 2013, 1-5

Adnyana, Benny. 2011. Optimasi Penentuan Tata Letak Parkir STIKOM Bali Menggunakan Algoritma Ant Colony System. Program Studi Sistem Komputer, STMIK STIKOM Bali