PENERAPAN ALGORITMA FLOYD WARSHALL UNTUK PENYELESAIAN KASUS MENGGUNAKAN PROGRAM C++

MAKALAH

Disusun Untuk Memenuhi Tugas Matakuliah Teori Graf

Oleh :

Akrom Khasani (161351028)

Anneke Gystary Pratiwi ( )

Denada Anggia Putri ( )

Indra Setiaji ( )

Yudiana Muharam ( )

JURUSAN TEKNIK INFORMATIKA

STT. WASTUKANCANA PURWAKARTA

2017

BAB I
DASAR TEORI

1.1 Pengertian Algoritma Floyd Warshall

Salah satu persoalan optimasi yang sering ditemui dalam kehidupan sehari-hari adalah pencarian lintasan terpendek (shortest path). Berbagai kalangan menemui permasalahan serupa dengan variasi yang berbeda, contohnya seorang pengemudi yang mencari jalur terpendek dari tempat asal ke tempat tujuan, pengantar pesanan makanan cepat saji yang juga mencari jalur terpendek dari tempat asal ke tempat tujuan, dan juga seorang desainer jaringan komputer yang harus mendesain skema perutean pada jaringan yang dia tangani agar memaksimalkan performa jaringan dan meminimalkan beban yang harus ditangani oleh jaringan tersebut.

Seiring dengan waktu yang berjalan dan juga perkembangan ilmu pengetahuan dan teknologi, permasalahan pencarian lintasan terpendek ini telah terpecahkan dengan berbagai algoritma. Salah satu algoritma yang populer yang memecahkan persoalan pencarian lintasan terpendek tersebut adalah Floyd-Warshall.

Algoritma Floyd-Warshall adalah salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

Hal yang membedakan pencarian solusi menggunakan pemrograman dinamis dengan algoritma greedy adalah bahwa keputusan yang diambil pada tiap tahap pada algoritma greedy hanya berdasarkan pada informasi yang terbatas sehingga nilai optimum yang diperoleh pada saat itu. Jadi pada algoritma greedy, kita tidak memikirkan konsekuensi yang akan terjadi seandainya kita memilih suatu keputusan pada suatu tahap.

Dalam beberapa kasus, algoritma greedy gagal memberikan solusi terbaik karena kelemahan yang dimilikinya tadi. Di sinilah peran pemrograman dinamis yang mencoba untuk memberikan solusi yang memiliki pemikiran terhadap konsekuensi yang ditimbulkan dari pengambilan keputusan pada suatu tahap. Pemrograman dinamis mampu mengurangi pengenumerasian keputusan yang tidak mengarah ke solusi.

Prinsip yang dipegang oleh pemrograman dinamis adalah prinsip optimalitas, yaitu jika solusi total optimal, maka bagian solusi sampai suatu tahap (misalnya tahap ke-i) juga optimal.

Algoritma Floyd-Warshall membandingkan semua kemungkinan lintasan pada graf untuk setiap sisi dari semua simpul. Menariknya, algoritma ini mampu mengerjakan proses perbandingkan ini sebanyak V3 kali(bandingkan dengan kemungkinan jumlah sisi sebanyak V2 (kuadrat jumlah simpul) pada graf, dan setiap kombinasi sisi diujikan). Hal tersebut bisa terjadi karena adanya perkiraan pengambilkan keputusan (pemilihan jalur terpendek) pada setiap tahap antara dua simpul, hingga perkiraan tersebut diketahui sebagai nilai optimal.

Misalkan terdapat suatu graf G dengan simpul-simpul V yang masing-masing bernomor 1 s.d. N (sebanyak N buah). Misalkan pula terdapat suatu fungsi shortestPath(i, j, k) yang mengembalikan kemungkinan jalur terpendek dari i ke j dengan hanya memanfaatkan simpul 1 s.d. k sebagai titik perantara. Tujuan akhir penggunaan fungsi ini adalah untuk mencari jalur terpendek dari setiap simpul i ke simpul j dengan perantara simpul 1 s.d. k+1.

Ada dua kemungkinan yang terjadi:

1. Jalur terpendek yang sebenarnya hanya berasal dari simpul-simpul yang berada antara 1 hingga k.
2. Ada sebagian jalur yang berasal dari simpul-simpul i s.d. k+1, dan juga dari k+1 hingga j

Perlu diketahui bahwa jalur terpendek dari i ke j yang hanya melewati simpul 1 s.d. k telah didefinisikan pada fungsi shortestPath(i, j, k) dan telah jelas bahwa jika ada solusi dari i s.d. k+1 hingga j, maka panjang dari solusi tadi adalah jumlah (konkatenasi) dari jalur terpendek dari i s.d. k+1 (yang melewati simpul-simpul 1 s.d. k), dan jalur terpendek dari k+1 s.d. j (juga menggunakan simpul-simpul dari 1 s.d. k).

Algoritma ini bekerja dengan menghitung shortestPath(i,j,1) untuk semua pasangan (i,j), kemudian hasil tersebut akan digunakan untuk menghitung shortestPath(i,j,2) untuk semua pasangan (i,j), dst. Proses ini akan terus berlangsung hingga k = n dan kita telah menemukan jalur terpendek untuk semua pasangan (i,j) menggunakan simpul-simpul perantara.

BAB II

ALGORITMA & PROGRAM

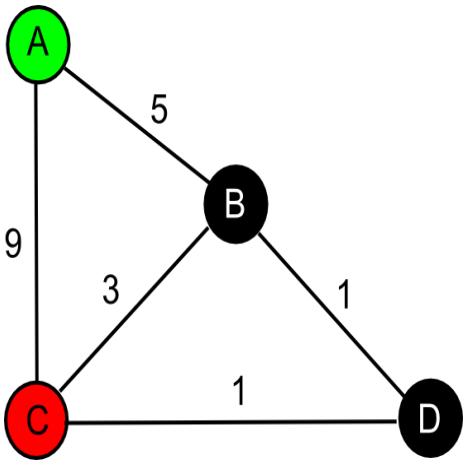
2.1 Algoritma Floyd Warshall

Algoritma Floyd Warshall adalah salah satu varian dari pemrograman dinamis, metode untuk memecahkan masalah pencarian rute terpendek (sama seperti Algoritma Dijkstra).

Metode ini melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Maksudnya, solusi-solusi dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

Algoritma ini juga bisa diterapkan pada sebuah aplikasi pencari rute jalan yang terdekat dari suatu daerah ke daerah lainnya. dengan metode ini hasil yang di dapat bisa lebih optimal namun memerlukan resource yang cukup besar jika dipakai untuk pencarian yang kompleks.

Sebagai contoh adalah mencari rute terpendek dari titik **A** menuju titik **C**dari grafik berikut :



1. Langkah pertama kita jadikan grafik di atas menjadi sebuah tabel atau matriks.

Dari, Ke	A	B	C	D	
A	0	5	9	∞	1
B	5	0	3	1	2
C	9	3	0	1	3
D	∞	1	1	0	4
	1	2	3	4	

2. Kotak abjad berwarna hijau disamping kiri adalah *titik awal* dan kotak abjad berwarna merah yang ada di atas adalah *titik tujuan*-nya. Sedangkan kotak angka hijau dan merah berfungsi untuk menentukan sebuah *index proses* ***(R0=1,R1=2,R2=3,dan R3=4)*** dan memudahkan posisi angka-angka yang ada didalam tabel dengan mengkombinasikan-nya dengan kotak abjad yang sama dengan warnaya. sebagai contoh :

   A(3),B(2) (*titik awal, titik tujuan*) = **9,0**

3. Karena dalam grafik diatas terdapat 4 buah titik, yaitu **A, B, C, D** maka akan ada 5 proses yang akan dilewati yaitu **R0, R1, R2, R3,** dan **R4** sebagai hasil akhir. perlu diingat bahwa hasil dari sebuah proses akan digunakan untuk proses berikutnya.

4. Rumusnya adalah :

   r = Index proses. => **R0** = 1**, R1** = 2**, R2** = 3**, R3** = 4**, R4** adalah hasil akhir.

   S = Titik awal. => A, B, C, dan D *(kotak hijau).*

   E = Titik tujuan. => A, B, C, dan D *(kotak merah).*

   Jika hasil penjumlahan nilai titik awal *S(r)* dan nilai titik tujuan *E(r)* **lebih kecil** daripada nilai jarak yang sebenarnya *S(E),* maka ganti nilai jarak sebenarnya dengan hasil penjumlahan nilai titik awal dan nilai titik tujuan *[S(E) = S(r) + E(r)]* .

5. Berikut perhitungannya :

a) proses pertama yaitu **R0.**

r = R0 = 1

S = **{** A(r) = 0 **;** B(r) = 5 **;** C(r) = 9 **;** D(r) = *tak hingga* **}**

E = **{** A(r) = 0 **;** B(r) = 5 **;** C(r) = 9 **;** D(r) = *tak hingga* **}**

- Titik awal A ke titik tujuan A=> *A(A) = 0*

  A(r) + A(r) = 0 + 0 = 0 => 0 sama dengan *A(A)* <tidak diganti>
- Titik awal A ke titik tujuan B => *A(B) = 5*

  A(r) + B(r) = 0 + 5 = 5 => 5 sama dengan *A(B)* <tidak diganti>
- Titik awal A ke titik tujuan C => *A(C) = 9*

  A(r) + C(r) = 0 + 9 = 9 => 9 sama dengan *A(C)* <tidak diganti>
- Titik awal A ke titik tujuan C => *A(D) = tak hingga*

  A(r) + D(r) = 0 + *tak hinggga = tak hingga => tak hingga* sama dengan *A(D)* <tidak diganti>

- Titik awal B ke titik tujuan A => *B(A) = 5*

  B(r) + A(r) = 5 + 0 = 5 => 0 sama dengan *B(A)* <tidak diganti>
- Titik awal B ke titik tujuan B => *B(B) = 0*

  B(r) + B(r) = 5 + 5 = 10 => 10 lebih besar dari *B(B)* <tidak diganti>
- Titik awal B ke titik tujuan C => *B(C) = 3*

  B(r) + C(r) = 5 + 9 = 14 => 14 lebih besar dari *B(C)* <tidak diganti>
- Titik awal B ke titik tujuan C => *B(D) = 1*

  B(r) + D(r) = 5 + *tak hinggga = tak hingga => tak hingga* lebih besar dari *B(D)* <tidak diganti>

- Titik awal C ke titik tujuan A => *C(A) = 9*

  C(r) + A(r) = 9 + 0 = 9 => 9 sama dengan *C(A)* <tidak diganti>
- Titik awal C ke titik tujuan B => *C(B) = 3*

  C(r) + B(r) = 9 + 5 = 14 => 14 lebih besar dari *C(B)* <tidak diganti>
- Titik awal C ke titik tujuan C => *C(C) = 0*

  C(r) + C(r) = 9 + 9 = 18 => 18 lebih besar dari *C(C)* <tidak diganti>

- Titik awal C ke titik tujuan C => *C(D) = 1*

  C(r) + D(r) = 9 + *tak hinggga* = *tak hingga* => *tak hingga* lebih besar dari *C(D)* <tidak diganti>
- Titik awal D ke titik tujuan A => *D(A) = tak hingga*

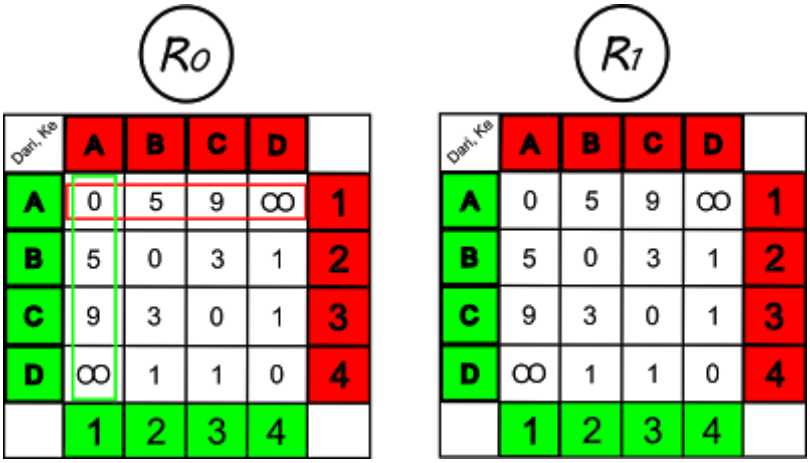
  D(r) + A(r) = *tak hingga* + 0 = *tak hingga* => *tak hingga* sama dengan *D(A)* <tidak diganti>
- Titik awal D ke titik tujuan B => *D(B) = 1*

  D(r) + B(r) = *tak hingga* + 5 = *tak hingga* => *tak hingga* lebih besar dari *D(B)* <tidak diganti>
- Titik awal D ke titik tujuan C => *D(C) = 1*

  D(r) + C(r) = *tak hingga* + 9 = *tak hingga* => *tak hingga* lebih besar dari *D(C)* <tidak diganti>
- Titik awal D ke titik tujuan C => *D(D) = 0*

  D(r) + D(r) = *tak hingga* + *tak hinggga* = *tak hingga* => *tak hingga* lebih besar dari *D(D)* <tidak diganti>

Demikianlah proses dari **R0**. Tidak ada yang diganti karna tidak ada hasil penjumlahan yang menunjukkan lebih dari nilai jarak sebenarnya *S(E).* Untuk proses selanjutnya sampai proses hasil akhir **R4** kami presentasikan melalui gambar di bawah ini.



R1

Dari, Ke	A	B	C	D	
A	0	5	9	∞	1
B	5	0	3	1	2
C	9	3	0	1	3
D	∞	1	1	0	4
	1	2	3	4	

R2

Dari, Ke	A	B	C	D	
A	0	5	8	6	1
B	5	0	3	1	2
C	8	3	0	1	3
D	6	1	1	0	4
	1	2	3	4	

R2

Dari, Ke	A	B	C	D	
A	0	5	8	6	1
B	5	0	3	1	2
C	8	3	0	1	3
D	6	1	1	0	4
	1	2	3	4	

R3

Dari, Ke	A	B	C	D	
A	0	5	8	6	1
B	5	0	3	1	2
C	8	3	0	1	3
D	6	1	1	0	4
	1	2	3	4	

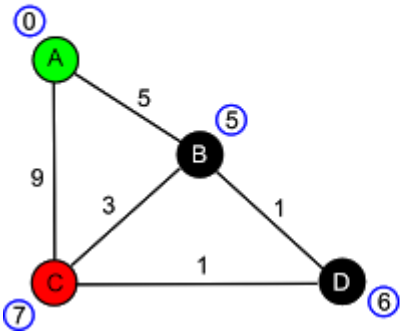
R3

Dari, Ke	A	B	C	D	
A	0	5	8	6	1
B	5	0	3	1	2
C	8	3	0	1	3
D	6	1	1	0	4
	1	2	3	4	

R4

Dari, Ke	A	B	C	D	
A	0	5	7	6	1
B	5	0	2	1	2
C	7	2	0	1	3
D	6	1	1	0	4
	1	2	3	4	

Dari hasil tabel matriks **R4** dapat diketahui rute terpendek manakah yang harus ditempuh dari titik **A** menuju ke titik **C**. kemudian kita tulis nilai-nilai yang ada di table matriks **R4** disamping titik dalam grafik sesuai dengan abjad mereka.



R4

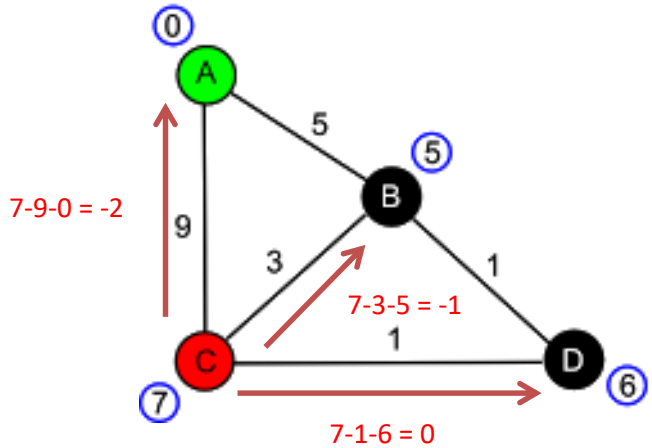
Dari, Ke	A	B	C	D	
A	0	5	7	6	1
B	5	0	2	1	2
C	7	2	0	1	3
D	6	1	1	0	4
	1	2	3	4	

Cara mengetahui rute yang harus dilewati

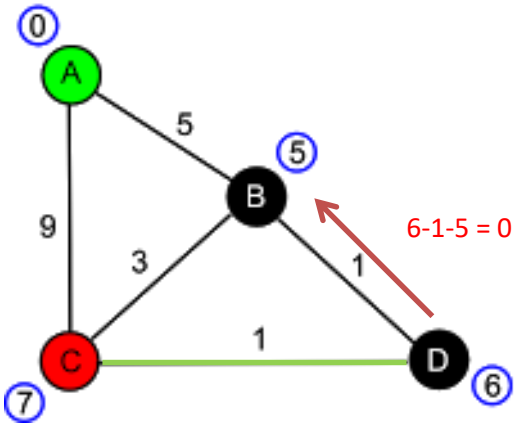
Untuk mengetahui rute manakah yang harus dilewati adalah dengan menelusuri kembali dari titik tujuan ke titik awal. tuliskan *label jarak* di samping setiap titik.

Titik mana sajakah yang dapat dihungi langsung dari titik **C** ?, Yakni titik **A, B** dan **D**. maka, untuk menentukan titik manakah yang seharusnya dilewati adalah dengan cara mengurangkan *label jarak* titik **C** dengan jaraknya ke titik tujuan serta *label jarak* titik tersebut. jika hasilnya kurang dari **0** maka titik tersebut tidak layak untuk dilewati, dan jika hasilnya lebih dari **0** serta lebih mendekati **0** maka titik tersebut yang seharusnya dilewati.

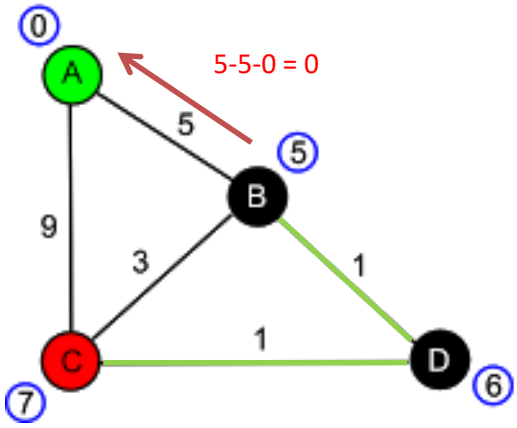
Langkah pertama :



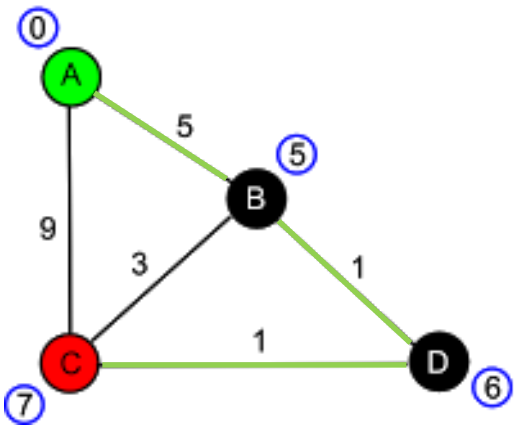
Langkah kedua :



Langkah ketiga :



Dengan begitu diketahui rute yang harus dilewati dan memiliki jarak terpendek dari titik **A** menuju titik **C** adalah **A** -> **B** -> **D** -> **C**



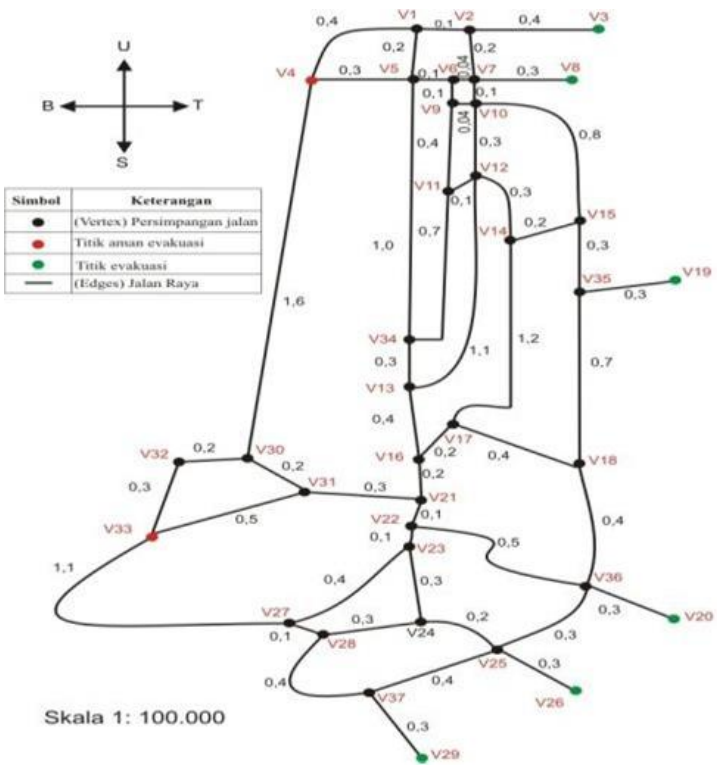
2.2 Contoh Kasus

Jurnal Riset Komputer (JURIKOM), Vol. 3 No. 6, Desember 2016 ISSN 2407-389X (Media Cetak) Hal : 20-24

Shortest Path Problem

Jalur pendek adalah suatu jaringan pengarahan perjalanan dimana seseorang pengarah jalan ingin menentukan jalur terpendek antara dua perusahaan, berdasarkan beberapa jalur alternatif yang tersedia, dimana titik tujuan hanya satu.

Data kuantitaf dalam penelitian ini adalah hasil pengukuran jarak jalur-jalur dari titik evakuasi menuju zona aman. Data kualitatif dalam penelitian ini berupa peta evakuasi Tsunami yang diperolah dari Badan penaggulangan bencana daerah (BPBD) Propinsi Bali. Teknik pengumpulan data yang digunakan dalam penelitian ini meliputi, observasi, dokumentasi, literatur, dan wawancara. Variabel yang digunakan dalam penelitian ini adalah jarak dari setiap jalur-jalur yang mungkin dapat dilalui dari pantai-pantai yang berada dikelurahan Sanur yaitu Pantai Segara (V3), Pantai Shindu (V8), Pantai Karang (V19), Pantai Duyung (V20), Pantai Semawang (V26), Pantai Cemara (V29), untuk menuju zona aman di Puskesmas III Denpasar Selatan (V4), dan SMK Negeri 3 Denpasar (V33).



Gambar 1 : Representasi Peta Kelurahan Sanur kedalam bentuk graf (Satuan Kilometer)

2.3 Penyelesaian Kasus

Karena pada kasus di atas merupakan masalah yang kompleks, maka akan membutuhkan resource yang besar apabila harus dihitung secara manual. Untuk itu akan lebih mudah penyelesaiannya jika menggunakan program. Di bawah ini adalah Script program C++ yang kami buat :

DAFTAR PUSTAKA

https://achmad-asrori.blogspot.co.id/2013/01/algoritma-dijkstra.html

https://achmad-asrori.blogspot.co.id/2013/01/algoritma-floyd-warshall.html

http://ankitstar.blogspot.co.id/2011/06/floyd-warshall-algortithm.html