9. Algoritma Path

Oleh : Ade Nurhopipah

Pokok Bahasan :

1. Algoritma Fleury
2. Algoritma Shortest Path
3. Studi Kasus

Sumber :

Aldous, Joan M. ,Wilson, Robin J. 2004. *Graph and Applications*. Springer: UK.

Pada bab ini, kita akan mempelajari beberapa masalah yang melibatkan pencarian sebuah path yang memiliki beberapa sifat khusus dari graph atau digraph yang disajikan. Pertama, kita akan mempelajari algoritma Fleury untuk menemukan trail Euler pada graph Euler yang diberikan. Kemudian kita akan beralih pada graph dan digraph berbobot dan menjelaskan algoritma untuk menemukan sebuah path terpendek (shortest path) dari suatu vertex ke vertex lain. Terakhir kita akan meninjau studi kasus masalah rute optimal untuk *Chinese postman problem*.

Algoritma Fleury

Pada bab sebelumnya kita mempelajari bagaimana membangun sebuah trail Euler pada graph Euler dengan cara menyatukan beberapa cycle. Cara lain untuk membangun trail Euler adalah dengan menggunakan algoritma Fleury sebagai berikut.

Algoritma Fleury
Mulai dengan sebuah graph Euler G

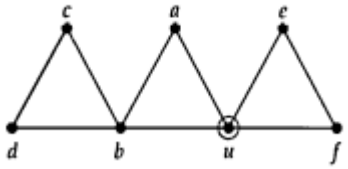
Langkah 1
Pilih vertex awal
Langkah 2
Di mulai dari vertex tersebut lintasi edge yang mungkin, pemilihan sebuah jembatan dilakukan hanya jika tidak ada alternatif lain. Kemudian hapus edge tersebut dan vertex yang terisolasi.

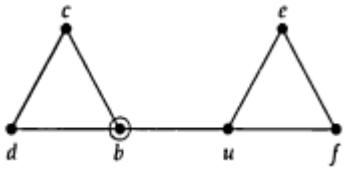
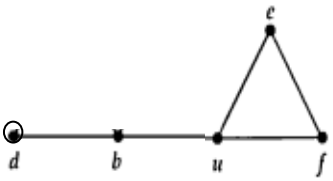
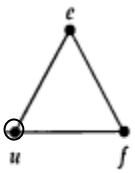
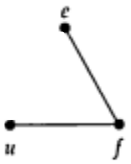
Ulangi langkah 2 sampai tidak ada lagi edge, dan berhentilah.

Ingat kembali bahwa jembatan adalah edge yang apa bila dihapus menyebabkan suatu graph tidak terhubung.

Contoh 9.1

Kita ilustrasikan penggunaan algoritma Fleury dengan menerapkannya pada graph berikut.

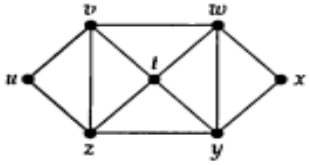


Langkah 1 Awali dengan vertex *u*. Langkah 2 Lintasi edge *ua* dan hapuslah edge tersebut. Lintasi edge *ab* dan hapuslah edge tersebut.	
Langkah 2 Kita tidak bisa memakai edge *bu* karena *bu* adalah sebuah jembatan, jadi kita pilih edge *bc* dan hapuslah edge tersebut. Lintasi edge *cd* dan hapuslah edge tersebut, bersama dengan vertex *c* yang terisolasi.	
Langkah 2 Lintasi edge *db* dan hapuslah edge tersebut, bersama dengan vertex *d* yang terisolasi. Lintasi jembatan *bu* dan hapus jembatan tersebut, bersama dengan vertex *b*	
Langkah 2 Lintasi edge *ue* dan hapus edge tersebut. Lintasi jembatan *ef* dan hapus jembatan tersebut, bersama vertex *e*. Lintasi jembatan *fu* dan hapus jembatan tersebut, bersama vertex *f.*	

Kita telah melintasi semua edge, sehingga kita berhenti dan mendapatkan trail Eulerian *u a b c d b u e f u.*

Latihan 9.1

Gunakan algoritma Fleury untuk memperoleh sebuah trail Euler dari graph berikut, dimulai dengan edge *uv,vz.*



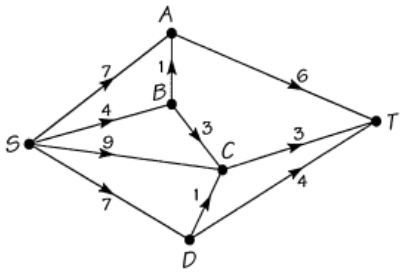
Algoritma Shortest Path

Algoritma path terpendek (*shortest path*) digunakan untuk mencari sebuah jalur terpendek dari sebuah vertex *S* pada sebuah graph atau digraph berbobot ke vertex lain *T.* Kita mulai dari titik *S* dan menghitung jarak terkecil dari *S* ke setiap vertex tetangganya. Pada setiap tahap algoritma, kita memperhatikan semua vertex yang dapat dicapai oleh sebuah edge atau arc dari vertex tersebut. Tandai semua vertex tersebut dengan sebuah label sementara, yang merepresentasikan jarak terdekat dari *S* ke vertex tersebut. Pada akhirnya setiap vertex akan memperoleh label tetap, yang disebut *potensial*, yang merepresentasikan jarak terdekat dari *S* ke vertex tersebut.

Saat *T* telah ditandai sebagai sebuah potensial, kita menemukan jalur terpendek dari *S* ke *T* dengan melacak kembali melalui label-label yang sudah dibuat. Algoritma ini dapat digunakan untuk graph ataupun digraph. Kali ini kita akan ilustrasikan dengan menerapkannya pada digraph.

Contoh 9.2

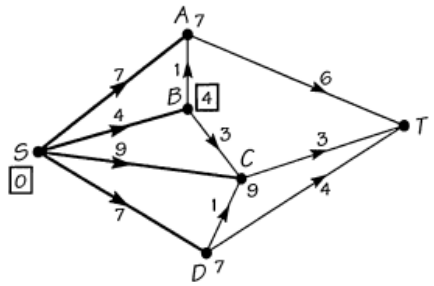
Masalah yang kita miliki adalah mencari jalur terpendek dari *S ke T.* Panjang setiap arc ditunjukan dengan nilai disamping arc. Kita akan menetapkan potensial pada setiap vertex yang diterapkan sepanjang path dari *S* ke *T.* Kita notasikan sebuah potensial dengan nilai di dalam sebuah segiempat. Kita mulai dengan menandai potensial *S* dengan nilai 0.



Ilustrasi terhadap algoritma ini dapat dilakukan dengan menggambar sebuah diagram. Namun begitu, kadang-kadang lebih nyaman bagi kita, untuk memakai metode tabel daripada dengan diagram. Metode tabel memungkinkan setiap langkah algoritma diperlihatkan secara jelas. Metode ini juga membuat pekerjaan lebih akurat dan memudahkan pengecekan untuk contoh dengan skala besar. Metode ini dilakukan dengan membuat tabel yang berkoresponden dengan setiap iterasi.

Iterasi pertama : Vertex yang langsung dicapai dari *S.*

Kita mulai dengan vertex *S* dan menyajikan setiap vertex yang dapat dicapai dari *S* dengan sebuah arc yaitu vertex A,B,C,D. Kita melabeli setiap vertex tersebut dengan jarak sepanjang arc dari S, seperti ditunjukan sebagai berikut.



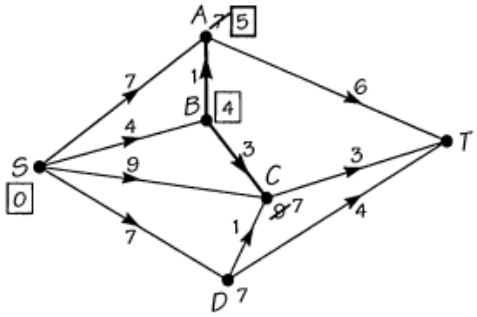
B has potential 4 assigned from S.

Sekarang kita mengambil nilai terkecil dari label vertex-vertex tersebut yang merupakan jarak terkecil dari *S,* dan menandainya sebagai sebuah potensial. Yang terkecil dari ke-empat jarak *SA, SB, SC*, dan *SD* adalah *SB* yang memiliki jarak 4. Tandai label B sebagai potensial. Dengan metode tabel kita dapat merekam informasi yang dihasilkan di iterasi pertama sebagai berikut.

iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	7	[4]	9	7	

Iterasi Kedua : Vertex yang dicapai langsung dari *B.*

Kita mulai dengan vertex B yang baru ditandai sebagai potensial, dan mengulangi prosedur sebelumnya. Dari vertex B, vertex yang dapat dicapai adalah *A* dan *C.* Jarak dari *S* ke vertex *A* dan *C* melalui vertex *B* masing-masing adalah 5 dan 7 seperti yang ditunjukan sebagai berikit.



A has potential 5 assigned from B.

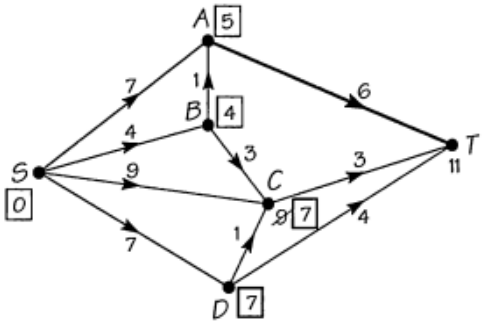
Jarak ke vertex *A* melalui *B* adalah 5. Nilai ini lebih kecil dibandingkan label sebelumnya untuk B yaitu 7. Jadi kita hapus label 7 dan manggantinya dengan 5. Aplikasikan cara yang sama pada vertex *C.* Jika label sebelumnya lebih kecil dari label baru, maka kita membiarkannya, karena kita menandai potensial sebagai jarak terpendek.

Jarak yang ditandai terkecil yang belum menjadi potensial adalah *A,* sehingga kita menandainya sebagai potensial baru seperti ditunjukan sebagai berikut.

iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	~~7~~	[4]	~~9~~	7	
2	B	[5]		7		

Iterasi Ketiga : Vertex yang dapat dicapai secara langsung dari A.

Kita mulai dengan vertex *A,* yang baru ditandai sebagai potensial. Hanya vertex *T* yang dapat dicapai dari vertex *A,* sehingga kita melabeli *T* dengan jarak dari *S* via *A.* Ini dicapai dengan menjumlahkan potensial pada *A* dengan jarak *AT,* yaitu 5+6=11.



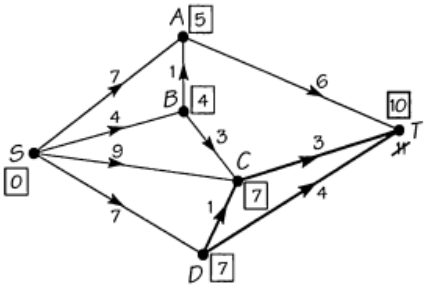
C has potential 7 assigned from *B*.
D has potential 7 assigned from *S*.

Kini ada dua vertex yaitu *C* dan *D* dengan label terkecil 7 sehingga dapat kita jadikan potensial. Karena label pada titik C dan D berasal dari iterasi sebelumnya, maka kita menambahkan busur dari potensial yang ditandai segiempat ke label *C* pada iterasi dua dan ke *D* pada iterasi 1.

iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	~~7~~	[4]	~~9~~	7	
2	B	[5]		7		
3	A			[7]	[7]	11

Iterasi keempat : Vertex yang dapat dicapai secara langsung dari C atau D.

Kita mulai dari vertex C dan D dan melihat vertex yang dapat dicapai dengan sebuah arc yaitu vertex *T.* Path melalui D memiliki total jarak 7+4=11, di mana memiliki nilai yang sama dengan label pada *T* yang berkoresponden dengan path melalui *A*. Namun, path via *C* memiliki jarak 10, dan lebih kecil dari pada path melalui A dan D, sehingga label pada *T* diganti 10, dan *T* dan menjadi potensial seperti ditunjukan sebagai berikut.

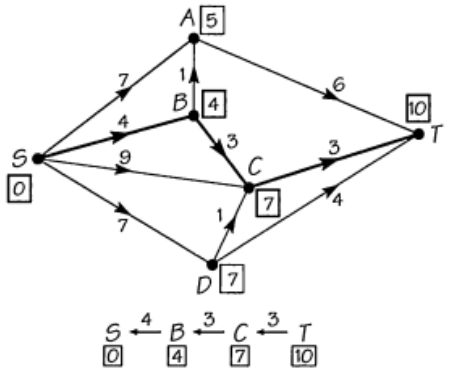


T has potential 10 assigned from C.

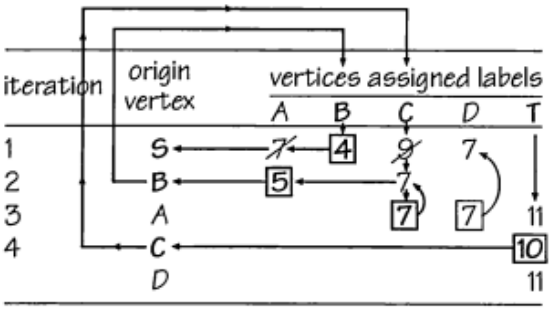
Menggunakan metode tabel, iterasi 4 ditunjukan dengan menambah dua baris untuk potensial C dan D, sebagai berikut.

iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	~~7~~ 5	4	~~9~~ 7	7	
2	B			7	7	
3	A					11
4	C					10
	D					11

Kita telah menemukan jarak terkecil dari S ke T yaitu 10, dan mendapatkan suatu path terpendek. Kita menemukan path dengan melacak balik dari vertex *T*. Pelacakan balik yang dilakukan dari *T* ke *C* bernilai 7 (potensial dari *C),* yang merupakan hasil dari 10 (potensial dari T) dikurangi 3 (panjang arc CT). Setelah itu dilakukan pelacakan balik dari C ke B dan dari B ke S, sehingga path terpendek adalah SBCT.



Menggunakan metode tabel, kita dapat membaca path terpendek dari tabel sebagai berikut.
T memiliki potensial 10 yang ditandai dari C
C memiliki potensial 7 yang ditandai dari B
B memiliki potensial 4 yang ditandai dari S
Jadi
Path terpendek adalah SBCT dengan panjang 10.
Kita melacak path melalui tabel sebagai berikut.



Algoritma shortest path secara formal ditunjukan sebagai berikut.

Algoritma Shortest Path
Mulai :
Tandai potensial 0 untuk *S.*

Langkah umum :
Perhatikan vertex-vertex yang baru ditandai sebagai potensial. Untuk setiap vertex *V* tersebut, perhatikan setiap vertex *W* yang dapat dicapai dari *V* melalui sebuah arc dan tandai W dengan label : (potensial dari *V*) + (jarak *VW*),
kecuali *W* telah memiliki label yang lebih kecil dari iterasi sebelumnya.
Ketika semua vertex *W* telah dilabeli, pilih label vetex terkecil yang belum menjadi potensial, dan jadikan ia potensial.

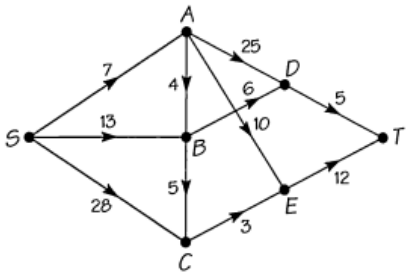
Ulangi langkah umum dengan potensial yang baru.

Berhenti :
Ketika T telah ditandai sebagai potensial.
Jarak terkecil dari S ke T adalah potensial dari T.

Untuk menemukan path terpendek, lakukan penelusuran balik dari T dan masukan sebuah arc VW, dengan (potensial W) – (potensial V) = jarak VW, sampai S dicapai.

Latihan 9.2

Gunakan algoritma shortest path untuk menemukan path terpendek dari S ke T pada graph berikut.



Studi Kasus

Masalah Chinese Postman

Masalah Chinese Postman

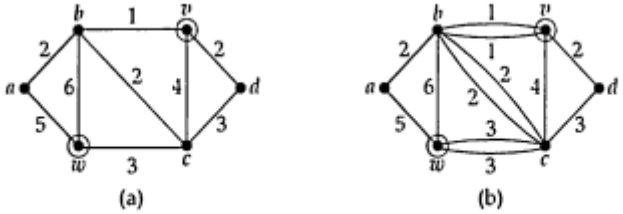
Seorang tukang pos ingin mengirim surat ke semua jalan di areanya dan kemudian kembali ke depot. Bagaimana ia dapat merencanakan sebuah rute, agar mendapatkan total jarak yang paling kecil.

Kata "China" pada masalah ini, tidak mengacu pada tukang pos di daerah Cina, melainkan karena masalah ini disajikan oleh Mei-ko Kwan pada tahun 1962.

Jika peta area berkorespenden dengan sebuah graph Euler, tidak ada kesulitan yang kita hadapi, karena kita hanya harus mencari trail Euler (menggunakan algoritma Fleury, jika perlu) dan trail tersebut melibatkan total jarak. Situasi ini muncul ketika semua vertex memiliki derajat yang genap. Namun yang biasanya terjadi adalah graph representasi rute yang ada bukanlah graph Euler. Oleh karena itu, tukang pos tersebut perlu mengunjungi beberapa bagian rute lebih dari sekali, dan ingin meminimalkan jumlah *retracing*.

Masalah ini secara umum diselesaikan menggunakan algoritma kombinasi algoritma Fleury dan algoritma Shortest Path. Jika graph yang kita miliki bukan graph Euler, maka terdapat vertex dengan derajat ganjil. Menurut lemma handshaking, jumlah vertex yang berderajat ganjil tersebut adalah genap atau sama dengan *2k*. Kita dapat membuat graph Euler dengan menambah edge sepanjang path *k* yang menghubungkan pasangan vertex tersebut. Kita memastikan bahwa kita memperoleh sebuah walk tertutup dengan bobot minimal sehingga jumlah panjang path *k* ini minimal.

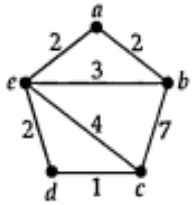
Perhatikan graph dibawah ini yang memiliki vertex berderajat ganjil, yaitu *v* dan *w.* Kita akan mencari path dari kedua pasang vertex, dan menambah edge sepanjang path tersebut sehingga kita menghasilkan graph Euler.



Path terpendek dari *v* ke *w* dengan total bobot 1+2+3=6. Jika kita menambahkan edge pada path ini, kita mendapatkan graph Euler pada gambar (b). Walk tertutup yang diminta dengan total bobot minimum diperoleh dengan menemukan trail Euler dalam graph (dengan algoritma Fleury jika perlu), seperti *abvdcvbcbwcwd*. Edge yang perlu dilewati kembali hanyalah *vb, bc* dan *cw*.

Terdapat pula cara alternatif yaitu kita menemukan sebuah path semi-Euler dari *v* ke *w* (menggunakan modifikasi algoritma Fleury) dan kemudian menemukan path terpendek dari *w* kembali ke *v*. Kombinasi dari path ini adalah trail dengan bobot minimum. Contoh untuk masalah diatas, kita memperoleh rute *vdcvbcwbawcbv*.

Latihan 9.3
Selesaikan masalah Chinese postman pada graph berbobot berikut.



Kini anggaplah masalah tersebut adalah masalah pembersihan jalan atau pembersihan salju, dan kendaraan perlu melalui kedua belah sisi jalan, sekali pada masing-masing arah. Kita dapat mengganti setiap edge dengan dua buah arc, satu pada masing-masing arah. Kita akhirnya memiliki digraph terhubung kuat dimana derajat masuk sama dengan derajat keluar pada setiap vertex. Menurut Teorema 4.2 digraph seperti itu adalah digraph Euler, hingga terdapat trail Euler dengan bobot minimal yang dapat ditemukan dengan modifikasi algoritma Fleury.