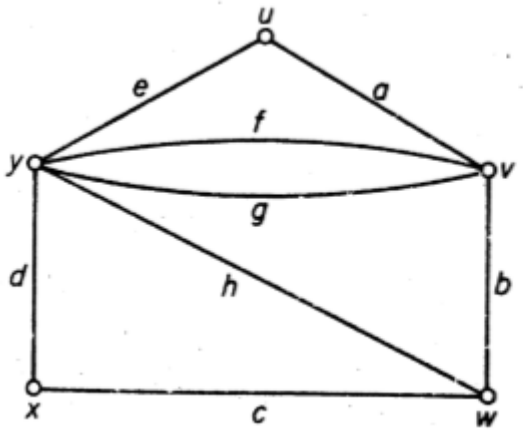
ALUR DAN KONEKSI

Jalan (*walk*) dalam G adalah barisan tidak nol berbatas W= $v_0e_1v_1e_2...e_kv_k$, yang suku-sukunya berupa simpul dan rusuk yang muncul secara bergantian, sedemikian sehingga, untuk 1<=i<=k, akhir e_i adalah v_{i-1} dan v_i. Kita menyebut bahwa W adalah **jalan** dari v_0 ke v_k, atau jalan(v_o,v_k). Simpul v_0 dan v_k masing-masingnya disebut dengan **titik asal** dan **terminal** dari W, dan $v_1,v_2,...,v_{k-1}$ disebut **simpul internal**. Bilangan bulat *k* disebut **panjang** W.

Jika $W=v_0e_1v^1...e_kv_k$ dan $W'=v_ke_{k+1}...e_1v_1$ adalah jalan, maka jalan $v_ke_kv_{k-1}e_{k-1}...e_1v_0$ diperoleh dengan membalikan W dinyatakan dengan W^{-1}. **Section** jalan $W=v_0e_1v_1...e_kv_1$ adalah jalan yang merupakan bagian dari $v_ie_{i+1}v_{i+1}...e_jv_j$ dari suku-suku berurut W.

Dalam sebuah graph sederhana, sebuah jalan $v_0e_1v_1...e_kv_k$ ditentukan oleh barisan $v_0v_1...v_k$. Dengan demikian sebuah graph sederhana dapat disederhanakan menjadi barisan simpul.

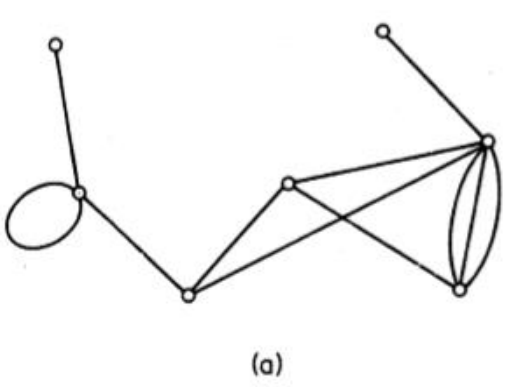
Jika rusuk $e_1,e_2,...,e_k$ dari sebuah jalan W berbeda, maka W disebut **trail**. Jika simpul $v_0,v_1,...,v_k$ dari sebuah jalan W berbeda, maka W disebut **alur** (*path*).



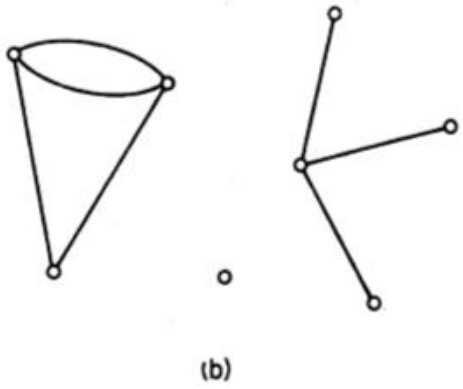
Walk: *uavfyfvgyhwbv*
Trail: *wcxdyhwbvgy*
Path: *xcwhyeuav*

Dua simpul u dan v pada graph G disebut terhubung jika terdapat alur(u,v) dalam G. Jadi jika V dibagi atas *non-empty* V_1, $V_2,...,V_n$ sedemikian sehingga dua simpul u dan v terhubung jika u dan v kepunyaan himpunan V_i yang sama.

Sub graph $G[V_1]$, $G[V_2],...,G[V_n]$ disebut **komponen** jika G mempunyai tepat satu komponen, maka G disebut terhubung, kalau tidak disebut tidak terhubung. Jumlah komponen G dinyatakan dengan $\omega(G)$. Contoh graph terhubung atau tidak terhubung dapat dilihat pada gambar dibawah.



Graph terhububung

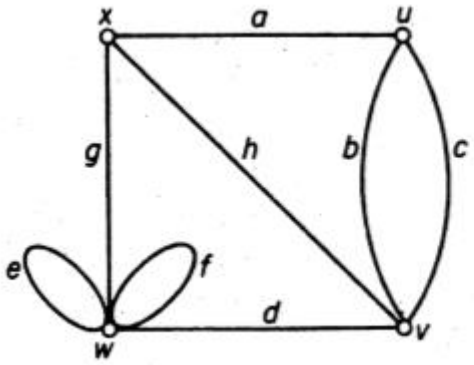


Graph tidak terhubung terdiri dari 3 komponen

SIKLIS

Suatu jalan disebut **tertutup** jika mempunyai panjang positif dimana simpul awal dan akhir sama. Trail tertutup yang simpul awal dan simpul internal berbeda disebut **siklis**.

Siklis dengan pajang k disebut k-siklis, yang panjangnya bisa ganjil atau genap tergantung kepada k genap atau ganjil. Contoh trail tertutup dan siklis seperti gambar berikut:



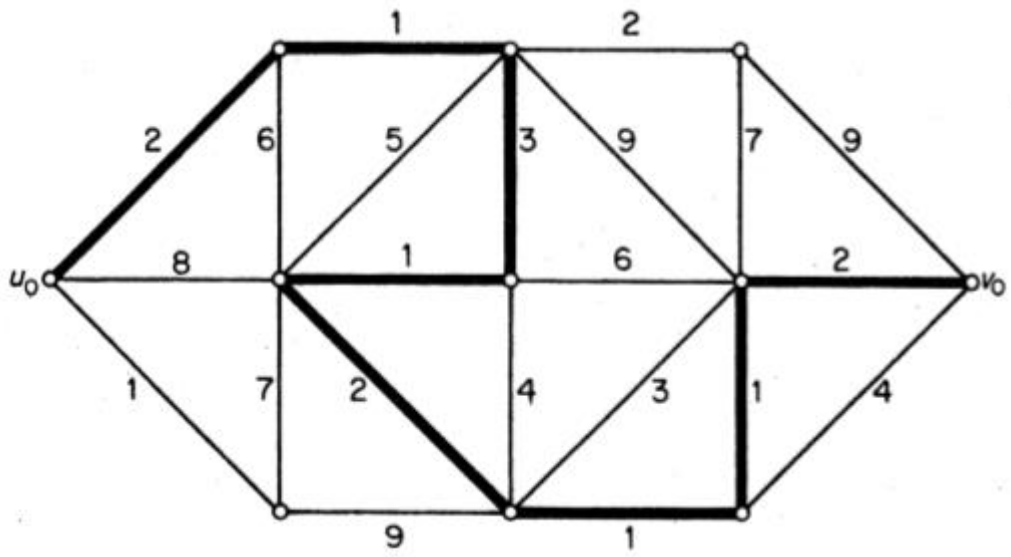
Closed trail: *ucvhxgwfwdvbu*
Cycle: *xaubvhx*

APLIKASI: ALGORITMA ALUR TERPENDEK

Adalah sebuah algoritma untuk yang digunakan untuk mendapatkan jarak terpendek dari dua simpul.

Graph berbobot adalah grah dimana graph mempunyai nilai berupa angka pada setiap rusuknya. Graph dibawah ini adalah graph berbobot. Terdapat beberapa persoalan optimasi pada graph berbobot, salah satunya adalah **persoalah alur terpendek** (*shortest path*

problem): diberikan suatu jaringan kereta yang menghubungkan antar kota, buatlah jalur terpendek yang menghubungkan dua buah kota dalam jaringan tersebut.



Algoritma Djikra

Adalah algoritma yang digunakan untuk menyelesaikan persoalan alur terpendek. Beberapa istilah yang digunakan dalam algoritma djikra adalah sebagai berikut:

1. Simpul awal
2. Simpul yang dapat dilihat
3. Simpul dikunjungi
4. terminal

. Inilah urutan logika dari algoritma Dijkstra:

1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada simpul awal dan nilai tak hingga terhadap simpul lain (belum terisi)
2. Set semua simpul “Belum dikunjungi ” dan set simpul awal sebagai “simpul keberangkatan”
3. Dari simpul keberangkatan, pertimbangkan simpul yang dapat dilihat (yaitu simpul yang dihubungkan oleh satu rusuk kepada simpul keberangkatan) yang belum dikunjungi dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan A ke B memiliki bobot jarak 6 dan dari B ke simpul C berjarak 2, maka jarak ke C melewati B menjadi 6+2=8. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.

4. Saat kita selesai mempertimbangkan setiap jarak terhadap simpul yang dapat dilihat, tandai simpul yang telah dikunjungi sebagai ‘simpul yang telah dikunjungi. Simpul yang telah dikunjungi tidak akan pernah tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Set “simpul yang belum dikunjungi” dengan jarak terkecil (dari simpul keberangkatan) sebagai “simpul keberangkatan” selanjutnya dan lanjutkan dengan kembali ke step 3

Referensi

J.A. Bondy, U.S. R. Murty, Graph Theory with applications