LAPORAN ANALISIS

“IMPLEMENTASI ALGORITMA *FLOYD WARSHALL* PADA *TRAVELLING SALESMAN PROBLEM*”

Disusun oleh :

Nama : Marde Fasma

NRP : 05111640000046

Mata Kuliah : Riset Operasi

Jurusan : Informatika

Daftar Isi

[Surat Pernyataan 3](#_Toc531515980)

[Latar Belakang 4](#_Toc531515981)

[Diagram Alur 5](#_Toc531515982)

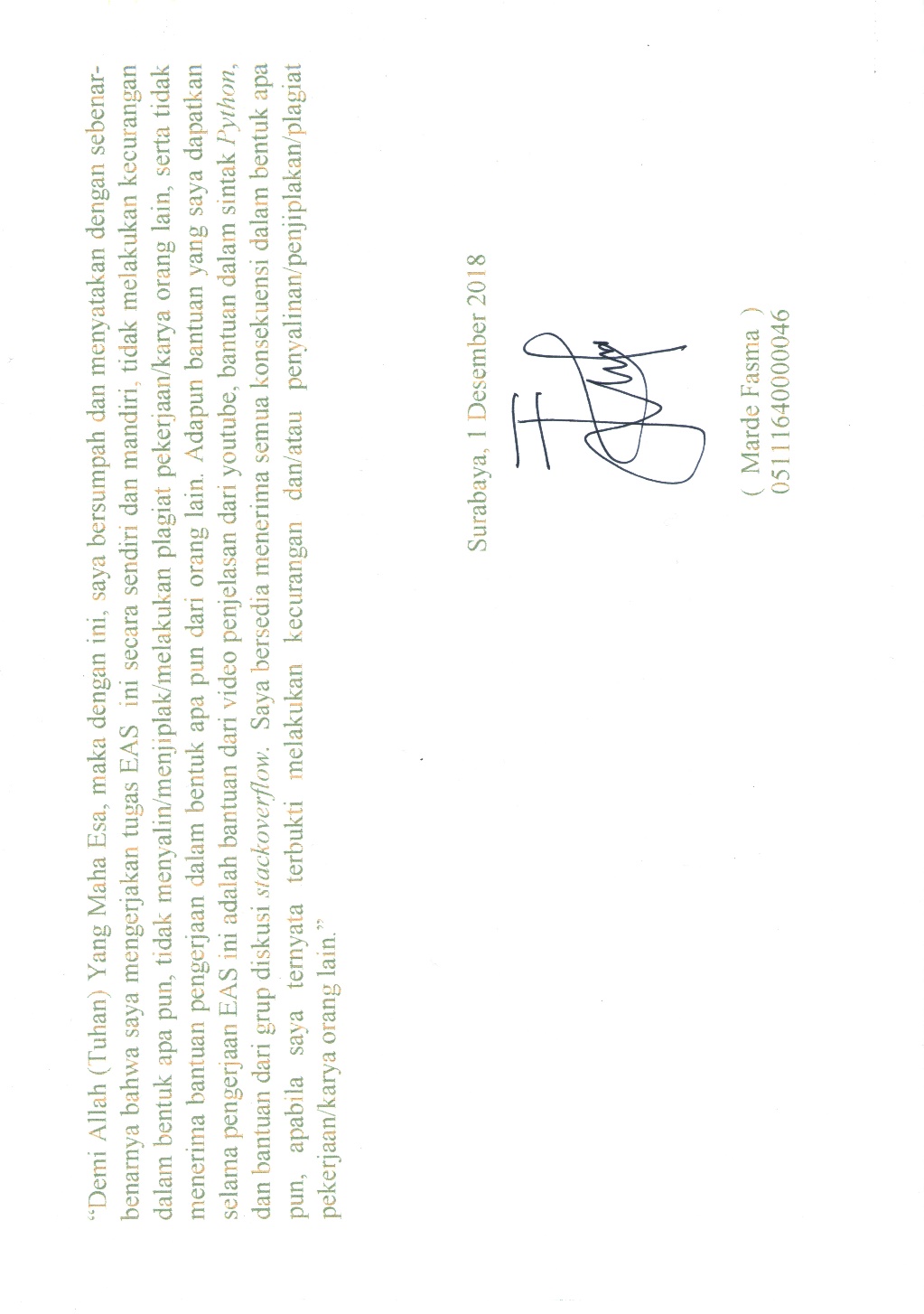
[Metode Penyelesaian 5](#_Toc531515983)

[Graph 5](#_Toc531515984)

[Floyd( graph ) 6](#_Toc531515985)

[Analisis 8](#_Toc531515986)

# Surat Pernyataan

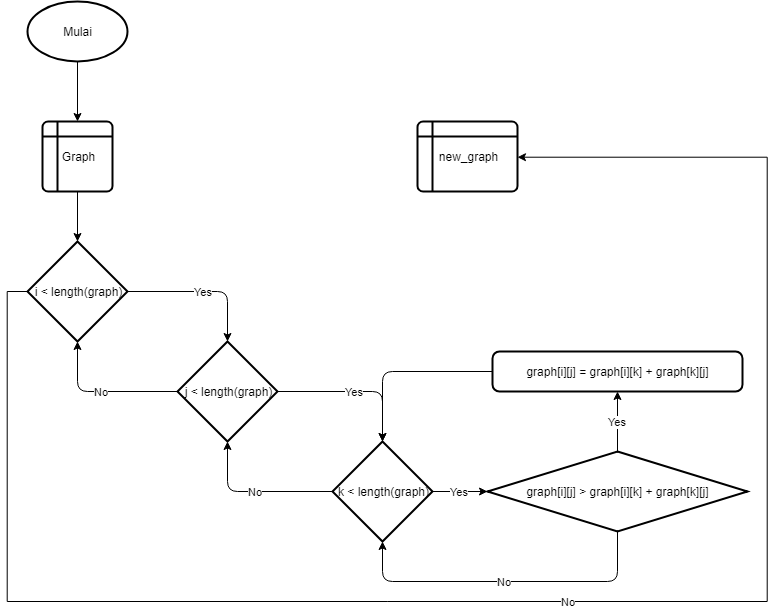


# Latar Belakang

Dalam pencarian jalur terpendek tentunya merupakan suatu permasalahan untuk menentukan sebuah jalur yang mempunyai dua simpul atau node yaitu node awal dan node akhir atau tujuan dengan jumlah bobot minimal. Dalam proses pencarian rute terpendek dari satu lokasi kelokasi lain tentu kita membutuhkan suatu peta pembantu, dan dalam kasus pencarian rute terpendek pada sebuah peta sebuah node akan merepresentasikan suatu lokasi pada peta tersebut, dan suatu bobot akan merepresentasikan jarak atau waktu tempuh untuk melakukan perjalanan dari dua lokasi yaitu lokasi asal kelokasi tujuan.

Salah satu metode yang bisa digunakan untuk menyelesaikan suatu masalah dalam proses pencerian jalur terpendek yaitu dengan menggunakan algoritma *Floyd Warshall*. Algoritma *Floyd Warshall* yaitu salah satu metode untuk mencari lintasan terpendek dalam graf berbobot (dengan bobot sisi negative atau positif). Pengertian *Wikipedia* : Algoritma *Floyd-Warshall* memiliki input graf berarah dan berbobot (V,E), yang berupa daftar titik (node/vertex V) dan daftar sisi (edge E). Jumlah bobot sisi-sisi pada sebuah jalur adalah bobot jalur tersebut. Sisi pada E diperbolehkan memiliki bobot negatif, akan tetapi tidak diperbolehkan bagi graf ini untuk memiliki siklus dengan bobot negatif. Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan melakukannya sekaligus untuk semua pasangan titik.

# Diagram Alur



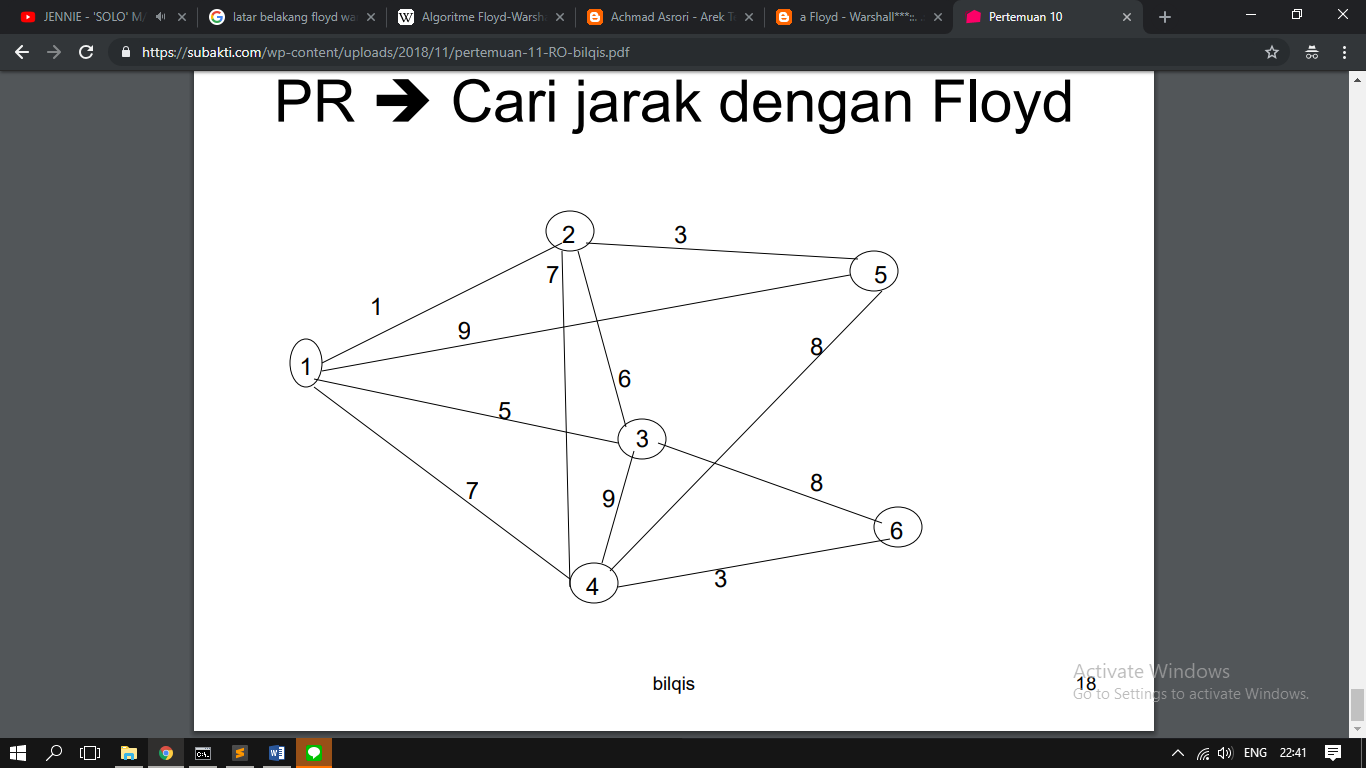
# Metode Penyelesaian

Dalam melakukan penyelesaian TSP dengan metode Floyd Warshall saya menggunakan  
Bahasa pemrograman *python* dan *matplotlib* sebagai bahasa implementernya. Untuk source code dapat dilihat di link berikut :

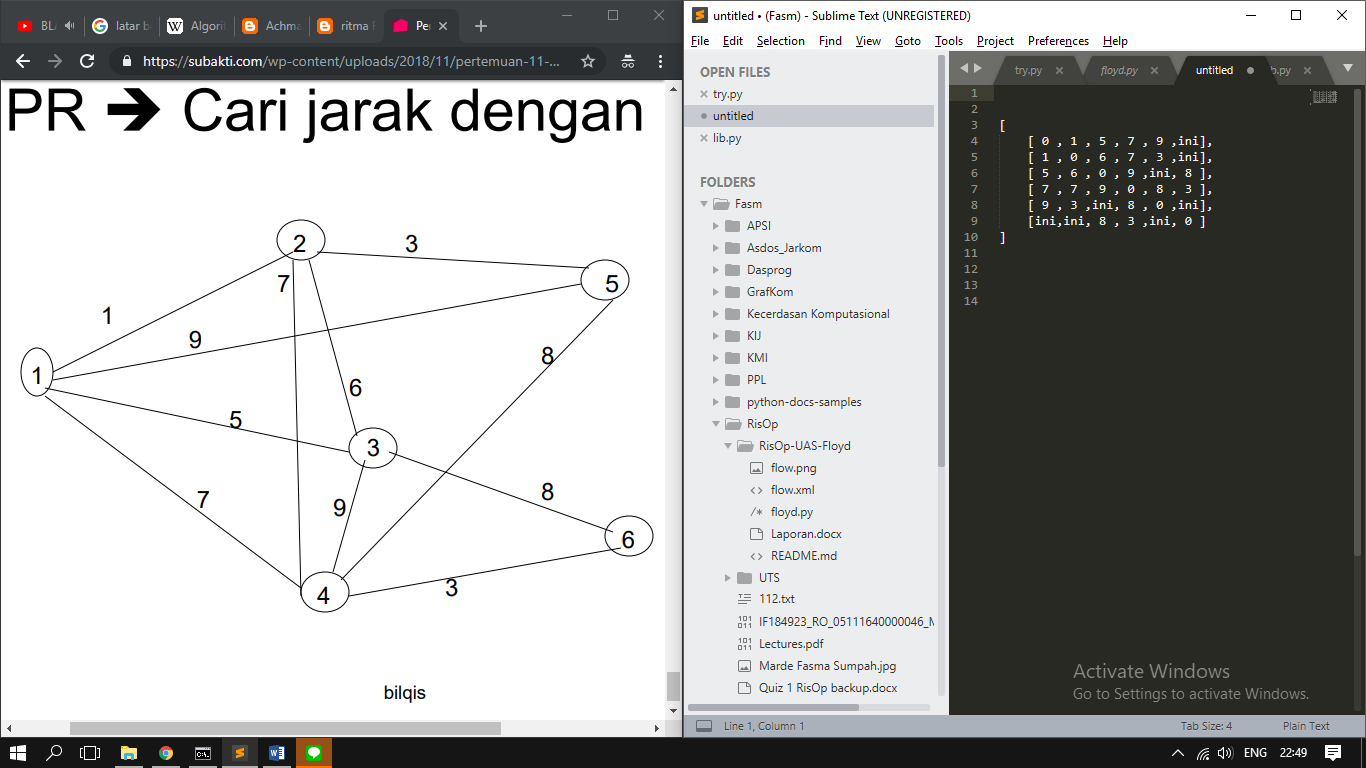
(<https://github.com/marde12345/RisOp-UAS-Floyd>)

## Graph

Dalam melakukan perhitungan *Floyd* diperlukan sebuah graph yang sudah di implementasikan kedalam array 2D. Dalam laporan berikut saya menggunakan bentuk *graph undirected* yang diambil dari link (<https://subakti.com/wp-content/uploads/2018/11/pertemuan-11-RO-bilqis.pdf>) halaman 18. Berikut bentuk graph yang terdapat dari sumber diatas :



Setelah memiliki graph tersebut, maka akan di representasikan kedalam array 2D. Berikut array 2D yang menjadi representasi *graph undirected* dalam Bahasa *python* :

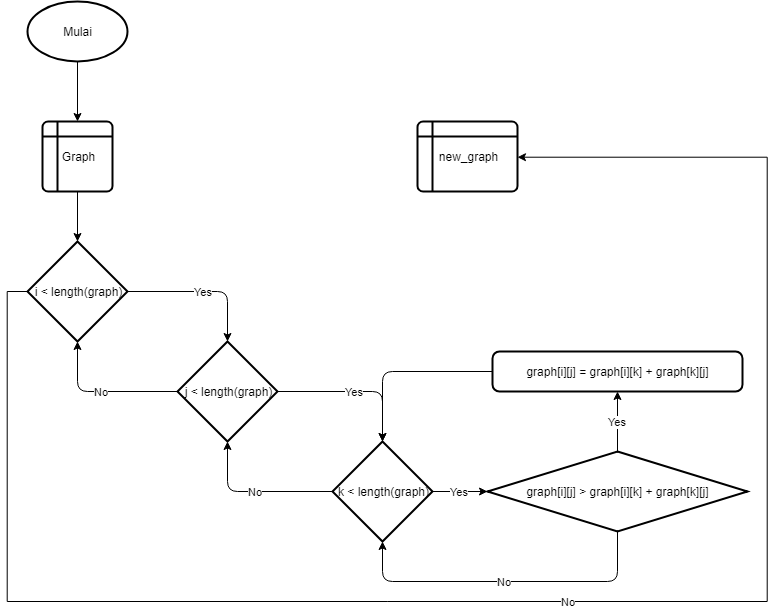


Keterangan :

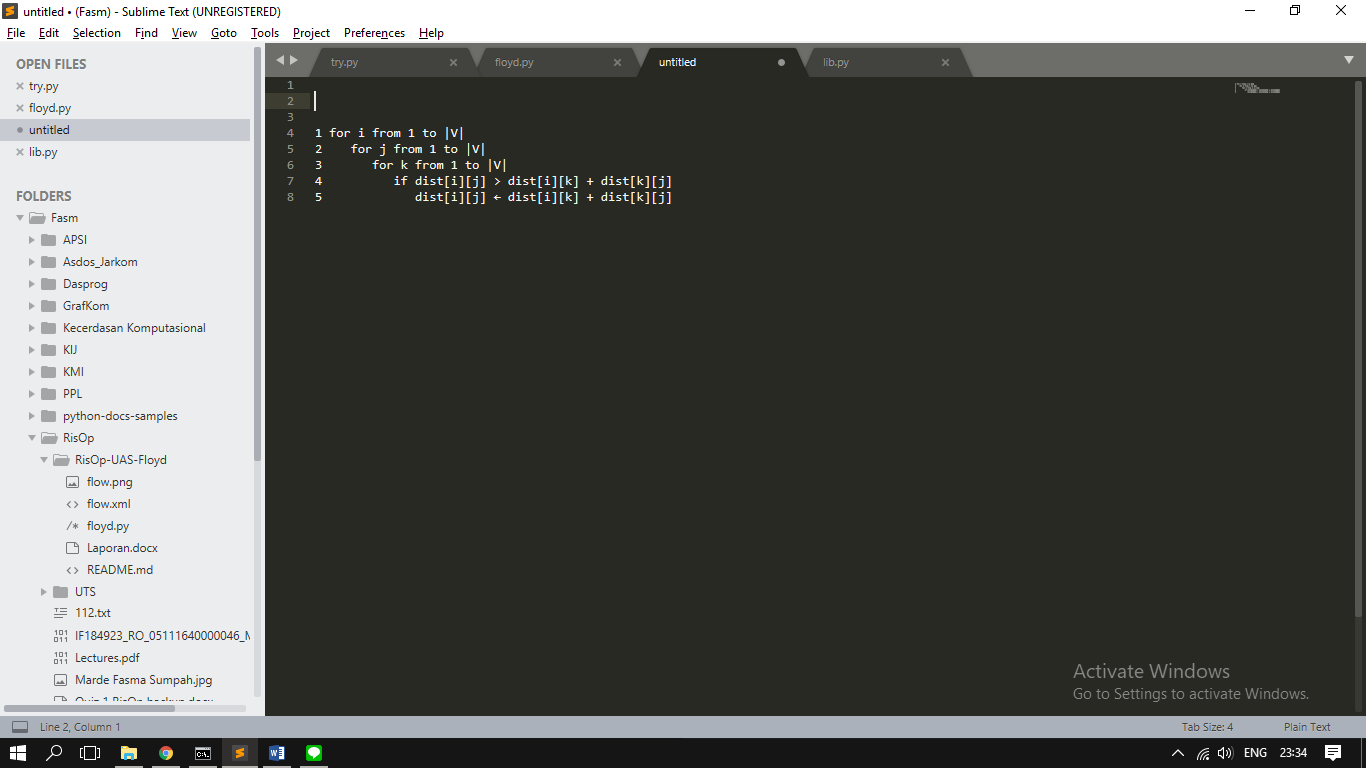
* Ini : representasi dari nilai *infinite* (∞)

## Floyd( graph )

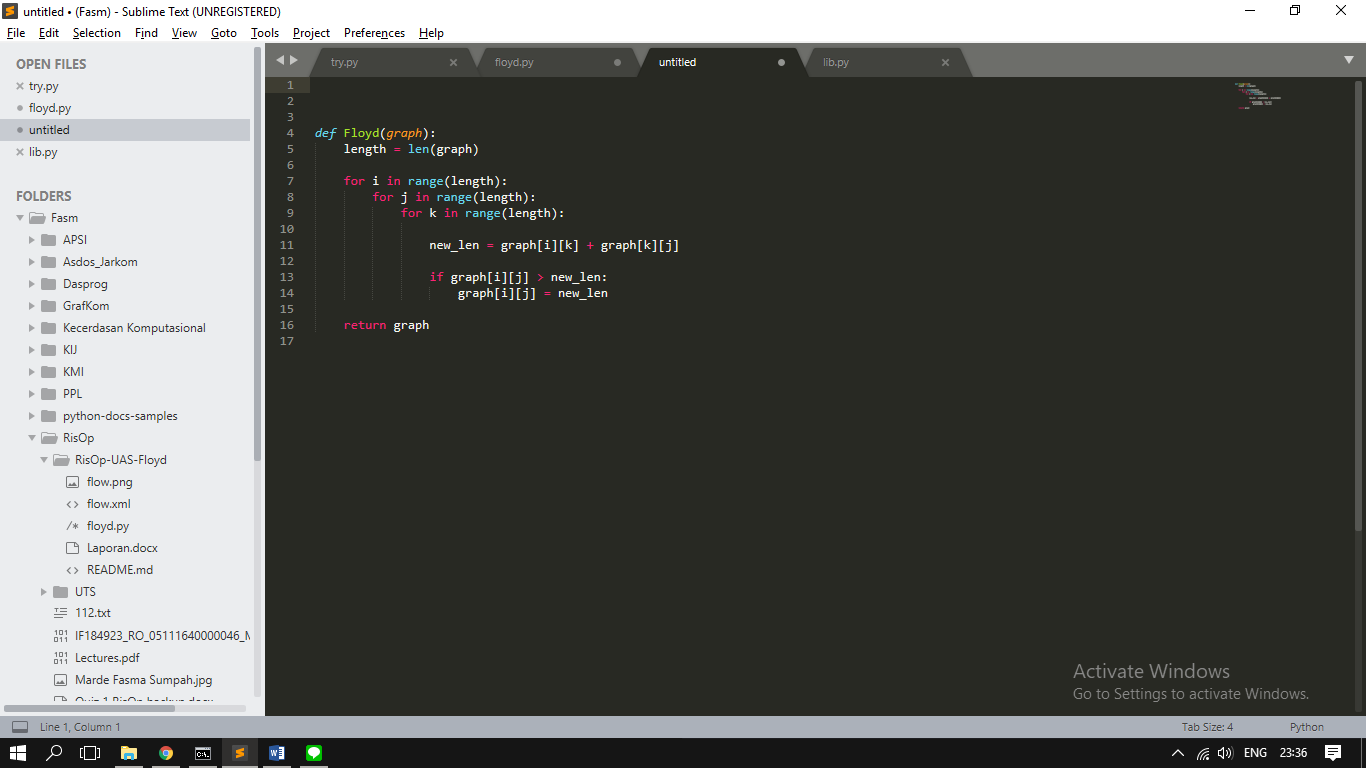
Fungsi Floyd( graph ) adalah sebuah procedure dengan nilai pengembalian *new\_graph* yang merupakan bentuk identic dari graph sebelumnya namun dengan perubahan yang sesuai dengan algoritma *Floyd*. Secara umum, *flowchart* dari fungsi ini adalah sebagai berikut :



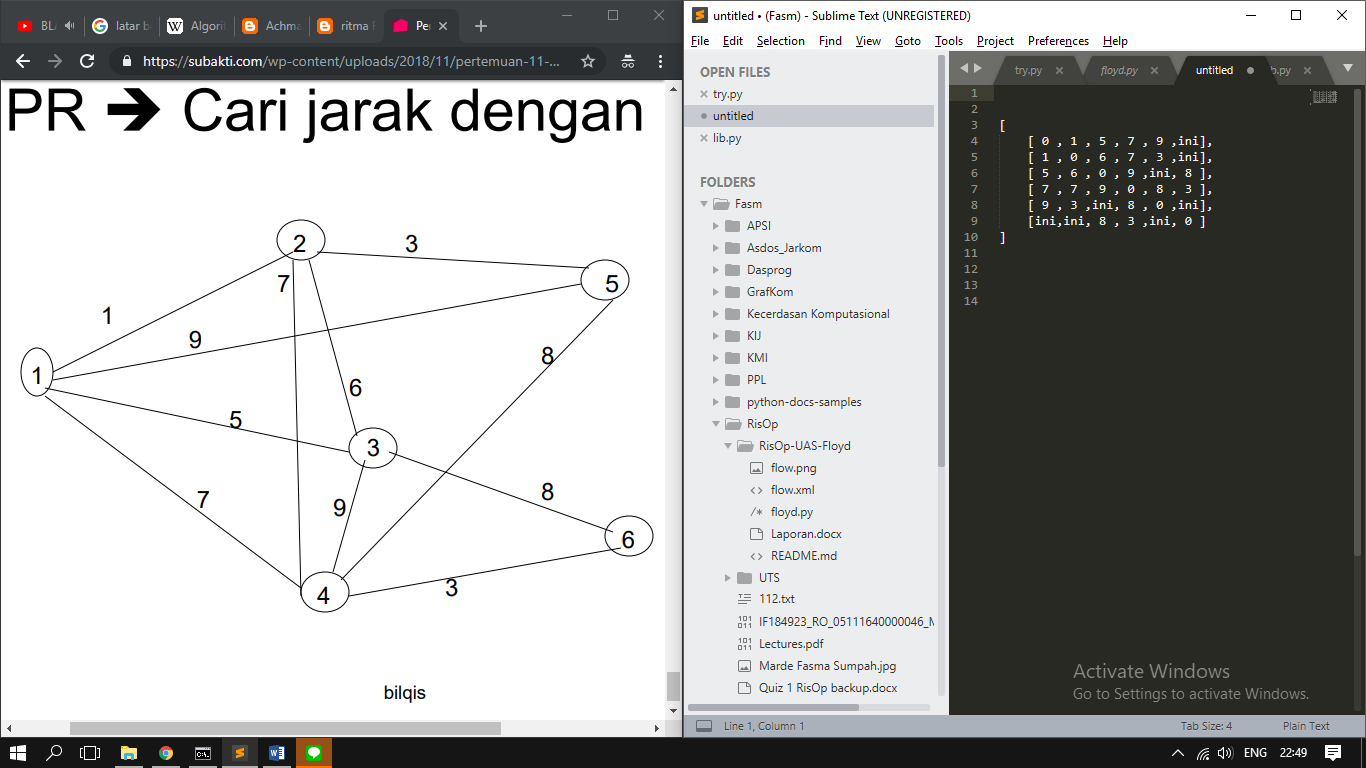
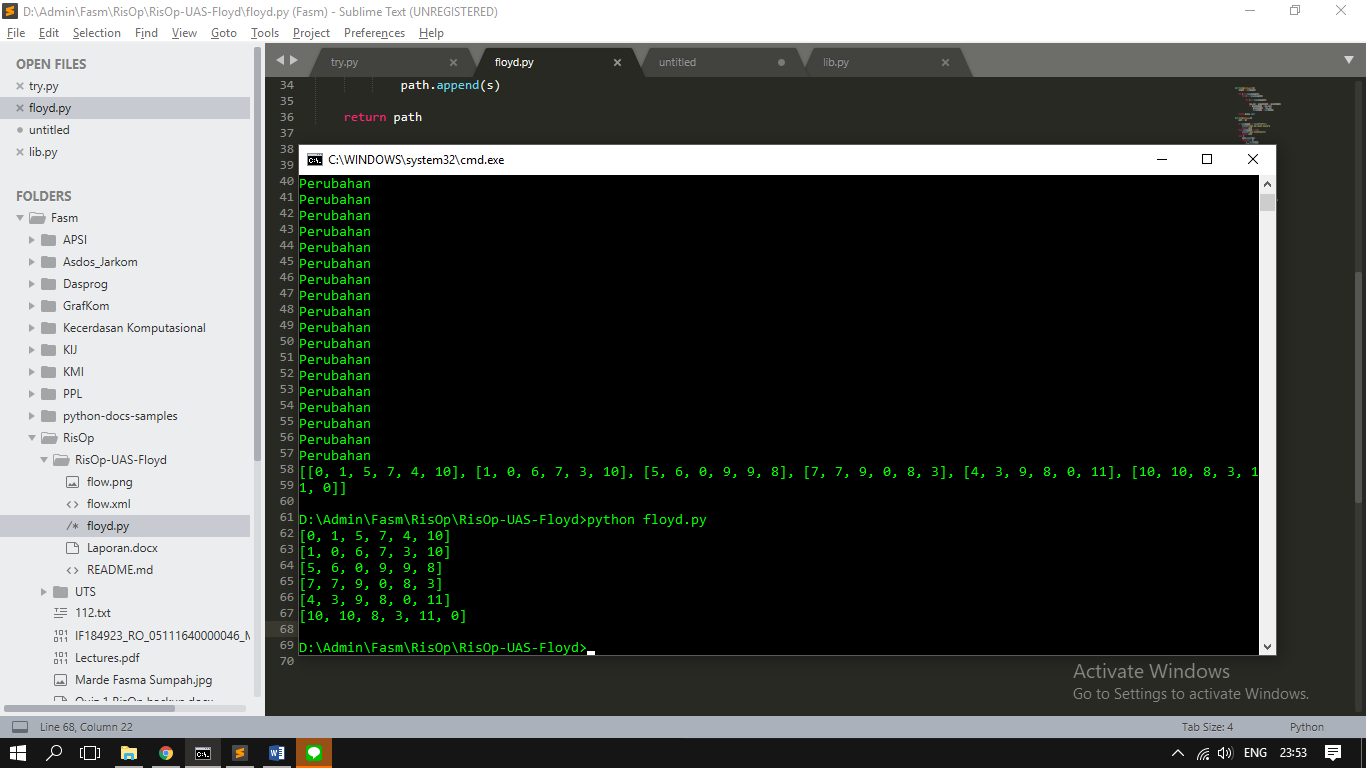
Dan *pseudocode* sebagai berikut :



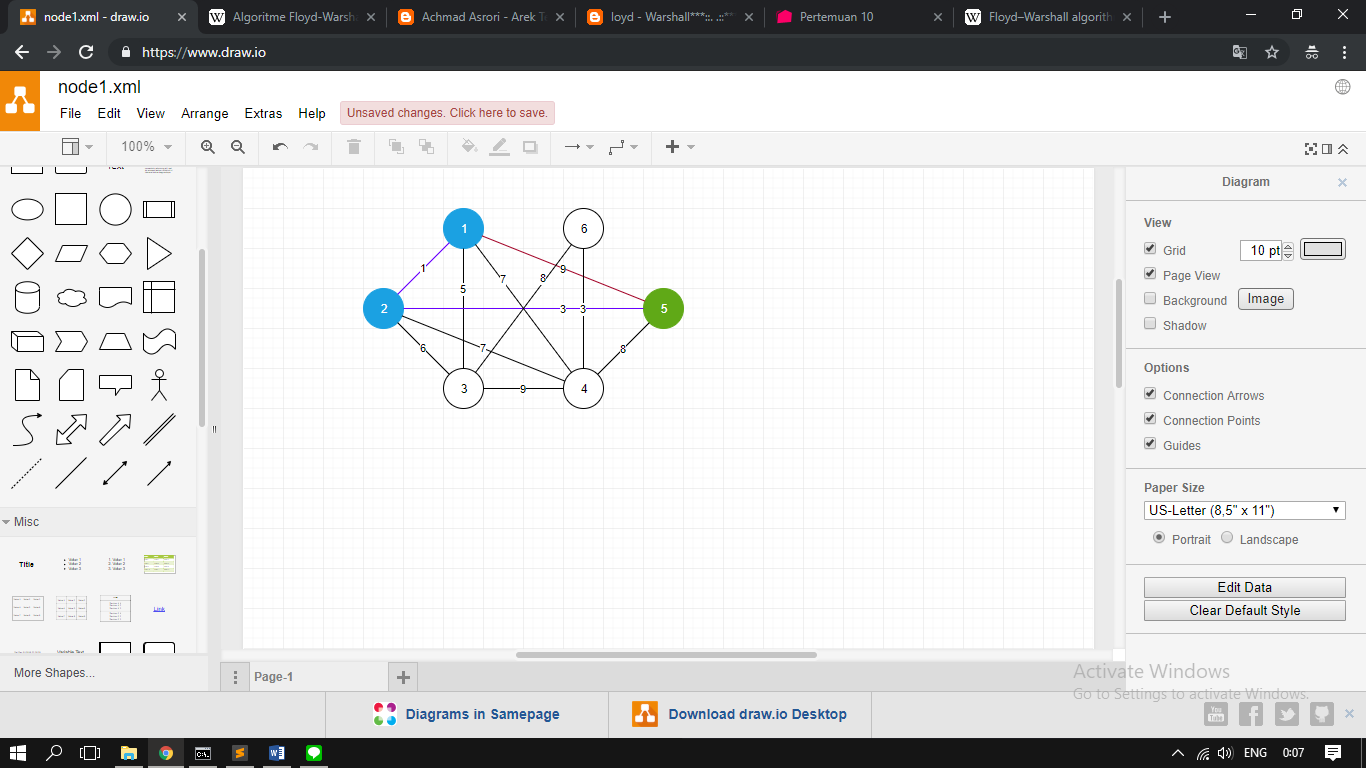
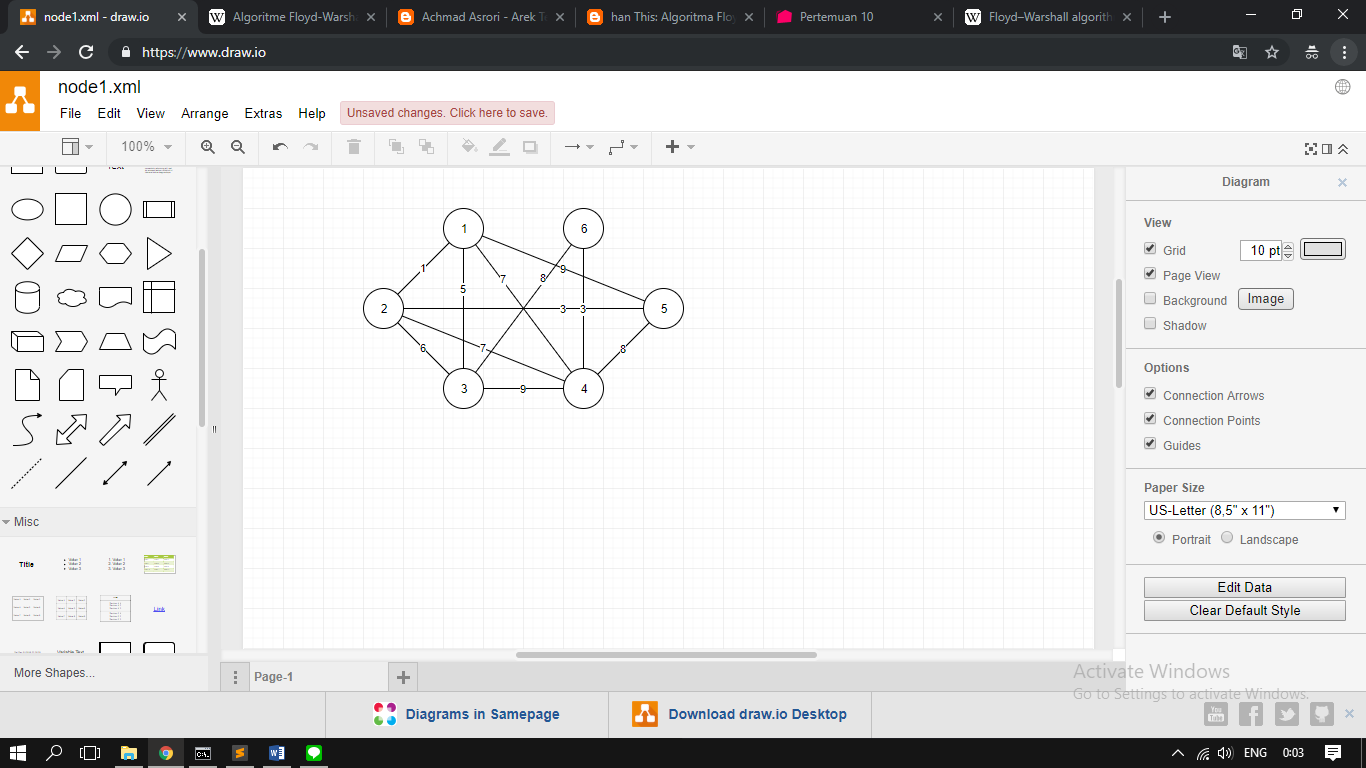
Untuk implementasinya dalam Bahasa *python* sebagai berikut :

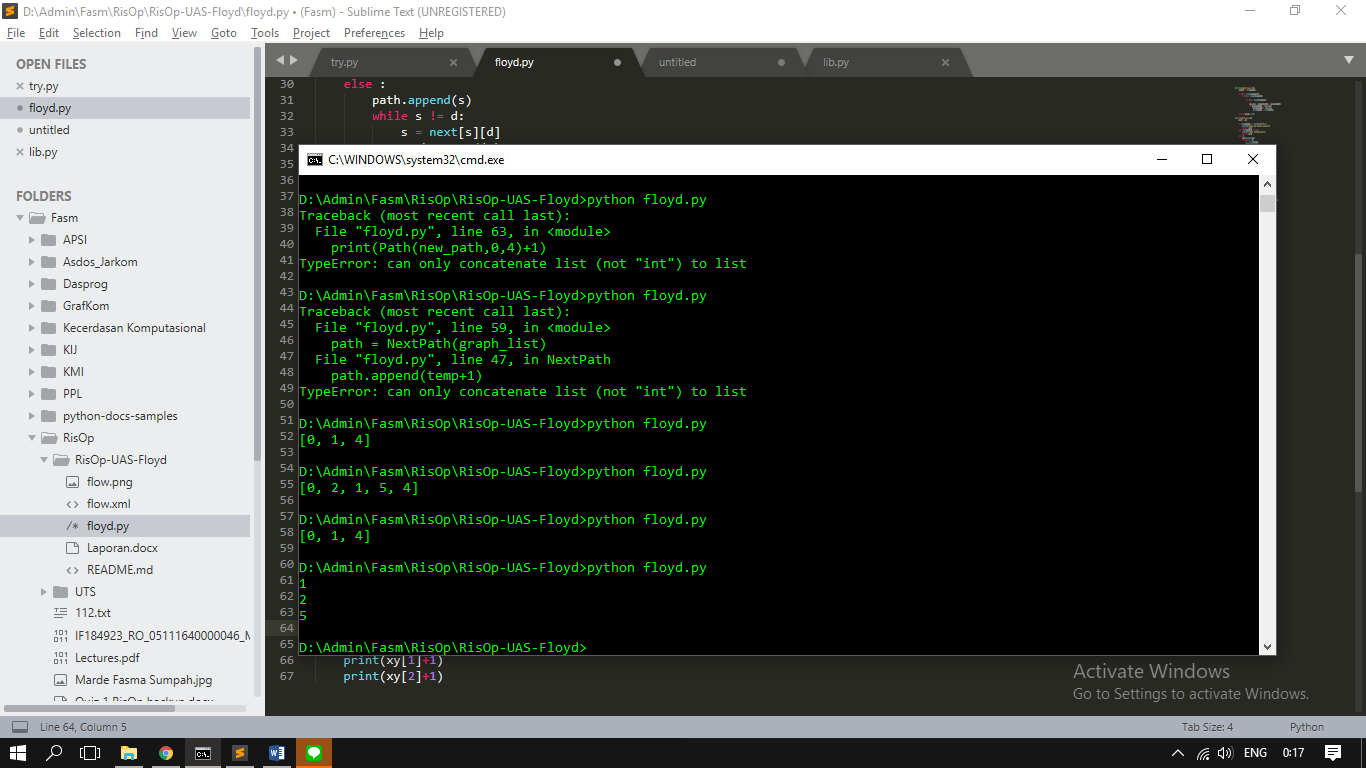


Contoh hasil penggunanaannya sebagai berikut :

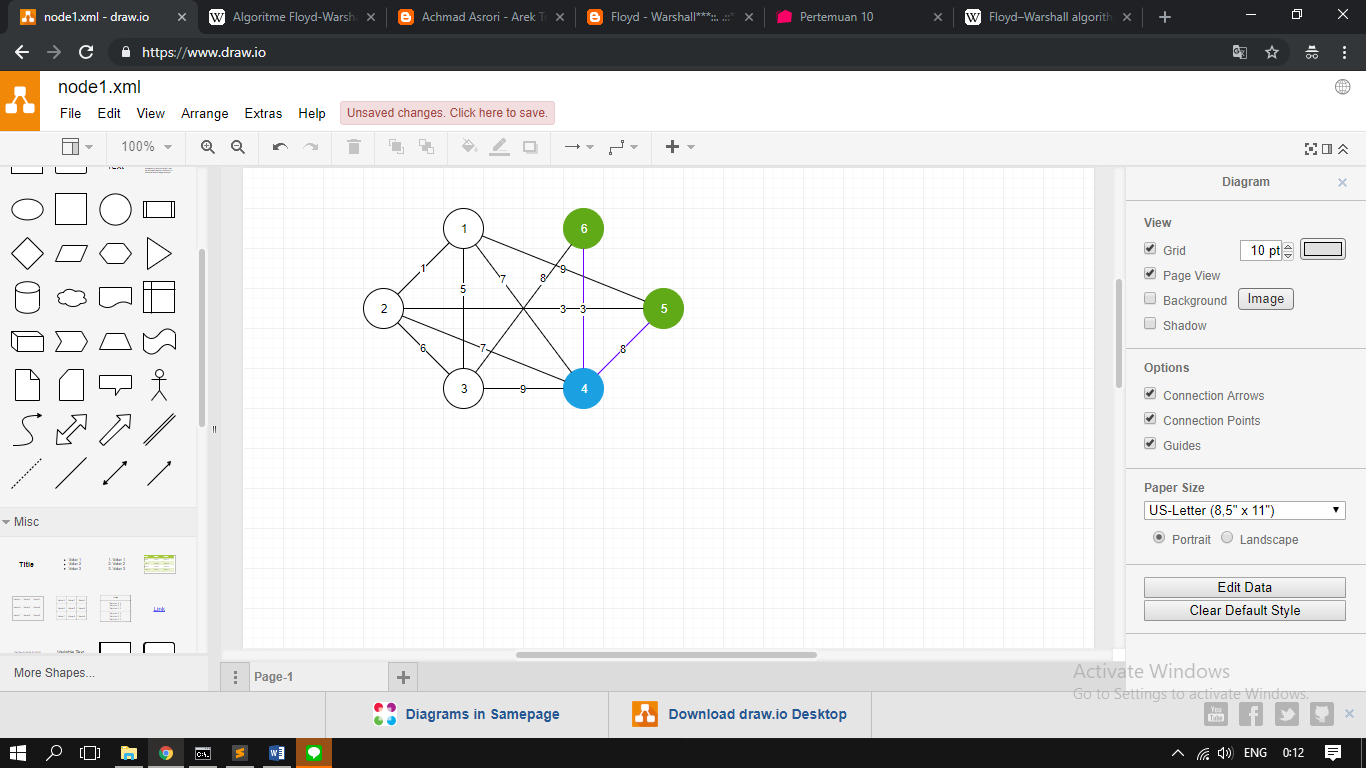
 

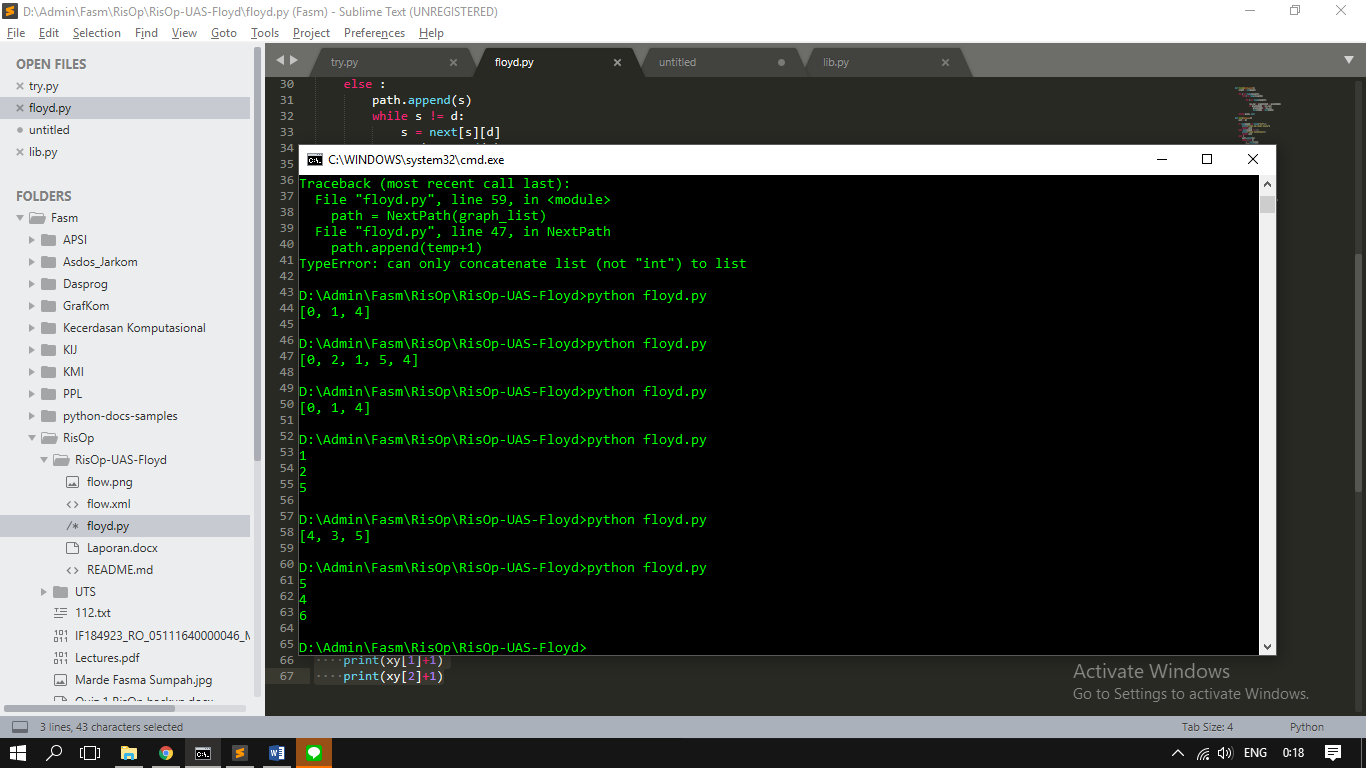
Antara graph dan new\_graph diatas terjadi sebuah perubahan isi matriks. Jika diperhatikan, *‘ini’* pada matrik graph berubah menjadi sebuah nilai serta nilai dari jarak node-1 ke node-5. Dimulai dari node-1 ke node-5, mengapa terjadi demikian? Dilihat dari graph yang ada, terdapat sebuah *shortcut* jalan yang memiliki jarak lebih kecil ketimbang nilai yang sekarang. Dari node-1 ke node-5 memiliki jarak 9 bila langsung menuju ke node-5. Namun, terdapat jalur lain yang memiliki nilai lebih kecil dari pada langsung menuju ke node-5, yaitu melalui node 1 -> 2 -> 5 dengan total jarak yang ditempuh adalah 4.





Lalu untuk nilai dari *ini* berubah menjadi sebuah nilai yang memiliki jarak terpendek melalui node lainnya. Dikarenakan graph yang di gunakan adalah *graph undirected* yang dimana *edge* mampu dilalui dari kedua sisi. Maka semua node mampu dicapai dengan syarat node tersebut memiliki tetangga. Sebagai contoh kita ambil dari node-5 ke node-6. Dari graph diatas, tidak ada jalan menuju ke node-6. Namun terdapat jalan lainnya dengan cara melalui node lainnya.





# Analisis

Algoritma Floyd-Warshall membandingkan semua kemungkinan lintasan pada graf untuk setiap sisi dari semua simpul. 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. 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.