LAPORAN ANALISIS

“IMPLEMENTASI ALGORITMA *DIFFERENTIAL EVOLUTION* DENGAN PURWARUPA DARI *PARTICLE SWARM OPTIMIZATION* MENGGUNAKAN *ROSENBROCK FUNCTION*”

Disusun oleh :

Nama : Marde Fasma

NRP : 05111640000046

Mata Kuliah : Topik Khusus

Kelas : B

Jurusan : Informatika

Daftar Isi

[Surat Pernyataan 3](#_Toc530243415)

[Latar Belakang 4](#_Toc530243416)

[Diagram Alur 5](#_Toc530243417)

[Metode Penyelesaian 5](#_Toc530243418)

[Kelas Ant 6](#_Toc530243419)

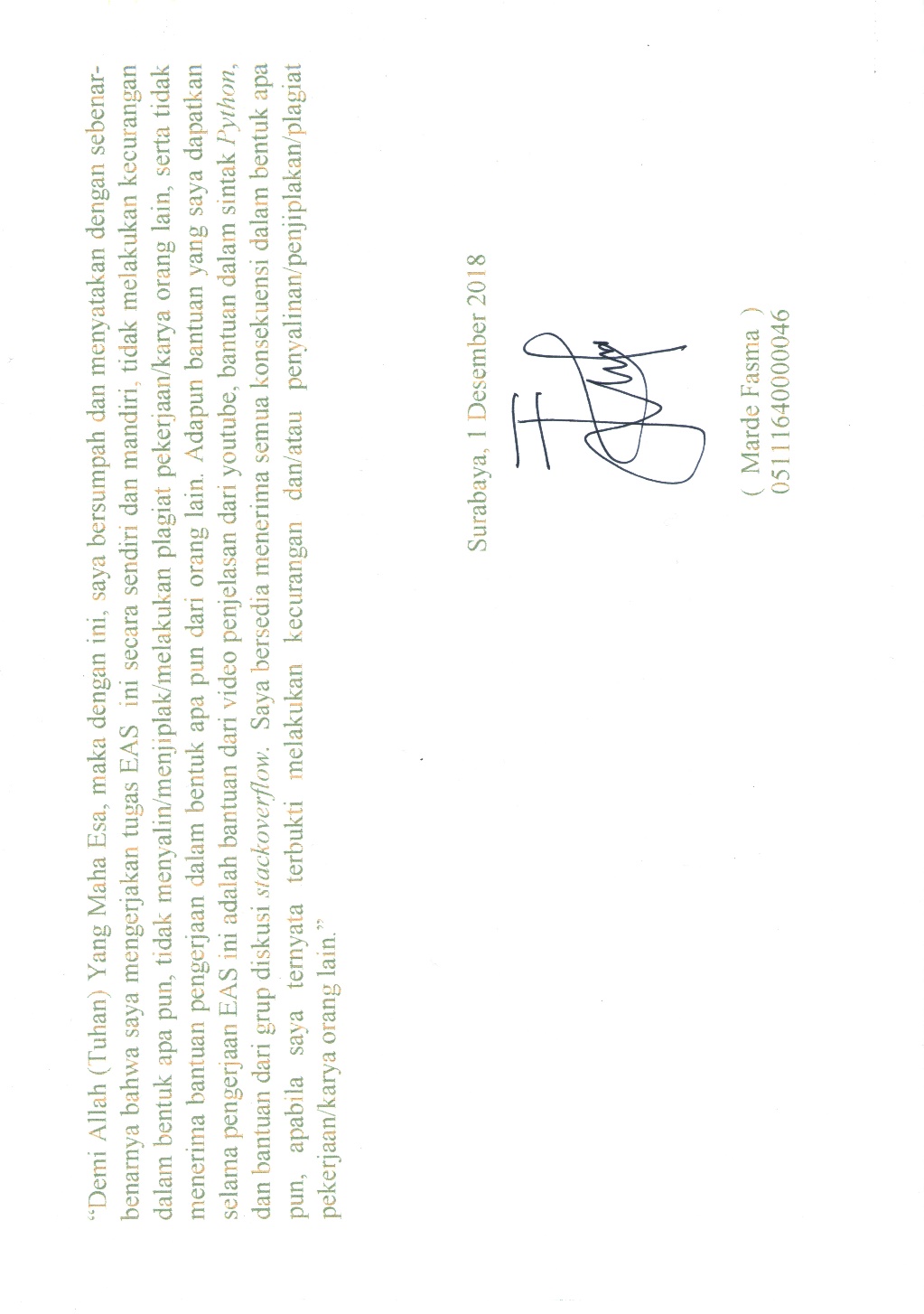
[Pembuatan Node 6](#_Toc530243420)

[Menghitung *Eucledian Distance* 7](#_Toc530243421)

[Generation 8](#_Toc530243422)

[Analisis 8](#_Toc530243423)

# Surat Pernyataan



# Latar Belakang

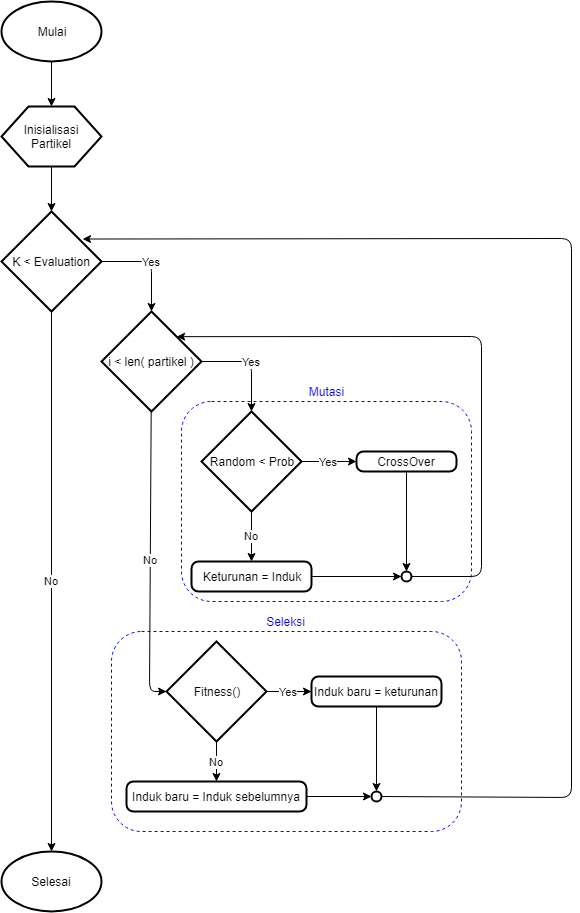
Particle Swarm Optimization (PSO) merupakan metode komputasi yang mengoptimalkan masalah dengan cara iterative serta meningkatkan solusi dari kandidat sebelumnya. PSO didasarkan pada perilaku sekawanan burung atau ikan. Algoritma PSO meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata partikel menunjukkan, misalnya, seekor burung dalam kawanan burung. Setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya (intelligence) sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh dikelompok tersebut.

*Genetic Algorithm* adalah algoritma komputasi yang diinspirasi teori evolusi yang kemudian diadopsi menjadi algoritma komputasi untuk mencari solusi suatu permasalahan dengan cara yang lebih “alamiah”. Salah satu aplikasi algoritma genetika adalah pada permasalahan optimasi kombinasi, yaitu mendapatkan suatu nilai solusi optimal terhadap suatu permasalahan yang mempunyai banyak kemungkinan solusi. Dalam tulisan ini akan dibahas teori dasar algoritma genetika beserta contoh aplikasinya dalam menyelesaikan suatu permasalahan optimasi kombinasi sederhana.

Bila dalam Penugasan PSO sebelumnya dalam membuat keturuan baru menggunakan variable *Global Best* dan *Local Best.* Maka dalam *Differential Evolution* ini dalam penurunan keturunan baru menggunakan fungsi *Genetic Algorithm*. Pada PSO kawanan di representasikan dengan titik koordinat pada 2D. Kawanan dari PSO tersebut akan diturunkan menggunakan algoritma GA.

Menggunakan *Rosenbrock* *Function* sebagai penentu *fitness values* dalam menentukan keturunan baru. Bila dalam penghitungan *fitness* nilai tersebut tidak sesuai maka keturunan selanjutnya akan mengikuti induk sebelumnya.

# Diagram Alur



# Metode Penyelesaian

## Dalam melakukan implementasi metode Differential Evolution saya menggunakan Bahasa pemrograman python dan matplotlib sebagai bahasa implementernya. Untuk source code dapat dilihat di link berikut :

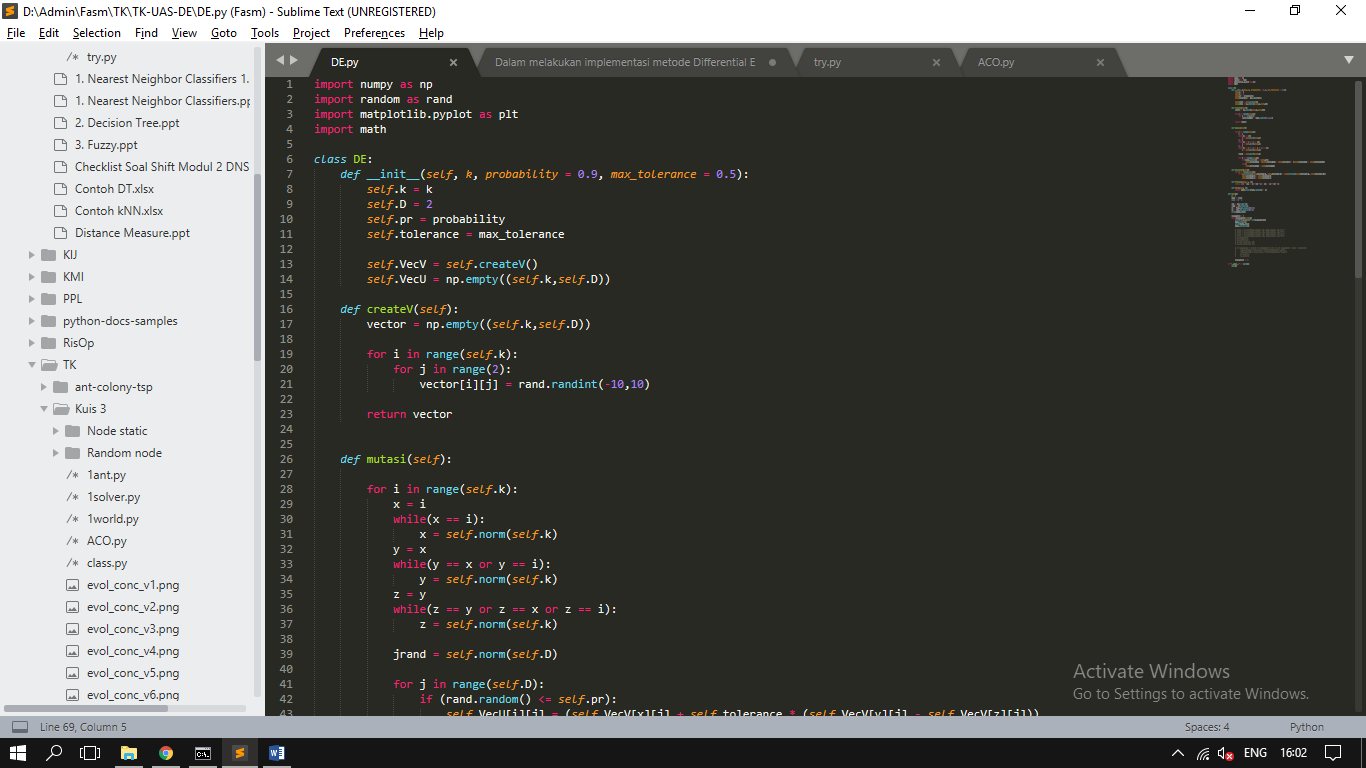
## (<https://github.com/marde12345/TK-UAS-DE>)

Serta menggunakan algoritma PSO yang digunakan sebagai tugas ETS :

(https://github.com/marde12345/TK-PSO)

## Kelas DE

Sebelum melakukan *flowchart* seperti pada gambar Diagram Alur, saya membuat sebuah kelas yang menampung variable yang terdapat pada *DE*. Berikut definisi kelas yang saya gunakan



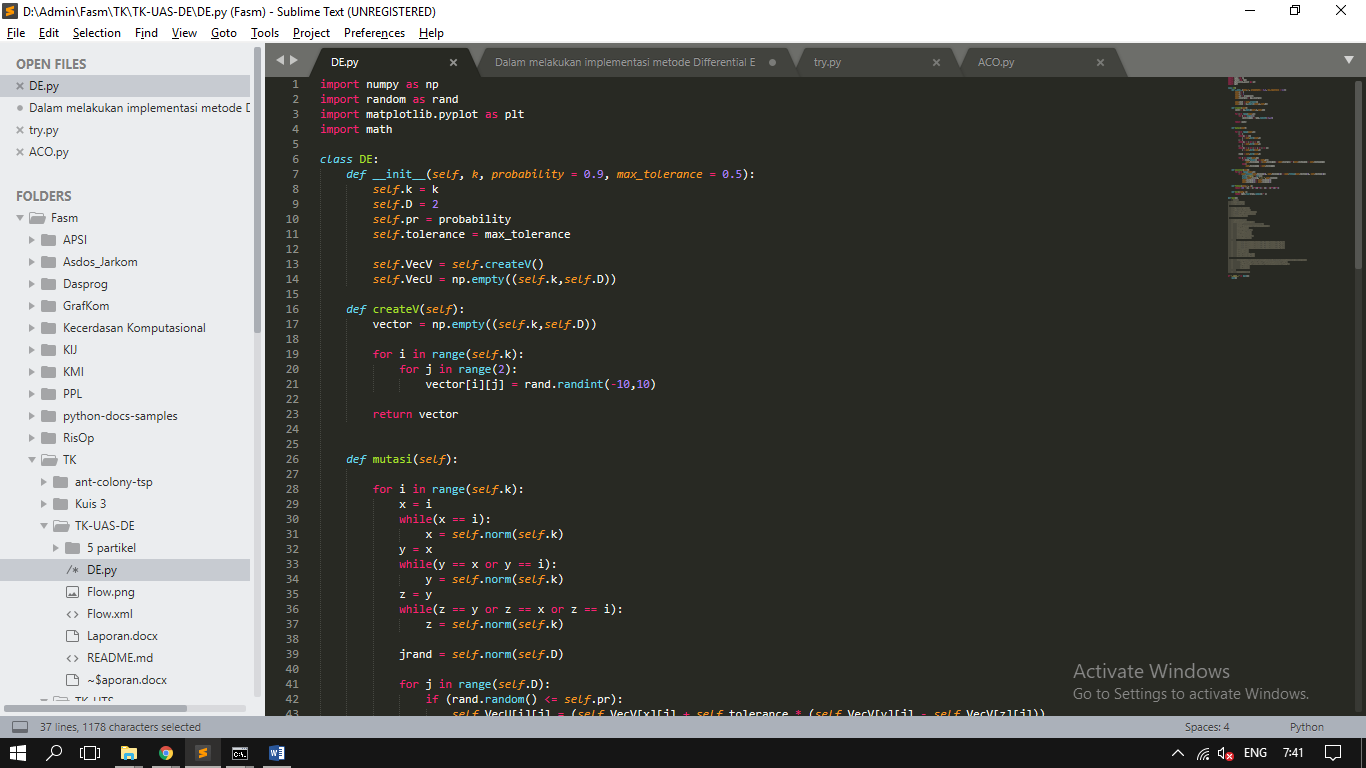
Penjelasan :

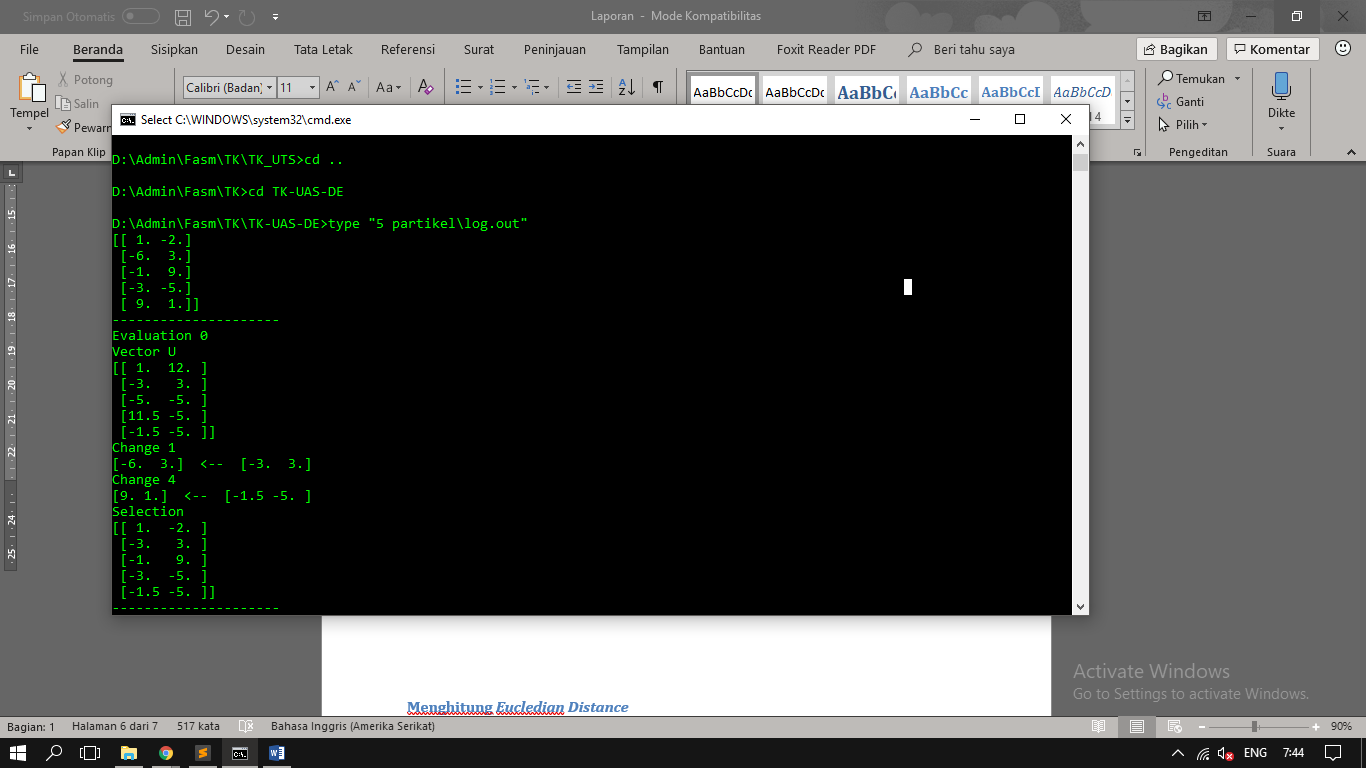
* K : banyaknya partikel didalam world
* D : dimensi dari sebuah partikel
* Pr : nilai *float* untuk melakukan probabilitas melakukan penurunan gen
* Tolerance : nilai *float* untuk mentoleransi nilai saat mutase gen
* VecV : array untuk menyimpan vector dari setiap titik
* VecU : array untuk menampung vector dari keturunan yang baru

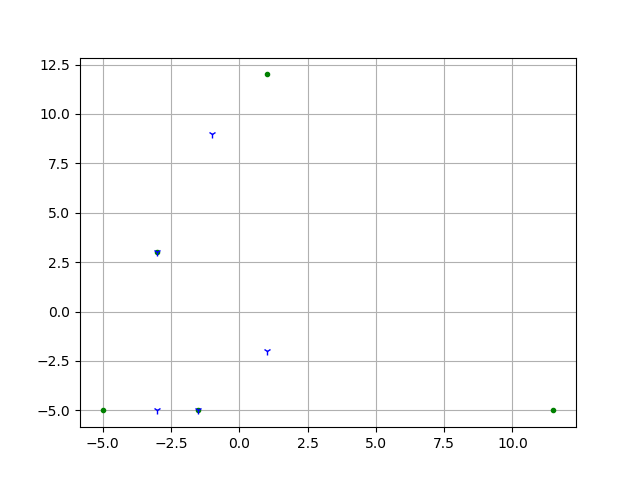
## createV()

Dalam menentukan partikel awal dalam algoritma ini, diperlukan sebuah titik untuk mempresentasikan setiap partikel dalam matrix dan titik yang didapat dari fungsi *random*.

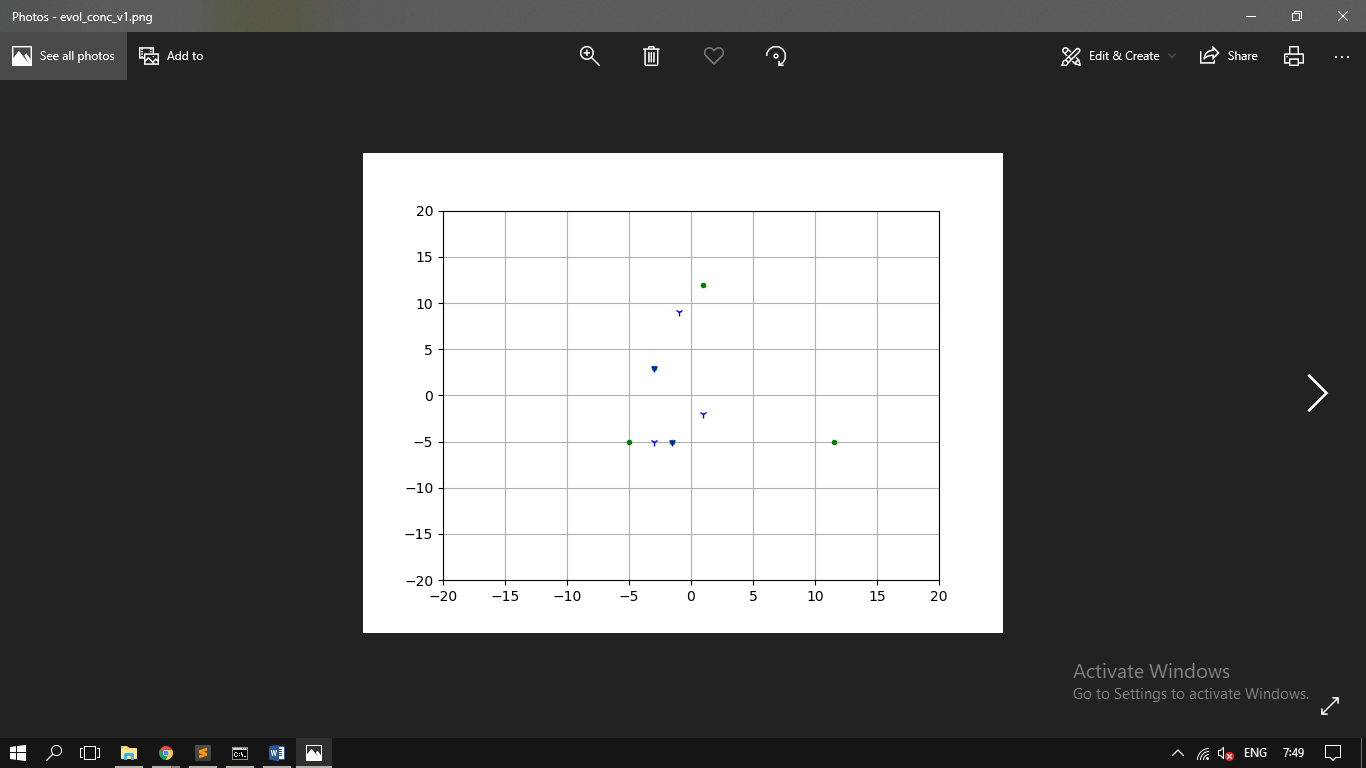
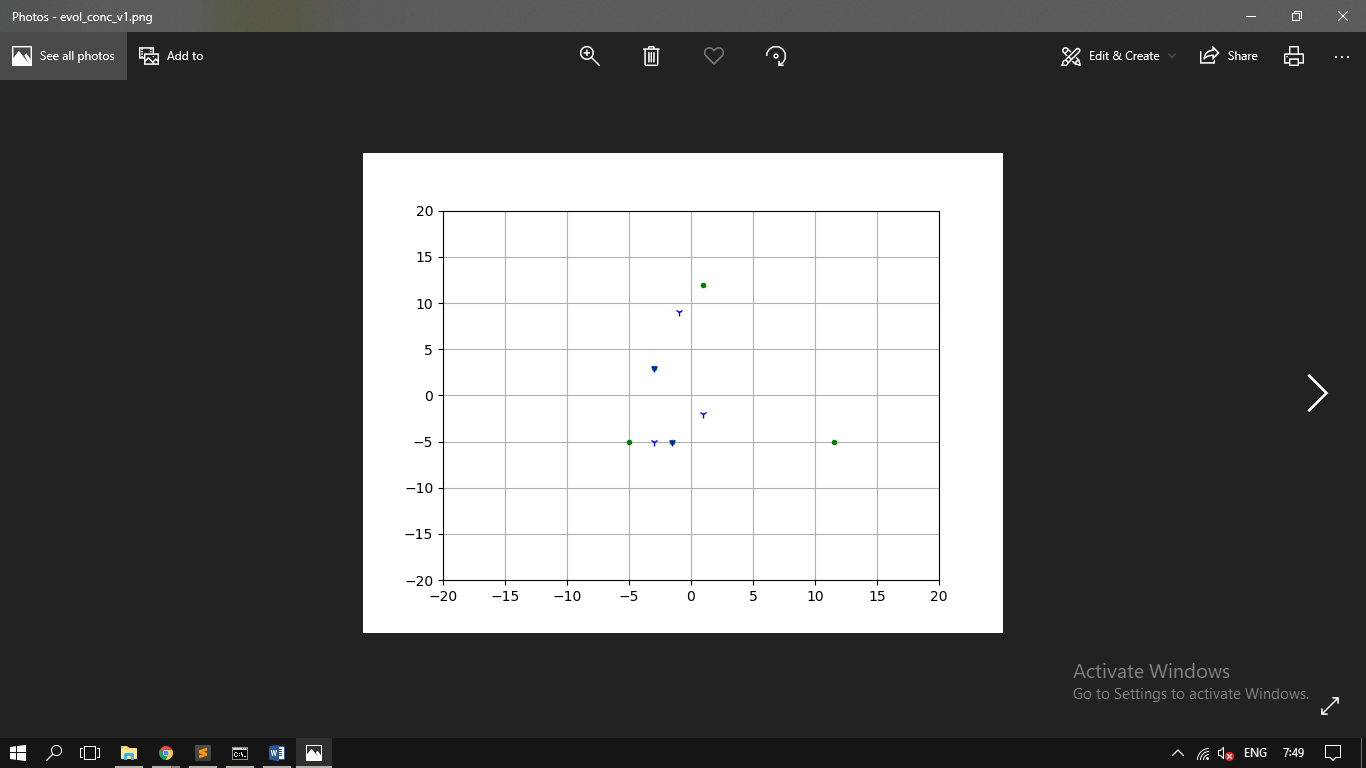
Grafik dengan 5 partikel dibuat secara *random* :





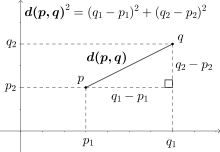


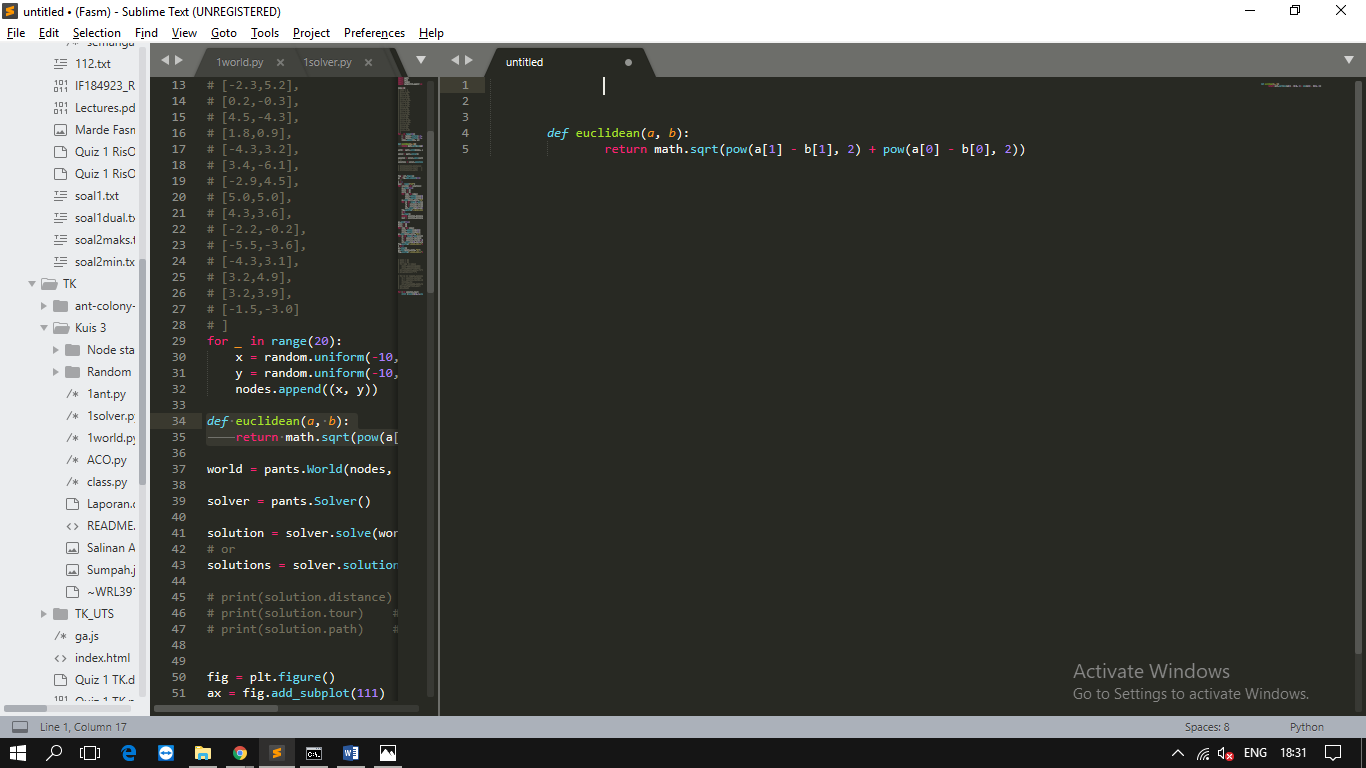
Dari grafik diatas, dapat dilihat bahwa terdapat 2 icon yang berbeda, yaitu:

*  : Digunakan sebagai representasi dari Vector induk
*  : Digunakan sebagai representasi dari Vector generasi selanjutnya

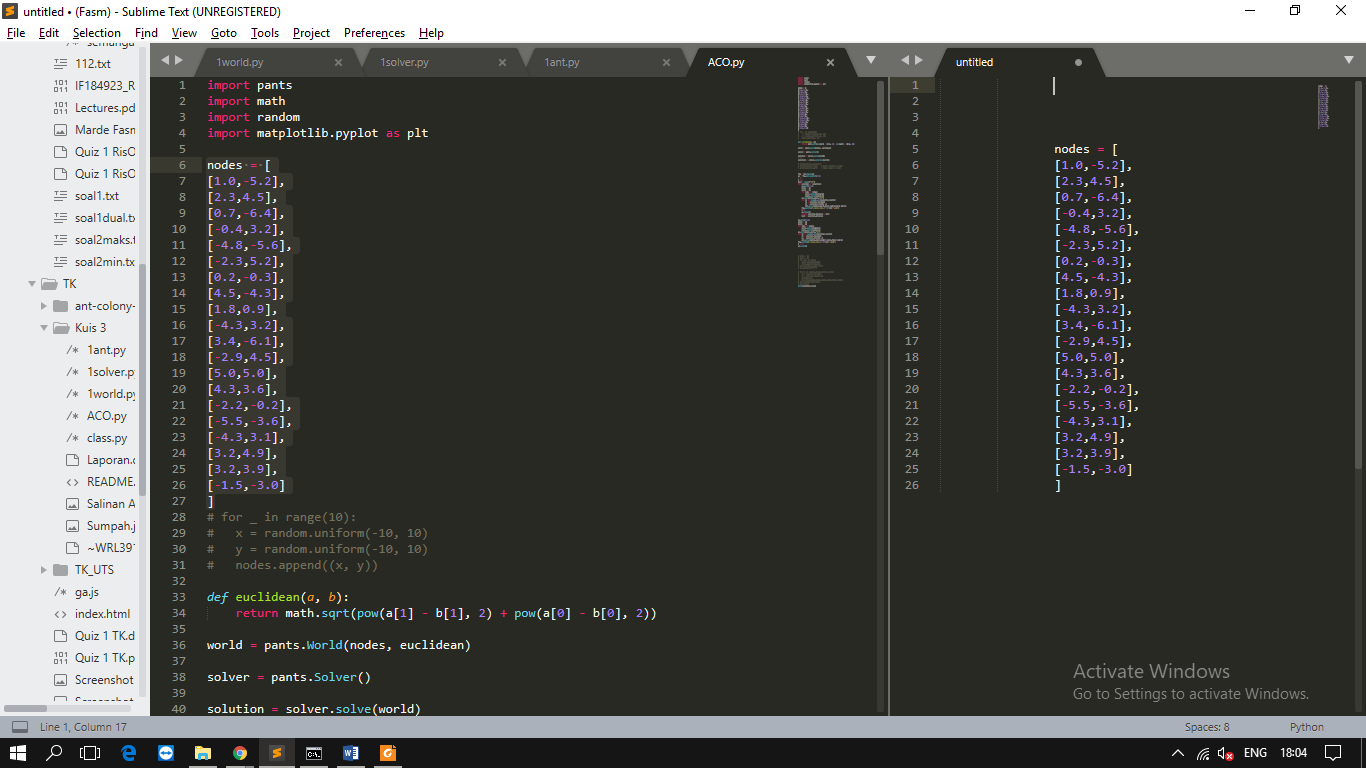
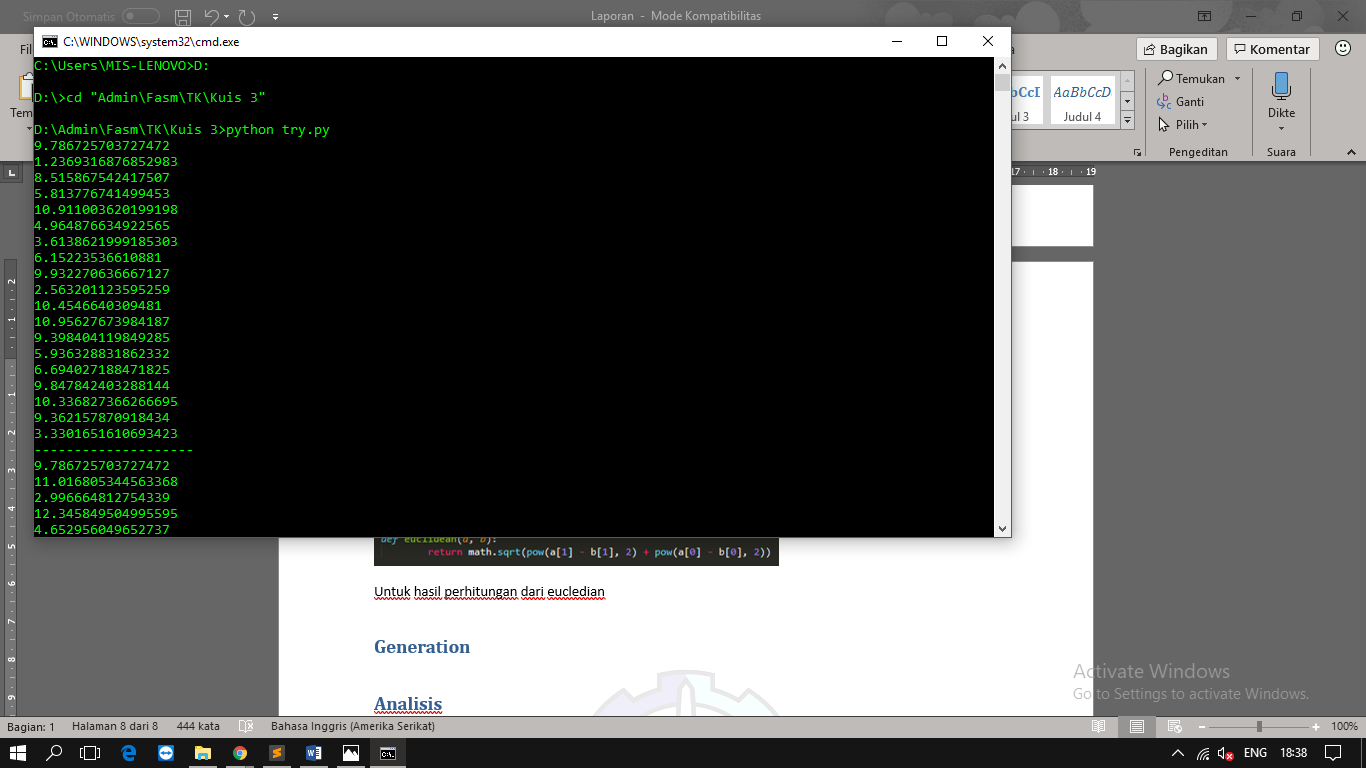
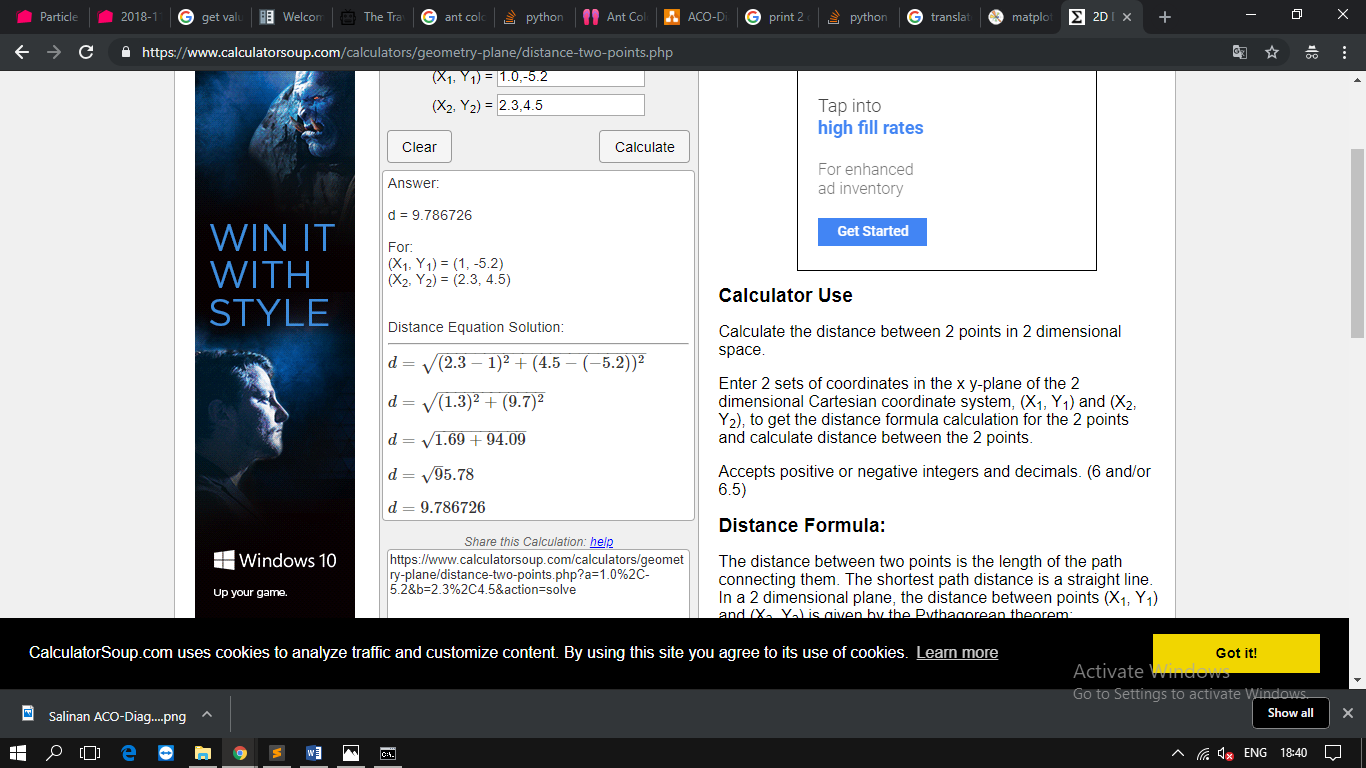
## Menghitung *Eucledian Distance*

Perlu ditentukan seluruh jarak dari sebuah node ke node yang lain. Maka dari itu, saya menggunakan Eucledian Distance sebagai perhitungannya dengan rumus sebagai berikut :





Untuk hasil perhitungan dari eucledian dengan node awal adalah node awal pada node static

# Generation

# Analisis