Tugas Kecil (Tucil) 2 IF 2211 Strategi Algoritma

**Membuat Program Mencari Pasangan Titik Terdekat 3D dengan Algoritma *Divide and Conquer***

Mencari sepasang titik terdekat dengan Algoritma *Divide and Conquer* sudah dijelaskan di dalam kuliah. Persoalan tersebut dirumuskan untuk titik pada bidang datar (2D). Pada Tucil 2 kali ini Anda diminta mengembangkan algoritma mencari sepasang titik terdekat pada bidang 3D. Misalkan terdapat *n* buah titik pada ruang 3D. Setiap titik *P* di dalam ruang dinyatakan dengan koordinat *P* = (*x*, *y*, *z*). Carilah sepasang titik yang mempunyai jarak terdekat satu sama lain. Jarak dua buah titk *P*1 = (*x*1, *y*1, *z*1) dan *P*2 = (*x*2, *y*2, *z*2) dihitung dengan rumus Euclidean berikut:



Buatlah program dalam dalam Bahasa C++/Java/Phyton/Ruby/Perl (pilih salah satu) untuk mencari sepasang titik yang jaraknya terdekat datu sama lain dengan menerapkan algoritma *divide and conquer* untuk penyelesaiannya, dan perbandingannya dengan Algoritma *Brute Force*.

Masukan program:

* *n*
* titik-titik (dibangkitkan secara acak)

Keluaran program

* sepasang titik yang jaraknya terdekat dan nilai jaraknya
* banyaknya operasi perhitungan rumus Eucledian
* waktu riil dalam detik (spesifikasikan komputer yang digunakan)
* (opsional, bonus: 10) penggambaran semua titik dalam bidang 3D, sepasang titik yang jaraknya terdekat ditunjukkan dengan warna yang berbeda dari titik lainnya.

Program dibuat per-1 orang atau per-2 orang, dibuat sendiri, dan dikumpulkan minggu depan (Hari Rabu) pada saat jam kuliah. Yang dikumpulkan adalah:

1. *Pseudo-code* algoritma
2. Kode program
3. Contoh masukan dan keluaran untuk *n* = 16, *n* = 64, *n* = 128, *n* = 1000

Tambahkan cek list berikut (centang dengan √) di dalam laporan anda untuk memudahkan Asisten dalam menilai:

|  |  |  |
| --- | --- | --- |
| Poin  | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa ada kesalahan.
 |  |  |
| 1. Program berhasil *running*
 |  |  |
| 1. Program dapat menerima masukan dan dan menuliskan keluaran.
 |  |  |
| 1. Keluaran sudah benar (solusi *closest pair* benar)
 |  |  |