Tugas Kecil 2 IF 2211 Strategi Algoritma

Semester II Tahun 2022/2023

Mencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

**Bab 1 Deskripsi Masalah**

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/C++/Java/Python/Golang/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) dalam koordinat (x, y, z)

Luaran program :

* sepasang titik yang jaraknya terdekat dan nilai jaraknya
* banyaknya operasi perhitungan rumus Euclidian
* waktu riil dalam detik (spesifikasikan komputer yang digunakan)
* Bonus 1 (Nilai = 7,5) penggambaran semua titik dalam bidang 3D, sepasang titik  
  yang jaraknya terdekat ditunjukkan dengan warna yang berbeda dari titik lainnya.
* Bonus 2 (nilai = 7,5): Generalisasi program anda sehingga dapat mencari sepasang titik terdekat untuk sekumpulan vektor di Rn, setiap vektor dinyatakan dalam bentuk **x** = (x1, x2,  
  …, xn)

**Bab 2 Algoritma**

Landasar teori yang akan digunakan pada penerapan strategi algoritma dalam mencari jarak minimum dari titik pada 3 dimesi yaitu: algoritma *Divide and Conquer* dan *Brute Force*

1. Definisi Algoritma *Divide and Conquer*

Algoritma *Divide and Conquer* merupakan algoritma umum yang digunakan untuk memecahkan suatu masalah besar menjadi beberapa masalah kecil dan menyelesaikan masalah kecil secara terpisah, kemudian menggabungkan solusi masalah kecil dan memecahkan masalah asli. Dari namanya Divide and Conquer dibagi menjadi tiga langkah dasar:

1. Divide: membagi masalah menjadi masalah-masalah yang lebih kecil. Biasanya  
   menjadi setengahnya atau mendekati setengahnya
2. Conquer: ketika sebuah masalah sudah cukup kecil untuk diselesaikan, langsung  
   selesaikan masalah tersebut.
3. Combine: ketika sebuah masalah sudah cukup kecil untuk diselesaikan, langsung  
   selesaikan masalah tersebut.

Algoritma *Divide and Conquer* digunakan dalam banyak aplikasi, termasuk komputasi paralel, optimasi dan pembuatan keputusan. Beberapa contoh dari masalah yang dapat dipecahkan dengan penedakat ini meliputi *sorting,* pencarian elemen terntentu, penghitungan nilai fungsi matematika. Dalam implementasinya, algoritma *Divide and* *Conquer* dapat memberikan solusi yang efektif daripada pendekatan algoritma lainnya.

1. Implementasi *Divide and Conquer* pada algoritma *quicksort*

Terdapat algoritma pengurutan yang memiliki kompleksitas cukup cepat yaitu *quicksort*. *Quicksort* menggunakan prinsip *divide and conquer* dalam pengurutannya. Tahapan dari pengurutan *quicksort* adalah sebagai berikut :

1. Divide: pilih suatu elemen yang kita sebut *pivot,* kemudian kita bagi *array* menjadi dua sehingga *object* dalam array *pivot* dan yang lainnya selalu *pivot.*
2. Conquer: ketika *array* hanya memiliki satu elemen, *array* tersebut sudah terurut
3. Combine: gabunglah *subarray* dengan menempelkan hasil quicsort bagian kiri dan bagian kanan

**Bab 4 Kode Program dalam Bahasa Python**

**Bab 5 Dokumentasi**

**Bab 6 Tabel Penilaian**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Poin | Ya | Tidak |
| 1 | Program berhasil dikompilasi tanpa ada kesalahan | V |  |
| 2 | Program berhasil running | V |  |
| 3 | Program dapat menerima masukan dan menuliskan luaran | V |  |
| 4 | Luaran program sudah benar (solusi *closest pair* sudah benar) | V |  |
| 5 | Bonus 1 dikerjakan | V |  |
| 6 | Bonus 2 dikerjakan | V |  |

**Bab 7 Kesimpulan**

**Bab 8 Referensi**