**LAPORAN PRAKTIKUM 5**

**ANALISIS ALGORITMA**



Disusun oleh :

**Alvin**

**140810180013**

**Kelas A**

Program Studi S-1 Teknik Informatika Departemen Ilmu Komputer

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Padjadjaran

2020

**Nama : Alvin**

**NPM : 140810180013**

**TUGAS-5**

**Studi Kasus 5: Mencari Pasangan Tititk Terdekat (Closest Pair of Points)**

1. Buatlah program untuk menyelesaikan problem closest pair of points menggunakan algoritma divide & conquer yang diberikan. Gunakan bahasa C++

Program :

*/\**

*Nama : Alvin*

*NPM : 140810180013*

*Kelas : A*

*Program : Closest Pair of Points*

*\*/*

#include <bits/stdc++.h>

**using** **namespace** std**;**

**class** Point **{**

**public:**

**int** x**,** y**;**

**};**

**int** compareX**(const** **void\*** a**,** **const** **void\*** b**){**

Point **\***p1 **=** **(**Point **\*)**a**,** **\***p2 **=** **(**Point **\*)**b**;**

**return** **(**p1**->**x **-** p2**->**x**);**

**}**

**int** compareY**(const** **void\*** a**,** **const** **void\*** b**){**

Point **\***p1 **=** **(**Point **\*)**a**,** **\***p2 **=** **(**Point **\*)**b**;**

**return** **(**p1**->**y **-** p2**->**y**);**

**}**

**float** dist**(**Point p1**,** Point p2**){**

**return** sqrt**(** **(**p1**.**x **-** p2**.**x**)\*(**p1**.**x **-** p2**.**x**)** **+** **(**p1**.**y **-** p2**.**y**)\*(**p1**.**y **-** p2**.**y**));**

**}**

**float** bruteForce**(**Point P**[],** **int** n**){**

**float** min **=** FLT\_MAX**;**

**for** **(int** i **=** 0**;** i **<** n**;** **++**i**)**

**for** **(int** j **=** i**+**1**;** j **<** n**;** **++**j**)**

**if** **(**dist**(**P**[**i**],** P**[**j**])** **<** min**)**

min **=** dist**(**P**[**i**],** P**[**j**]);**

**return** min**;**

**}**

**float** min**(float** x**,** **float** y**){**

**return** **(**x **<** y**)?** x **:** y**;**

**}**

**float** stripClosest**(**Point strip**[],** **int** size**,** **float** d**){**

**float** min **=** d**;** *//Inisiasi jarak minimum = d*

qsort**(**strip**,** size**,** **sizeof(**Point**),** compareY**);**

**for** **(int** i **=** 0**;** i **<** size**;** **++**i**)**

**for** **(int** j **=** i**+**1**;** j **<** size **&&** **(**strip**[**j**].**y **-** strip**[**i**].**y**)** **<** min**;** **++**j**)**

**if** **(**dist**(**strip**[**i**],**strip**[**j**])** **<** min**)**

min **=** dist**(**strip**[**i**],** strip**[**j**]);**

**return** min**;**

**}**

**float** closestUtil**(**Point P**[],** **int** n**){**

*//Jika ada 2 atau 3 points, gunakan brute force*

**if** **(**n **<=** 3**)**

**return** bruteForce**(**P**,** n**);**

**int** mid **=** n**/**2**;**

Point midPoint **=** P**[**mid**];**

**float** dl **=** closestUtil**(**P**,** mid**);**

**float** dr **=** closestUtil**(**P **+** mid**,** n **-** mid**);**

**float** d **=** min**(**dl**,** dr**);**

Point strip**[**n**];**

**int** j **=** 0**;**

**for** **(int** i **=** 0**;** i **<** n**;** i**++)**

**if** **(**abs**(**P**[**i**].**x **-** midPoint**.**x**)** **<** d**)**

strip**[**j**]** **=** P**[**i**],** j**++;**

**return** min**(**d**,** stripClosest**(**strip**,** j**,** d**)** **);**

**}**

**float** closest**(**Point P**[],** **int** n**){**

qsort**(**P**,** n**,** **sizeof(**Point**),** compareX**);**

**return** closestUtil**(**P**,** n**);**

**}**

**int** main**(){**

Point P**[]** **=** **{{**1**,** 3**},** **{**11**,** 29**},** **{**39**,** 49**},** **{**4**,** 2**},** **{**11**,** 9**},** **{**2**,** 5**}};**

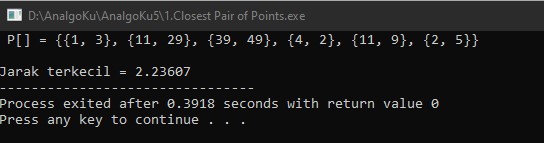
**int** n **=** **sizeof(**P**)** **/** **sizeof(**P**[**0**]);**

cout**<<" P[] = {{1, 3}, {11, 29}, {39, 49}, {4, 2}, {11, 9}, {2, 5}}"<<**endl**<<**endl**;**

cout**<<"Jarak terkecil = "<<**closest**(**P**,** n**);**

**}**

Screenshot :



1. Tentukan rekurensi dari algoritma tersebut, dan selesaikan rekurensinya menggunakan metode recursion tree untuk membuktikan bahwa algoritma tersebut memiliki Big-O (n lg n)

Jawab :

* **Kompleksitas Waktu**

Biarkan kompleksitas waktu dari algoritma di atas menjadi T (n). Mari kita asumsikan bahwa kita menggunakan algoritma pengurutan O (nLogn). Algoritma di atas membagi semua titik dalam dua set dan secara rekursif memanggil dua set. Setelah membelah, ia menemukan strip dalam waktu O (n), mengurutkan strip dalam waktu O (nLogn) dan akhirnya menemukan titik terdekat dalam strip dalam waktu O (n). Jadi T (n) dapat dinyatakan sebagai berikut

T (n) = 2T (n / 2) + O (n) + O (nLogn) + O (n)

T (n) = 2T (n / 2) + O (nLogn)

T (n) = T (n x Logn x Logn)

**Studi Kasus 6: Algoritma Karatsuba untuk Perkalian Cepat**

1. Buatlah program untuk menyelesaikan problem fast multiplication menggunakan algoritma divide & conquer yang diberikan (Algoritma Karatsuba). Gunakan bahasa C++

Program :

*/\**

*Nama : Alvin*

*NPM : 140810180013*

*Kelas : A*

*Program : Karatsuba*

*\*/*

#include<iostream>

#include<stdio.h>

**using** **namespace** std**;**

**int** makeEqualLength**(**string **&**str1**,** string **&**str2**){**

**int** len1 **=** str1**.**size**();**

**int** len2 **=** str2**.**size**();**

**if** **(**len1 **<** len2**){**

**for** **(int** i **=** 0 **;** i **<** len2 **-** len1 **;** i**++)**

str1 **=** '0' **+** str1**;**

**return** len2**;**

**}**

**else** **if** **(**len1 **>** len2**){**

**for** **(int** i **=** 0 **;** i **<** len1 **-** len2 **;** i**++)**

str2 **=** '0' **+** str2**;**

**}**

**return** len1**;** *// If len1 >= len2*

**}**

string addBitStrings**(** string first**,** string second **){**

string result**;**

**int** length **=** makeEqualLength**(**first**,** second**);**

**int** carry **=** 0**;**

**for** **(int** i **=** length**-**1 **;** i **>=** 0 **;** i**--){**

**int** firstBit **=** first**.**at**(**i**)** **-** '0'**;**

**int** secondBit **=** second**.**at**(**i**)** **-** '0'**;**

**int** sum **=** **(**firstBit **^** secondBit **^** carry**)+**'0'**;**

result **=** **(char)**sum **+** result**;**

carry **=** **(**firstBit**&**secondBit**)** **|** **(**secondBit**&**carry**)** **|** **(**firstBit**&**carry**);**

**}**

**if** **(**carry**)** result **=** '1' **+** result**;**

**return** result**;**

**}**

**int** multiplyiSingleBit**(**string a**,** string b**){**

**return** **(**a**[**0**]** **-** '0'**)\*(**b**[**0**]** **-** '0'**);**

**}**

**long** **int** multiply**(**string X**,** string Y**){**

**int** n **=** makeEqualLength**(**X**,** Y**);**

**if** **(**n **==** 0**)** **return** 0**;**

**if** **(**n **==** 1**)** **return** multiplyiSingleBit**(**X**,** Y**);**

**int** fh **=** n**/**2**;**

**int** sh **=** **(**n**-**fh**);**

string Xl **=** X**.**substr**(**0**,** fh**);**

string Xr **=** X**.**substr**(**fh**,** sh**);**

string Yl **=** Y**.**substr**(**0**,** fh**);**

string Yr **=** Y**.**substr**(**fh**,** sh**);**

**long** **int** P1 **=** multiply**(**Xl**,** Yl**);**

**long** **int** P2 **=** multiply**(**Xr**,** Yr**);**

**long** **int** P3 **=** multiply**(**addBitStrings**(**Xl**,** Xr**),** addBitStrings**(**Yl**,** Yr**));**

**return** P1**\*(**1**<<(**2**\***sh**))** **+** **(**P3 **-** P1 **-** P2**)\*(**1**<<**sh**)** **+** P2**;**

**}**

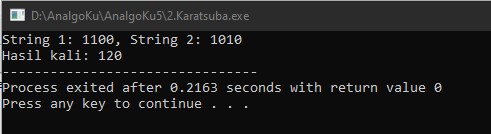
**int** main**(){**

cout**<<"String 1: 1100, String 2: 1010"<<**endl**;**

cout**<<"Hasil kali: "<<**multiply**("1100",** **"1010");**

**}**

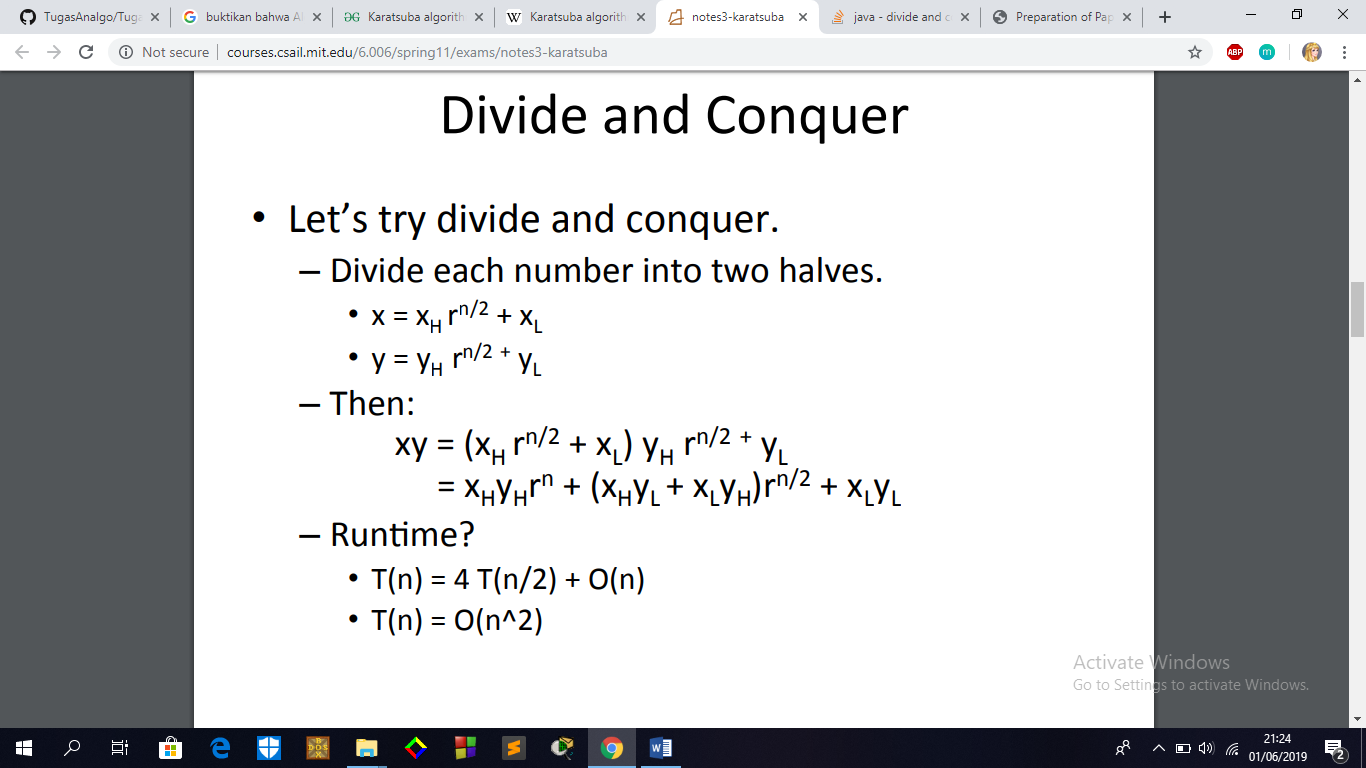
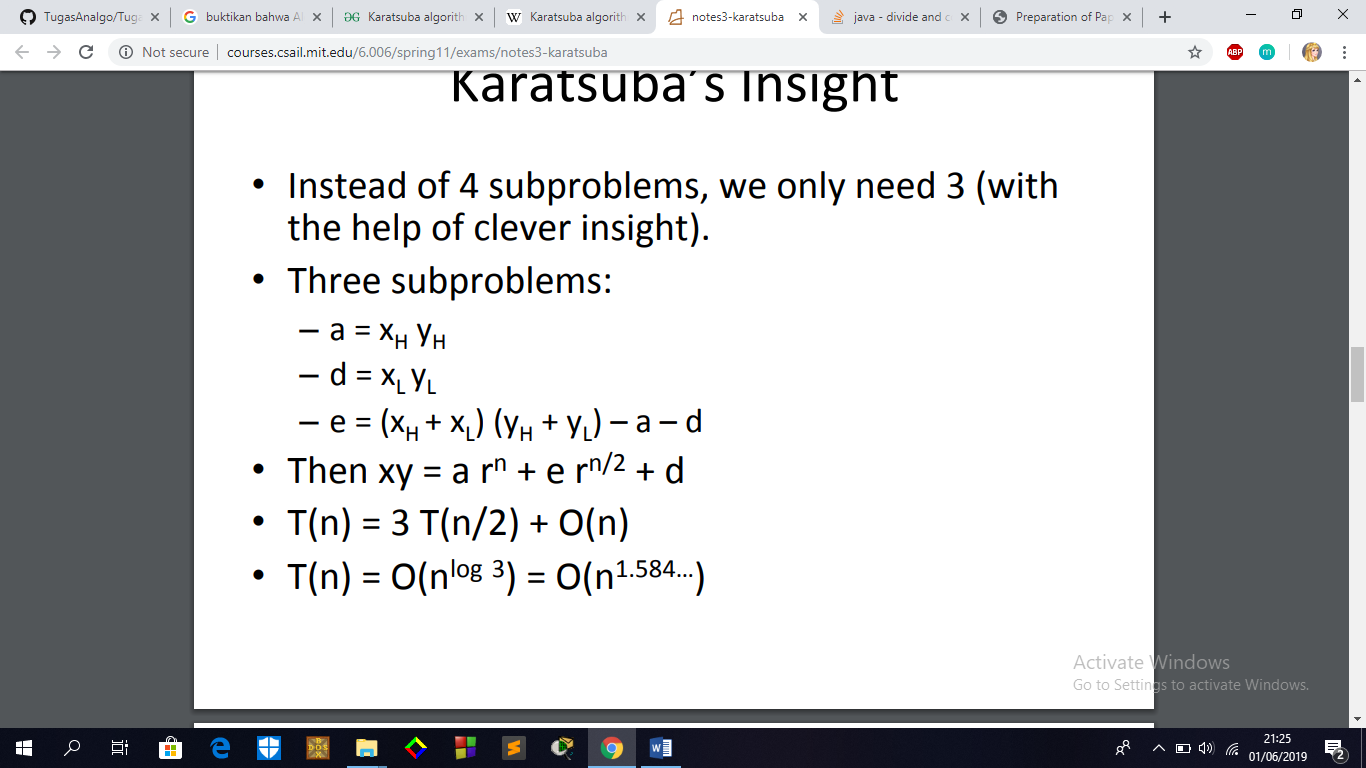
Screenshot :



1. Rekurensi dari algoritma tersebut adalah T (n) = 3T (n / 2) + O (n), dan selesaikan rekurensinya menggunakan metode substitusi untuk membuktikan bahwa algoritma tersebut memiliki Big-O (n lg n)

Jawab :

* Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 6 titik input: 187 ms

Pembuktian dari algoritma:

Menggunakan algoritma Divide dan Conquer, kita dapat melipatgandakan dua bilangan bulat dalam kompleksitas waktu yang lebih sedikit. Bagi angka yang diberikan dalam dua bagian. Biarkan angka yang diberikan menjadi X dan Y.

Untuk kesederhanaan, mari kita asumsikan bahwa n adalah genap:

X = Xl\*2n/2 + Xr [Xl dan Xr mengandung n/2 bit paling kiri dan paling kanan X]

Y = Yl\*2n/2 + Yr [Yl dan Yr mengandung n/2 bit paling kiri dan paling kanan Y]

Hasilnya seperti ini:

XY = (Xl\*2n/2 + Xr)(Yl\*2n/2 + Yr)

= 2n XlYl + 2n/2(XlYr + XrYl) + XrYr

Jika kita melihat rumus di atas, ada empat perkalian ukuran n / 2, jadi pada dasarnya kita membagi masalah ukuran n menjadi empat sub-masalah ukuran n / 2. Tetapi itu tidak membantu karena solusi pengulangan T (n) = 4T (n / 2) + O (n) adalah O (n ^ 2). Bagian rumit dari algoritma ini adalah mengubah dua bagian tengah ke bentuk lain sehingga hanya satu perkalian tambahan yang cukup. Berikut ini adalah ekspresi sulit untuk dua bagian tengah:

XlYr + XrYl = (Xl + Xr) (Yl + Yr) - XlYl- XrYr

Jadi nilai akhir XY menjadi:

XY = 2n XlYl + 2n / 2 \* [(Xl + Xr) (Yl + Yr) - XlYl - XrYr] + XrYr

Dengan trik di atas, perulangan menjadi T (n) = 3T (n / 2) + O (n) dan solusi dari perulangan ini adalah O (n1.59).

Untuk menangani kasus panjang yang berbeda, tambahkan 0 di awal. Untuk menangani panjang ganjil, tempatkan bit bawah (n / 2) di setengah kiri dan atas (n / 2) bit di setengah kanan. Jadi ekspresi untuk XY berubah menjadi berikut:

XY = 22bawah(n/2) XlYl + 2bawah(n/2) \* [(Xl + Xr)(Yl + Yr) - XlYl - XrYr] + XrYr

**T(n) = O (n1.59)**

**Studi Kasus 7: Permasalahan Tata Letak Keramik Lantai (Tilling Problem)**

1. Buatlah program untuk menyelesaikan problem tilling menggunakan algoritma divide & conquer yang diberikan. Gunakan bahasa C++

Jawab :

// n adalah ukuran kotak yang diberikan, p adalah lokasi sel yang hilang

Tile (int n, Point p)

1) Kasus dasar: n = 2, A 2 x 2 persegi dengan satu sel yang hilang tidak ada apa-apanya

   tapi ubin dan bisa diisi dengan satu ubin.

2) Tempatkan ubin berbentuk L di tengah sehingga tidak menutupi

   subsquare n / 2 \* n / 2 yang memiliki kuadrat yang hilang. Sekarang keempatnya

   subskuen ukuran n / 2 x n / 2 memiliki sel yang hilang (sel yang tidak

   perlu diisi). Lihat gambar 2 di bawah ini.

3) Memecahkan masalah secara rekursif untuk mengikuti empat. Biarkan p1, p2, p3 dan

   p4 menjadi posisi dari 4 sel yang hilang dalam 4 kotak.

   a) Ubin (n / 2, p1)

   b) Ubin (n / 2, p2)

   c) Ubin (n / 2, p3)

   d) Ubin (n / 2, p3)

1. Relasi rekurensi untuk algoritma rekursif di atas dapat ditulis seperti di bawah ini. C adalah konstanta. T (n) = 4T (n / 2) + C. Selesaikan rekurensi tersebut dengan Metode Master

Jawab :

Kompleksitas Waktu:

Relasi perulangan untuk algoritma rekursif di atas dapat ditulis seperti di bawah ini. C adalah konstanta.

T (n) = 4T (n / 2) + C

Rekursi di atas dapat diselesaikan dengan menggunakan Metode Master dan kompleksitas waktu adalah O (n2)

Pengerjaan algoritma Divide and Conquer dapat dibuktikan menggunakan Mathematical Induction. Biarkan kuadrat input berukuran 2k x 2k di mana k> = 1.

Kasus Dasar: Kita tahu bahwa masalahnya dapat diselesaikan untuk k = 1. Kami memiliki 2 x 2 persegi dengan satu sel hilang.

Hipotesis Induksi: Biarkan masalah dapat diselesaikan untuk k-1.

Sekarang perlu dibuktikan untuk membuktikan bahwa masalah dapat diselesaikan untuk k jika dapat diselesaikan untuk k-1. Untuk k, ditempatkan ubin berbentuk L di tengah dan memiliki empat subsqure dengan dimensi 2k-1 x 2k-1 seperti yang ditunjukkan pada gambar 2 di atas. Jadi jika dapat menyelesaikan 4 subskuares, dapat menyelesaikan kuadrat lengkap.