Nama : Muhamad Farid Ridho Rambe  
NPM : 140810180033  
Kelas : A  
  
**Studi Kasus 5: Mencari Pasangan Titik Terdekat (Closest Pair of Points)**

Tugas:

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

Program :

/\*

Nama : Muhamad Farid Ridho Rambe

NPM : 140810180033

Kelas : A

Tanggal : 30 Maret 2020

Program : Closest Pair of Point

\*/

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

float min = 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]);

}

}

}

}

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

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);

}

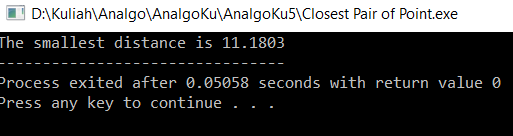
int main(){

Point P[]={{6,1},{4,12},{44,56}};

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

cout<<"The smallest distance is "<<closest(P,n);

return 0;

}  


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)

Catatan

1. Kompleksitas waktu dapat ditingkatkan menjadi O (nLogn) dengan mengoptimalkan langkah 5 dari algoritma di atas.
2. Kode menemukan jarak terkecil. Dapat dengan mudah dimodifikasi untuk menemukan titik dengan jarak terkecil.
3. Kode ini menggunakan pengurutan cepat yang bisa O (n ^ 2) dalam kasus terburuk. Untuk memiliki batas atas sebagai O (n (Logn) ^ 2), algoritma pengurutan O (nLogn) seperti pengurutan gabungan atau pengurutan tumpukan dapat digunakan

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

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

Jawab :

Program :

/\*

Nama : Muhamad Farid Ridho Rambe

NPM : 140810180033

Kelas : A

Tanggal : 30 Maret 2020

Program : Problem Fast Multiplication Karatsuba Algorithm

\*/

#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;

}

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(){

printf ("%ld\n", multiply("1111", "0010"));

printf ("%ld\n", multiply("1100", "0011"));

printf ("%ld\n", multiply("1100", "1010"));

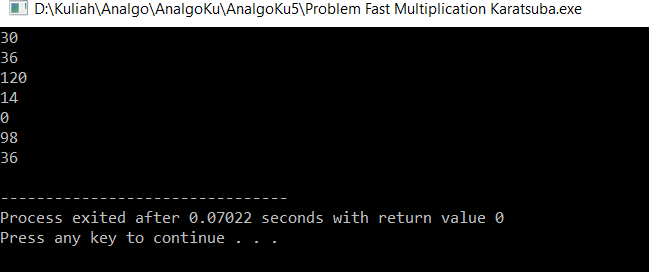
printf ("%ld\n", multiply("0001", "1110"));

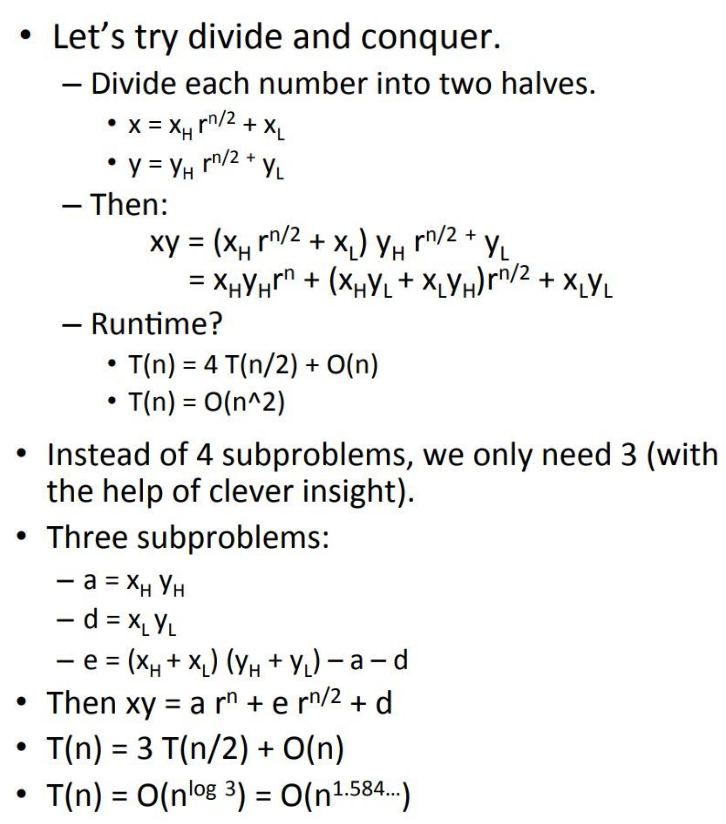
printf ("%ld\n", multiply("0000", "1010"));

printf ("%ld\n", multiply("0111", "1110"));

printf ("%ld\n", multiply("0011", "1100"));

}

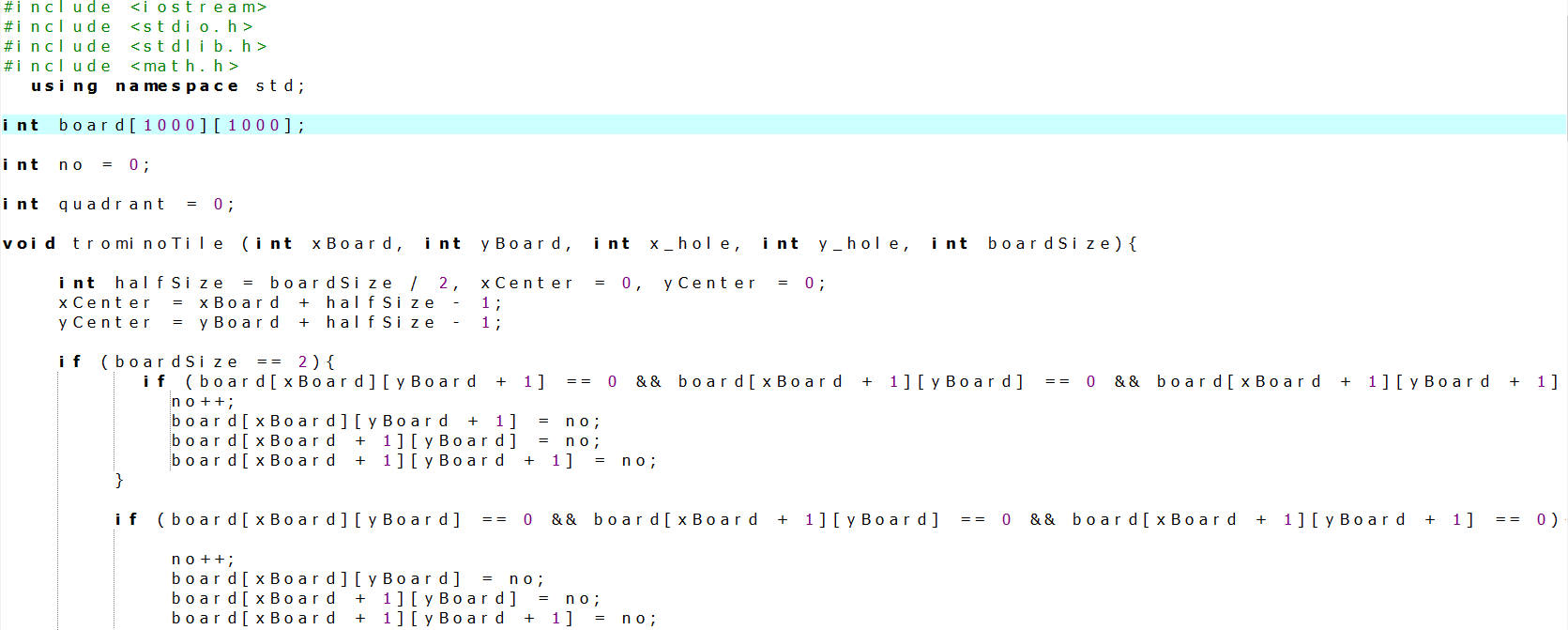
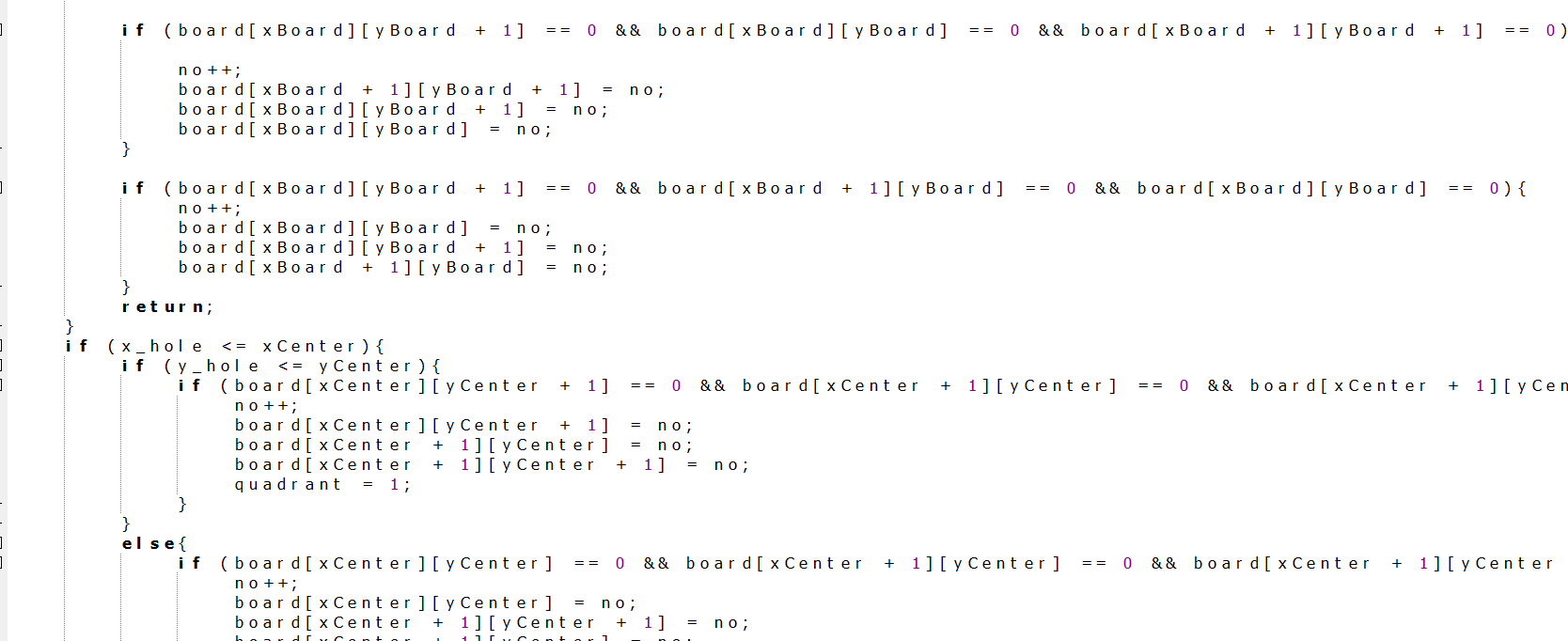
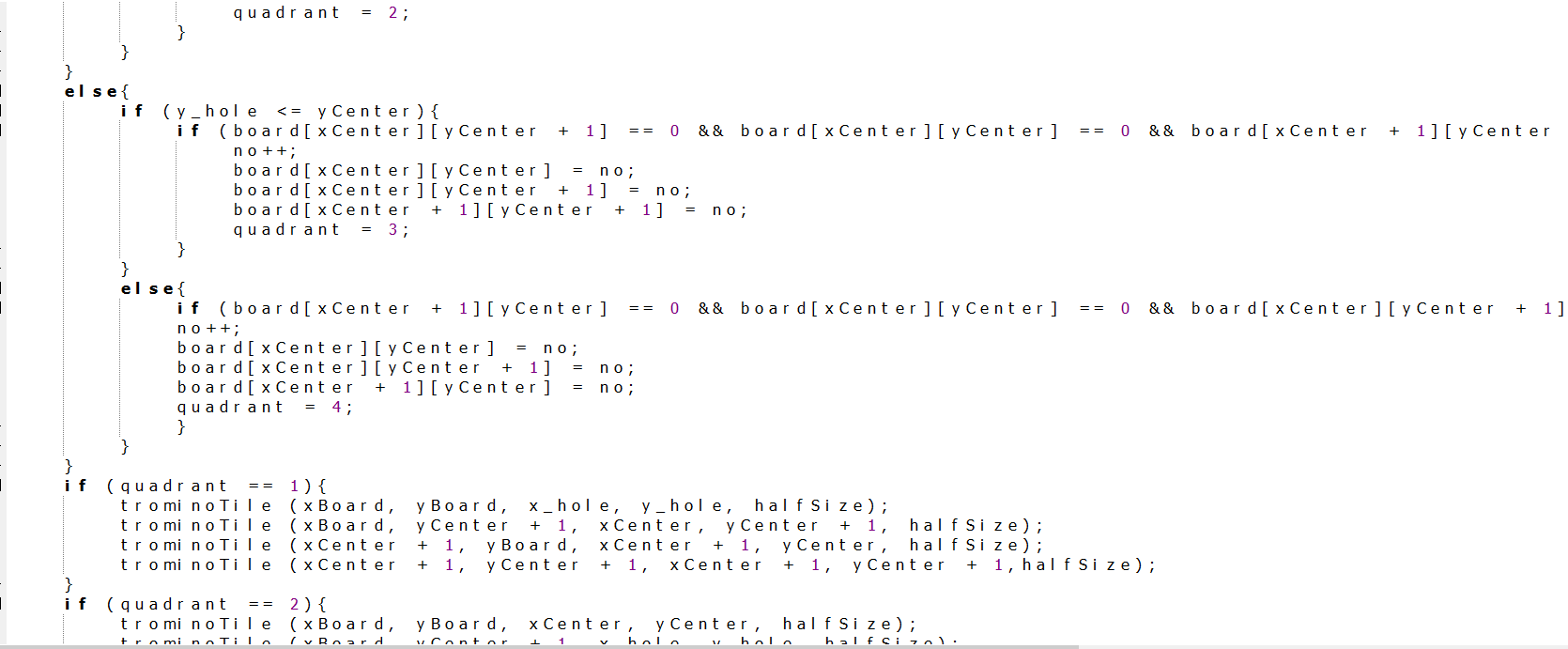
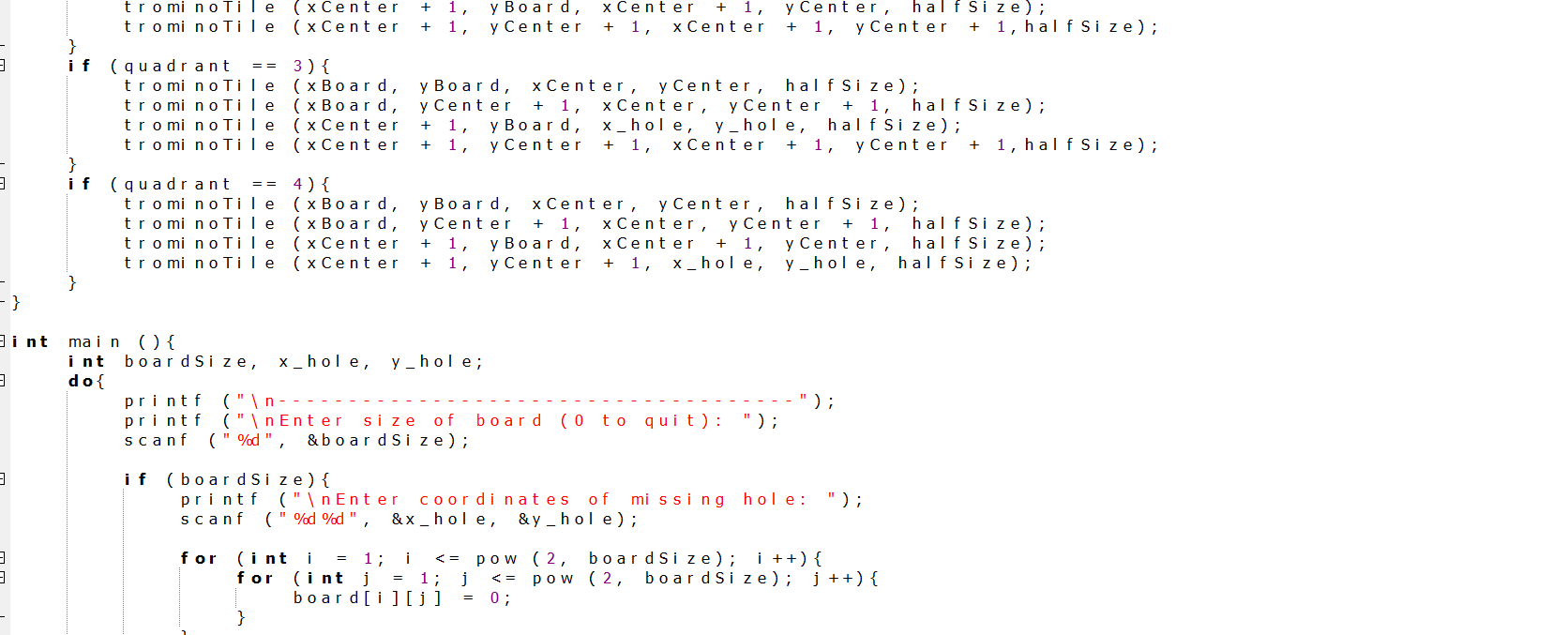
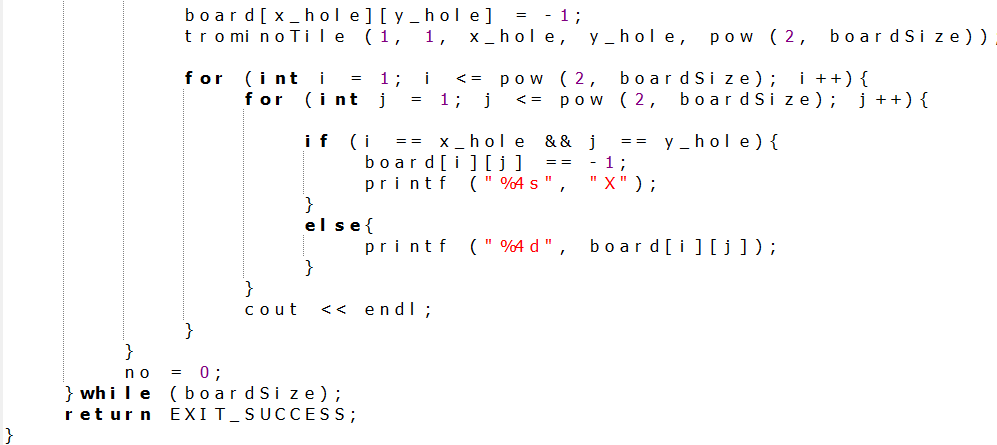
1.   
     
   Rekurensi dari algoritma tersebut adalah T (n) = 3T (n / 2) + O (n), dan selesaikan rekurensinya menggunakan metode substitusi untuk membuktikan bahwa algoritma tersebut memiliki Big-O (n lg n)

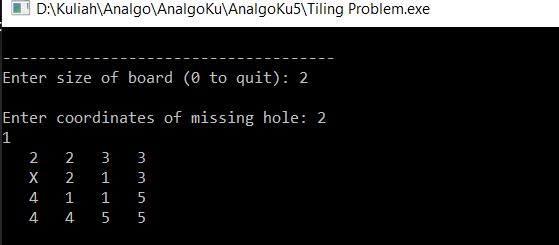
Jawab :

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

Tugas:

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

Program :  
  
  
  
  


Screenshot :  


* 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.\

1. Memecahkan masalah secara rekursif untuk mengikuti empat. Biarkan p1, p2, p3

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

* 1. Ubin (n / 2, p1)
  2. Ubin (n / 2, p2)
  3. Ubin (n / 2, p3)
  4. 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)

Bagaimana cara kerjanya?

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.