Alvina Vania Kirana
140810180010
Tugas 5
Praktikum Analgo

Kasus 5 Mencari Pasangan Titik Terdekat

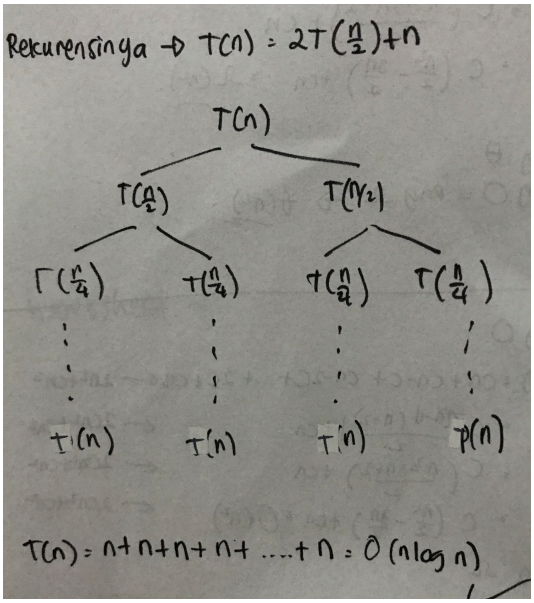
```
/*
Alvina Vania Kirana
140810180010
Praktikum Analisis Algoritma
Worksheet 5 kasus 5
Closest Pair of Points
*/

#include <iostream>
#include <float.h>
#include <stdlib.h>
#include <math.h>
using namespace std;


class Point
{
    public:
    int x, y;
};

int compareX(const void* a, const void* b);
int compareY(const void* a, const void* b);
float dist(Point p1, Point p2);
float bruteForce(Point P[], int n);
float min(float x, float y) ;
float stripClosest(Point strip[], int ukuran, float d);
float closestUtil(Point P[], int n);
float closest(Point P[], int n);

int main() {
    Point P[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3,
4}};
    int n = sizeof(P) / sizeof(P[0]);
```

```
    cout << "Maka jarak terdekatnya adalah " << closest(P,
n)<<endl;
    return 0;
}

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 ukuran, float d) {
    float min = d;

    qsort(strip, ukuran, sizeof(Point), compareY);

    for (int i = 0; i < ukuran; ++i)
```

```
        for (int j = i+1; j < ukuran && (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) {

    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);
}
```

```
Maka jarak terdekatnya adalah 1.41421
alvinas-MacBook-Pro:~ alvinavania$
```

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

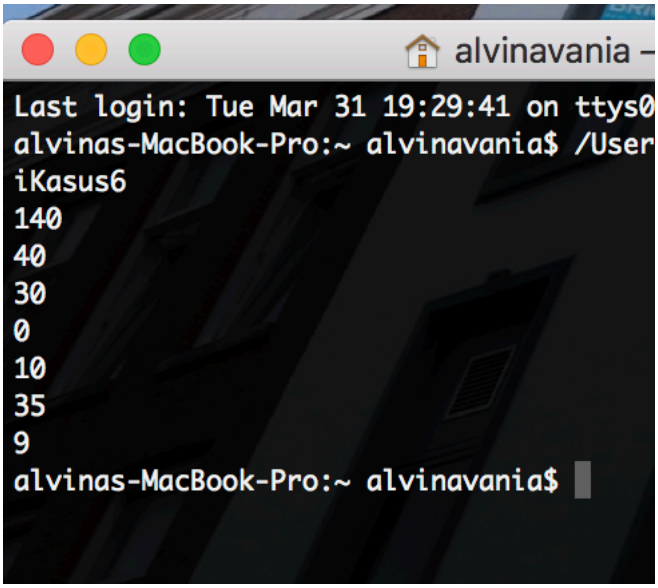


Kasus 6 Algoritma Karatsuba untuk perkalian cepat

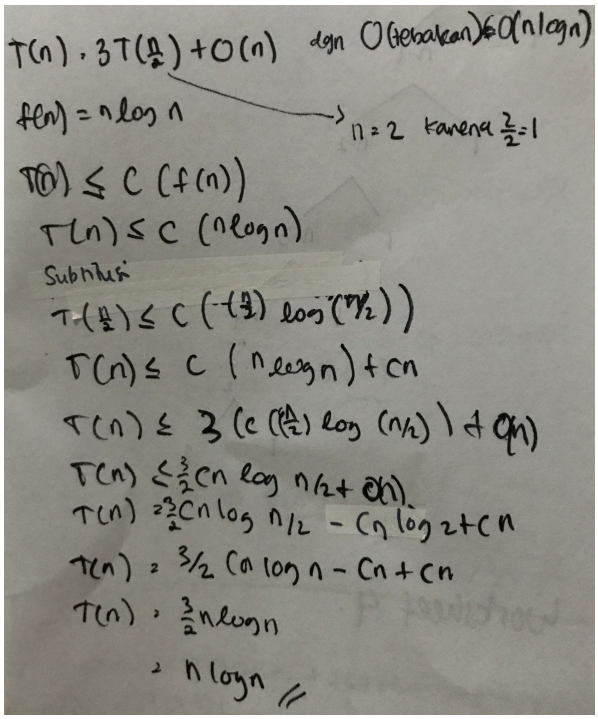
```
/*
Alvina Vania Kirana
140810180010
Praktikum Analisis Algoritma
Worksheet 5 kasus 6
Algoritma Karatsuba
*/

#include <iostream>
using namespace std;

int makeEqualLength(string &a, string &b);
string addBitStrings( string first, string second );
int multiplyiSingleBit(string a, string b);
long int multiply(string X, string Y);

int main()
{
    printf ("%ld\n", multiply("1110", "1010"));
```

```
    printf ("%ld\n", multiply("100", "1010"));
    printf ("%ld\n", multiply("11", "1010"));
    printf ("%ld\n", multiply("0", "1010"));
    printf ("%ld\n", multiply("1", "1010"));
    printf ("%ld\n", multiply("101", "111"));
    printf ("%ld\n", multiply("11", "11"));
}

int makeEqualLength(string &a, string &b){
    int p1 = a.size();
    int p2 = b.size();
    if (p1 < p2)
    {
        for (int i = 0 ; i < p2 - p1 ; i++)
            a = '0' + a;
        return p2;
    }
    else if (p1 > p2)
    {
        for (int i = 0 ; i < p1 - p2 ; i++)
            b = '0' + b;
    }
    return p1;
}

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;
}
```



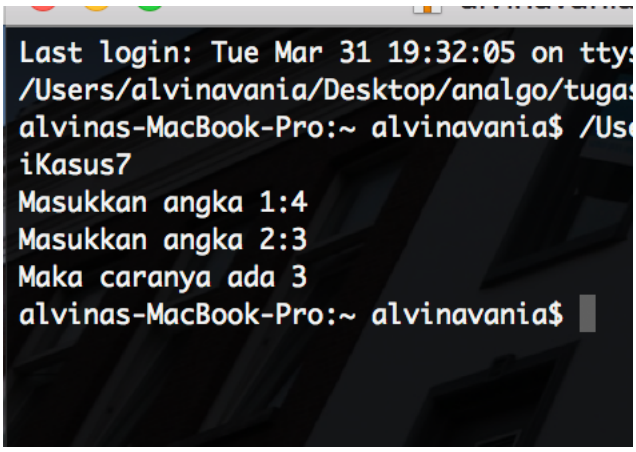
Rekurensi dari algoritma tersebut adalah T (n) = 3T (n / 2) + O (n), dan selesaikan rekurensinya menggunakan metode substitusi untuk membuktikan bahwa algoritma tersebut memiliki Big-O (n lg n)!



Kasus 7 Permasalahan Tata Letak Keramik Lantai (Tilling Problem)

```
/*
Alvina Vania Kirana
140810180010
Praktikum Analisis Algoritma
Worksheet 5 kasus 7
Tilling Problem
*/

#include <iostream>
using namespace std;

int hitung(int a, int b);

int main()
{
    int a, b;
    cout << "Masukkan angka 1:"; cin>> a;
    cout << "Masukkan angka 2:"; cin>>b;
    cout << "Maka caranya ada "<< hitung(a, b)<<endl;

    return 0;
}

int hitung(int a, int b)
{

    int count[a + 1];
    count[0] = 0;

    for (int i = 1; i <= a; i++) {
        if (i > b)
            count[i] = count[i - 1] + count[i - b];
        else if (i < b)
            count[i] = 1;
        else
            count[i] = 2;
    }
    return count[a];
```

```
}
```



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 !