LAPORAN PRAKTIKUM

ANALISIS ALGORITMA

Disusun Oleh:

Delanika Olympiani T. C. 140810180026

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2020

Nama : Delanika Olympiani

NPM : 140810180026

Tugas 5

Studi Kasus 5: Closest Pair of Points

1) Program C++

```
/*
Nama    : Delanika Olympiani
NPM     : 140810180026
Program : mencari pasangan poin terdekat
*/

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

struct Point
{
    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) +
```

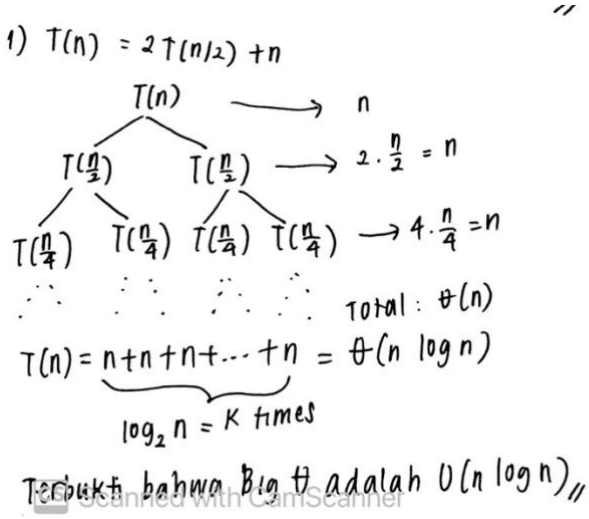
```
            (p1.y - p2.y)*(p1.y - p2.y)
        );
}

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;

    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 Px[], Point Py[], int n)
{
    if (n <= 3)
        return bruteForce(Px, n);

    int mid = n/2;
    Point midPoint = Px[mid];

    Point Pyl[mid+1];
```

```
    Point Pyr[n-mid-1];
    int li = 0, ri = 0;
    for (int i = 0; i < n; i++)
    {
    if (Py[i].x <= midPoint.x)
        Pyl[li++] = Py[i];
    else
        Pyr[ri++] = Py[i];
    }

    float dl = closestUtil(Px, Pyl, mid);
    float dr = closestUtil(Px + mid, Pyr, n-mid);

    float d = min(dl, dr);

    Point strip[n];
    int j = 0;
    for (int i = 0; i < n; i++)
        if (abs(Py[i].x - midPoint.x) < d)
            strip[j] = Py[i], j++;

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

float closest(Point P[], int n)
{
    Point Px[n];
    Point Py[n];
    for (int i = 0; i < n; i++)
    {
        Px[i] = P[i];
        Py[i] = P[i];
    }

    qsort(Px, n, sizeof(Point), compareX);
    qsort(Py, n, sizeof(Point), compareY);

    return closestUtil(Px, Py, n);
}
```

```
int main()
{
    Point P[] = {{1, 5}, {23, 35}, {44, 55}, {3, 1}, {12, 20}, {13, 4}};
    int n = sizeof(P) / sizeof(P[0]);
    cout << "The smallest distance is " << closest(P, n);
    cout << endl;
    return 0;
}
```

```
The smallest distance is 4.47214
Delanikas-MacBook-Air:AnalgoKu5 delanikaotc$
```

2) Rekurensi menggunakan Recursion Tree



Studi Kasus 6: Karatsuba

1) Program C++

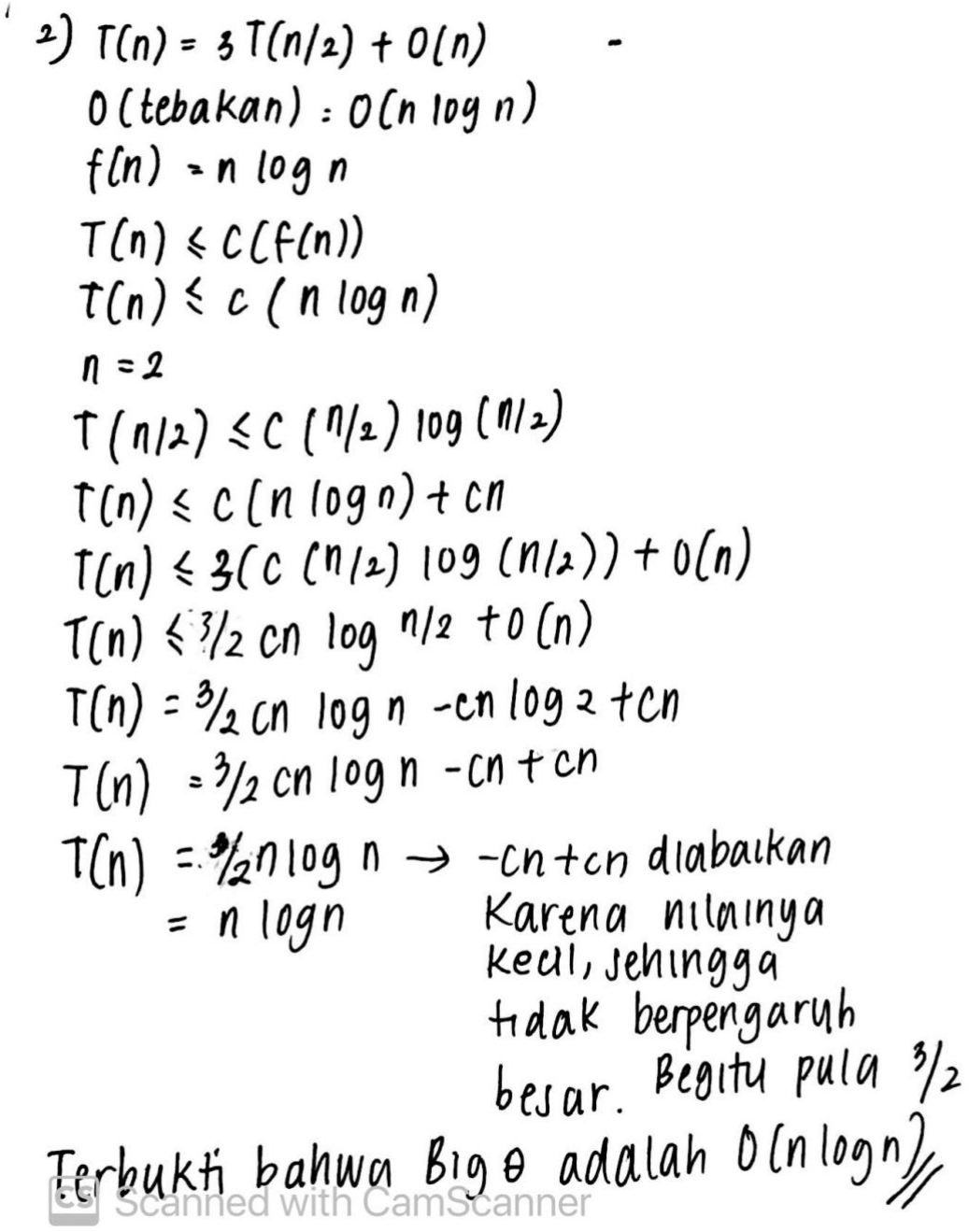
```
/*
Nama    : Delanika Olympiani
NPM     : 140810180026
Program : mengalikan cepat dua buah string biner
*/

#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("1101", "0101"));
    printf ("%ld\n", multiply("111", "1011"));
    printf ("%ld\n", multiply("00", "1111"));
    printf ("%ld\n", multiply("0", "00110"));
    printf ("%ld\n", multiply("11", "0100"));
    printf ("%ld\n", multiply("101", "010"));
    printf ("%ld\n", multiply("00", "11"));
}
```

```
65
77
0
0
12
10
0
Delanikas-MacBook-Air:AnalgoKu5 delanikaotc$
```

2) Pembuktian Big O = O(n log n) menggunakan Metode Substitusi



Studi Kasus 7: Tiling Problem

1) Program C++

```cpp
/*
Nama    : Delanika Olympiani
NPM     : 140810180026
Program : tiling problem
*/

#include<iostream>
using namespace std;

int countWays(int n, int m)
{

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

    for (int i = 1; i <= n; i++) {
        if (i > m)
            count[i] = count[i - 1] + count[i - m];

        else if (i < m)
            count[i] = 1;

        else
            count[i] = 2;
    }

    return count[n];
}

int main()
{
    int n = 10, m = 8;
    cout << "Number of ways = "
         << countWays(n, m);
    cout << endl;
    return 0;
```

```
}
```

```
Number of ways = 4
Delanikas-MacBook-Air:AnalgoKu5 delanikaotc$
```

2) Menyelesaikan rekurensi dengan Metode Master