# LAPORAN PRAKTIKUM 5
## Analisis Algoritma


**Disusun oleh :**

**Asep Budiyana Muharam**
**140810180029**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**
**UNIVERSITAS PADJADJARAN**

**2020**

Asep Budiyana M
140810180029
Tugas 5

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

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

```cpp
#include <iostream>
#include <float.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 distance(Point p1, Point p2)
{
    return sqrt(pow(p1.x-p2.x, 2) + pow(p1.y-p2.y, 2));
}

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 (distance(P[i], P[j]) < min)
                min = distance(P[i], P[j]);
    return min;
}

float min(float x, float y)
{
    if (x < y)
        return x;
```

Asep Budiyana M
140810180029
Tugas 5

```
    else
        return 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 (distance(strip[i], strip[j]) < min)
                min = distance(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 mid_point = 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 <= mid_point.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-mid_point.x) < d)
            strip[j] = Py[i], j++;

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

Asep Budiyana M
140810180029
Tugas 5

```
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[] = {{4, 1}, {15, 20}, {30, 40}, {8, 4}, {13, 11}, {5, 6}};
    int n = sizeof(P)/sizeof(P[0]);
    cout<< "Jarak terkecil adalah "<<closest(P, n);
    return 0;
}
```

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

(5) Closest Pair of Points

- Algoritma 'Closest pair of points' membagi semua titik dalam dua set dan secara rekursif menghasil dua set .... $2T(n/2)$
- menemukan array strip ... $O(n)$
- membagi array Py di sekitar garis tengah ... $O(n)$
- menemukan titik terdekat dalam array strip ... $O(n)$

maka rekurensi dari algoritma tsb, adalah :

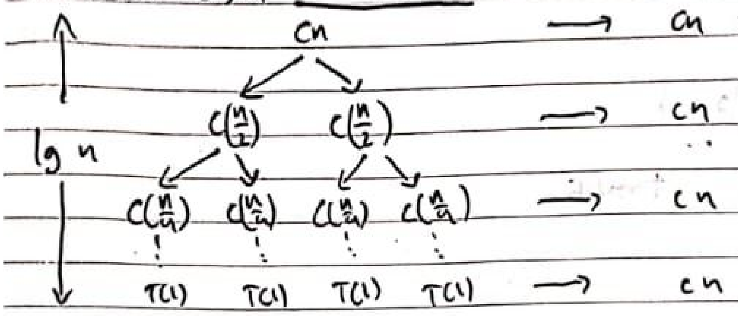
$T(n) = 2T(n/2) + O(n) + O(n) + O(n)$

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

$T(n) = 2T(n/2) + cn$

Asep Budiyana M
140810180029
Tugas 5

Pembuktian dengan 'recursion tree'




Cost utk keseluruhan Tree yaitu :

$$T(n) = \underbrace{cn + cn + cn + \dots + cn}_{\lg n}$$

$$= \sum_{i=0}^{\lg n} cn$$

$$= cn \lg n$$

$$= O(n \lg n)$$

Terbukti

**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++

   **karatsuba.cpp*

```
#include <iostream>
using namespace std;

int equalyzingLength(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 addBit(string first, string second)
```

Asep Budiyana M
140810180029
Tugas 5

```
{
    string result;
    int length = equalyzingLength(first, second);
    int carry = 0;

    for (int i = length-1; i >= 0; i--) {
        int first_bit = first.at(i) - '0';
        int second_bit = second.at(i) - '0';
        int sum = (first_bit ^ second_bit ^ carry) + '0';

        result = (char)sum + result;
        carry = (first_bit & second_bit) | (second_bit & carry) | (first_bit
& carry);
    }

    if (carry)
        result = '1' + result;
    return result;
}

int multiplySingleBit(string a, string b)
{
    return (a[0] - '0')*(b[0] - '0');
}

long int multiply(string X, string Y)
{
    int n = equalyzingLength(X, Y);

    if (n == 0) return 0;
    if (n == 1) return multiplySingleBit(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(addBit(Xl, Xr), addBit(Yl, Yr));

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

```
}

int main()
{
    cout<<multiply("101001", "101010")<<endl;
    return 0;
}
```

2. 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)

(6) karatsuba untuk Perkalian Cepat

Pembuktian rekurensi dengan '<u>Metode Subsitusi</u>'

$T(n) = 3T(n/2) + O(n)$

$T(n) = 3T(n/2) + cn$

$T(n) \leq 3\left(c\left(\frac{n}{2}\right)\lg\left(\frac{n}{2}\right)\right) + cn$

$\leq \frac{3}{2}cn\lg\left(\frac{n}{2}\right) + cn$

$= cn\lg n - cn\lg 2 + cn$

$= cn\lg n - cn + cn$

$= cn\lg n$

$= O(n\lg n)$

∴ terbukti

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

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

**tiling-problem.cpp*

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <climits>
using namespace std;
```

Asep Budiyana M
140810180029
Tugas 5

```
int ** tromino_tile;

int power_2(int k)
{
    int result = 1;
    for (int i = 0; i < k; i++) {
        result = result * 2;
    }
    return result;
}

void allocate(int n, int x, int y)
{
    tromino_tile = new int * [n];
    for (int i = 0; i < n; i++) {
        tromino_tile[i] = new int[n];
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            tromino_tile[i][j] = -1;
        }
    }
    tromino_tile[x][y] = 100;
}

void print_tromino(int n)
{
    for (int i = 0; i < n; i++) {
        cout<<"\n";
        for (int j = 0; j < n; j++) {
            if (tromino_tile[i][j] == 100)
                cout<<"\t"<<"X";
            else
                cout<<"\t"<<tromino_tile[i][j];
        }
        cout<<"\n";
    }
}

void free_memory(int n)
{
    for (int i = 0; i < n; ++i) {
        delete [] tromino_tile[i];
    }
```

Asep Budiyana M
140810180029
Tugas 5

```
    delete [] tromino_tile;
}

void trominoTile(int n, int hole_row, int hole_col, int x, int y)
{
    int i;
    int half = n/2;

    if (n == 2) {
        if (hole_row < x + half && hole_col < y + half) {
            i = rand() % 10;
            tromino_tile[x + half][y + half - 1] = i;
            tromino_tile[x + half][y + half] = i;
            tromino_tile[x + half - 1][y + half] = i;
        }

        else if (hole_row < x + half && hole_col >= y + half) {
            i = rand() % 10;
            tromino_tile[x + half][y + half - 1] = i;
            tromino_tile[x + half][y + half] = i;
            tromino_tile[x + half - 1][y + half - 1] = i;
        }

        else if (hole_row >= x + half && hole_col < y + half) {
            i = rand() % 10;
            tromino_tile[x + half - 1][y + half] = i;
            tromino_tile[x + half][y + half] = i;
            tromino_tile[x + half - 1][y + half - 1] = i;
        }

        else {
            i = rand() % 10;
            tromino_tile[x + half - 1][y + half] = i;
            tromino_tile[x + half][y + half - 1] = i;
            tromino_tile[x + half - 1][y + half - 1] = i;
        }
    } else {
        if (hole_row < x + half && hole_col < y + half) {
            i = rand() % 10;
            tromino_tile[x + half][y + half - 1] = i;
            tromino_tile[x + half][y + half] = i;
            tromino_tile[x + half - 1][y + half] = i;

            trominoTile(half, hole_row, hole_col, x, y);
            trominoTile(half, x + half, y + half - 1, x + half, y);
```

Asep Budiyana M
140810180029
Tugas 5

```
            trominoTile(half, x + half, y + half, x + half, y + half);
            trominoTile(half, x + half - 1, y + half, x, y + half);
        } else if (hole_row < x + half && hole_col >= y + half) {
            i = rand() % 10;
            tromino_tile[x + half][y + half - 1] = i;
            tromino_tile[x + half][y + half] = i;
            tromino_tile[x + half - 1][y + half - 1] = i;

            trominoTile(half, hole_row, hole_col, x, y + half);
            trominoTile(half, x + half, y + half - 1, x + half, y);
            trominoTile(half, x + half, y + half, x + half, y + half);
            trominoTile(half, x + half - 1, y + half - 1, x, y);
        } else if(hole_row >= x + half && hole_col < y + half) {
            i = rand() % 10;
            tromino_tile[x + half - 1][y + half] = i;
            tromino_tile[x + half][y + half] = i;
            tromino_tile[x + half - 1][y + half - 1] = i;

            trominoTile(half, hole_row, hole_col, x + half, y);
            trominoTile(half, x + half - 1, y + half, x, y + half);
            trominoTile(half, x + half, y + half, x + half, y + half);
            trominoTile(half, x + half - 1, y + half - 1, x, y);
        } else {
            i = rand() % 10;
            tromino_tile[x + half - 1][y + half] = i;
            tromino_tile[x + half][y + half - 1] = i;
            tromino_tile[x + half - 1][y + half - 1] = i;

            trominoTile(half, hole_row, hole_col, x + half, y + half);
            trominoTile(half, x + half - 1, y + half, x, y + half);
            trominoTile(half, x + half, y + half - 1, x + half, y);
            trominoTile(half, x + half - 1, y + half - 1, x, y);
        }
    }
}

int main()
{
    int k = 2, hole_row = 2, hole_col = 2;
    srand(time(NULL));
    if (k < 1)
        cout<<"\n<nilai k> harus positive\n";
    else {
        if(hole_row == 0 || hole_col == 0)
```

Asep Budiyana M
140810180029
Tugas 5

```
            cout<<"\nThe <jumlah baris> dan <jumlah kolom> harus positive
dan integer\n";
        else {
            int n = power_2(k);

            if (n >= hole_row && n >= hole_col) {
            hole_row--;
            hole_col--;

            allocate(n, hole_row, hole_col);

            trominoTile(n, hole_row, hole_col, 0, 0);
            print_tromino(n);
            free_memory(n);
            }
            else
                cout<<"\nBaris dan Kolom tidak boleh lebih dari 2^k\n";
        }
    }
    return 0;
}
```

2. 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.

7 Tiling Problem

Pembuktian rekurensi dengan 'metode master'

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

$T(n) = 4T(n/2) + 1$

$a = 4, \ b = 2, \ f(n) = 1$

$n^{\log_b a} = n^{\log_2 4}$

$f(n) = 1 = O(n^{\log_2 4 - \varepsilon})$ untuk $\varepsilon = 2$

Case 1 applies

$T(n) = O(n^2)$