Praktikum Analisis Algoritma **Naufal Aulia**
Worksheet 5 **140810180074**

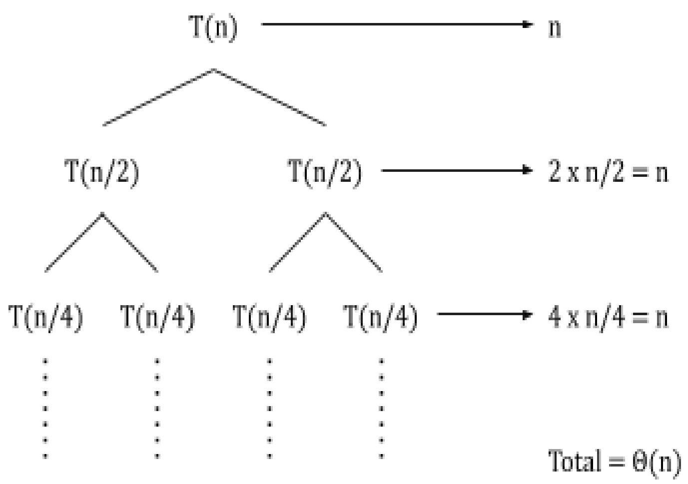
Studi Kasus 5 : Closest Pair of Points

Kode

```
#include <iostream>
#include <math.h>
#include <float.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; // Initialize the minimum distance as 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)
{
    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[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}, {1, 20}, {7, 2},
{4, 5}, {12, 4}, {2, 11}};
    int n = sizeof(P) / sizeof(P[0]);
    cout << "The smallest distance is " << closest(P, n);
    return 0;
}
```

Rekurensi dengan Recursion Tree



$T(n) = n + n + n + \cdots + n = \Theta(n \; log \; n)$

Terbukti bahwa big Θ adalah O(n log n)

Studi Kasus 6 : Karatsuba Fast Multiplication

Kode

```
#include<iostream>

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()
{
    cout << multiply("1001", "0010") << "\n";
    cout << multiply("1101", "1110") << "\n";
    cout << multiply("00001010", "00011010") << "\n";
    cout << multiply("11010100", "11011011") << "\n";
```

Rekurensi dengan Substitusi

$T(n) = 3(T(\frac{n}{2}) + O(n)$

$O(tebakan) \; = \; O(n \; log \; n)$

$f(n) \; = \; n \; log \; n$

$T(n) \leq C(f(n))$

$T(n) \leq C(n \; log \; n)$

$n = 2$

$T(\frac{n}{2}) \leq c(\frac{n}{2}) \; log(\frac{n}{2})$

$T(n) \leq c(n \; log \; n) \; + \; cn$

$T(n) \leq 3(c(\frac{n}{2}) \; log(\frac{n}{2})) + O(n)$

$T(n) \leq \frac{3}{2} \; cn \; log \; \frac{n}{2} \; + O(n)$

$T(n) \; = \frac{3}{2} \; cn \; log \; n \; - \; cn \; log \; 2 \; + cn$

$T(n) \; = \frac{3}{2} \; cn \; log \; n \; - \; cn + cn$

$T(n) \; = \frac{3}{2} \; cn \; log \; n \quad \rightarrow - \; cn + cn$ dan $\frac{3}{2}$ diabaikan karena nilainya kecil, tidak berpengaruh

$T(n) \; = n \; log \; n$

Terbukti bahwa big Θ adalah O(n log n)

Studi Kasus 7 : Tiling Problem

Kode

```
#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 = 8, m = 4;
    cout << "Number of ways = "
         << countWays(n, m);
    return 0;
}
```

Rekurensi dengan Metode Master

$T(n) = 4T(\frac{n}{2}) + c$

$a = 4 \; ; b = 2 \; ; \; f(n) = c$

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

$\quad\quad = n^2$

$f(n) = n = \Theta(n^{log_2 4 - \varepsilon})$

$\varepsilon = 1 \longrightarrow f(n) \;\; = \;\; \Theta(n^{log_2 4 - 1})$

$f(n) \;\; = \;\; \Theta(n^{log_2 3})$

$f(n) \;\; = \;\; \Theta(n^{1.585})$

$f(n) \;\; = \;\; \Theta(n^2)$