Nama : Prayudha A.L

NPM : 140810180008

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++

```
#include <bits/stdc++.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 size, 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 << "The smallest distance is " << closest(P, n);
    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 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]);

    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);
}
```

```
C:\Users\Libra\Documents\MEGA\Semester 4\Analgo\Praktikum\Analgoku\Analgoku5>.\"closestPairofPoints.exe"
The smallest distance is 1.41421
```

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

kompleksitas waktu = T(n)

O (nlogn) membagi dan menata strip→ ditemukan titik terdekat dalam strip dalam waktu O (n)

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)

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

Diberikan dua string biner yang mewakili nilai dua bilangan bulat, cari produk (hasil kali) dari dua string. Misalnya, jika string bit pertama adalah "1100" dan string bit kedua adalah "1010", output harus 120. Supaya lebih sederhana, panjang dua string sama dan menjadi n.

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

```
#include <iostream>
#include <stdio.h>

using namespace std;

int makeEqualLength(string &str1, string &str2);
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("1100", "1010"));
    printf("%ld\n", multiply("110", "1010"));
    printf("%ld\n", multiply("11", "1010"));
    printf("%ld\n", multiply("1", "1010"));
    printf("%ld\n", multiply("0", "1010"));
    printf("%ld\n", multiply("111", "111"));
    printf("%ld\n", multiply("11", "11"));
}

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 & car
ry);
    }

    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;
}
```

```
C:\Users\Libra\Documents\MEGA\Semester 4\Analgo\Praktikum\Analgoku\Analgoku5>.\"karatsuba.exe"
120
60
30
10
0
49
9
```

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).

Bagi tiap nomor menjadi 2 bagian

- $x = x_H r^{n/2} + x_L$
- $y = y_H r^{n/2} + y_L$

lalu:

$xy = (x_H r^{n/2} + x_L)\, y_H r^{n/2} + y_L \rightarrow x_H y_H r^{n/2} + (x_H y_L + x_L y_H) r^{n/2} + x_L y_L$

- $T(n) = 4\, T(n/2) + O(n)$
- $T(n) = O(n^2)$

Subproblems:

- $a = x_H y_H$
- $d = x_L y_L$
- $e = (x_H + x_L)\,(y_H + y_L) - a - d$

$xy = a\, r^n + e\, r^{n/2} + d$

- $T(n) = 3\, T(n/2) + O(n)$
- $T(n) = O(n^{\log 3}) = O(n^{1.584\ldots})$

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

Diberikan papan berukutan n x n dimana n adalah dari bentuk 2k dimana k ≥ 1 (Pada dasarnya n adalah pangkat dari 2 dengan nilai minimumnya 2). Papan memiliki satu sel yang hilang (ukuran 1 x 1). Isi papan menggunakan ubin berbentuk L. Ubin berbentuk L berukuran 2 x 2 persegi dengan satu sel berukuran 1 × 1 hilang.

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

```cpp
#include <bits/stdc++.h>

using namespace std;

int countWays(int n, int m);

int main()
{
    int n = 4, m = 2;
    cout << "Number of ways = "
         << countWays(n, m);
    return 0;
}

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];
}
```

```
C:\Users\Libra\Documents\MEGA\Semester 4\Analgo\Praktikum\Analgoku\Analgoku5>.\"tilling.exe"
Number of ways = 5
```

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.

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

a = 4, b =2, f(n) =1
$n^{logb\hat{}a} = n^{log2\hat{}4}$

f(a) = 1 = $O(n^{log2\hat{}4 - \varepsilon})$ untuk ε = 2
case 1 applies

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