LAPORAN PRAKTIKUM ANALISIS ALGORITMA

PARADIGMA DIVIDE AND CONQUER

Oleh

Shalvina Zahwa Aulia (140810180052)

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJAJARAN

2020

Studi kasus 5 : Mencari pasangan titik terdekat (closest pair of points)

Tugas:

1) Buatlah program untuk menyelesaikan problem closest pair of points menggunakan algoritma divide & conquer yang diberikan. Gunakan bahasa C++
2) Tentukan rekurensi dari algoritma tersebut, dan selesaikan rekurensinya menggunakan metode recursion tree untuk membuktikan bahwa algoritma tersebut memiliki Big-O (n lg n)

1) Program closest pair of points

```
/*
Nama    : Shalvina Zahwa Aulia
NPM     : 140810180052
Tugas 5 Closest Pair of Points
*/
#include<iostream>
#include <bits/stdc++.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;

	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[] = {{1, 9}, {10, 20}, {12, 56}, {34, 23}, {9, 13}, {3, 10}};
```

```
	int n = sizeof(P) / sizeof(P[0]);
	cout << "Closest pair of points : " << closest(P, n);
	return 0;
}
```

Hasil :

```
"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisisAlgoritma\Praktikum\Tugas\Analgo
Closest pair of points : 2.23607
Process returned 0 (0x0)   execution time : 0.386 s
Press any key to continue.
```

2) Kompleksitas waktu (T(n))

Misal O(nlogn)

Closest pair of points membagi titik ke dalam 2 set dan memanggil 2 set secara rekursif. Setelah membelah, strip ditemukan dalam waktu O(n), waktu mengurutkan strip O(nlogn) dan menemukan closest pair of points dengan wakktu O(n).

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

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

= T(n x log n x log n)

Studi kasus 6 : Algoritma Karatsuba

Tugas:

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

1) Program Karatsuba

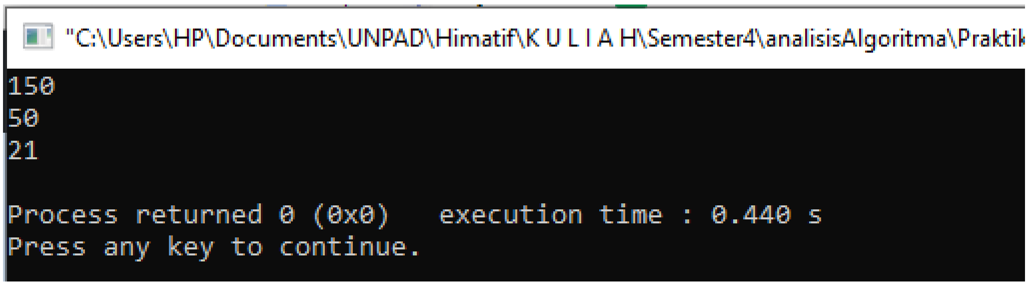
```
/*
Nama    : Shalvina Zahwa Aulia
NPM     : 140810180052
Tugas 5 karatsuba
*/
#include<iostream>
#include<stdio.h>
```

```
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; // If len1 >= len2
}

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);

    // Base cases
    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("1111", "1010"));
        printf ("%ld\n", multiply("101", "1010"));
        printf ("%ld\n", multiply("111", "11"));
}
```

Hasil :



2) Pembuktian :

- Divide setiap angka menjadi 2 bagian

  $X = X_H r^{n/2} + X_L$

  $Y = Y_H r^{n/2} + Y_L$

- $XY = (X_H r^{n/2} + X_L)\ Y_H r^{n/2} + Y_L$

  $\quad = X_H Y_H r^n + (X_H Y_L + X_L Y_H) r^{n/2} + X_L Y_L$

- Runtime :

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

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

- 3 subproblem :

  $a = X_H Y_H$

  $d = X_L Y_L$

  $e = (X_H + X_L)(Y_H + Y_L) – a – d$

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

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

$T(n) = O(n^{\log 3})$

Studi kasus 7 : Permasalahan tata letak keramik lantai (tilling problem)

Tugas:

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

1) Program tilling problem

```
/*
Nama    : Shalvina Zahwa Aulia
NPM     : 140810180052
Tugas Tilling Problem
*/
#include <bits/stdc++.h>

using namespace std;

int hitungCara(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 = 7, m = 3;
    cout << "Hasil = "
         << hitungCara(n, m);
    return 0;
}
```

Hasil :

```
"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisisAlgoritma\Praktikum\Tug
Hasil = 9
Process returned 0 (0x0)   execution time : 0.514 s
Press any key to continue.
```

2) Kompleksitas Waktu:

Relasi perulangan untuk algoritma rekursif di atas dapat ditulis seperti di bawah ini. C adalah konstanta. T (n) = 4T (n / 2) + C

Rekursi di atas dapat diselesaikan dengan menggunakan Metode Master dan kompleksitas waktu adalah O (n2)

Pengerjaan algoritma Divide and Conquer dapat dibuktikan menggunakan Mathematical Induction. Biarkan kuadrat input berukuran 2k x 2k di mana k> = 1.

Kasus Dasar: Kita tahu bahwa masalahnya dapat diselesaikan untuk k = 1. Kami memiliki 2 x 2 persegi dengan satu sel hilang. Hipotesis Induksi: Biarkan masalah dapat diselesaikan untuk k-1.

Sekarang perlu dibuktikan untuk membuktikan bahwa masalah dapat diselesaikan untuk k jika dapat diselesaikan untuk k-1. Untuk k, ditempatkan ubin berbentuk L di tengah dan memiliki empat subsqure dengan dimensi 2k-1 x 2k-1 seperti yang ditunjukkan pada gambar 2 di atas. Jadi jika dapat menyelesaikan 4 subskuares, dapat menyelesaikan kuadrat lengkap.