TUGAS 5 PRAKTIKUM

ANALISIS ALGORITMA

Disusun Oleh:

Kefilino Khalifa Filardi

140810180028

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2020

Studi Kasus (Lanjutan)

Studi Kasus 5: Mencari Pasangan Tititk Terdekat (Closest Pair of Points) Identifikasi

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)

Jawaban:

1. Kode Program:

```
/*
 * Kefilino Khalifa Filardi
 * 140810180028
 * Tugas 5
 */

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

struct point {
    int x, y;
};

void input(int &n)
{
    cout << "Banyak titik dalam array : ";
    cin >> n;
}

void input(point P[], int n)
{
    cout << endl;
    for (int i = 0; i < n; i++) {
        cout << "Masukkan nilai x titik ke-" << i+1 << " : ";
        cin >> P[i].x;
        cout << "Masukkan nilai y titik ke-" << i+1 << " : ";
        cin >> P[i].y;
    }
}
```

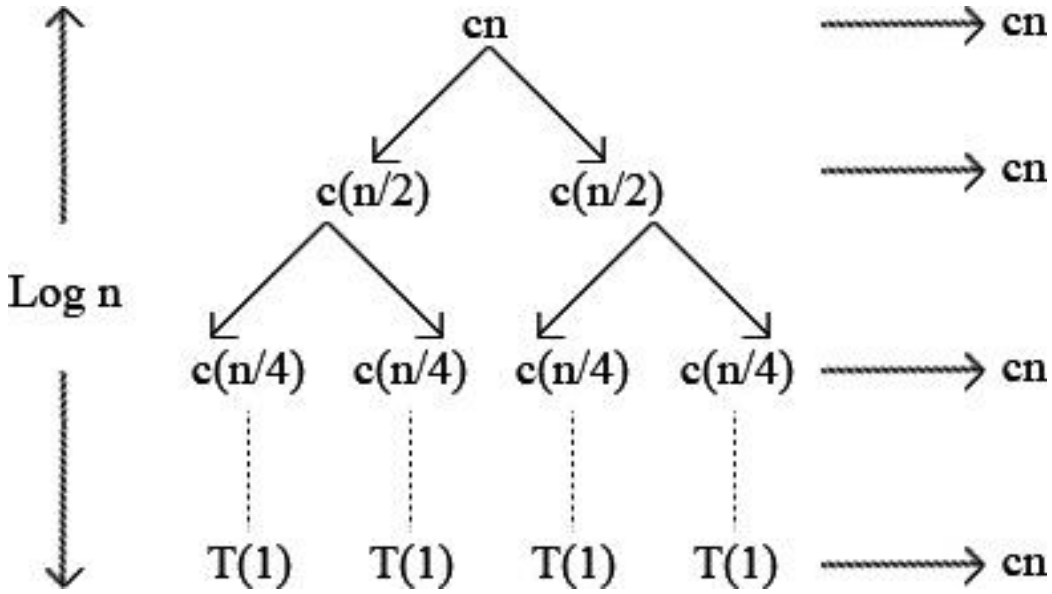
```
int compareX(const void* a, const void* b)
{
    point *p1 = (point *)a, *p2 = (point *)b;
    return (p1->x - p2->x);
}

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()
{
    int n = 0;
    input(n);

    point P[n];
    input(P, n);

    cout << "\nJarak pasangan titik terdekat adalah " << closest(P, n) << end
l;
}
```

2. Rekurensi algoritma Closest Pair of Points menggunakan metode recursion tree:
   - Algoritma ini menerapkan algoritma divide and conquer dimana algoritma ini membagi array titik menjadi dua bagian dan secara rekursif memanggil dua bagian tersebut. Ketika suatu bagian hanya memiliki 2 titik maka jarak kedua titik tersebut dibandingkan dengan jarak titik lainnya sampai ditemukan jarak minimum. Waktu yang dibutuhkan dari langkah ini adalah 2T(n/2).
   - Setelah didapatkan jarak minimum dibuatlah array strip ditengah dengan jarak minimum tadi ke kanan dan ke kiri. Pembuatan array strip ini membutuhkan waktu O(n)
   - Array strip ini lalu diurutkan sehingga membutuhkan waktu O(n).
   - Setelah array strip terurut, dicarilah jarak pasangan titik terdekat dari dalam array strip. Langkah ini membutuhkan waktu O(n).

   Didapat rekurensi:
   $T(n) = 2T(n/2) + O(n) + O(n) + O(n)$
   $T(n) = 2T(n/2) + O(n)$
   $T(n) = 2T(n/2) + cn$

   Pembuktian menggunakan recursion tree:



Studi Kasus 6: Algoritma Karatsuba untuk Perkalian Cepat Identifikasi

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)

Jawaban:

1. Kode Program

```
/*
 * Kefilino Khalifa Filardi
 * 140810180028
 * Tugas 5
 */

#include <iostream>
using namespace std;

void input(string &X, string &Y)
{
    cout << "String pertama\t: ";
    getline(cin, X);
    cout << "String kedua\t: ";
    getline(cin, Y);
}

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 & ca
rry);
    }

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

    return result;
}

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

```
long int multiplyBits(string X, string Y)
{
    for (int i = 0; i < X.length(); i++)
        if (X[i] != '0' && X[i] != '1') {
            cout << "String pertama bukan kode biner!" << endl;
            return 0;
        }
    for (int i = 0; i < Y.length(); i++)
        if (Y[i] != '0' && Y[i] != '1') {
            cout << "String pertama bukan kode biner!" << endl;
            return 0;
        }

    int n = makeEqualLength(X, Y);

    if (n == 0)
        return 0;
    else if (n == 1)
        return multiplyBit(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 = multiplyBits(Xl, Yl);
    long int P2 = multiplyBits(Xr, Yr);
    long int P3 = multiplyBits(addBitStrings(Xl, Xr), addBitStrings(Yl, Yr));

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

int main()
{
    string X = "\0", Y = "\0";
    input(X, Y);

    cout << "\nHasil perkalian kedua bit adalah " << multiplyBits(X, Y);
}
```

2. Rekurensi algoritma Karatsuba menggunakan metode substitusi
   $T(n) = 3T(n/2) + O(n)$
   $T(n) = 3T(n/2) + cn$

   $T(n) \leq 3(c(n/2)\log(n/2)) + cn$
   $T(n) \leq (3/2)(n \log(n/2)) + cn$
   $T(n) = cn \log n - cn \log 2 + cn$
   $T(n) = cn \log n - cn + cn$
   $T(n) = cn \log n$
   $T(n) = O(n \log n)$

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

Jawaban:

1. Kode Program

```
/*
 * Kefilino Khalifa Filardi
 * 140810180028
 * Tugas 5
 */

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

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];
    }
    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);
            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)
            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";
        }
    }
}
```

2. Rekurensi algoritma Tilling Problem menggunakan 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, $T(n) = O(n^2)$