LAPORAN PRAKTIKUM
ANALISIS ALGORITMA

Dibuat oleh:
AHMAD IRFAN FADHOLI
140810180034

PROGRAM STUDI S-1 TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN

2020

Ahmad Irfan Fadholi
140810180034
Tugas 5

Studi Kasus 5

1. Program Closest Pair of Points C++

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

Identifikasi Problem:

Diberikan array n poin dalam bidang kartesius, dan problemnya adalah mencari tahu pasangan poin terdekat dalam bidang tersebut dengan merepresentasikannya ke dalam array. Masalah ini muncul di sejumlah aplikasi. Misalnya, dalam kontrol lalu lintas udara, kita mungkin ingin memantau pesawat yang terlalu berdekatan karena ini mungkin menunjukkan kemungkinan tabrakan. Ingat rumus berikut untuk jarak antara dua titik p dan q.

$$\|pq\| = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

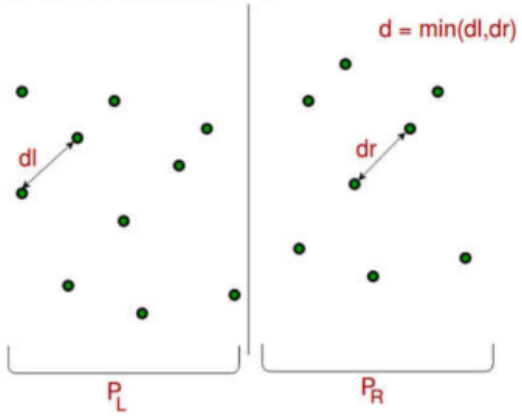
Solusi

Solusi umum dari permasalahan tersebut adalah menggunakan algoritma **Brute force** dengan O (n ^ 2), hitung jarak antara setiap pasangan dan kembalikan yang terkecil. Namun, kita dapat menghitung jarak terkecil dalam waktu **O (nLogn) menggunakan strategi Divide and Conquer.** Ikuti algoritma berikut:

Input: An array of n points P[]
Output: The smallest distance between two points in the given array.

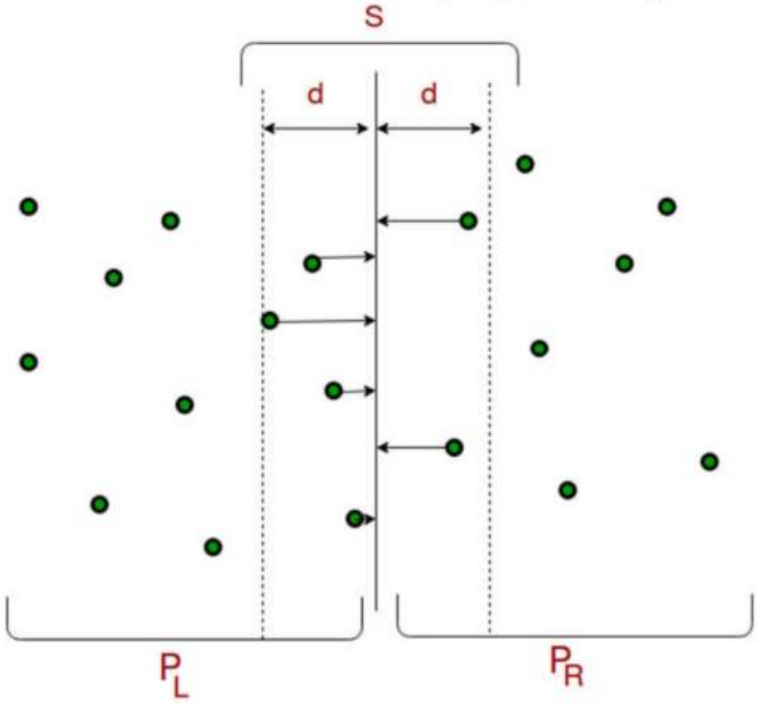
As a pre-processing step, input array is sorted according to x coordinates.

1) Find the middle point in the sorted array, we can take *P[n/2]* as middle point.
2) Divide the given array in two halves. The first subarray contains points from P[0] to P[n/2]. The second subarray contains points from P[n/2+1] to P[n-1].
3) Recursively find the smallest distances in both subarrays. Let the distances be dl and dr. Find the minimum of dl and dr. Let the minimum be d.



Ahmad Irfan Fadholi
140810180034
Tugas 5

4) From above 3 steps, we have an upper bound d of minimum distance. Now we need to consider the pairs such that one point in pair is from left half and other is from right half. Consider the vertical line passing through passing through P[n/2] and find all points whose x coordinate is closer than d to the middle vertical line. Build an array strip[] of all such points.



5) Sort the array strip[] according to y coordinates. This step is O(nLogn). It can be optimized to O(n) by recursively sorting and merging.

6) Find the smallest distance in strip[]. This is tricky. From first look, it seems to be a O(n^2) step, but it is actually O(n). It can be proved geometrically that for every point in strip, we only need to check at most 7 points after it (note that strip is sorted according to Y coordinate). See this for more analysis.

7) Finally return the minimum of d and distance calculated in above step (step 6)

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.  /*
2.      Nama        : Ahmad Irfan Fadholi
3.      Kelas       : B
4.      NPM         : 140810180034
5.      Deskripsi   : Closest Pair of Points
6.  */
7.
8.  #include <iostream>
9.  #include <float.h>
10. #include <stdlib.h>
11. #include <cmath>
12.
13. using namespace std;
14.
15. struct Point{
16.     int x, y;
```

Ahmad Irfan Fadholi
140810180034
Tugas 5

```
};

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[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};
```

```
84.     int n = sizeof(P) / sizeof(P[0]);
85.     cout << "The smallest distance is " << closest(P, n);
86.     return 0;
87. }
```

```
The smallest distance is 1.41421
--------------------------------
Process exited after 0.04823 seconds with return value 0
Press any key to continue . . .
```

2. Kompleksitas Waktu

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

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

   T (n) = T (n * Log n * Log n)

Studi Kasus 6: Algoritma Karatsuba untuk Perkalian Cepat
Identifikasi Problem:
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.

Solusi:
Salah satu solusinya adalah dengan naïve approach yang pernah kita pelajari di sekolah. Satu per satu ambil semua bit nomor kedua dan kalikan dengan semua bit nomor pertama. Akhirnya tambahkan semua perkalian. Algoritma ini membutuhkan waktu O (n ^ 2).

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.  /*
2.      Nama        : Ahmad Irfan Fadholi
3.      Kelas       : B
4.      NPM         : 140810180034
5.      Deskripsi   : Karatsuba Fast Multiplication Algorithm
6.  */
7.  #include<iostream>
8.
9.  using namespace std;
10.
11. int makeEqualLength(string &str1, string &str2) {
12.     int len1 = str1.size();
13.     int len2 = str2.size();
14.     if (len1 < len2) {
15.         for (int i = 0 ; i < len2 - len1 ; i++)
16.             str1 = '0' + str1;
17.         return len2;
18.     }
```

Ahmad Irfan Fadholi
140810180034
Tugas 5

```
    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", "0110")<<endl;
    cout<<multiply("1101", "1010")<<endl;
    cout<<multiply("1101", "1010")<<endl;
    cout<<multiply("1010", "1010")<<endl;
    cout<<multiply("0101", "0010")<<endl;
    cout<<multiply("0011", "1110")<<endl;
    cout<<multiply("1010", "1001")<<endl;
}
```

Ahmad Irfan Fadholi
140810180034
Tugas 5

```
54
130
130
100
10
42
90
```

2. Komplesitas Waktu

- Let's try divide and conquer.
  - Divide each number into two halves.
    - $x = x_H r^{n/2} + x_L$
    - $y = y_H r^{n/2} + y_L$
  - Then:

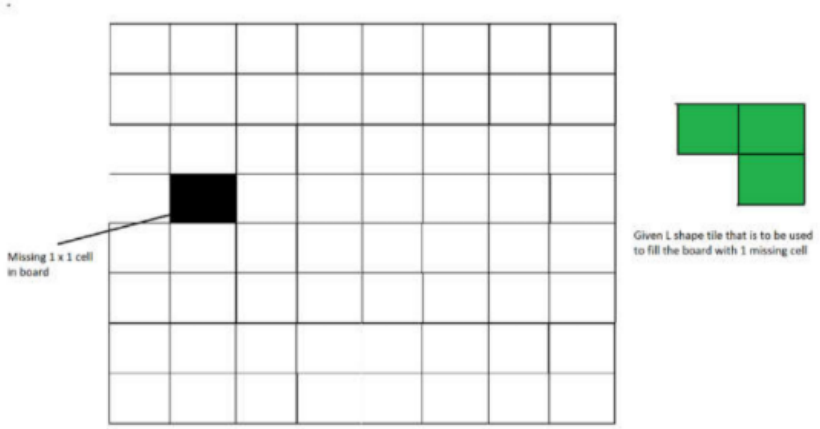
    $xy = (x_H r^{n/2} + x_L)\, y_H r^{n/2} + y_L$

    $= x_H y_H r^n + (x_H y_L + x_L y_H) r^{n/2} + x_L y_L$
  - Runtime?
    - $T(n) = 4\,T(n/2) + O(n)$
    - $T(n) = O(n^2)$

- Instead of 4 subproblems, we only need 3 (with the help of clever insight).
- Three subproblems:
  - $a = x_H y_H$
  - $d = x_L y_L$
  - $e = (x_H + x_L)(y_H + y_L) - a - d$
- Then $xy = a\, r^n + e\, r^{n/2} + d$
- $T(n) = 3\,T(n/2) + O(n)$
- $T(n) = O(n^{\log 3}) = O(n^{1.584\ldots})$

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



Gambar 2. Ilustrasi tilling problem

Ahmad Irfan Fadholi
140810180034
Tugas 5

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.  /*
2.      Nama        : Ahmad Irfan Fadholi
3.      Kelas       : B
4.      NPM         : 140810180034
5.      Deskripsi   : Tilling Problem
6.  */
7.  #include <iostream>
8.  using namespace std;
9.
10. // function to count the total number of ways
11. int countWays(int n, int m){
12.     int count[n + 1];
13.     count[0] = 0;
14.     for (int i = 1; i <= n; i++)
15.     {
16.         if (i > m)
17.             count[i] = count[i - 1] + count[i - m];
18.         else if (i < m)
19.             count[i] = 1;
20.         else
21.             count[i] = 2;
22.     }
23.     return count[n];
24. }
25.
26. int main(){
27.     int n = 8, m = 2;
28.     cout << "Number of ways = "
29.          << countWays(n, m);
30.     return 0;
31. }
```

```
Number of ways = 34
--------------------------------
Process exited after 0.05493 seconds with return value 0
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)

   Bagaimana cara kerjanya?

   Pengerjaan algoritma Divide and Conquer dapat dibuktikan menggunakan Mathematical

Ahmad Irfan Fadholi
140810180034
Tugas 5

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.