TUGAS 5 PRAKTIKUM
ANALISIS ALGORITMA

Disusun Oleh :
Bunga Azizha N
140810180016

Asisten Praktikum:
Faradilla Azranur, Felia Sri Indriyani, Agnes Hata

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

2020

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

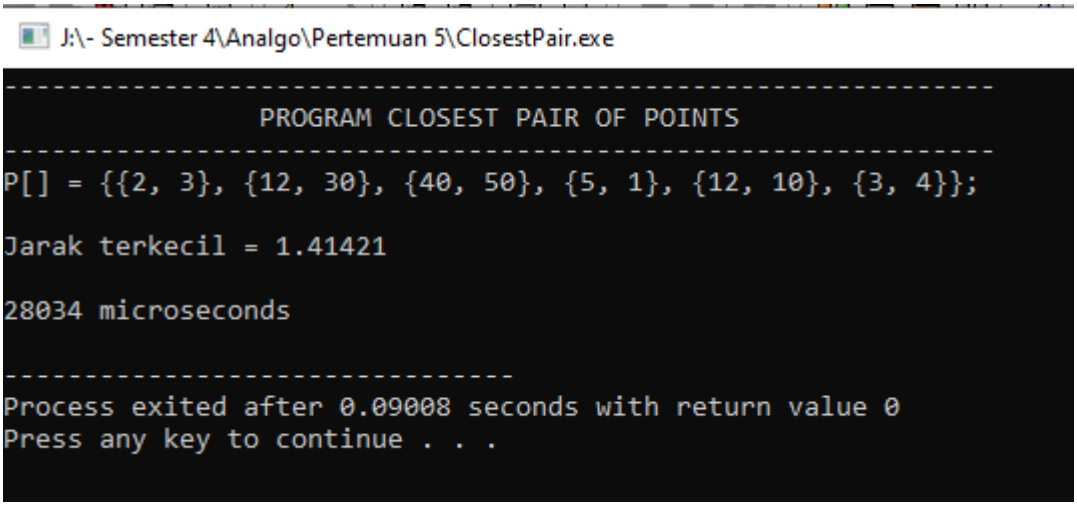
- Source Code

```
/* Bunga Azizha N
140810180016 - Kelas B
Program Program Closest Pair of Points */

#include <bits/stdc++.h>
#include <chrono>

using namespace std;
using namespace std::chrono;

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(){
   high_resolution_clock::time_point t1 = high_resolution_clock::now();
   Point P[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};
   int n = sizeof(P) / sizeof(P[0]);
```

```
    cout<<"-------------------------------------------------------------"<<endl;
    cout<<"\t\tPROGRAM CLOSEST PAIR OF POINTS"<<endl;
    cout<<"-------------------------------------------------------------"<<endl;
    cout<<"P[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};"<<endl<<endl;
    cout<<"Jarak terkecil = "<<closest(P, n);

    high_resolution_clock::time_point t2 = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>( t2 - t1 ).count();
    cout<<endl<<endl<<duration<<" microseconds" <<endl;
}
```

- Screenshoot Hasil Program



- Rekurensi

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

- Source Code

```
/* Bunga Azizha N
140810180016 - Kelas B
Program Karatsuba */

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

using namespace std;
using namespace std::chrono;

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(){
    high_resolution_clock::time_point t1 = high_resolution_clock::now();

    cout<<"-----------------------------------"<<endl;
    cout<<"\tPROGRAM KARATSUBA"<<endl;
    cout<<"-----------------------------------"<<endl;
    cout<<"String 1: 1100, String 2: 1010"<<endl;
    cout<<"String 1: 11, String 2: 11"<<endl;
    cout<<"\nHasil kali: "<<multiply("1100", "1010");
    cout<<"\nHasil kali: "<<multiply("11", "11");

    high_resolution_clock::time_point t2 = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>( t2 - t1 ).count();
    cout<<endl<<endl<<duration<<" microseconds" <<endl;
}
```

- Screenshoot Hasil Program

- Rekurensi

- 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)

- Source Code

```
/* Bunga Azizha N
140810180016 - Kelas B
Program Tilling */

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

int board[1000][1000];
int no = 0;
int quadrant = 0;

void trominoTile(int xBoard,
        int yBoard,
        int x_hole,
        int y_hole,
        int boardSize);

void trominoTile(int xBoard, int yBoard, int x_hole, int y_hole, int boardSize){
```

```
  int halfSize = boardSize / 2,
     xCenter = 0,
     yCenter = 0;

  xCenter = xBoard + halfSize - 1;
  yCenter = yBoard + halfSize - 1;

  if (boardSize == 2){
     if (board[xBoard][yBoard + 1] == 0 && board[xBoard + 1][yBoard] == 0 &&
board[xBoard + 1][yBoard + 1] == 0){
        no++;
        board[xBoard][yBoard + 1] = no;
        board[xBoard + 1][yBoard] = no;
        board[xBoard + 1][yBoard + 1] = no;
     }

     if (board[xBoard][yBoard] == 0 && board[xBoard + 1][yBoard] == 0 &&
board[xBoard + 1][yBoard + 1] == 0){
        no++;
        board[xBoard][yBoard] = no;
        board[xBoard + 1][yBoard] = no;
        board[xBoard + 1][yBoard + 1] = no;
     }

     if (board[xBoard][yBoard + 1] == 0 && board[xBoard][yBoard] == 0 &&
board[xBoard + 1][yBoard + 1] == 0){
        no++;
        board[xBoard + 1][yBoard + 1] = no;
        board[xBoard][yBoard + 1] = no;
        board[xBoard][yBoard] = no;
     }

     if (board[xBoard][yBoard + 1] == 0 && board[xBoard + 1][yBoard] == 0 &&
board[xBoard][yBoard] == 0){
        no++;
        board[xBoard][yBoard] = no;
        board[xBoard][yBoard + 1] = no;
        board[xBoard + 1][yBoard] = no;
     }

     return;
  }

  if (x_hole <= xCenter){
     if (y_hole <= yCenter){
```

```
        if (board[xCenter][yCenter + 1] == 0 && board[xCenter + 1][yCenter] == 0 &&
board[xCenter + 1][yCenter + 1] == 0){
            no++;
            board[xCenter][yCenter + 1] = no;
            board[xCenter + 1][yCenter] = no;
            board[xCenter + 1][yCenter + 1] = no;
            quadrant = 1;
        }
      }
      else{
        if (board[xCenter][yCenter] == 0 && board[xCenter + 1][yCenter] == 0 &&
board[xCenter + 1][yCenter + 1] == 0){
            no++;
            board[xCenter][yCenter] = no;
            board[xCenter + 1][yCenter + 1] = no;
            board[xCenter + 1][yCenter] = no;
            quadrant = 2;
        }
      }
    }
    else{
      if (y_hole <= yCenter){
        if (board[xCenter][yCenter + 1] == 0 && board[xCenter][yCenter] == 0 &&
board[xCenter + 1][yCenter + 1] == 0){
            no++;
            board[xCenter][yCenter] = no;
            board[xCenter][yCenter + 1] = no;
            board[xCenter + 1][yCenter + 1] = no;
            quadrant = 3;
        }
      }
      else{
        if (board[xCenter + 1][yCenter] == 0 && board[xCenter][yCenter] == 0 &&
board[xCenter][yCenter + 1] == 0){
            no++;
            board[xCenter][yCenter] = no;
            board[xCenter][yCenter + 1] = no;
            board[xCenter + 1][yCenter] = no;
            quadrant = 4;
        }
      }
    }

    if (quadrant == 1){
```

```
    trominoTile(xBoard, yBoard, x_hole, y_hole, halfSize);
    trominoTile(xBoard, yCenter + 1, xCenter, yCenter + 1, halfSize);
    trominoTile(xCenter + 1, yBoard, xCenter + 1, yCenter, halfSize);
    trominoTile(xCenter + 1, yCenter + 1, xCenter + 1, yCenter + 1,
            halfSize);
  }

  if (quadrant == 2){
    trominoTile(xBoard, yBoard, xCenter, yCenter, halfSize);
    trominoTile(xBoard, yCenter + 1, x_hole, y_hole, halfSize);
    trominoTile(xCenter + 1, yBoard, xCenter + 1, yCenter, halfSize);
    trominoTile(xCenter + 1, yCenter + 1, xCenter + 1, yCenter + 1,
            halfSize);
  }

  if (quadrant == 3){
    trominoTile(xBoard, yBoard, xCenter, yCenter, halfSize);
    trominoTile(xBoard, yCenter + 1, xCenter, yCenter + 1, halfSize);
    trominoTile(xCenter + 1, yBoard, x_hole, y_hole, halfSize);
    trominoTile(xCenter + 1, yCenter + 1, xCenter + 1, yCenter + 1,
            halfSize);
  }

  if (quadrant == 4){
    trominoTile(xBoard, yBoard, xCenter, yCenter, halfSize);
    trominoTile(xBoard, yCenter + 1, xCenter, yCenter + 1, halfSize);
    trominoTile(xCenter + 1, yBoard, xCenter + 1, yCenter, halfSize);
    trominoTile(xCenter + 1, yCenter + 1, x_hole, y_hole, halfSize);
  }
}

int main(){
  int boardSize, x_hole, y_hole;
  do{
    printf("\n----------------------------------");
    printf("\nEnter size of board (0 to quit): ");
    scanf("%d", &boardSize);
    if (boardSize){
      printf("\nEnter coordinates of missing hole: ");
      scanf("%d%d", &x_hole, &y_hole);
      for (int i = 1; i <= pow(2, boardSize); i++){
        for (int j = 1; j <= pow(2, boardSize); j++){
          board[i][j] = 0;
        }
      }
```

```
        board[x_hole][y_hole] = -1;

        trominoTile(1, 1, x_hole, y_hole, pow(2, boardSize));

        for (int i = 1; i <= pow(2, boardSize); i++){
           for (int j = 1; j <= pow(2, boardSize); j++){
              if (i == x_hole && j == y_hole){
                 board[i][j] == -1;
                 printf("%4s", "X");
              }
              else{
                 printf("%4d", board[i][j]);
              }
           }cout << endl;
        }
     }no = 0;
  }

  while (boardSize);

  return EXIT_SUCCESS;
}
```

- Rekurensi

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)

$T\ (n) = 4T\ (n\ 2$
$) + C$
$a = 4, b = 2, \quad f(n) = 1$

$n\log b\ a = n\log 2\ 4 = n2$
$f(n) = 1 = O(n\log 2\ 4{-}\epsilon)$ untuk $\epsilon = 1$
Case 1 applies

Maka, solusinya adalah
$T(n) = O(n\log b\ a) = O(n\log 2\ 4) = O(n2)$