Nama : Naufal Ariful Amri

NPM : 140810180009

1. Studi Kasus 5

- Buatlah program untuk menyelesaikan problem closest pair of points menggunakan metode divide and conquer dan gunakan bahasa C++
- Tentukan rekurensi algoritma tersebut dan selesaikan rekurensinya menggunakan metode recursion tree untuk membuktikan bahwa algortma tersebut Big-O(n lg n)

Jawab :

```
#include <iostream>
#include <cfloat>
#include <cstdlib>
#include <cmath>
using namespace std;

struct Point {
    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 small_dist(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 stripClosest(Point strip[], int size, float d) {
     float min = d;
     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 Px[], Point Py[], int n)
{
     if (n <= 3){
         return small_dist(Px, n);
     }

     int mid = n / 2;
     Point midPoint = Px[mid];
     Point Pyl[mid + 1];
     Point Pyr[n - mid - 1];

     int li = 0, ri = 0;
     for (int i = 0; i < n; i++) {
          if (Py[i].x <= midPoint.x){
               Pyl[li++] = Py[i];
          }
          else{
               Pyr[ri++] = Py[i];
          }
      }

     float dl = closestUtil(Px, Pyl, mid);
     float dr = closestUtil(Px + mid, Pyr, n-mid);
     float d = min(dl, dr);
     Point strip[n];
     int j = 0;
     for (int i = 0; i < n; i++) {
          if (abs(Py[i].x - midPoint.x) < d){
               strip[j] = Py[i], j++;
          }
     }
     return min(d, stripClosest(strip, j, d));
}

float closest(Point P[], int n) {
    Point Px[n];
    Point Py[n];
    for (int i = 0; i < n; i++){
          Px[i] = P[i];
          Py[i] = P[i];
    }
    qsort(Px, n, sizeof(Point), compareX);
    qsort(Py, n, sizeof(Point), compareY);
    return closestUtil(Px, Py, 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 << "Jarak terdekat adalah " << closest(P, n);
    return 0;
}
```

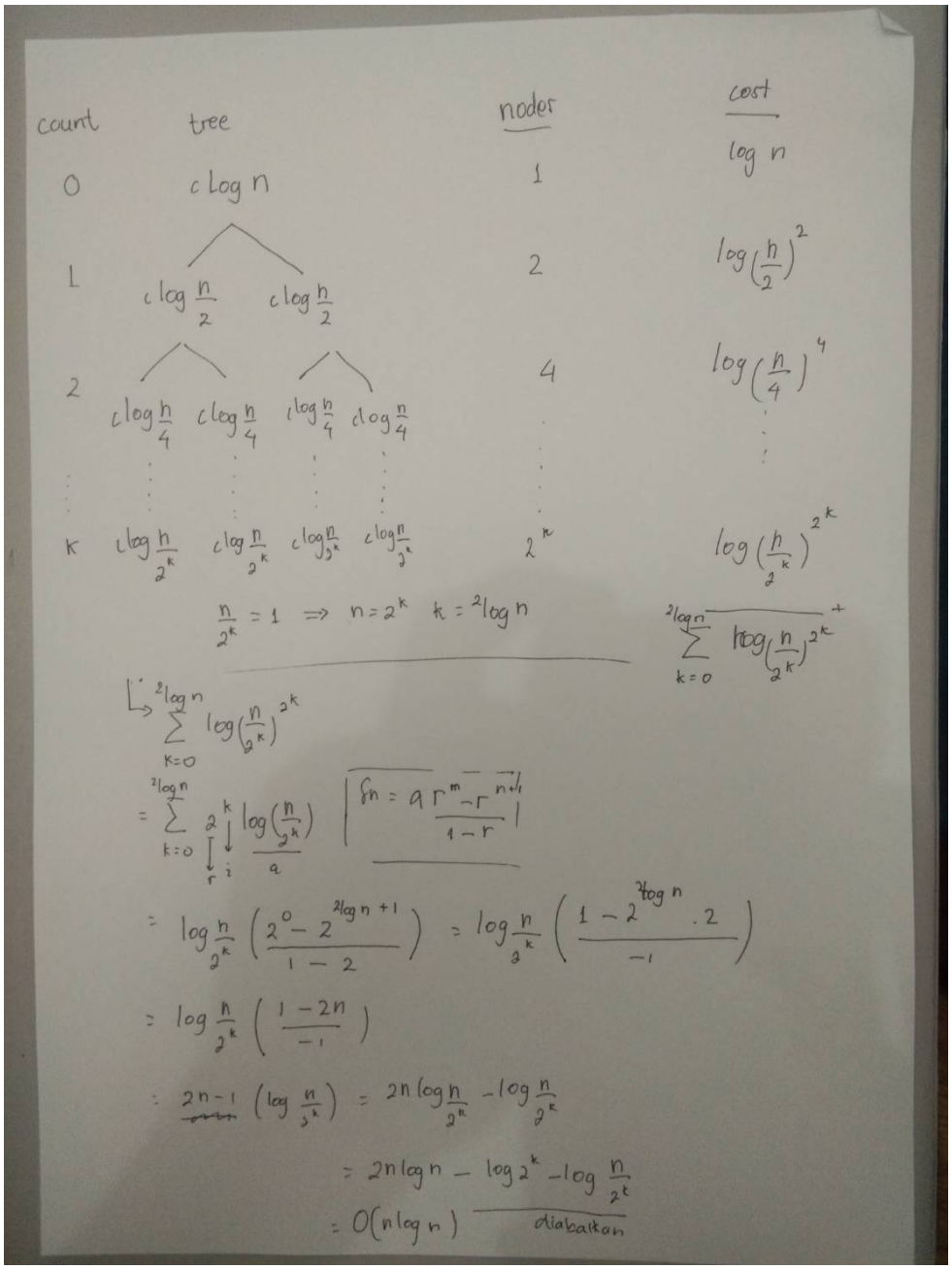
Buktikan bahwa big O dari closest pair of points adalah n log n.

Rekurensi = $2T(\frac{n}{2})$

Waktu membelah = $O(n)$

Waktu mengurutkan = $\log n$ (looping rekurensi)

Waktu mengurutkan = $O(n)$



2. Studi Kasus 6

- Buatlah program untuk menyelesaikan problem fast multipication menggunakan metode divide and conquer dan gunakan bahasa C++
- Rekurensi dari algoritma tersebut adalah $T(n) = 3T\left(\frac{n}{2}\right) + O(n)$. dan selesaikan rekurensinya menggunakan metode subtitusi untuk membuktikan bahwa algortma tersebut Big-O(n lg n)

Jawab :

```
#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);

    // Find the first half and second half of second string
    string Yl = Y.substr(0, fh);
    string Yr = Y.substr(fh, sh);

    // Recursively calculate the three products of inputs of size n/2
    long int P1 = multiply(Xl, Yl);
    long int P2 = multiply(Xr, Yr);
    long int P3 = multiply(addBitStrings(Xl, Xr), addBitStrings(Yl, Yr));

    // Combine the three products to get the final result.
    return P1*(1<<(2*sh)) + (P3 - P1 - P2)*(1<<sh) + P2;
}

// Driver program to test above functions
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"));
}
```

Tebakan awal = n log n (karena untuk membuktikan)

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

$T(n) \leq cn$

$T(n) \leq c(n \log n)$

$n = 2$

$T\left(\frac{n}{2}\right) \leq c((\frac{n}{2})\log(\frac{n}{2}))$

$T(n) \leq c(n \log n) + cn$

$T(n) \leq 3(c\left(\frac{n}{2}\right)\log\left(\frac{n}{2}\right) + cn)$

$T(n) \leq \frac{3}{2}cn \log\left(\frac{n}{2}\right) + cn$

$T(n) \leq \frac{3}{2}cn \log n - \frac{3}{2}cn \log 2 + cn$

$T(n) \leq \frac{3}{2}cn \log n$

$O(n \log n)$

3. Studi Kasus 7

- Buatlah program untuk menyelesaikan problem tilling menggunakan algoritma divide and conquer yang diberikan dan gunakan bahasa C++
- Relasi rekurensi untuk aloritma rekrusif diatas adalah $T(n) = 4T\left(\frac{n}{2}\right) + C$. Dengan C adalah konstanta. Selesaikan dengan metode master

Jawab :

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

int ** board;
int cnt=0;

int ** createBoard(int n,int hr,int hc){
     int ** array = (int**)malloc(n*sizeof(int *));
     int i,j;
     for(i = 0;i < n; i++) {
          array[i] = (int*)malloc(n*sizeof(int));
          for(j = 0; j < n; j++) {
               if((i==hr) && (j==hc))
                    array[i][j]=-1;
               else
                    array[i][j] = 0;
          }
     }
     return array;
}


void printBoard(int n){
     int i,j;
     for(i=0;i<n;i++){
          for(j=0;j<n;j++) {
               printf(" %d ",board[i][j]);
          }
          printf("\n");
     }
}
void putTromino(int x1,int y1,int x2,int y2,int x3, int y3) {
     cnt++;
     board[x1][y1] = cnt;
     board[x2][y2] = cnt;
     board[x3][y3] = cnt;
}

/*
* TrominoTile Recursive function
*/
int trominoTileRec(int n,int x,int y) {
     int i,j,hr,hc;
     if(n == 2){
          cnt++;
          for(i=0;i<n;i++) {
               for(j=0;j<n;j++){
                    if(board[x+i][y+j]==0)
                         board[x+i][y+j]=cnt;
               }
          }
          return 0;
     }
```

```
     //Search the hole's location
     for(i=x;i<n;i++) {
          for(j=y;j<n;j++){
               if(board[i][j]!=0) {
                    hr=i; hc=j;
               }
          }
     }

     //If missing Tile is in 1st quadrant
     if(hr< x + n/2 &&hc < y+ n/2) {
          putTromino(x+n/2,y+ (n/2) - 1,x+n/2,y+n/2,x+n/2-1,y+n/2);
     }
     //If missing Tile is in 2st quadrant
     else if(hr>=x+ n/2 && hc < y + n/2) {
          putTromino(x+n/2,y+ (n/2) - 1,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
     }
     //If missing Tile is in 3st quadrant
     else if(hr < x + n/2 && hc >= y + n/2) {
          putTromino(x+(n/2) - 1,y+ (n/2),x+(n/2),y+n/2,x+(n/2)-1,y+(n/2) -1);
     }
     //If missing Tile is in 4st quadrant
     else if(hr >= x + n/2 && hc >= y + n/2) {
          putTromino(x+(n/2) -1, y+ (n/2),x+(n/2),y+(n/2) -1,x+(n/2)-1,y+(n/2)-1);
     }
     trominoTileRec(n/2, x, y+n/2);
     trominoTileRec(n/2, x, y);
     trominoTileRec(n/2, x+n/2, y);
     trominoTileRec(n/2, x+n/2, y+n/2);

     return 0;

}


void trominoTile(int size) {
     trominoTileRec(size,0,0);
}

void freeMemory(int n){
     int i;
     for(i=0;i<n;i++)
          free(board[i]);
     free(board);
}

int main(int argc,char ** argv) {
     int i,k,hr,hc,size=1;

     k=atoi(argv[1]);
     hr=atoi(argv[2]);
     hc=atoi(argv[3]);
     for(i=0;i<k;i++){
          size = 2*size;
     }
     board = createBoard(size,hr,hc);
     trominoTile(size);
     printf("\n\n");
     printBoard(size);

     freeMemory(size);
     return 0;
}
```

1) Kasus dasar: n = 2, A 2 x 2 persegi dengan satu sel yang hilang tidak ada apa-apanya tapi ubin dan bisa diisi dengan satu ubin.

2) Tempatkan ubin berbentuk L di tengah sehingga tidak menutupi subsquare n / 2 * n / 2 yang memiliki kuadrat yang hilang. Sekarang keempatnya subskuen ukuran n / 2 x n / 2 memiliki sel yang hilang (sel yang tidak perlu diisi). Lihat gambar 2 di bawah ini.

3) Memecahkan masalah secara rekursif untuk mengikuti empat. Biarkan p1, p2, p3 dan p4 menjadi posisi dari 4 sel yang hilang dalam 4 kotak.

a) Ubin (n / 2, p1)

b) Ubin (n / 2, p2)

c) Ubin (n / 2, p3)

d) Ubin (n / 2, p3)

$T(n) = 4T\left(\frac{n}{2}\right) + c$

$a = 4, b = 2, f(n) = c$

$n^{\log_b a} = n^{\log_2 4} = n^2$

$f(n) = c = O\left(n^{\log_2(4-\varepsilon)}\right)$ untuk $\varepsilon = 1$

Case 1 applies.