Nama : Angga Kresnabayu

NPM : 140810160001

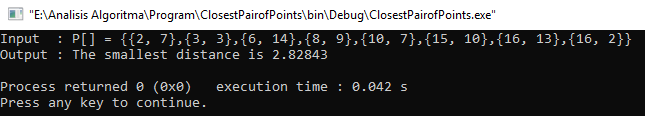
Kelas : A

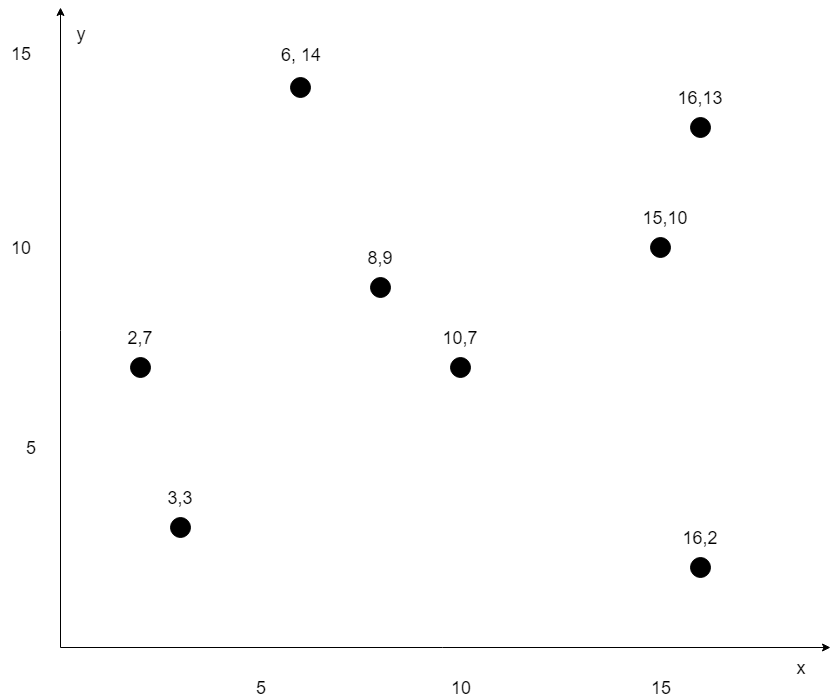
1. **Closest Pair of Points**
2. Buatlah program untuk menyelesaikan problem closest pair of points menggunakan algoritma divide & conquer yang diberikan. Gunakan bahasa C++

**Jawab: ClosestPairofPoints.cpp**

1. Telusuri algoritma Closest Pair of Points dan hitung kompleksitas

**Jawab:**





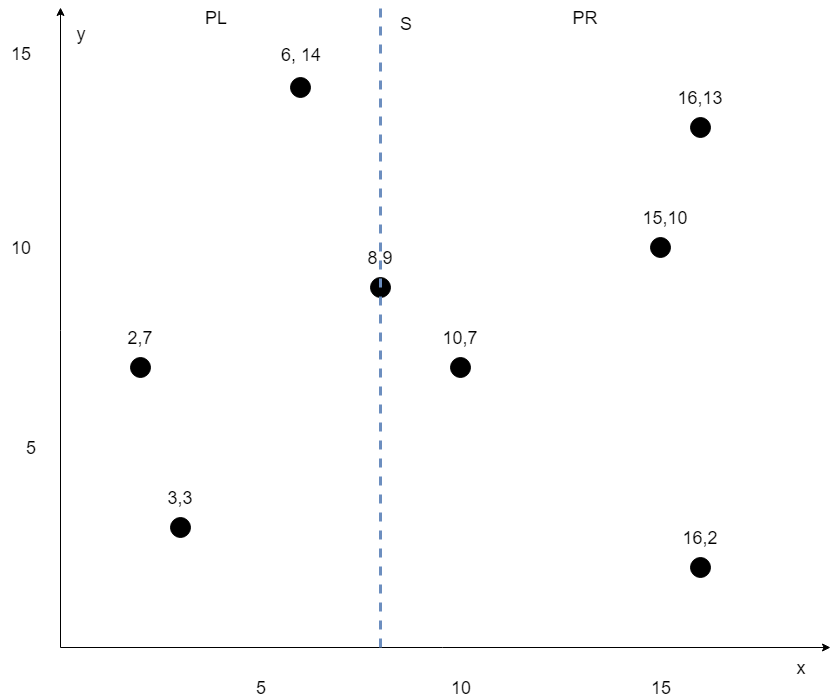
* Find the middle point in the sorted array, we can take P[n/2] as middle point.

P[] = {{2, 7}, {3, 3}, {6, 14}, **{8, 9}**, {10, 7}, {15, 10}, {16, 2}, {16, 13}} sorted

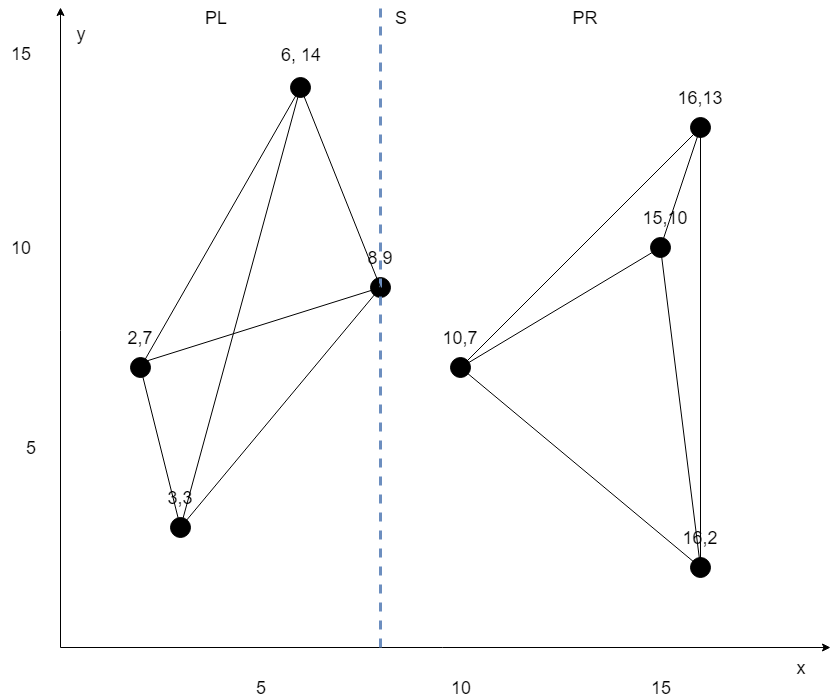
* Divide the given array in two halves. The first subarray contains points from P[0] to P[n/2]. The second subarray contains points from P[n/2+1] to P[n-1].

PL[] = {{2, 7}, {3, 3}, {6, 14}, **{8, 9}**}

PR[] = {{10, 7}, {15, 10}, {16, 2}, {16, 13}}



* 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.





**||{2, 7}, {3, 3}||**

||{2, 7}, {6, 14}||

||{2, 7}, {8, 9}||

||{3, 3}, {6, 14}||

||{3, 3}, {8, 9}||

||{8, 9}, {6, 14}||

**||10, 7}, {15, 10}||**

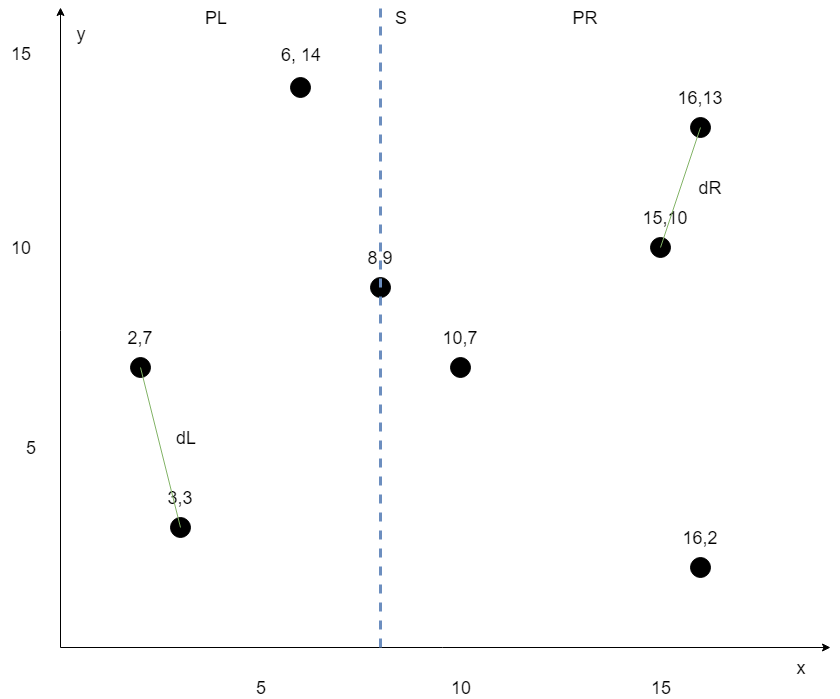
||10, 7}, {16, 2}||

||10, 7}, {16, 13}||

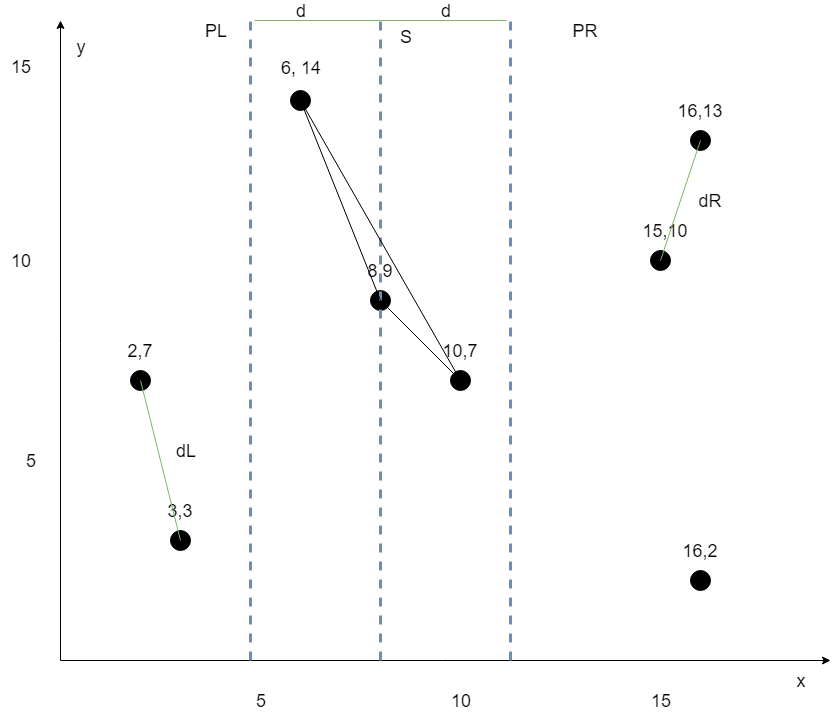
||15, 10}, {16, 2}||

||16, 13}, {16, 2}||

**dl dr d = min(dl,dr) =**

****

* 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.

****

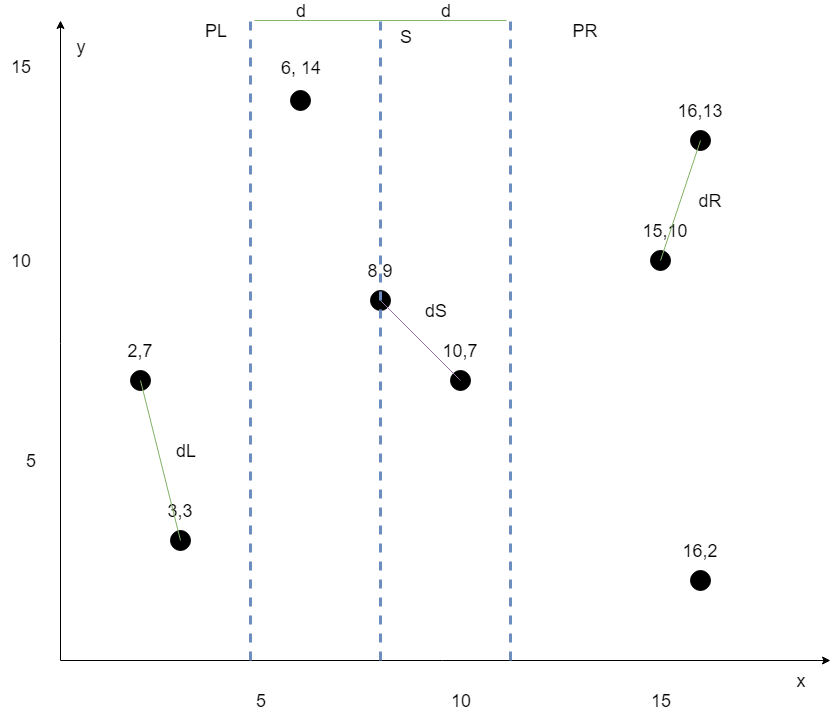
* Sort the array strip[] according to y coordinates. This step is O(nLogn). It can be optimized to O(n) by recursively sorting and merging
* Find the smallest distance in strip[]. This is tricky. From first look, it seems to be a O(n^2) step, but it is actually O(n). It can be proved geometrically that 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

||6, 14}, {8, 9}||

||6, 14}, {10, 7}||

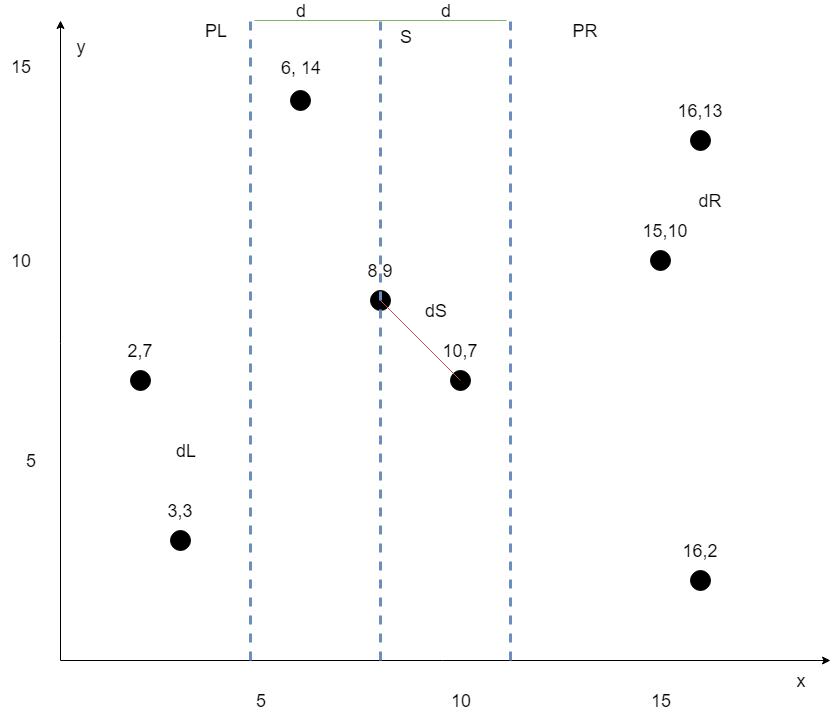
**||8, 9}, {10, 7}||**

dS



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

**d = min(dS,dl,dr) = 2,82843**

****

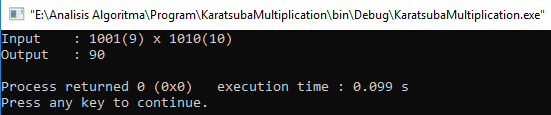
**Kompleksitas:**  
Biarkan kompleksitas waktu dari algoritma di atas menjadi T (n). Mari kita asumsikan bahwa kita menggunakan algoritma pengurutan O (nLogn). Algoritma di atas membagi semua titik dalam dua set dan secara rekursif memanggil dua set. Setelah membagi, ia menemukan strip dalam waktu O (n), mengurutkan strip dalam waktu O (nLogn) dan akhirnya menemukan titik terdekat dalam strip dalam waktu O (n). Jadi T (n) dapat dinyatakan sebagai berikut T (n) = 2T (n / 2) + O (n) + O (nLogn) + O (n) T (n) = 2T (n / 2) + O (nLogn) T (n) = T (n x Logn x Logn) Catatan 1) Kompleksitas waktu dapat ditingkatkan menjadi O (nLogn) dengan mengoptimalkan langkah 5 dari algoritma di atas. Kami akan segera membahas solusi yang dioptimalkan dalam posting terpisah. 2) Kode menemukan jarak terkecil. Dapat dengan mudah dimodifikasi untuk menemukan titik dengan jarak terkecil. 3) Kode ini menggunakan pengurutan cepat yang bisa O (n ^ 2) dalam kasus terburuk. Untuk memiliki batas atas sebagai O (n (Logn) ^ 2), algoritma pengurutan O (nLogn) seperti pengurutan gabungan atau pengurutan tumpukan dapat digunakan

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

**Jawab: KaratsubaMultiplication.cpp**

1. Telusuri algoritma Karatsuba Multiplication dan hitung kompleksitas

**Jawab:**



1001(9)x1010(10)

= (10.22+ 01) (10.22+ 10)

= (10.10)24 + (10.10)22 + (01.10)22 + (01.10)

= (10.10)24 + ((10.10) + (01.10))22 + (01.10)

= ((1.2­­1 + 0)(1.2­­1 + 0))24 +

(((1.2­­1 + 0)(1.2­­1 + 0))+ (((0.2­­1 + 1)(1.2­­1 + 0)))22 + ((0.2­­1 + 1)(1.2­­1 + 0))

= ((1.1)22 + (0.1)2­­1+ (1.0)2­­1 + (0.0))24 +

(((1.1)22 + (0.1)2­­1+ (0.1)2­­1 + (0.0))) + ((0.1)22 + (0.0)2­­1+ (1.1)2­­1 + (1.0)))22 +

((0.1)22 + (0.0)2­­1+ (1.1)2­­1 + (1.0))

= ((1.1)22 + ((0.1) + (1.0))2­­1 + (0.0))24 +

(((1.1)22 + ((0.1)+ (0.1))2­­1 + (0.0))) + ((0.1)22 + ((0.0)+ (1.1))2­­1 + (1.0)))22 +

((0.1)22 + ((0.0)+ (1.1))2­­1 + (1.0))

= ((1)22 + ((0) + (0))2­­1 + (0))24 +

(((1)22 + ((0)+ (0))2­­1 + (0))) + ((0)22 + ((0)+ (1))2­­1 + (0)))22 +

((0)22 + ((0)+ (1))2­­1 + (0))

= (100+0+0)24 + ((100 + 0 + 0)+(0 + 1 + 0))22 + (0+10 +0)

= 100.24 + ((100 + 10)22) + 10

= 100 0000 + 1 1000 + 10

= **101 1010** (9)

Kompleksitas:  
Kompleksitas waktu dari solusi di atas adalah O (n1.59).

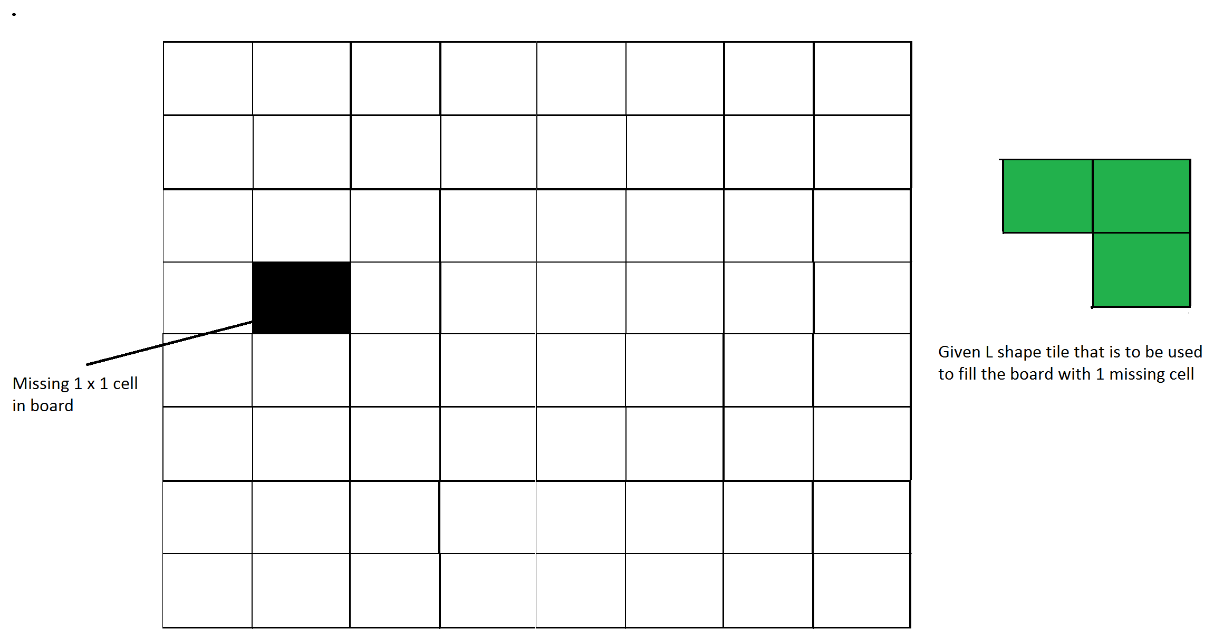
1. **Problem Tilling**
2. Buatlah program untuk menyelesaikan problem tilling menggunakan algoritma divide & conquer yang diberikan. Gunakan bahasa C++

**Jawab: ProblemTilling.cpp**

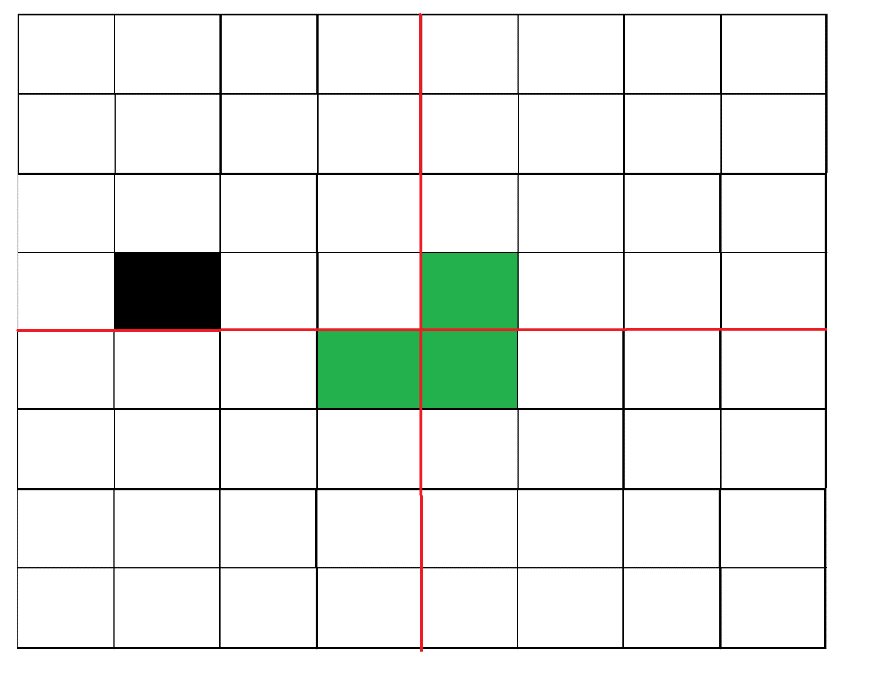
1. Telusuri algoritma Problem Tillingdan hitung kompleksitas

**Jawab:**

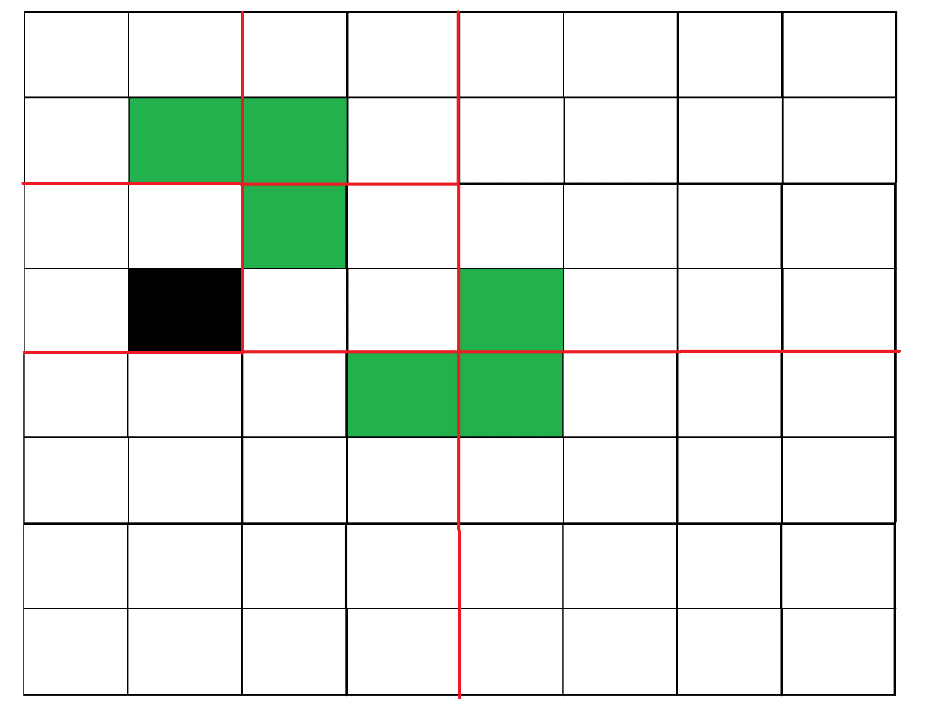
Missing 1x1 cell in board. Tile yang sempurna 2x2 maka devide hingga 2x2



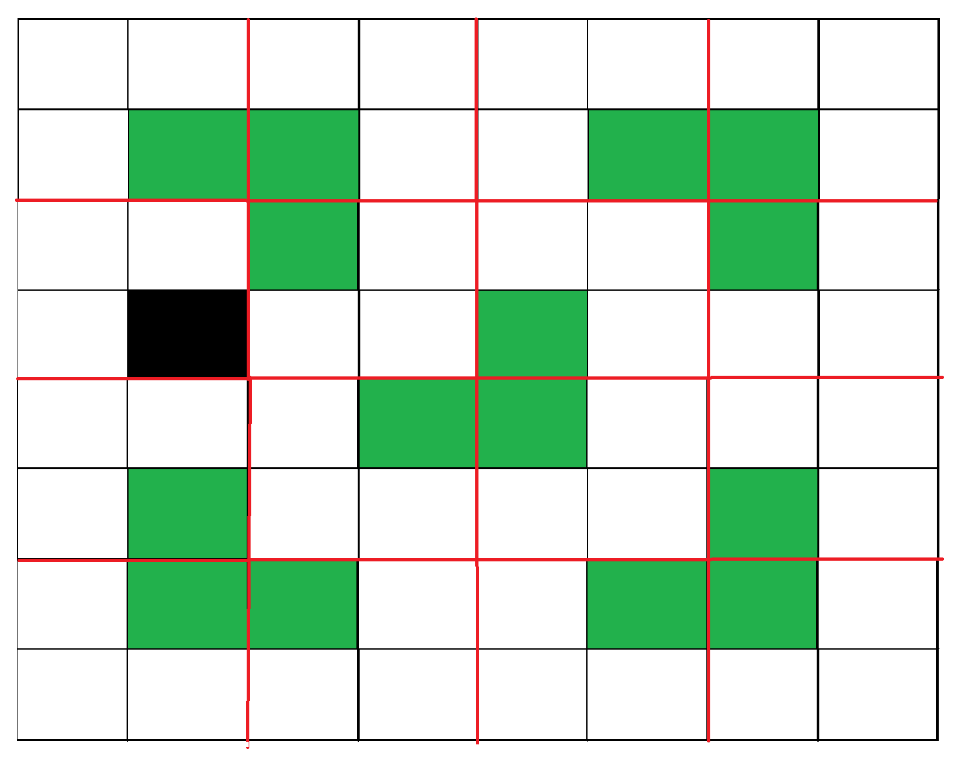
Letakan Tile dititik pusat, setiap 2x2 kotak hanya diisi satu bagian Tile (kotak hijau)



Kemudian bagi lagi, letakan Tile lagi di titik pusat, setiap 2x2 kotak hanya diisi satu bagian Tile (kotak hijau)



Secara Rekursif hingga setiap 2x2 hanya terisi 1 sel hijau



Kompleksitas:

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)