CEN360
ALGORITHMS DESIGN AND ANALYSIS
Fall – 2018
Lab 9: Divide and Conquer and Transform and Conquer Algorithms-4

1. Strassen's Matrix Multiplication: [20pts]

You have already learned brute force matrix multiplication, it is given in the following ($O(n^3)$):

```
void multiply(int A[][N], int B[][N], int C[][N])
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            C[i][j] = 0;
            for (int k = 0; k < N; k++)
            {
                C[i][j] += A[i][k]*B[k][j];
            }
        }
    }
}
```

The idea of **Strassen's method** is to reduce the number of recursive calls to 7. Strassen's method is similar to simple divide and conquer method in the sense that this method also divide matrices to sub-matrices of size N/2 x N/2, but in Strassen's method, the four sub-matrices of result are calculated using following formulae. Implement Strassen's method for 2x2 Matrices.

p1 = a(f - h) p2 = (a + b)h
p3 = (c + d)e p4 = d(g - e)
p5 = (a + d)(e + h) p6 = (b - d)(g + h)
p7 = (a - c)(e + f)

The A x B can be calculated using above seven multiplications.
Following are values of four sub-matrices of result C

$$\underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} e & f \\ g & h \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} p5 + p4 - p2 + p6 & p1 + p2 \\ p3 + p4 & p1 + p5 - p3 - p7 \end{bmatrix}}_{C}$$

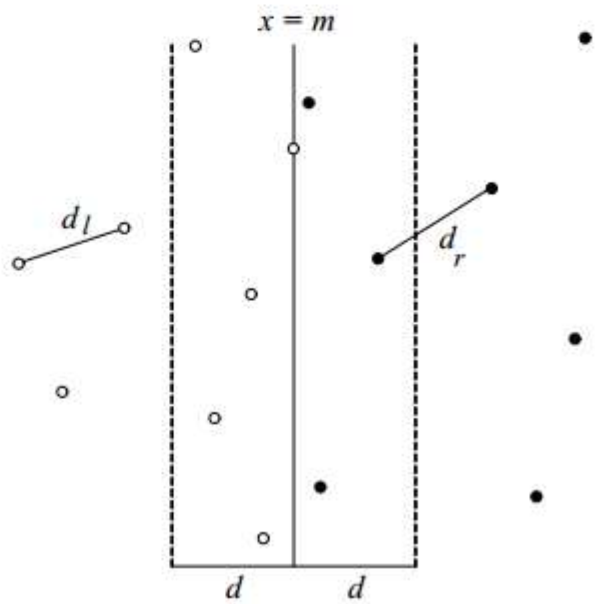
A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2
p1, p2, p3, p4, p5, p6 and p7 are submatrices of size N/2 x N/2

2. <u>Closest-Pair Problem by Divide-and-Conquer:</u> [50pts]

You have already solved this problem using brute force method. Remember that the efficiency of brute force solution is $O(n^2)$.

<u>Divide and Conquer Approach:</u>

Step 1: Divide the points given into two subsets P_l and P_r by a vertical line x = m so that half the points lie to the left or on the line and half the points lie to the right or on the line.



Step 2 Find recursively the closest pairs for the left and right subsets.
Step 3 Set $d = \min\{d_l, d_r\}$ We can limit our attention to the points in the symmetric vertical strip S of width 2d as possible closest pair. (The points are stored and processed in increasing order of their y coordinates.)
Step 4 Scan the points in the vertical strip S from the lowest up. For every point p(x,y) in the strip, inspect points in the strip that may be closer to p than d.

Running time of the algorithm is described by
$T(n) = 2T(n/2) + M(n)$, where $M(n) \in O(n)$
By the Master Theorem (with $a = 2$, $b = 2$, $d = 1$)
$T(n) \in O(n \log n)$

3. Element uniqueness with presorting (Transform and Conquer): [30pts]

Remember Brute force algorithm:
Compare all pairs of elements
Efficiency: $O(n^2)$

Presorting-based algorithm
Stage 1: sort by efficient sorting algorithm (e.g. mergesort)
Stage 2: scan array to check pairs of adjacent elements
Efficiency: $\Theta(n\log n) + O(n) = \Theta(n\log n)$

Create a random array with the size of N, check element uniqueness using presorting based algorithm.