分割統治法 ClosestPairを例に

計算機アルゴリズム特論：2017年度

只木進一

分割統治法(Divide and Conquer)と再帰

- 小さい問題に分割
- 分割された、非常に小さい問題を解決する。
- その後、小さな問題の統合を繰り返し、全体の問題を解く
- 再帰的／非再帰的手法

最近接点：問題設定

- 二次元空間内の点の集合

$$P = \{p_k | 0 \le k < n\}$$

- 最も距離の近い点の組を求める
- 単純に点の組の距離を調べたのでは、$n \times (\mathrm{n} - 1)/2$個の組について、調べる必要がある
- 分割統治に基づくアルゴリズムを検討

基本的方針

- x座標、y座標でソート
  - 「近い」領域だけ調べる
  - 領域に分割
- 小さな領域→力ずくで調べる
- 領域の統合
  - 既知の組と境界の組を比較

準備

- java.awt.geom.Point2D.Double
  - 2次元の点の座標(double)を保持
  - 距離を測るメソッドdistance()
- PointPair
  - 二つの点とその間の距離を保持

Brute Force
力ずく

```
public static PointPair findClosestBF(List<Point2D.Double> list) {
        int n = list.size();
        PointPair pp = new PointPair(list.get(0), list.get(1));
        double min = pp.distance;
        for (int i = 0; i < n - 1; i++) {
            Point2D.Double p = list.get(i);
            for (int j = i + 1; j < n; j++) {
                Point2D.Double q = list.get(j);
                if (p.distance(q) < min) {
                    min = p.distance(q);
                    pp = new PointPair(p, q);    }
            }}
    return pp;
}
```

全ての点の組を調べる

最近接点：準備

- 全ての頂点をx座標の順で並べたリストX_0
- 全ての頂点をy座標の順で並べたリストY_0
- それぞれ、$n \log_2 n$回の比較で生成できる

点のリストP

↓

x座標でソートしたリストX_0

y座標でソートしたリストY_0

ソートは一度しか行わない

最近接点：左右分割

- 組(X_i, Y_i)に対して、X_iを左右$\left(X_{i+1}^L, X_{i+1}^R\right)$に分割
- 対応してY_iを割り当て、$\left(Y_{i+1}^L, Y_{i+1}^R\right)$とする。

最近接点：上下分割

- 組(X_i, Y_i)に対して、Y_iを上下$\left(Y_{i+1}^D, Y_{i+1}^U\right)$に分割
- 対応してX_iを割り当て、$\left(X_{i+1}^D, X_{i+1}^U\right)$とする。

- 注：いずれのリストもソートされている

x座標でソートしたリストX_0

左右に分割

X_1^L

X_1^R

対応付け

Y_1^L

Y_1^R

y座標でソートしたリストY_0

$$P = \{(2,5),(1,6),(9,5),(3,1),(2,2),(8,7),(4,8),(7,1)\}$$

ソート

$$X_0 = \{(1,6),(2,5),(2,2),(3,1),(4,8),(7,1),(8,7),(9,5)\}$$

$$Y_0 = \{(3,1),(7,1),(2,2),(2,5),(9,5),(1,6),(8,7),(4,8)\}$$

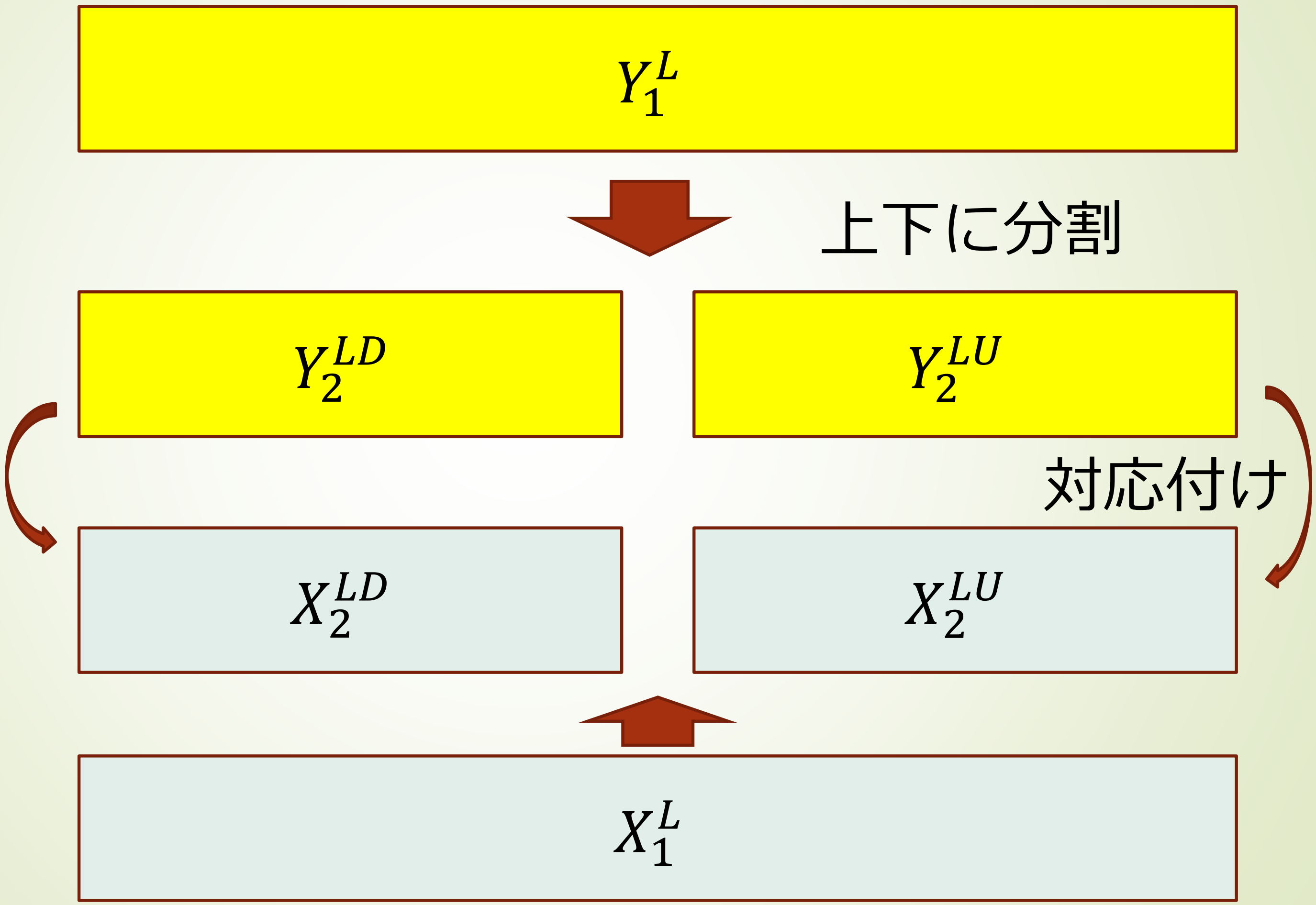
x方向への分割

$$X_1^L = \{(1,6),(2,5),(2,2),(3,1)\}$$

$$Y_1^L = \{(3,1),(2,2),(2,5),(1,6)\}$$

$$X_1^R = \{(4,8),(7,1),(8,7),(9,5)\}$$

$$Y_1^R = \{(7,1),(9,5),(8,7),(4,8)\}$$


Y_1^L
上下に分割
Y_2^{LD}
Y_2^{LU}
対応付け
X_2^{LD}
X_2^{LU}
X_1^L

$$X_1^{\ L} = \{(1,6),(2,5),(2,2),(3,1)\}$$

$$Y_1^{\ L} = \{(3,1),(2,2),(2,5),(1,6)\}$$

$$X_1^{\ R} = \{(4,8),(7,1),(8,7),(9,5)\}$$

$$Y_1^{\ R} = \{(7,1),(9,5),(8,7),(4,8)\}$$

y方向への分割

$$X_2^{\ LD} = \{(2,2),(3,1)\}$$

$$Y_2^{\ LD} = \{(3,1),(2,2)\}$$

$$X_2^{\ LU} = \{(1,6),(2,5)\}$$

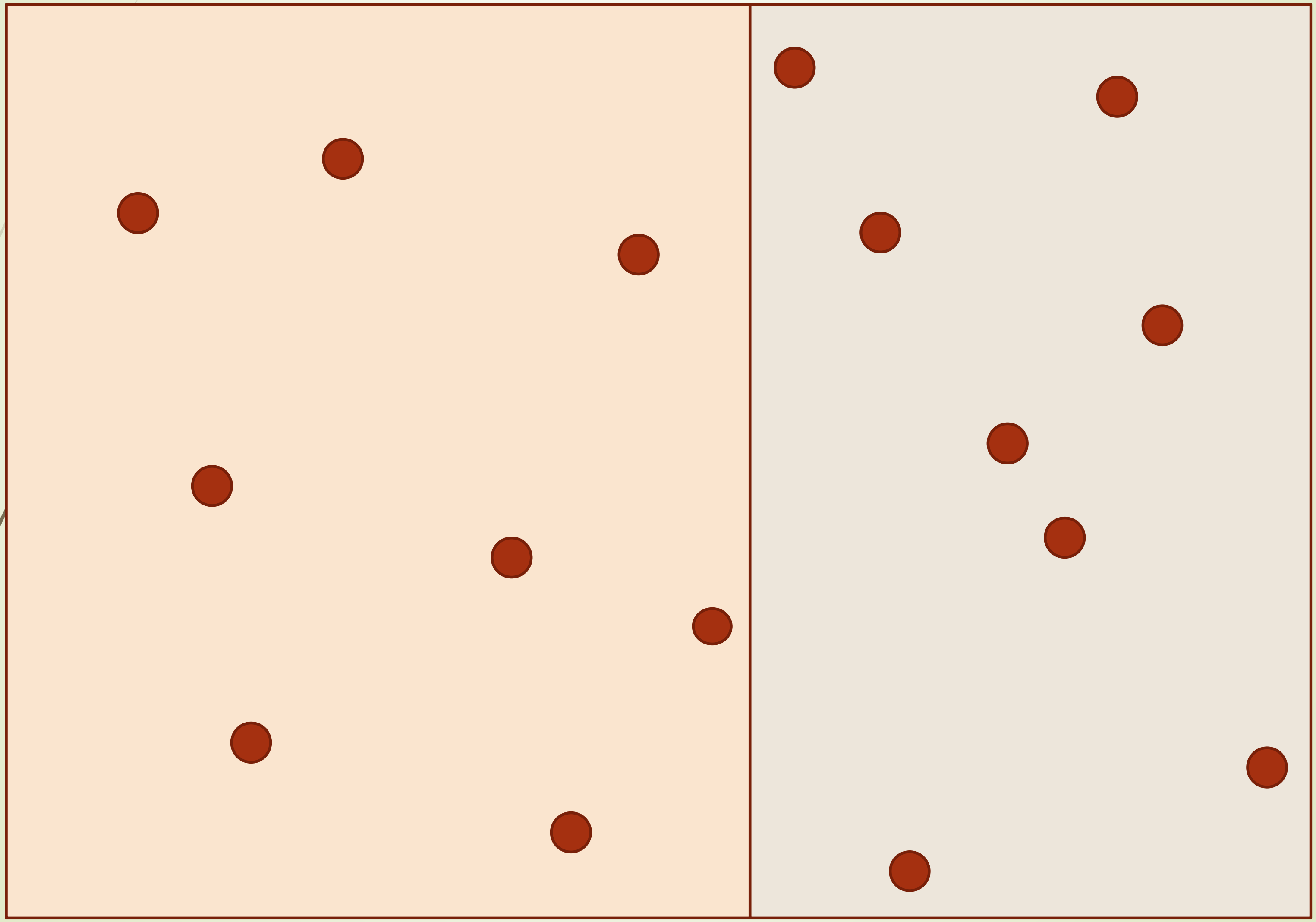
$$Y_2^{\ LU} = \{(2,5),(1,6)\}$$

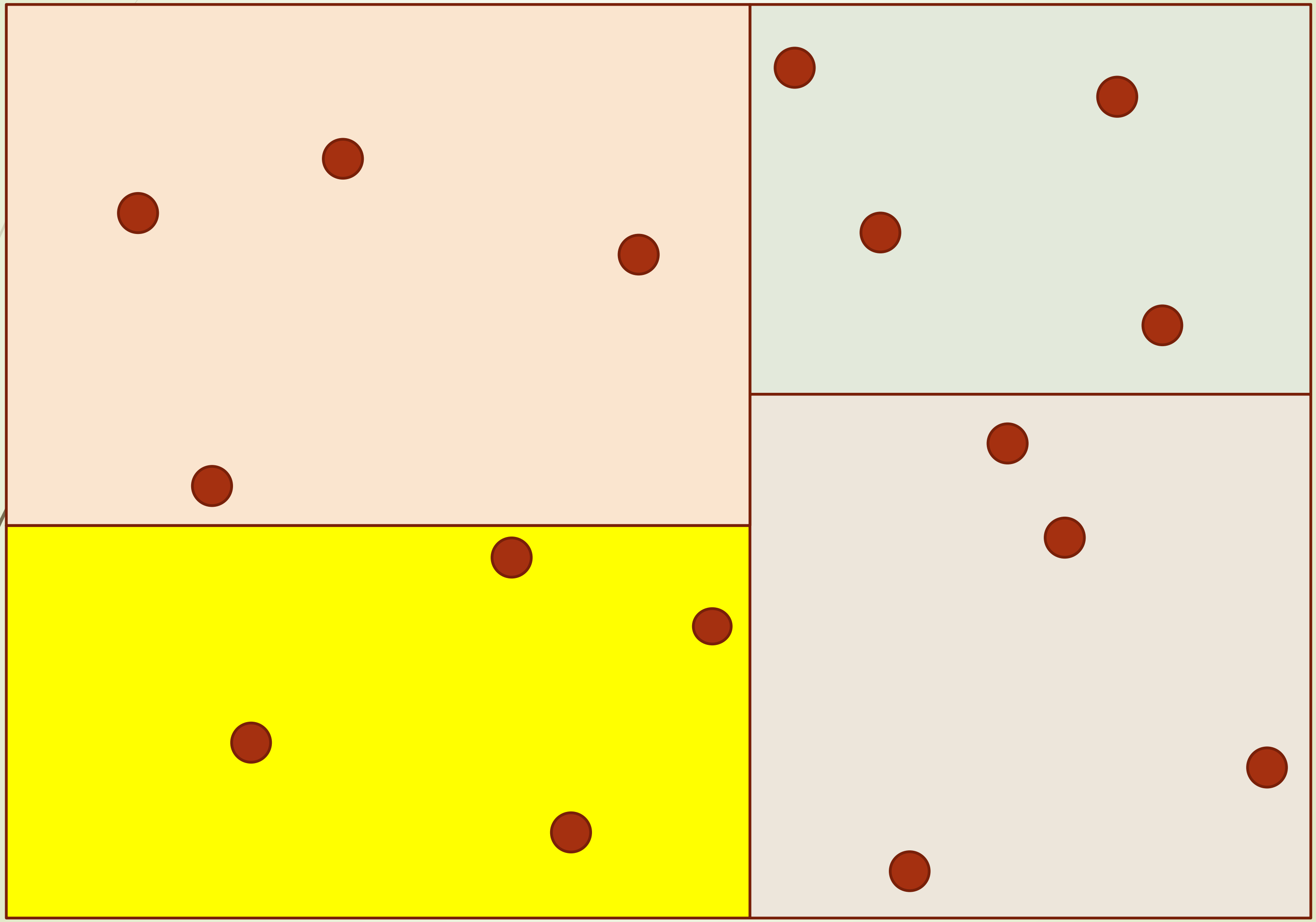
$$X_2^{\ RD} = \{(7,1),(9,5)\}$$

$$Y_2^{\ RD} = \{(7,1),(9,5)\}$$

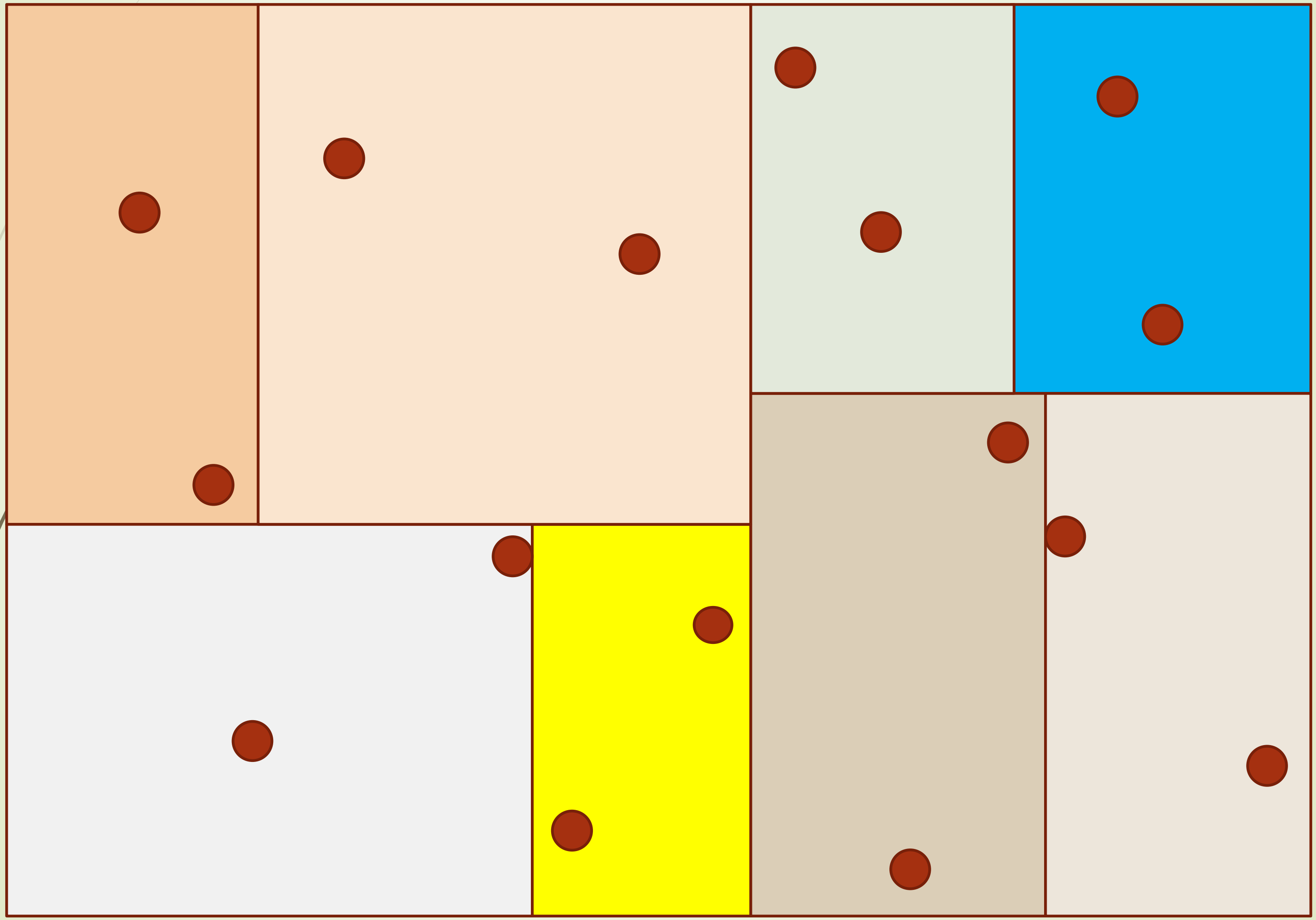
$$X_2^{\ RU} = \{(4,8),(8,7)\}$$

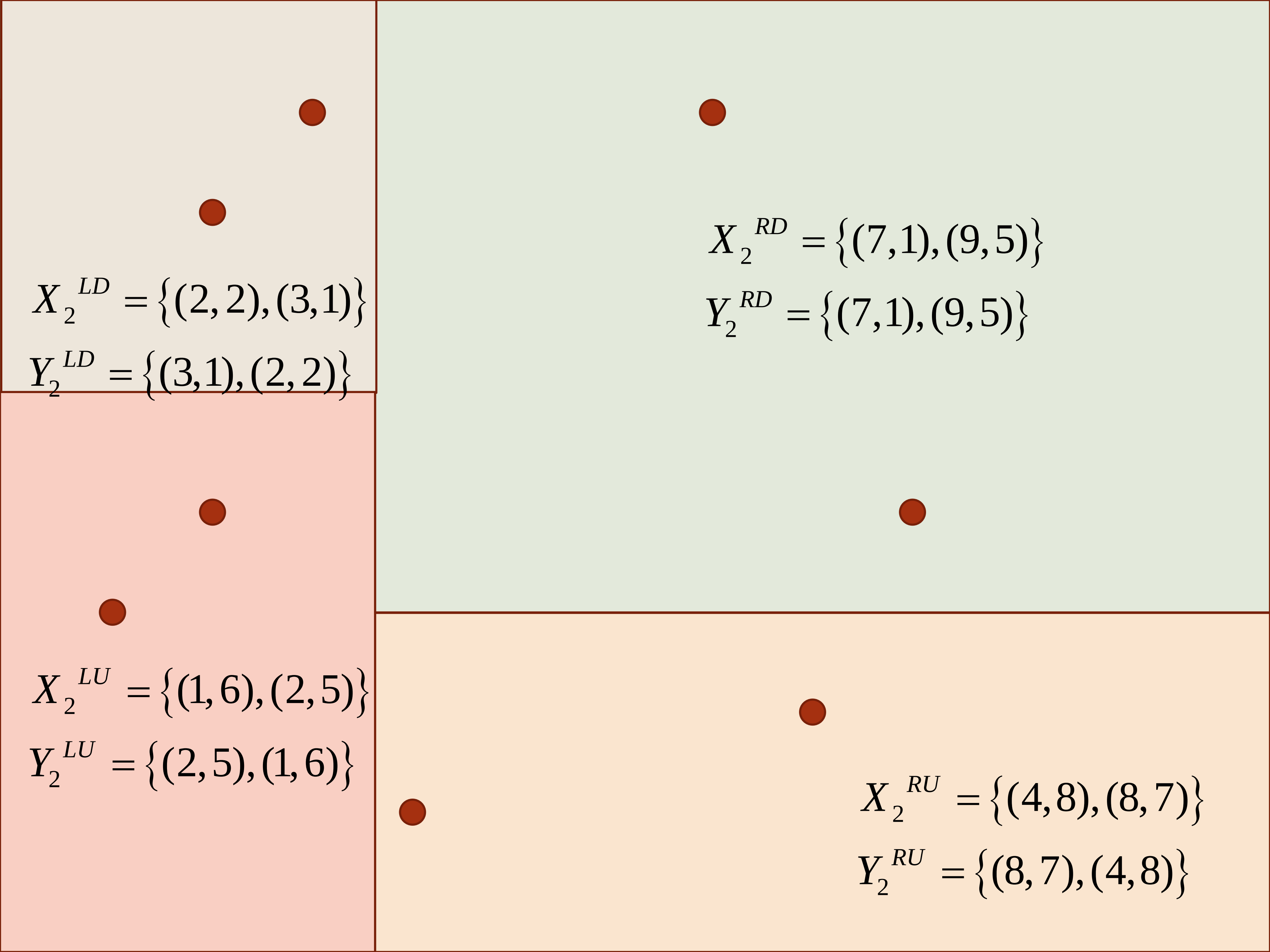
$$Y_2^{\ RU} = \{(8,7),(4,8)\}$$





領域内の点が3以下になるまで分割

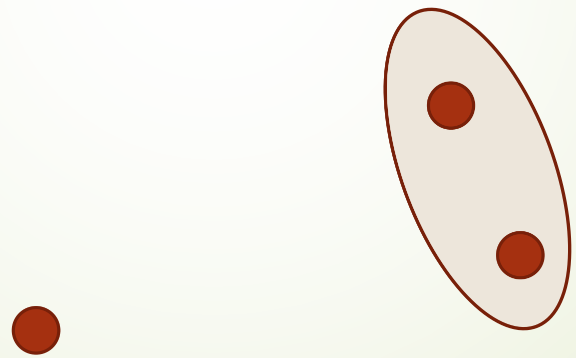



$X_2^{LD} = \{(2,2),(3,1)\}$
$Y_2^{LD} = \{(3,1),(2,2)\}$
$X_2^{RD} = \{(7,1),(9,5)\}$
$Y_2^{RD} = \{(7,1),(9,5)\}$
$X_2^{LU} = \{(1,6),(2,5)\}$
$Y_2^{LU} = \{(2,5),(1,6)\}$
$X_2^{RU} = \{(4,8),(8,7)\}$
$Y_2^{RU} = \{(8,7),(4,8)\}$

x-方向とy-方向の分割は交互であり、かつ対称的

- x-方向に分割し、 y-方向のリストを対応付け
- y-方向に分割し、 x-方向のリストを対応付け
- 違いは、点のどちらの座標を見るか

- 含まれている点の数が3以下になったら、最短距離の点の組を返す。
  - 比較は3回以下



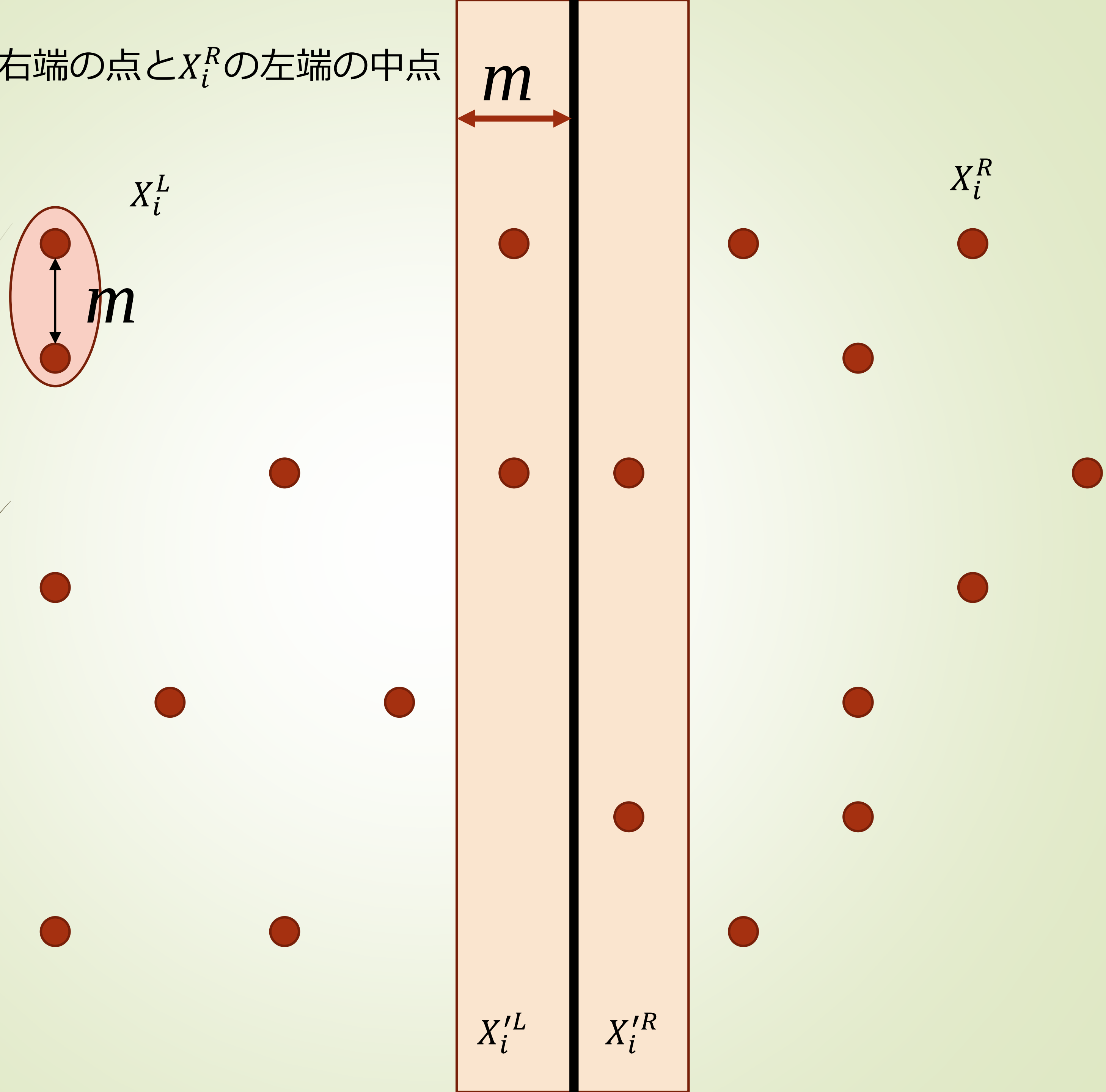
```
findSub(l_x, l_y, d){//ある方向のリスト、直交方向のリスト、方向
    if (|l_x| ≤ 3) 力づくで最近接点を探す
    l_xを分割→ l_l^x, l_r^x
    l_yを対応して分割→ l_l^y, l_r^y
    p_l =findSub(l_l^y, l_l^x, d+1)
    p_r =findSub(l_r^y, l_r^x, d+1)
    p = min(p_l, p_r)
    境界で最近接点を探す
}
```

例えば、$d\%2 = 0$の場合は水平方向、$d\%2 = 1$の場合は鉛直方向とする

最近接点：統合

- 二つに分割した空間内のそれぞれの最近節点の組pとqを得る。
- pがqより、距離が短いとし、その距離をmとする。
- 境界から距離m以内の点について、最近接点を求める

境界はX_i^Lの右端の点とX_i^Rの左端の中点



左右の場合の統合

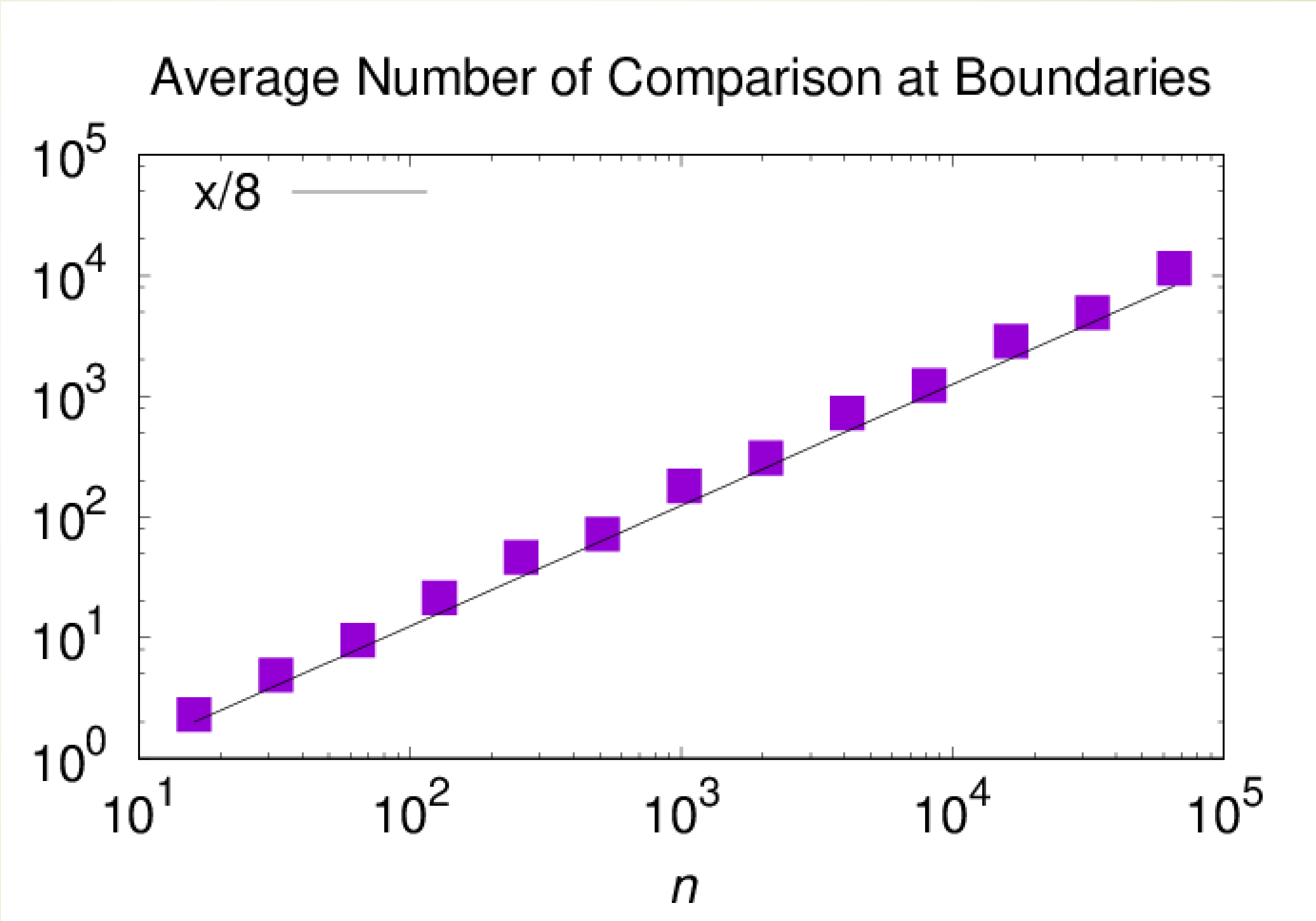
- 左領域の右端の点p^L
- 右領域の左端の点p^R
- それぞれのx座標の中点を境界とする
  - $x_c = \frac{1}{2}(p_x^L + p_x^R)$
- X'^L:X^Lのうち、x_cからの距離がm以下の点の集合
- X'^R:X^Rのうち、x_cからの距離がm以下の点の集合

- 左右分割
  - $\forall(a,b), a \in X_i'^L, b \in X_i'^R$について距離を計測
- 上下分割
  - $\forall(a,b), a \in Y_i'^T, b \in Y_i'^D$について距離を計測

境界部分の点のリスト生成 x方向分割の場合

X^R:右側リスト
X^L:左側リスト
$r = X^R$の先頭要素
$l = X^L$の先頭要素
$c = (r.x + l.x)/2$
if($|r - l| < m$){
    $ln = \{p \in X^L \mid |p.x - c.x| \leq m\}$
    $rn = \{p \in X^R \mid |p.x - c.x| \leq m\}$
}

```
PointPair findMin(左リスト l, 右リスト r, 既知の組 p){
    d_min=pの距離
    p_new=p;//点の組
    l'//境界からd_min以内の左の点リスト
    r'//境界からd_min以内の左の点リスト
    forall(a ∈ l'){
        forall(b ∈ r'){
            if ( distance(a,b)<d_min){
                d_min=distance(a,b);
                p_new=(a,b);
            }
        }
    }
    return p_new;
}
```


Average Number of Comparison at Boundaries
x/8
10^0
10^1
10^2
10^3
10^4
10^5
10^1
10^2
10^3
10^4
10^5
n