


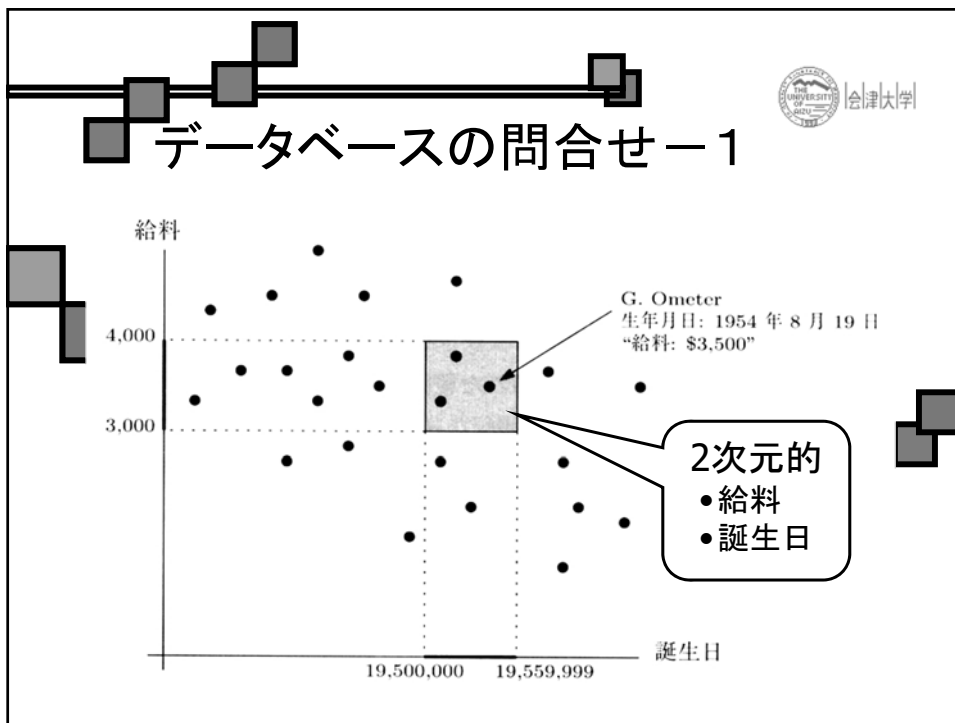
計算幾何学

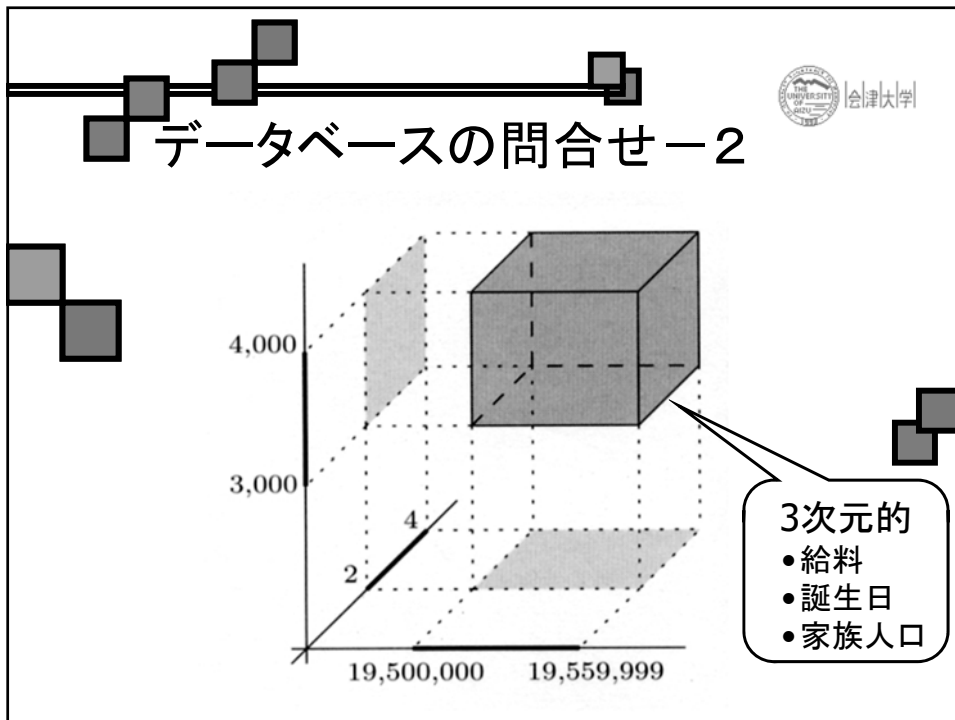
Computational Geometry



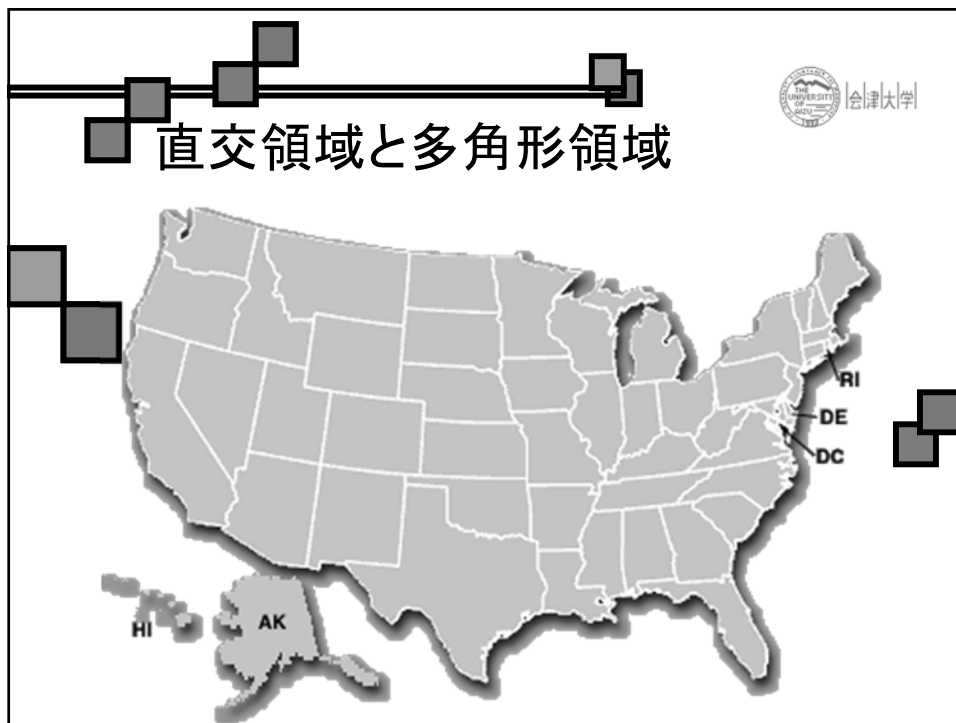
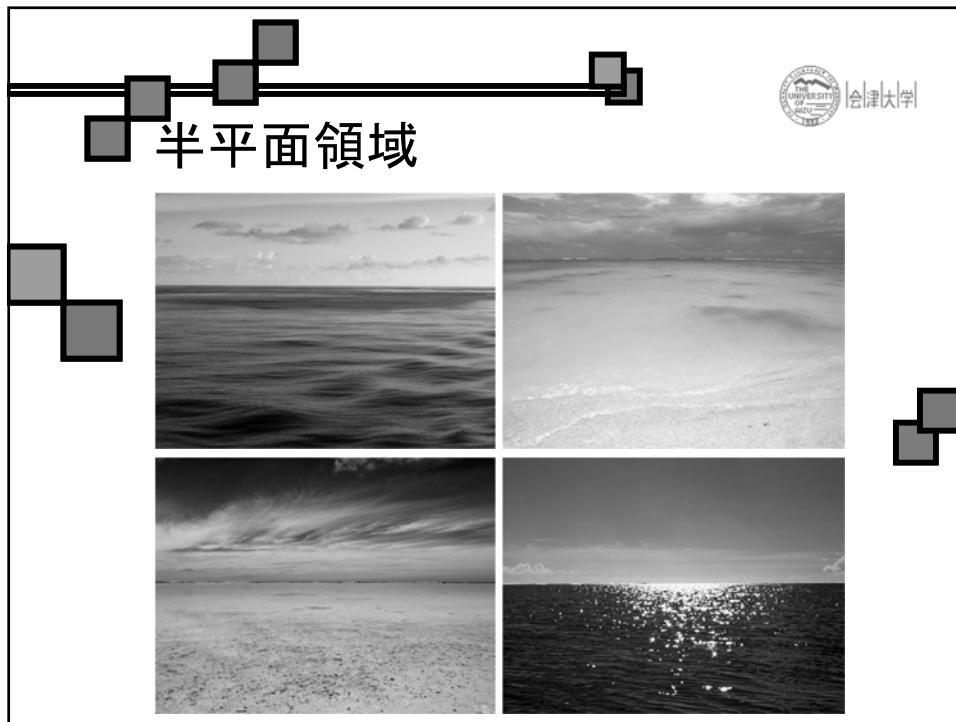
会津大学

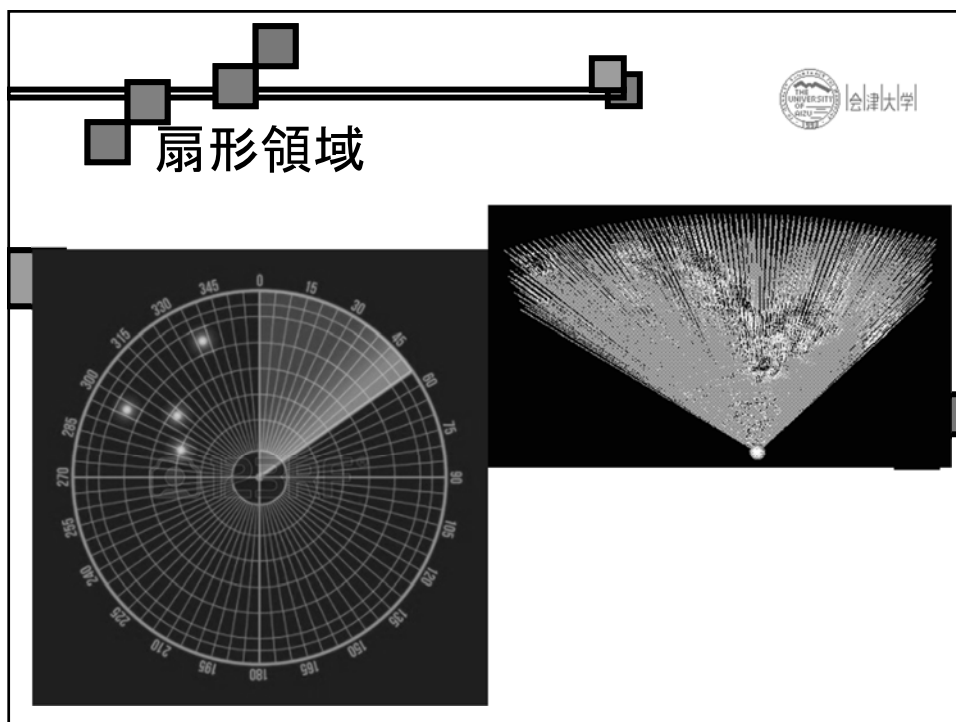
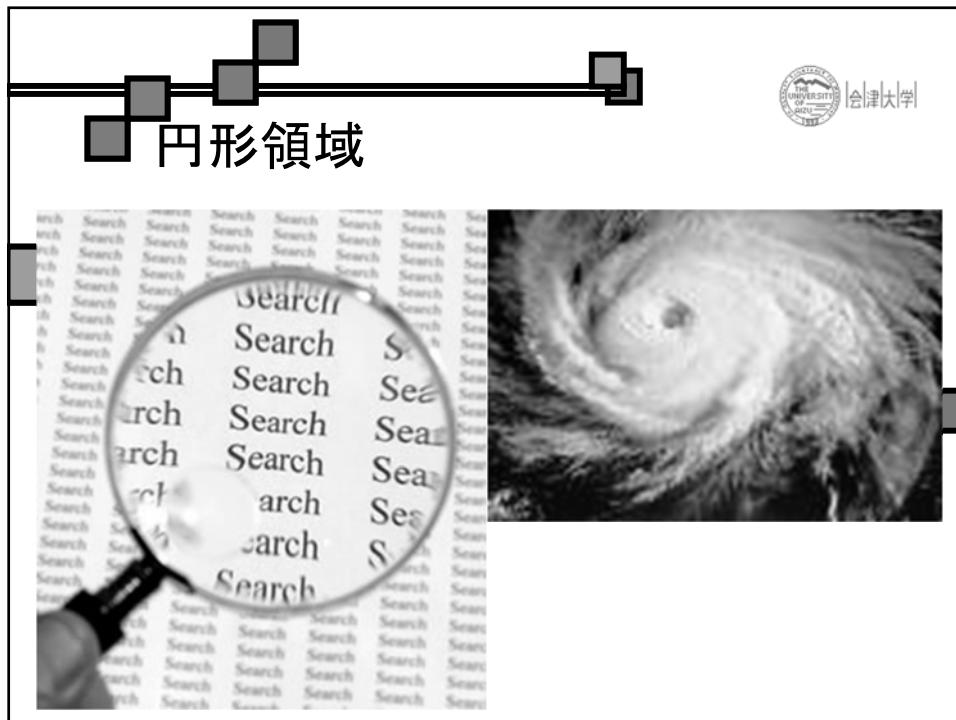
第六章 幾何的領域探索
Geometric range search






- ### 幾何的領域探索(range search)
- データベースの中から対応するフィールドが指定された範囲内にあるレコードをすべて報告せよという問合せは、
 - ① レコードの d 個のフィールドを d 次元空間に変換
 - ② レコードを d 次元空間の点に変換
 - ③ 範囲内にある点集合をすべて求める
 という幾何学的な問題になる
 - 幾何学的領域探索
 - 半平面領域探索
 - 長方形領域探索(直交領域探索)
 - 多角形領域探索
 - 円形領域探索
 - 扇形領域探索

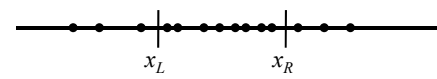







1次元領域探索

- 既知: 直線上の点集合 $P = \{p_1, p_2, \dots, p_n\}$
- 設問: 区間 $[x_L : x_R]$ に含まれる点集合を探せ

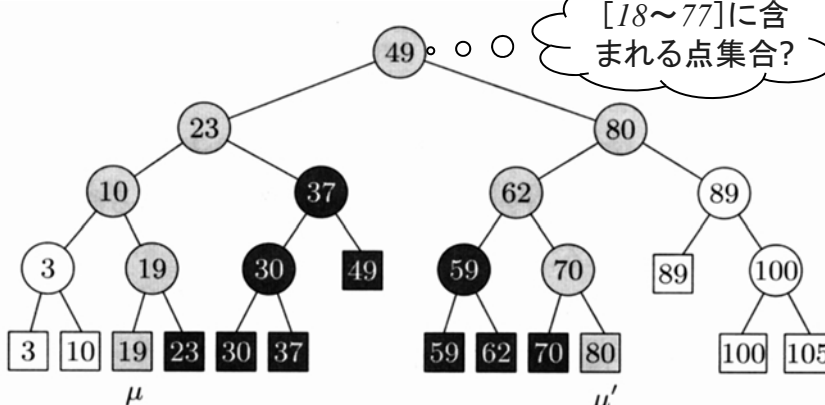



- 2分探索木 T に点集合を保存すると、高速な探索が可能
- T の構築方法
 1. T の外点 $\leftarrow P$
 2. T の内点 \leftarrow 探索を助けるための分割値
 3. 内点 v の分割値を x_v とする
 4. 左部分木節点の値 $\leq x_v$
 5. 右部分木節点の値 $> x_v$



点集合と2分探索木

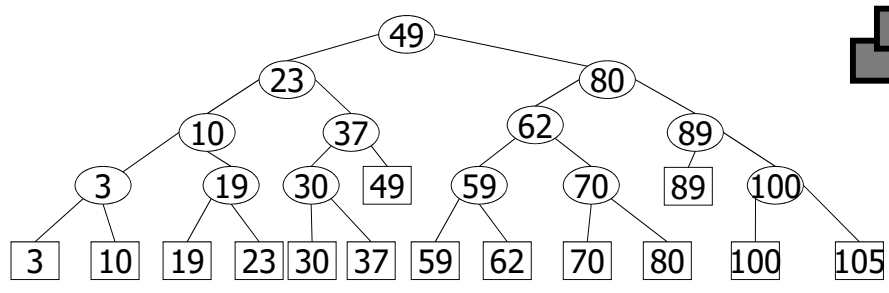
- 点集合 $P = \{3, 10, 19, 23, 30, 37, 49, 59, 62, 70, 80, 89, 100, 105\}$






Query [18:77]

- 分岐節点 V_{split} を探索する
- Report all values in left subtree on search path for μ
- Report all values in right subtree on search path for μ'
- 左右経路の先端にある外点を報告すべきか判定する

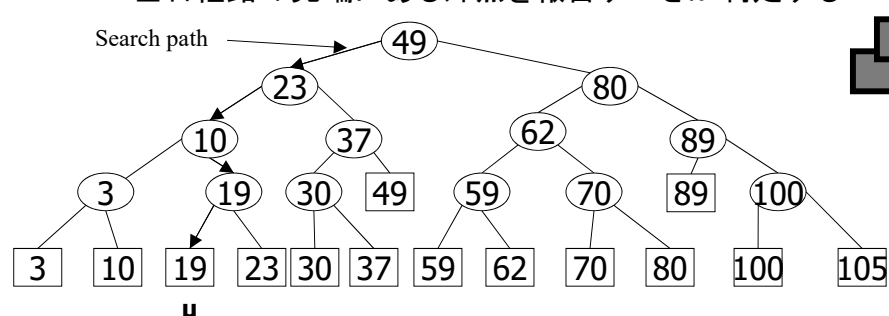





Query [18:77]

- 分岐節点 V_{split} を探索する
- Report all leaf values in left subtree on search path for μ
- Report all leaf values in right subtree on search path for μ'
- 左右経路の先端にある外点を報告すべきか判定する

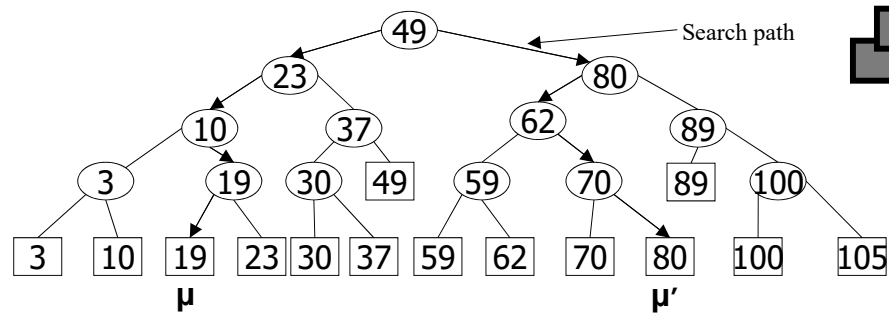
Search path →






Query [18:77]

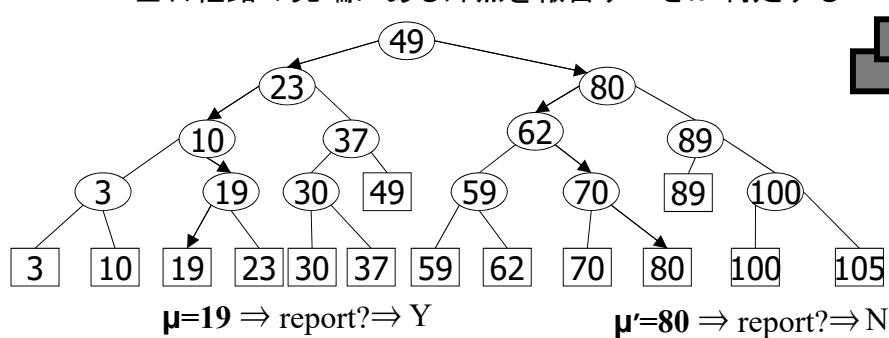
- 分岐節点 V_{split} を探索する
- Report all leaf values in left subtree on search path for μ
- Report all leaf values in right subtree on search path for μ'
- 左右経路の先端にある外点を報告すべきか判定する






Query [18:77]

- 分岐節点 V_{split} を探索する
- Report all values in left subtree on search path for μ
- Report all values in right subtree on search path for μ'
- 左右経路の先端にある外点を報告すべきか判定する





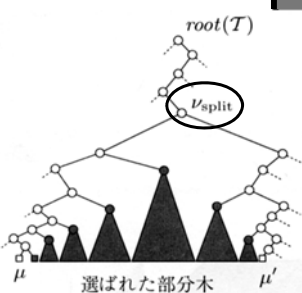
分岐節点を探索する

$\text{FindSplitNode}(T, x_L, x_R)$


- 入力: 木 T と2つの値 x_L と x_R 、ただし、 $x_L \leq x_R$
- 出力: ① x_L への探索経路と x_R への探索経路が分岐する内点 v 、又は、②両方の経路が共に終了する外点 v

1. $v \leftarrow \text{root}(T)$
2. while ($v \neq \text{leaf}$) and ($x_v \geq x_R$ or $x_v < x_L$)
3. do if $x_v \geq x_R$
4. then $v \leftarrow \text{lc}(v)$
5. else $v \leftarrow \text{rc}(v)$
6. return v

$\text{lc} = \text{left child}$
 $\text{rc} = \text{right child}$



選ばれた部分木

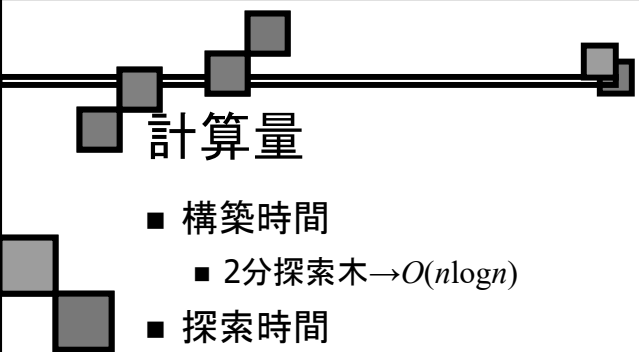



アルゴリズム

$\text{1DRangeQuery}(T, x_L, x_R)$

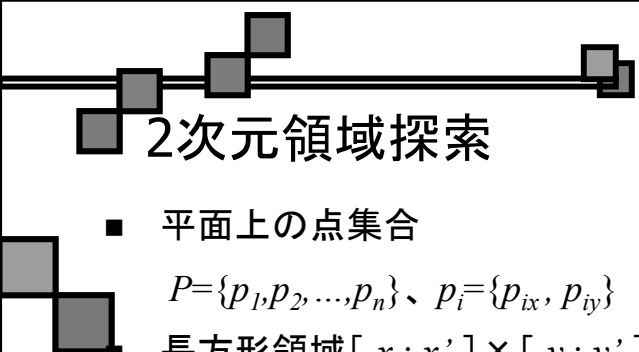

- 入力: 木 T と2つの範囲値 x_L と x_R 、ただし、 $x_L \leq x_R$
- 出力: 指定された範囲内にある点集合

1. $v_{\text{split}} \leftarrow \text{FindSplitNode}(T, x_L, x_R)$
2. if $v_{\text{split}} = \text{leaf}$
3. then v_{split} に蓄えられる点を報告すべきか判定する
4. else
5. ① x_L への経路をたどり、経路の右側にある部分木の外点を報告し、経路の先端にある外点を報告すべきか判定する
6. ② x_R への経路の左側について①と類似な処理

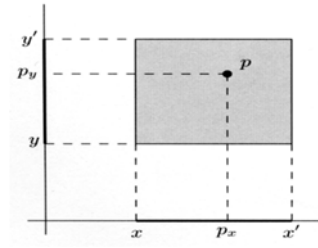
計算量


- 構築時間
 - 2分探索木 → $O(n \log n)$
- 探索時間
 - 区間内にある点の数 = k
 - ReportSubTreeの計算時間 = $O(k)$
 - FindSplitNodeの計算時間 = 木の高さ = $O(\log n)$
 - 全体の時間 = $O(\log n + k)$

2次元領域探索

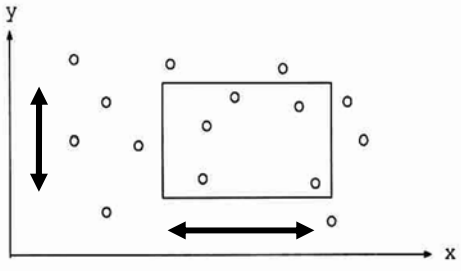
- 平面上の点集合
 $P = \{p_1, p_2, \dots, p_n\}$, $p_i = \{p_{ix}, p_{iy}\}$
- 長方形領域 $[x : x'] \times [y : y']$ に含まれる点集合を探索する
- 必要十分条件 $p_x \in [x : x']$
 and $p_y \in [y : y']$






基本的な考え方

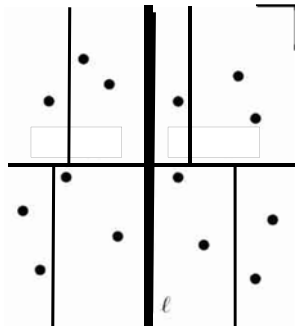
- $p_x \in [x : x'] \ \& \ p_y \in [y : y'] \rightarrow$ 必要十分条件
- 点の x 座標値と点の y 座標値、それぞれについて、2つの1次元問題にすれば・・・?
- 2分探索木を使う
- But, how to ...?
- 具体的に言えば...

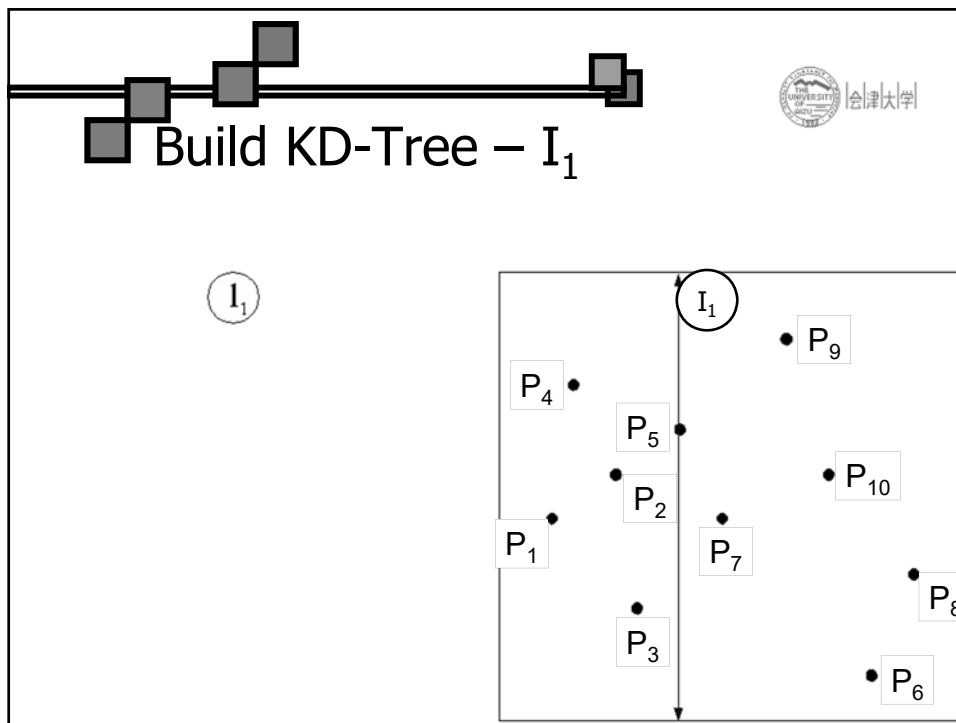
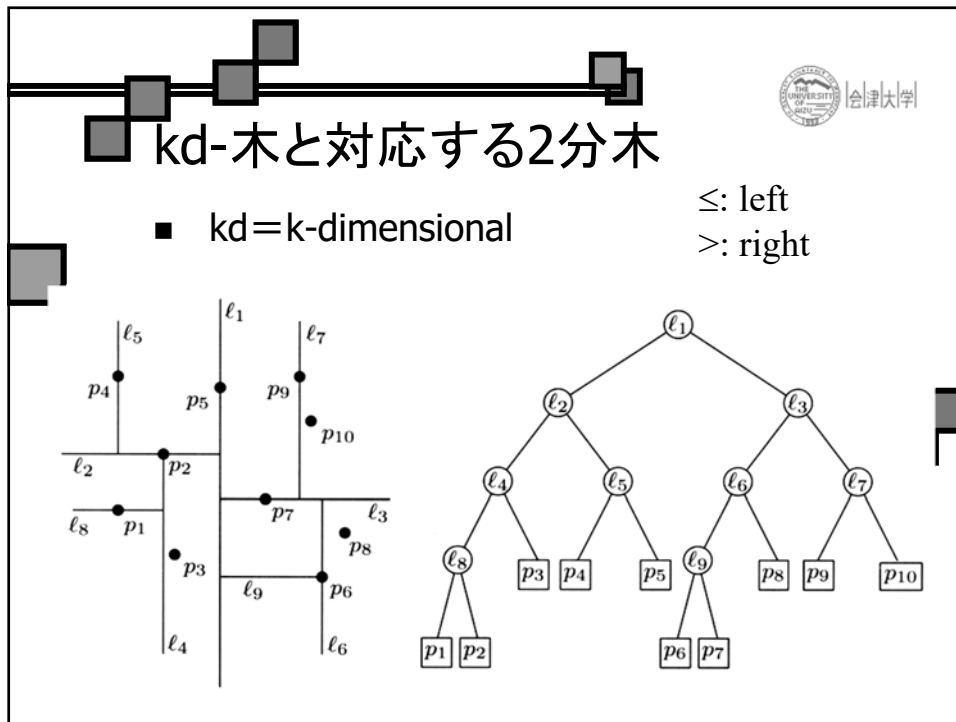


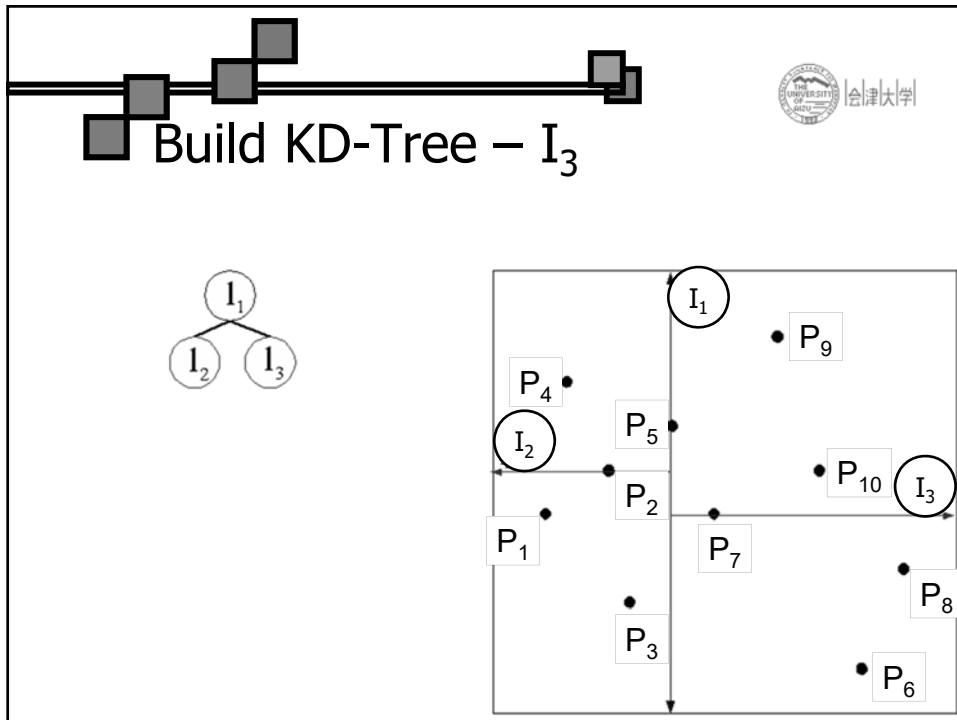
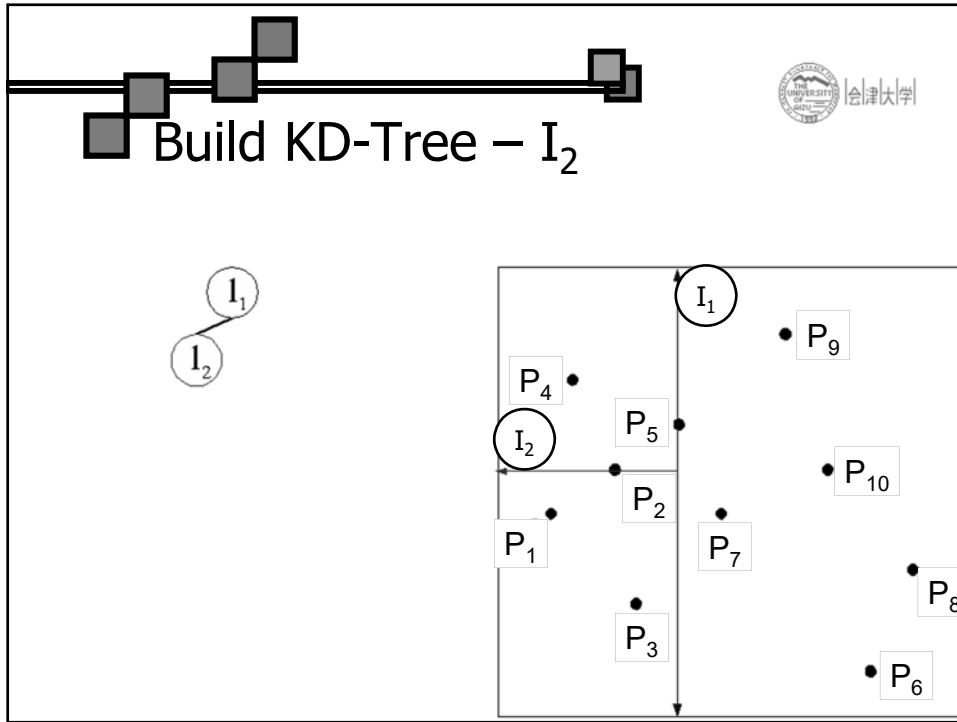


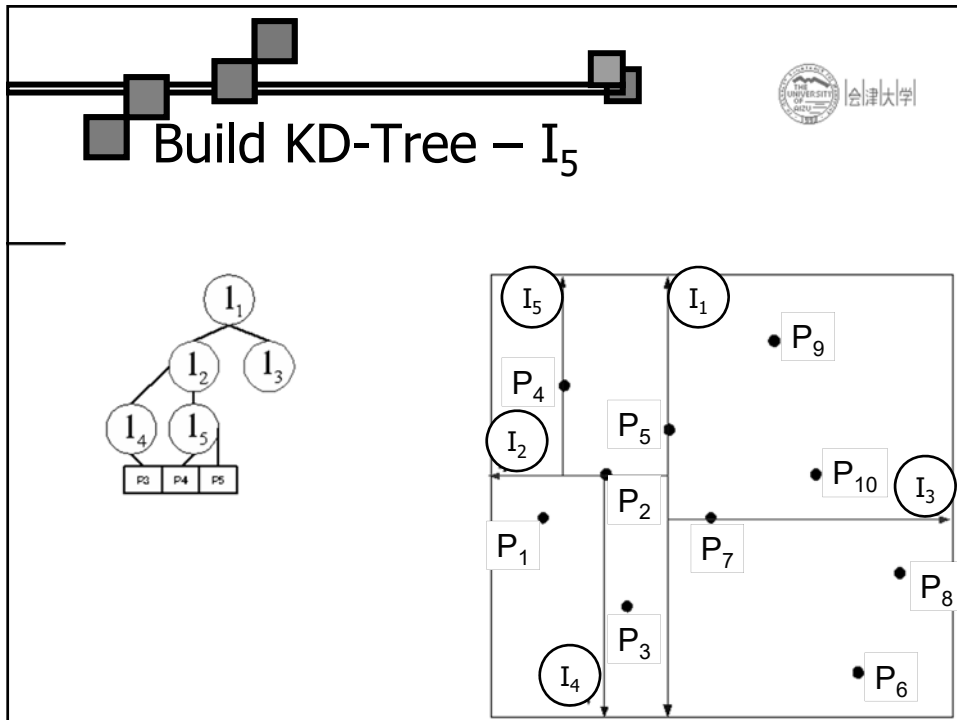
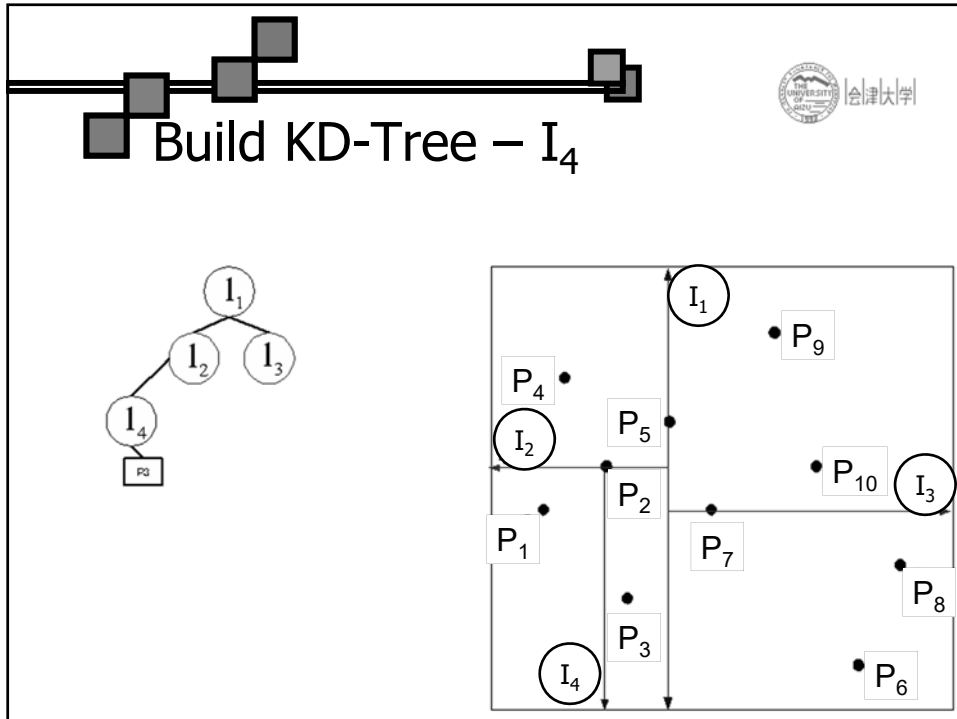
基本的な考え方 続き

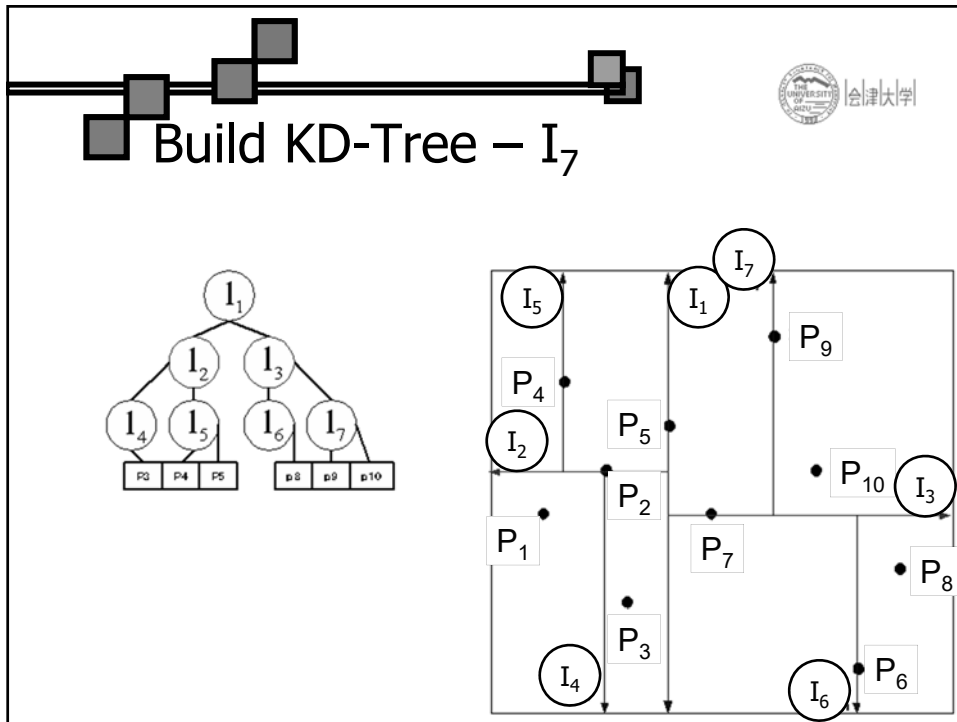
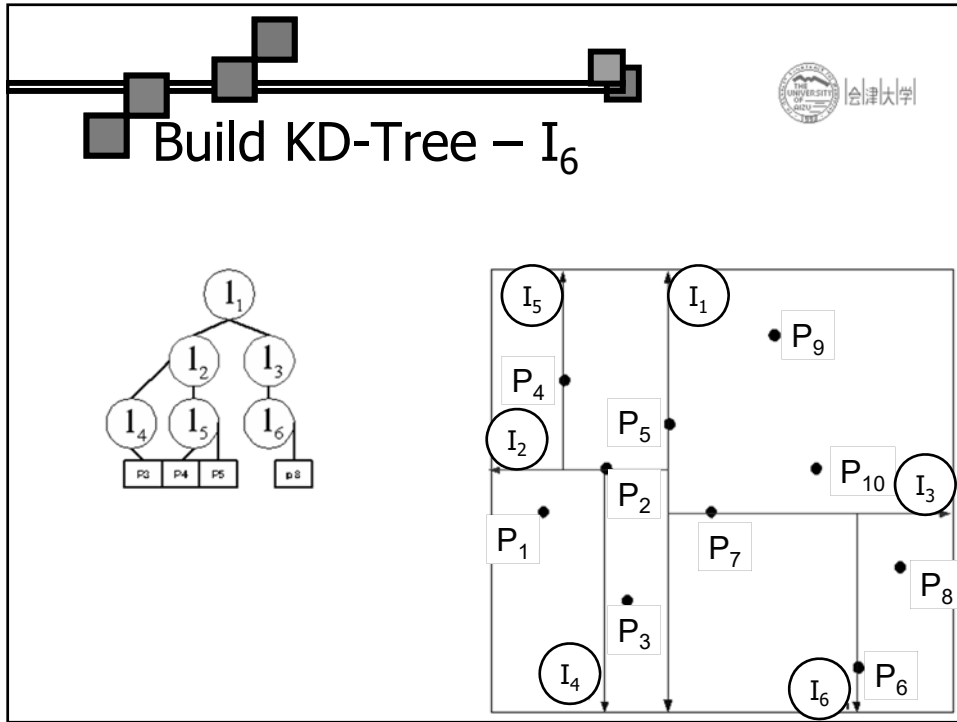
1. P を2分割する垂直線 $l \rightarrow$ 根
2. $P_{\text{left}} \rightarrow$ 左部分木、 $P_{\text{right}} \rightarrow$ 右部分木
3. P_{left} を水平線で2分割
 1. 上側 \rightarrow 右部分木
 2. 下側 \rightarrow 左部分木
4. P_{right} も同様
5. 根の孫節点では、再び垂直線で分割する
6. 垂直・水平分割を交互に繰り返す

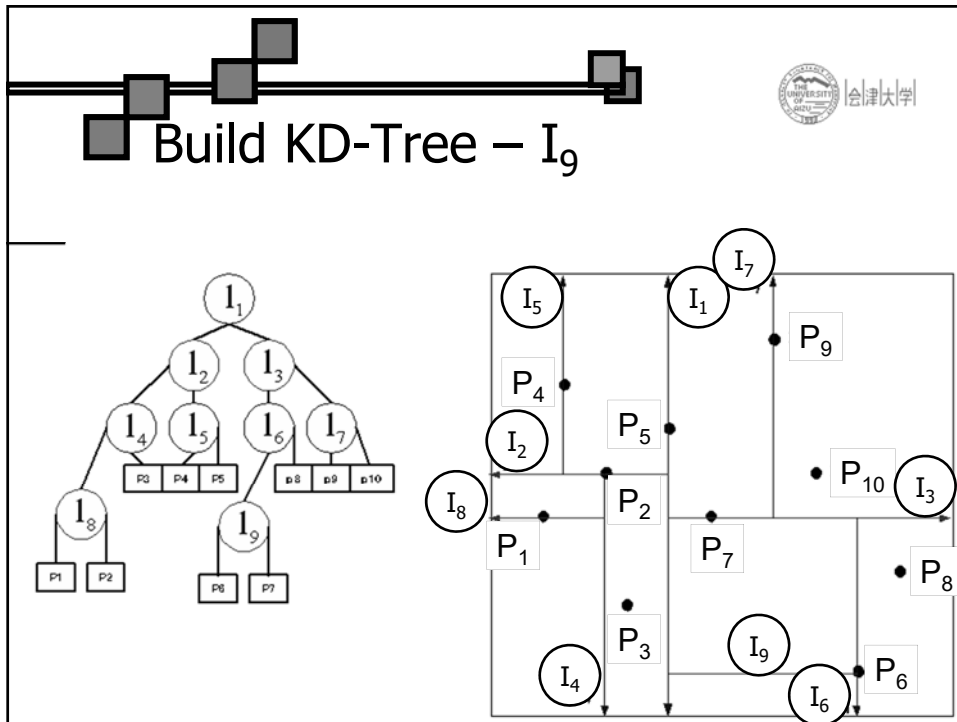
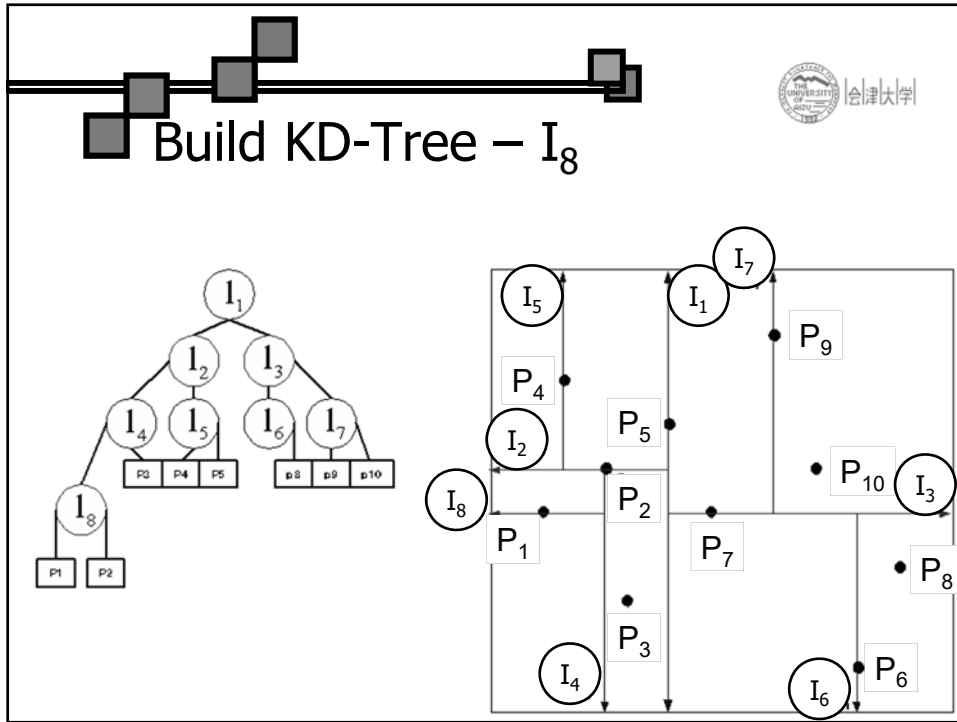












■ アルゴリズム


BuildKdTree($P, depth$)

- 入力: 点集合 P 、木の深さ $depth$
- 出力: P を蓄えるkd木の根

1. if P の点数=1 // 一点のみの場合
2. then return その点を蓄えている外点
3. else
4. if $depth = \text{even}$ // count from 0
5. then // split with a vertical line (垂直分割)
6. P の x 座標値の中央値 x_{median} を通る垂直線 l で P を2分割
 - 左側 P_1 点集合の x 座標値 $\leq x_{\text{median}}$
 - 右側 P_2 点集合の x 座標値 $> x_{\text{median}}$


■ アルゴリズム 続き

7. else // split with a horizontal line (水平分割)
8. P の y 座標値の中央値 y_{median} を通る水平線 l で P を2分割
 - 下側 P_1 点集合の y 座標値 $\leq y_{\text{median}}$
 - 上側 P_2 点集合の y 座標値 $> y_{\text{median}}$
9. $v_{\text{left}} \leftarrow \text{BuildKdTree}(P_1, depth+1)$
10. $v_{\text{right}} \leftarrow \text{BuildKdTree}(P_2, depth+1)$
11. l を蓄えている節点 v を作り、
 v の左子= v_{left} 、 v の右子= v_{right}
12. return v


 会津大学

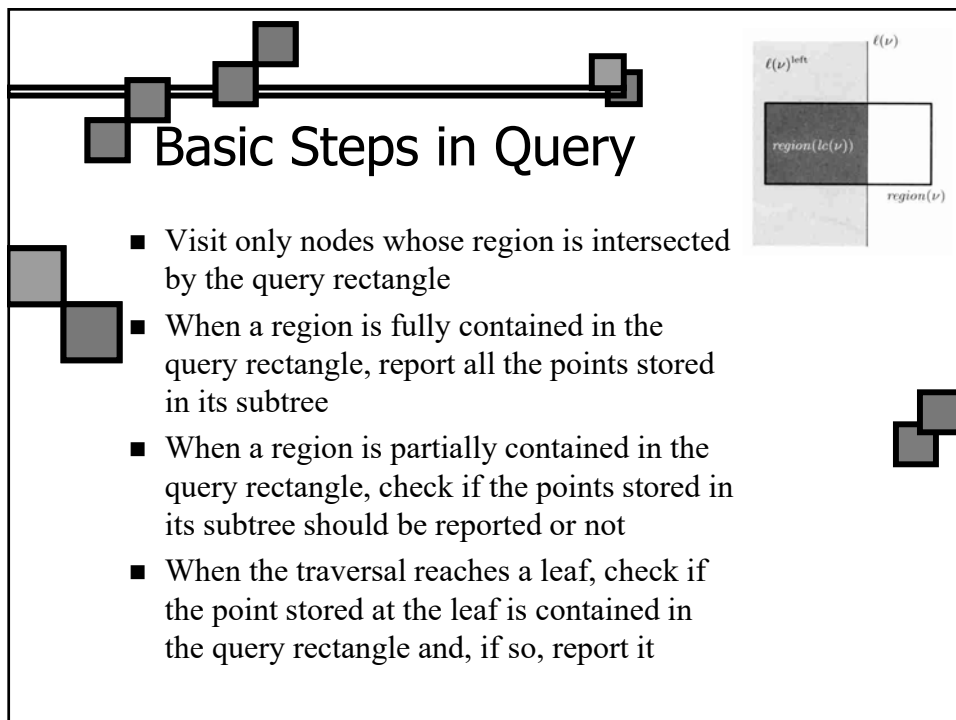
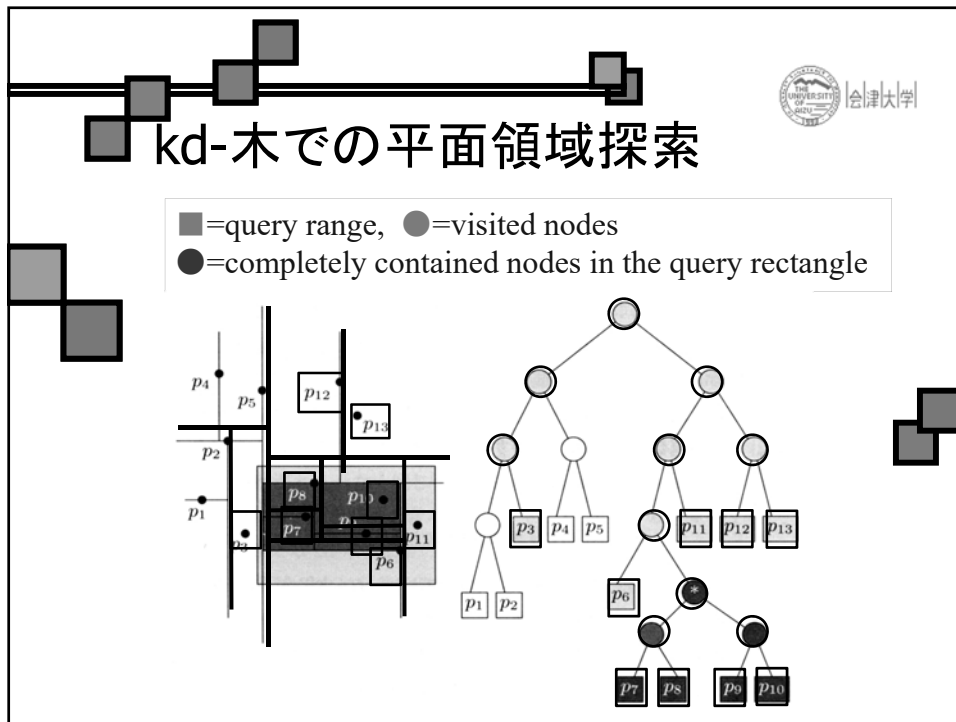
kd-木と平面領域の対応関係


- 根 $l_1 \Leftrightarrow$ 全平面
- 木の節点 $v \Leftrightarrow$ 平面領域 $region(v)$


 会津大学

節点 v と質問領域 R

- 主な判定は、質問領域 R がある節点 v に対応する領域と重なりを持つかどうかの判定。
- 垂直分割(深さが偶数)である節点 v の左子に対応する領域は、 $region(v)$ を用いると、次のような関係が成立。
 $region(lc(v)) = region(v) \cap l(v)^{left}$
- 水平分割(深さが奇数)である節点 v の左子に対応する領域は、 $region(v)$ を用いると、次のような関係が成立。
 $region(lc(v)) = region(v) \cap l(v)^{lower}$



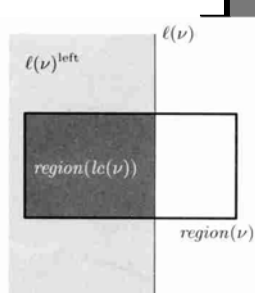


■ アルゴリズム


SearchKdTree(v, R)

- 入力: kd木の根(部分木の節点) v と質問領域 R
- 出力: この範囲に入っている v の下にある全ての外点

1. if $v = \text{leaf}$
2. then v に蓄えられた点を調べ、 R の内部にあれば、報告
- else
3. if $\text{region}(lc(v))$ が R に完全包含
4. then ReportSubtree($lc(v)$)
5. else if $\text{region}(lc(v))$ が R と部分包含
6. then SearchKdTree($lc(v), R$)
7. if $\text{region}(rc(v))$ が R に完全包含
8. then ReportSubtree($rc(v)$)
9. else if $\text{region}(rc(v))$ が R と部分包含
10. then SearchKdTree($rc(v), R$)



The diagram shows a node v with a vertical splitting line. The left region is labeled $\ell(v)$ and the right region is labeled $r(v)$. A query region R is shown as a rectangle. The region of the left child is labeled $\text{region}(lc(v))$.



■ 計算量


- n = 点集合の総点数
- k = 報告すべき領域内の点数
- 記憶Storage = $O(n)$
- 構築Construction = $O(n \log n)$
 - 分割線探索 $\rightarrow x$ と y について2分探索木を用意し、交互に中央値を再帰取り出す
 - 計算量 $T(n)$ の漸化式

中央値を求める

$$T(n) = \begin{cases} O(1), & n=1 \\ O(n) + 2T\left(\left\lceil \frac{n}{2} \right\rceil\right), & n > 1 \end{cases}$$
- 探索Query = $O(\text{sqrt}(n) + k)$
 - 水平と垂直分割線をそれぞれ用いて再帰探索 \rightarrow 一つ領域に高々 $\lceil n/4 \rceil$ 個の点をしか含まない
 - 計算量 $T(n)$ の漸化式


左右上下探索を含む

$$T(n) = \begin{cases} O(1), & n=1 \\ 2 + 2T\left(\left\lceil \frac{n}{4} \right\rceil\right), & n > 1 \end{cases}$$



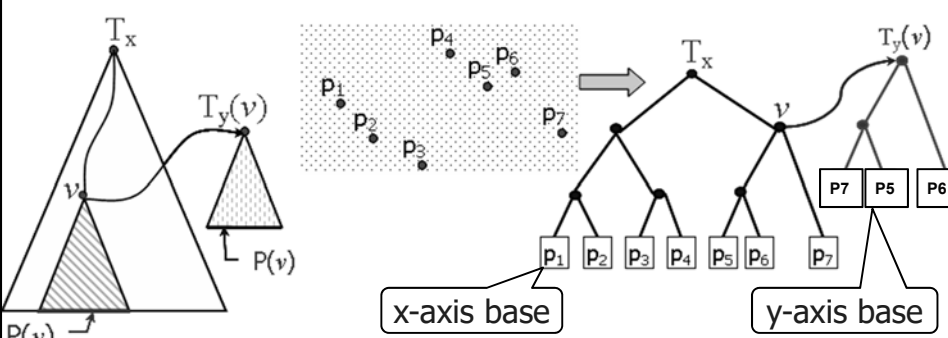
kd-木と領域木(Range tree)



- kd-木: x, y 座標交互に
 - ① 2つ1次元領域探索
 - ② x, y 座標値について交互に分割
- 領域木: x, y 座標順番に
 - ① x 座標値について→1次元領域探索
 - ② y 座標値について→1次元領域探索
- 共通点=2次元領域探索→2つ1次元問題



領域木の構築

- x 座標に基づき構築した木 T_x
- 節点 v を根とする部分木の葉の点集合 $P(v)$
- y 座標に基づき点集合 $P(v)$ を構築した木 $T_y(v)$
- Pointer from v to $T_y(v)$



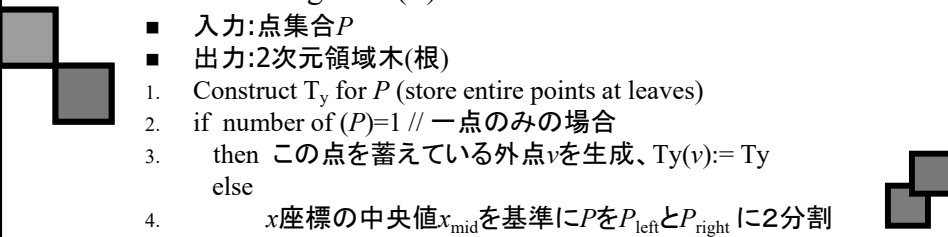





■ アルゴリズム(Build)

Build2DRangeTree(P)

- 入力: 点集合 P
- 出力: 2次元領域木(根)

1. Construct T_y for P (store entire points at leaves)
2. if number of (P)=1 // 一点のみの場合
3. then この点を蓄えている外点 v を生成、 $T_y(v) := T_y$
- else
4. x 座標の中央値 x_{mid} を基準に P を P_{left} と P_{right} に2分割
5. $v_{left} \leftarrow \text{Build2DRangeTree}(P_{left})$
6. $v_{right} \leftarrow \text{Build2DRangeTree}(P_{right})$
7. 新節点 v を生成し、 $x_v \leftarrow x_{mid}$ を蓄える
8. Leftchild(v) $\leftarrow v_{left}$
9. Rightchild(v) $\leftarrow v_{right}$
10. $T_y(v) := T_y$
11. return v

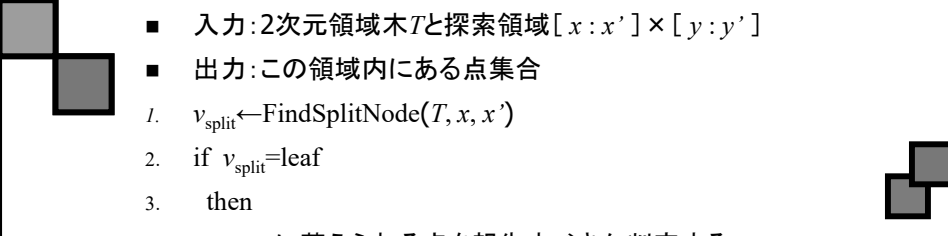




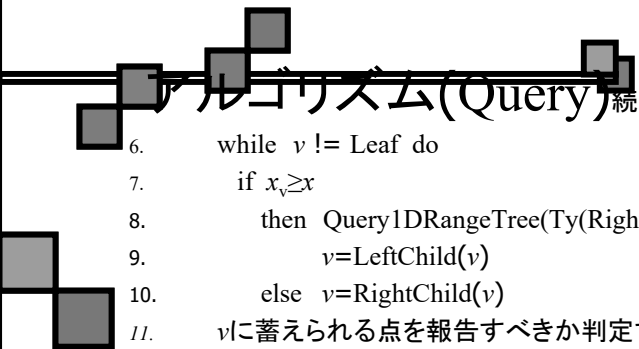
■ アルゴリズム(Query)


Query2DRangeTree($T, [x : x'] \times [y : y']$)

- 入力: 2次元領域木 T と探索領域 $[x : x'] \times [y : y']$
- 出力: この領域内にある点集合

1. $v_{split} \leftarrow \text{FindSplitNode}(T, x, x')$
2. if $v_{split} = \text{leaf}$
3. then
- v_{split} に蓄えられる点を報告すべきか判定する
4. else // x への経路をたどり、経路の右側にある部分木 T_y に関して Query1DRangeTree を呼び出す
5. $v = \text{LeftChild}(v_{split})$



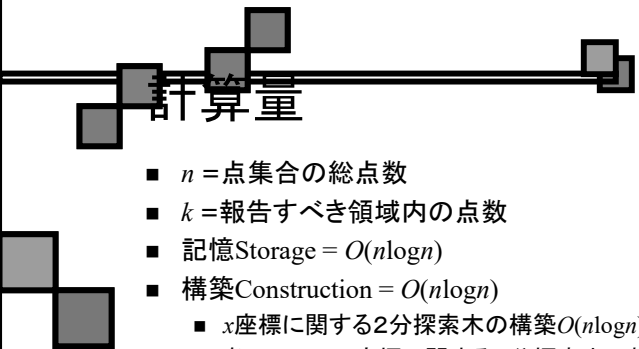




 会津大学

アルゴリズム(Query) 続き

```

6.   while v != Leaf do
7.       if  $x_v \geq x$ 
8.           then Query1DRangeTree(Ty(RightChild(v)), [y : y'])
9.               v=LeftChild(v)
10.          else v=RightChild(v)
11.          vに蓄えられる点を報告すべきか判定する
           //RightChild( $v_{split}$ )から  $x'$ への経路の左側にある部分木Tyに
           関してQuery1DRangeTreeを呼び出す
12.          v=RightChild( $v_{split}$ )
13.          while v != Leaf do
14.              if  $x_v \leq x'$ 
15.                  then Query1DRangeTree(Ty(LeftChild(v)), [y : y'])
16.                      v=RightChild(v)
17.                  else v=LeftChild(v)
18.                  vに蓄えられる点を報告すべきか判定する
    
```




 会津大学

計算量

- n = 点集合の総点数
- k = 報告すべき領域内の点数
- 記憶Storage = $O(n \log n)$
- 構築Construction = $O(n \log n)$
 - x 座標に関する2分探索木の構築 $O(n \log n)$
 - 各 $T_{assoc}(v)$ で y 座標に関する2分探索木の構築 →
一つの深さは $O(n)$ 、すべての深さは $O(n \log n)$
- 探索Query = $O((\log n)^2 + k)$
 - ノード v に対し $1DRangeQuery(rc(v), [y : y'])$ を呼び出すときの
計算量 $O\left(\log \frac{n}{2^{depth}} + k_v\right)$
- 総計算量 $\sum_{i=0}^{\log n} \left(\log \frac{n}{2^i} + k_v\right) = \sum_{i=0}^{\log n} \log \frac{n}{2^i} + \sum_{i=0}^{\log n} k_v = \sum_{i=0}^{\log n} \log n - \sum_{i=0}^{\log n} i + O(k)$
 $= (\log n)^2 - \frac{\log n(\log n + 1)}{2} + O(k) = O((\log n)^2 + k)$

