Ορθογωνική Εκτασιακή Αναζήτηση

Orthogonal Range Searching

Ερωτήματα σε Βάσεις Δεδομένων

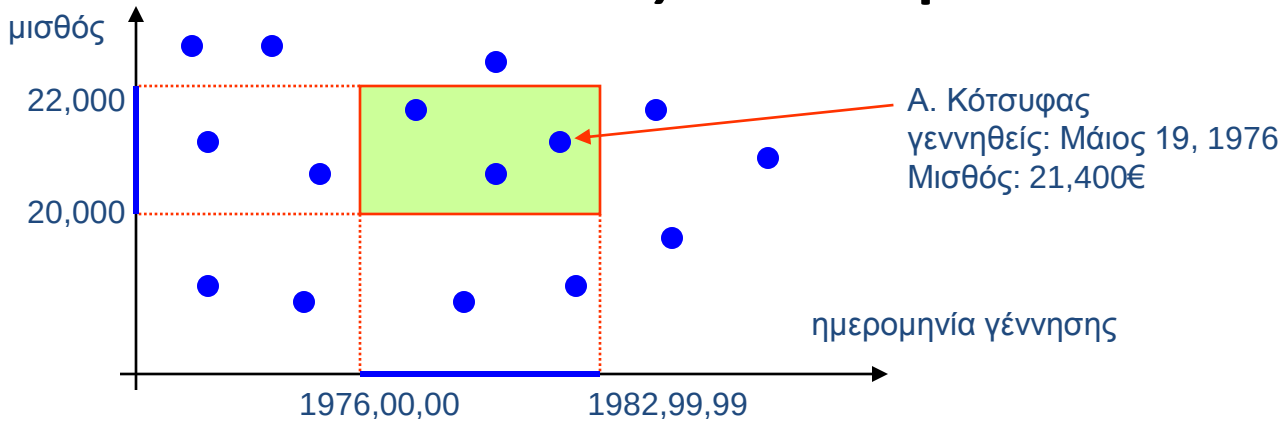
Αναφορές:

- [M. de Berge et al] Κεφάλαιο 5

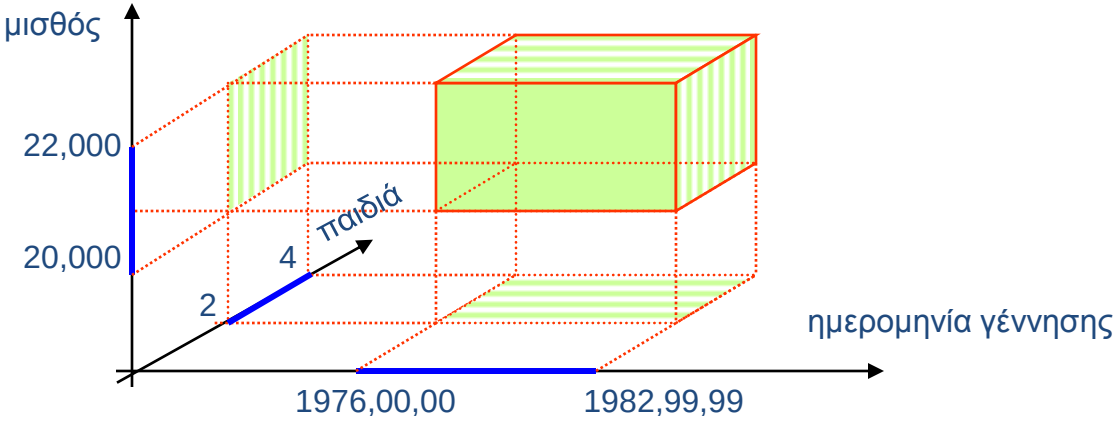
Εφαρμογές:

- Βάσεις Δεδομένων
- GIS, Γραφικά: αποκοπή και εστίαση, παράθυρα

Ορθογωνική Εκτασιακή Αναζήτηση: Βάσεις Δεδομένων

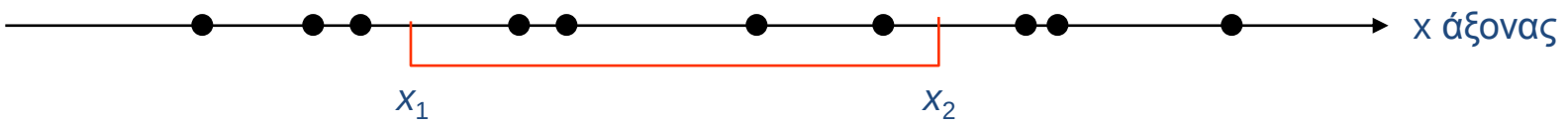


2D Ορθογώνιο Ερώτησης [1976,00,00 : 1982,99,99] × [20,000 : 22,000]



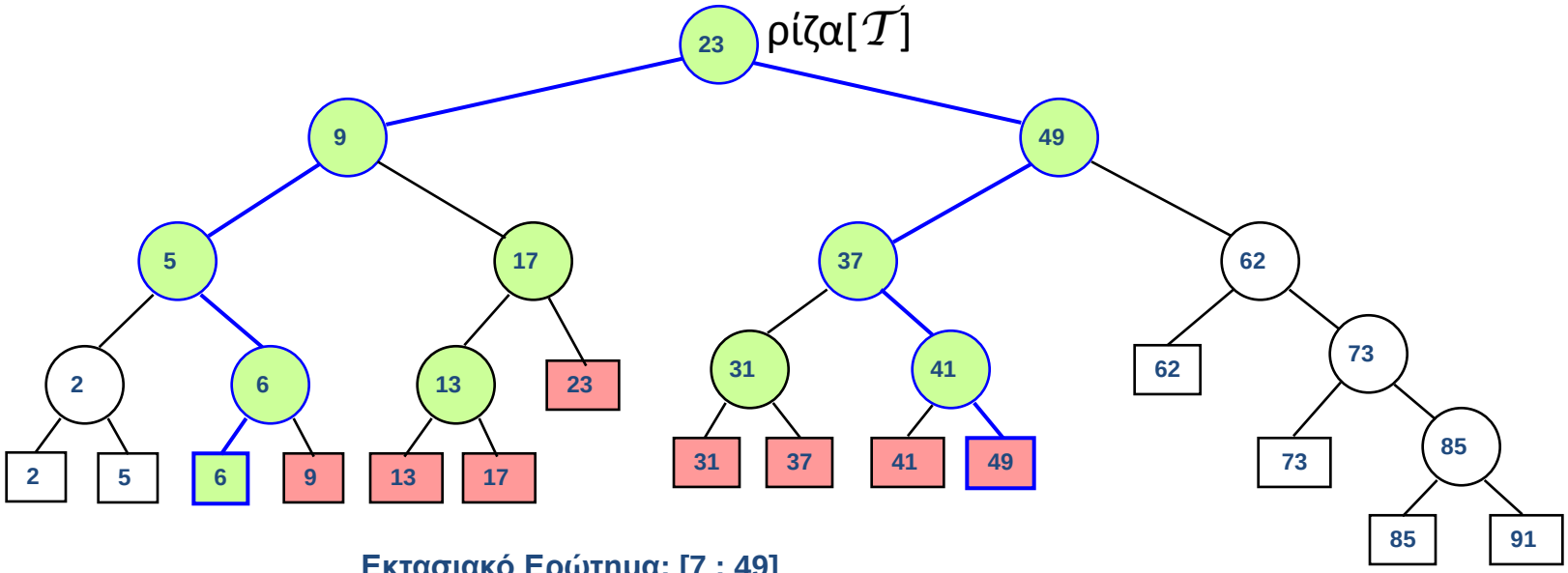
3D Ορθογωνικό Εκτασιακό Ερώτημα [1976,00,00 : 1982,99,99] × [20,000 : 22,000] × [2 : 4]

1D-Δένδρο: Μονοδιάστατη Εκτασιακή Αναζήτηση



Στατικό; Δυαδική Αναζήτηση σε ένα ταξινομημένο πίνακα

Δυναμικό; Αποθήκευση σημείων σε ένα ζυγισμένο δυαδικό δένδρο $\mathcal{T}$.



Εκτασιακό Ερώτημα: [7 : 49]

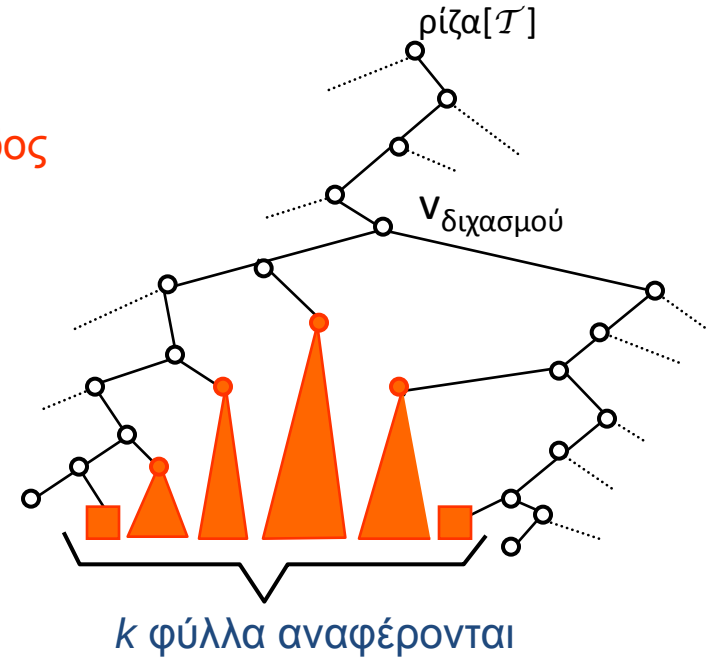
Εκτασιακή Ερώτηση

Αλγόριθμος 1DΕκτασιακήΕρώτηση (v, x_1, x_2)
Αν το *v* είναι φύλλο **τότε αν** $x_1 \le x(v) \le x_2$ **τότε** αναφορά δεδομένων στον *v*
αλλιώς
Αν $x_1 \le x(v)$ **τότε** 1DΕκτασιακήΕρώτηση($\alpha\theta(v)$, x_1, x_2)
Αν $x(v) < x_2$ **τότε** 1DΕκτασιακήΕρώτηση($\delta\theta(v)$, x_1, x_2)

Απόδοση:

Χρόνος Ερώτησης	$O(k + \log n)$ $\mathcal{T}$,$[x_1,x_2]$ → έξοδος
Χρόνος Κατασκευής	$O(n \log n)$ P → $\mathcal{T}$
Χώρος	$O(n)$ αποθήκευση $\mathcal{T}$

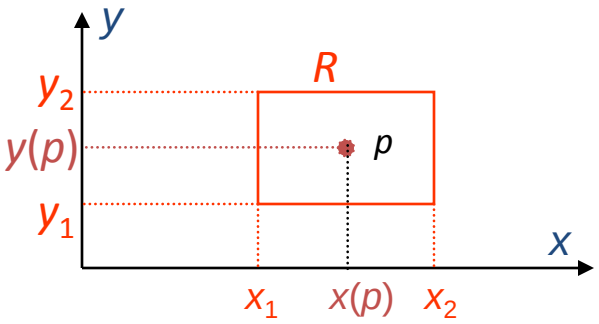
[Βέλτιστα]

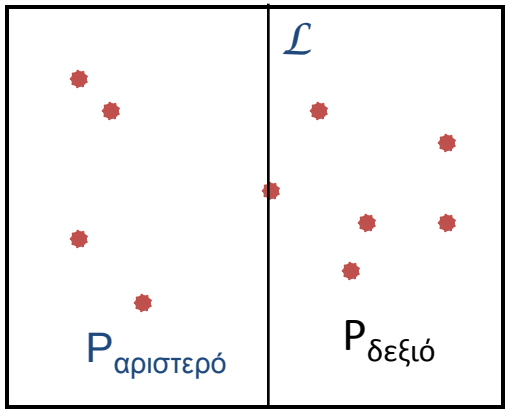


k φύλλα αναφέρονται

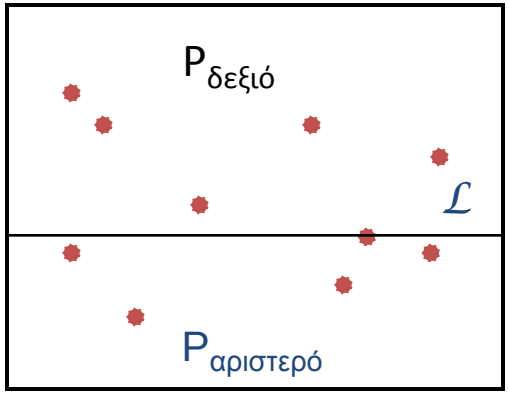
Διδιάστατο Δένδρο

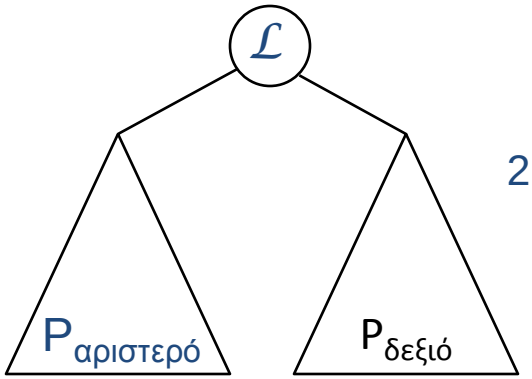
Διδιάστατο πρόβλημα *d*=2:
σημείο $p=(x(p), y(p))$, περιοχή $R = [x_1 : x_2] \times [y_1 : y_2]$
$p \in R \Leftrightarrow x(p) \in [x_1 : x_2]$ και $y(p) \in [y_1 : y_2]$.





Ή





2D-δένδρο

$\mathcal{L}$ = κάθετη/οριζόντια διχαστική ευθεία
Εναλλαγή μεταξύ κάθετης και οριζόντιας διχοτόμησης σε άρτια και περιττά βάθη αντίστοιχα.
(Υπόθεση: δεν υπάρχουν σημεία με ίδιες *x* ή *y* συντεταγμένες.)

Κατασκευή 2D-δένδρου

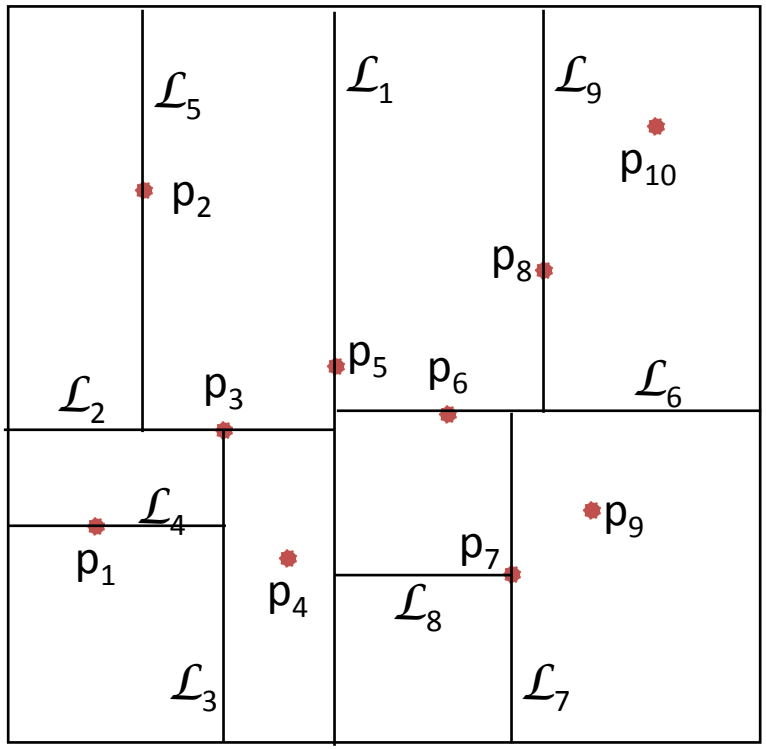
Είσοδος: $P = \{p_1, p_2, \ldots, p_n\} \subseteq \Re^2$
Έξοδος: 2D-δένδρο αποθηκεύοντας το P.

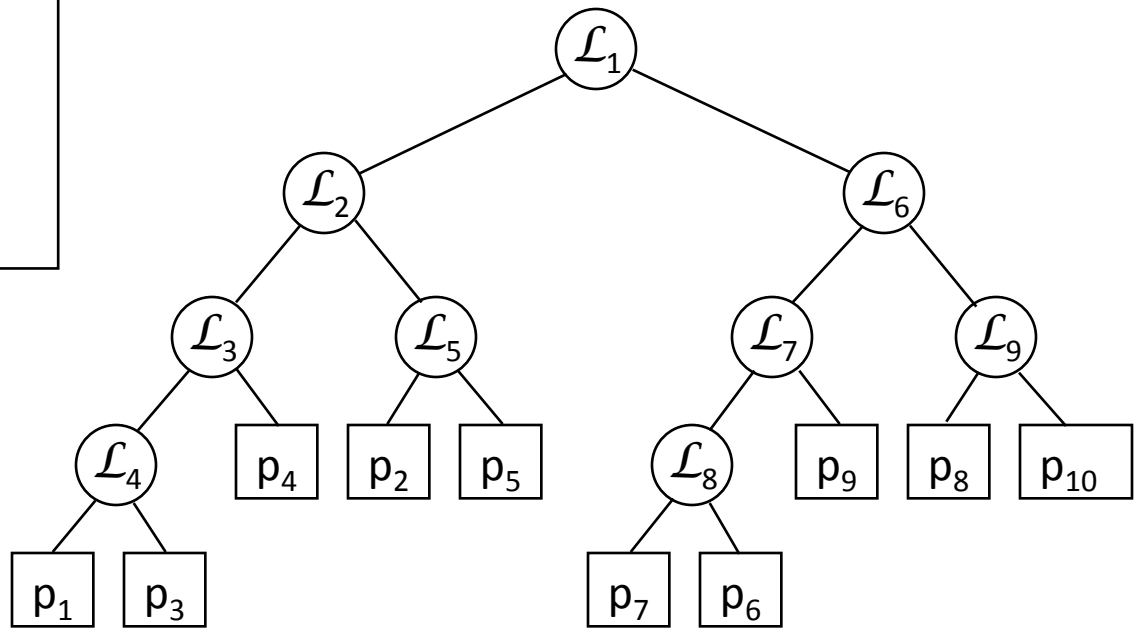
Βήμα 1: Ταξινόμηση του P ως προς x & y, Û = (Xsorted(P), Ysorted(P)).
Βήμα 2: root[$\mathcal{T}$] ← Κατασκευή2DΔένδρου(Û, 0)

Αλγόριθμος Κατασκευή2DΔένδρου(Û, βάθος)
Αν το Û περιέχει ένα σημείο **τότε return** το φύλλο που περιέχει το σημείο
αλλιώς
    Αν το βάθος είναι άρτιο
    τότε η διάμεση τετμημένη διασπά το Û σε $\hat{U}_{\text{αριστερά}}$ και $\hat{U}_{\text{δεξιά}}$
    αλλιώς η διάμεση τεταγμένη διασπά το Û σε $\hat{U}_{\text{αριστερά}}$ και $\hat{U}_{\text{δεξιά}}$
  $v \leftarrow$ ένας νέος κόμβος που αποθηκεύει την ευθεία διάσπασης $\mathcal{L}$
  $v_{\text{αριστερός}} \leftarrow$ Κατασκευή2DΔένδρου($\hat{U}_{\text{αριστερά}}$, 1+βάθος)
  $v_{\text{δεξιός}} \leftarrow$ Κατασκευή2DΔένδρου($\hat{U}_{\text{δεξιά}}$, 1+βάθος)
  return v
τέλος

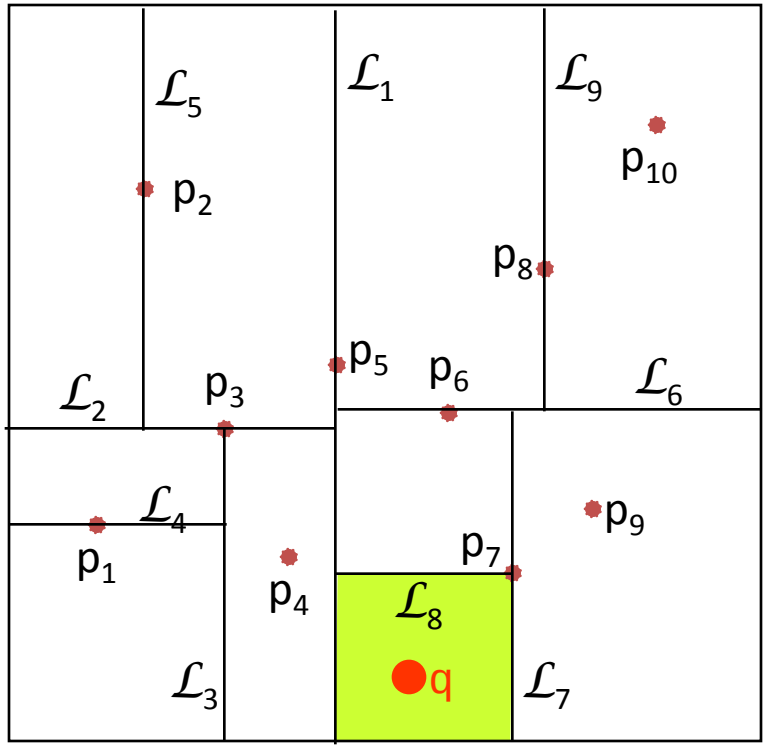
$T(n) = 2T(n/2) + O(n) = O(n\log n)$ χρόνος.

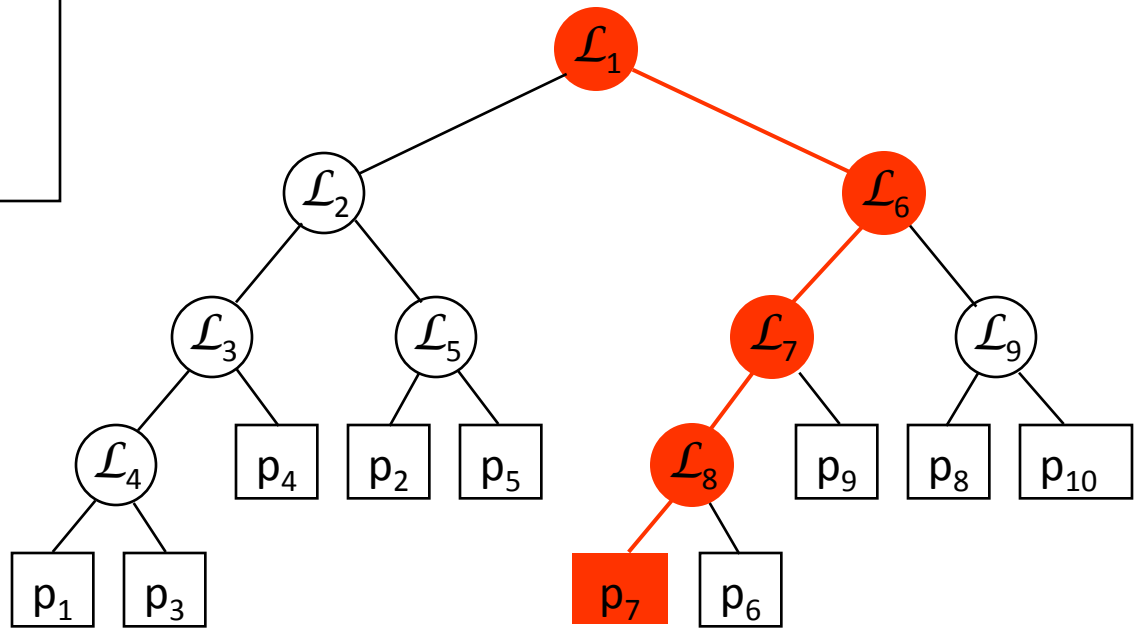
Παράδειγμα του 2D-δένδρου





Ερώτηση Σημείου σε 2D-Δένδρο





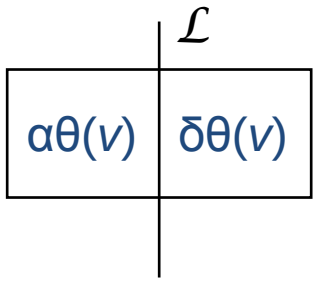
Περιοχές Κορυφών σε 2D-Δένδρο

περιοχή(v) = ορθογώνια περιοχή (πιθανώς μη φραγμένη) που καθορίζεται από το υποδένδρο με ρίζα το v.

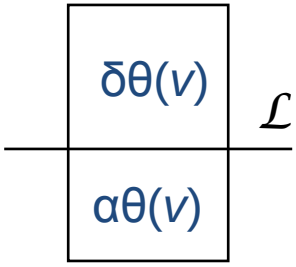
περιοχή(root[$\mathcal{T}$]) = $(-\infty : +\infty) \times (-\infty : +\infty)$

Έστω περιοχή(v) = $\langle x_1 : x_2 \rangle \times \langle y_1 : y_2 \rangle$
Ποιες είναι οι περιοχή(αθ(v)) και περιοχή(δθ(v));

Με διάσπαση ως προς x:
περιοχή(αθ(v)) = $\langle x_1 : x(\mathcal{L})] \times \langle y_1 : y_2 \rangle$
περιοχή(δθ(v)) = $(x(\mathcal{L}) : x_2 \rangle \times \langle y_1 : y_2 \rangle$



Με διάσπαση ως προς y:
περιοχή(αθ(v)) = $\langle x_1 : x_2 \rangle \times \langle y_1 : y(\mathcal{L})]$
περιοχή(δθ(v)) = $\langle x_1 : x_2 \rangle \times (y(\mathcal{L}) : y_2 \rangle$

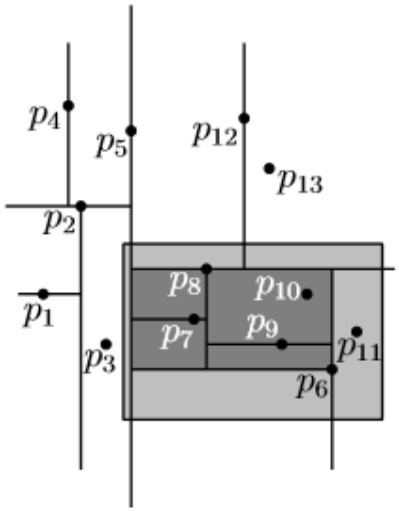


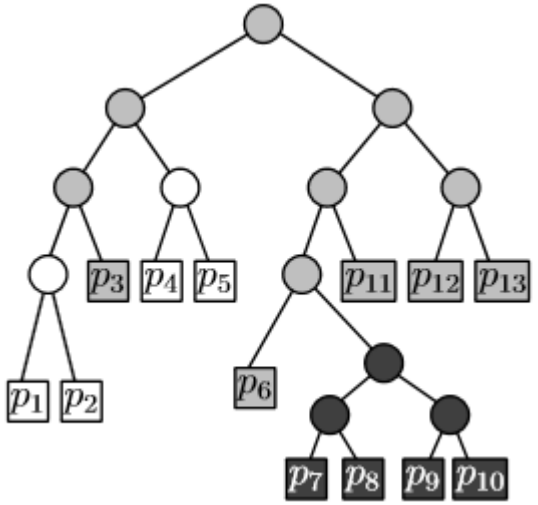
Εκτασιακή Ερώτηση σε 2D-Δένδρο

Για την περιοχή: R = $[x_1 : x_2] \times [y_1 : y_2]$ τρέξε **Διερεύνηση2DΔέντρου (ρίζα[$\mathcal{T}$] , R)**

ΑΛΓΟΡΙΘΜΟΣ Διερεύνηση2DΔέντρου (v , R)
1. **αν** v είναι φύλλο **και αν** p(v) ∈ R : ανέφερε p(v)
2. **αλλιώς αν** περιοχή(αθ(v)) ⊆ R
3. **τότε** **ΑναφοράΥποδέντρου(αθ(v))**
4. **αλλιώς αν** περιοχή(αθ(v)) ∩ R ≠ ∅
5. **τότε Διερεύνηση2DΔέντρου (αθ(v) ,R)**
6. **αν** περιοχή(δθ(v)) ⊆ R
7. **then** **ΑναφοράΥποδέντρου(δθ(v))**
8. **else if** περιοχή(δθ(v)) ∩ R ≠ ∅
9. **then Διερεύνηση2DΔέντρου (δθ(v) ,R)**

τέλος





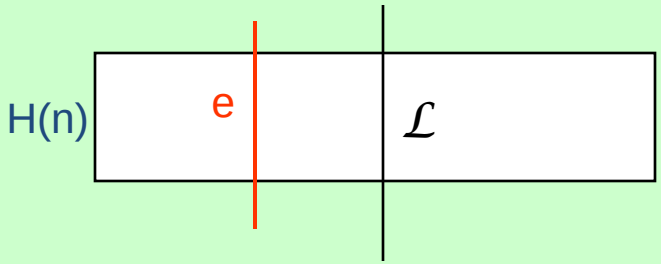
- Η **περιοχή(v)** μπορεί είτε να αποθηκεύεται σε κάθε κόμβο είτε να υπολογίζεται επί τόπου (βλ. προηγούμενη διαφάνεια)

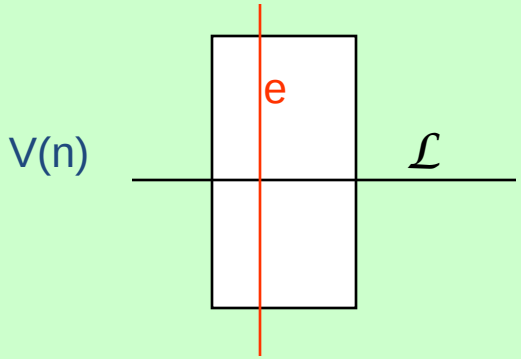
- Η **ΑναφοράΥποδέντρου(v)** είναι μια απλή in-order αναφορά των κόμβων που είναι απόγονοι του v και απαιτεί γραμμικό χρόνο

Χρόνος Απάντησης Εκτασιακού Ερωτήματος

- Ανάλυση μέσω πλήθους άσπρων, γκρίζων και μαύρων κόμβων
- Οι άσπροι κόμβοι δεν επηρεάζουν τον χρόνο εκτέλεσης
- Πλήθος μαύρων κόμβων = K
- Μένει να υπολογιστεί το πλήθος των γκρίζων κόμβων

- Αν αντί για ορθογωνικό εκτασιακό ερώτημα, έχουμε μια κατακόρυφη ευθεία, πόσοι κόμβοι θα είναι γκρίζοι στο δέντρο;
- Ορίζουμε αναδρομικά το πλήθος των γκρίζων κόμβων (προσοχή: η αναδρομή πρέπει να έχει ίδια αρχική κατάσταση –π.χ. κατακόρυφη ευθεία- οπότε μελετούμε τι συμβαίνει σε βάθος δύο επιπέδων)





$$\left.\begin{cases} H(n) = V(n/2)+1 \\ V(n) = 2H(n/2)+1 \\ (H(1) = V(1) = 1) \end{cases}\right\} \Rightarrow \left\{\begin{matrix} H(n) = 2H(n/4)+2 \\ V(n) = 2V(n/4)+3 \end{matrix}\right\} \Rightarrow \left\{\begin{matrix} H(n) = 3\sqrt{n}-2 \\ V(n) = 4\sqrt{n}-3 \end{matrix}\right.$$

$$\Rightarrow \text{RunningTime} = O(K+\sqrt{n}).$$

Πολυπλοκότητες dD-Δέντρου

2D-Δέντρο

- Χρόνος Ερώτησης : $O(K + \sqrt{n})$ χειρ. περίπτωση, $O(K + \log n)$ μέση περίπτωση
- Χρόνος κατασκευής : $O(n \log n)$
- Χώρος : $O(n)$

dD-Δέντρο d-διαστάσεις

Ακολουθούμε την ίδια διαδικασία χωρίζοντας σε κάθε επίπεδο με βάση μια από τις d διαστάσεις.

- Χρόνος Ερώτησης: $O(dK + d\, n^{1-1/d})$
- Χρόνος Κατασκευής: $O(d\, n \log n)$
- Χώρος: $O(dn)$

Range Trees

2D-Tree

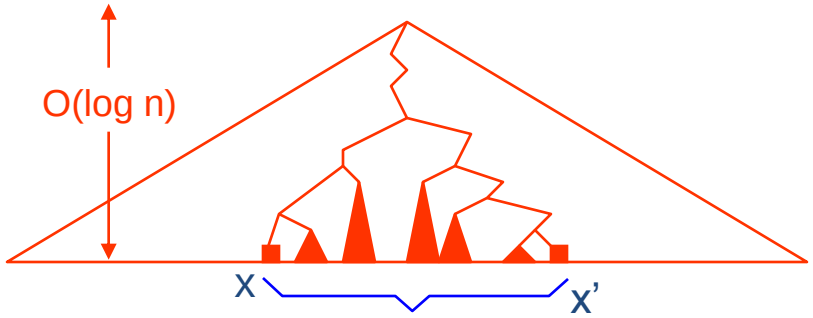
- Query Time: $O(K + \sqrt{n})$
- Construction Time: $O(n \log n)$
- Space: $O(n)$

→

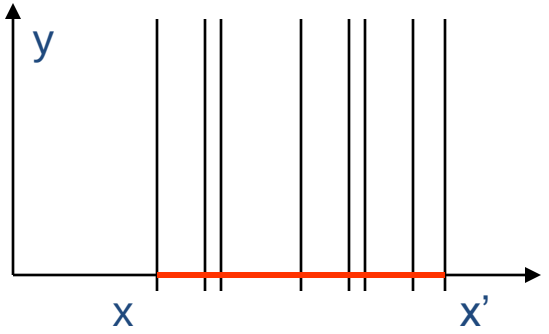
2D Range Tree

- Query Time: $O(K + \log^2 n)$
  $O(K + \log n)$ by Fractional Cascading
- Construction Time: $O(n \log n)$
- Space: $O(n \log n)$

Range $R = [x : x'] \times [y : y']$
1D Range Tree on x-coordinates:



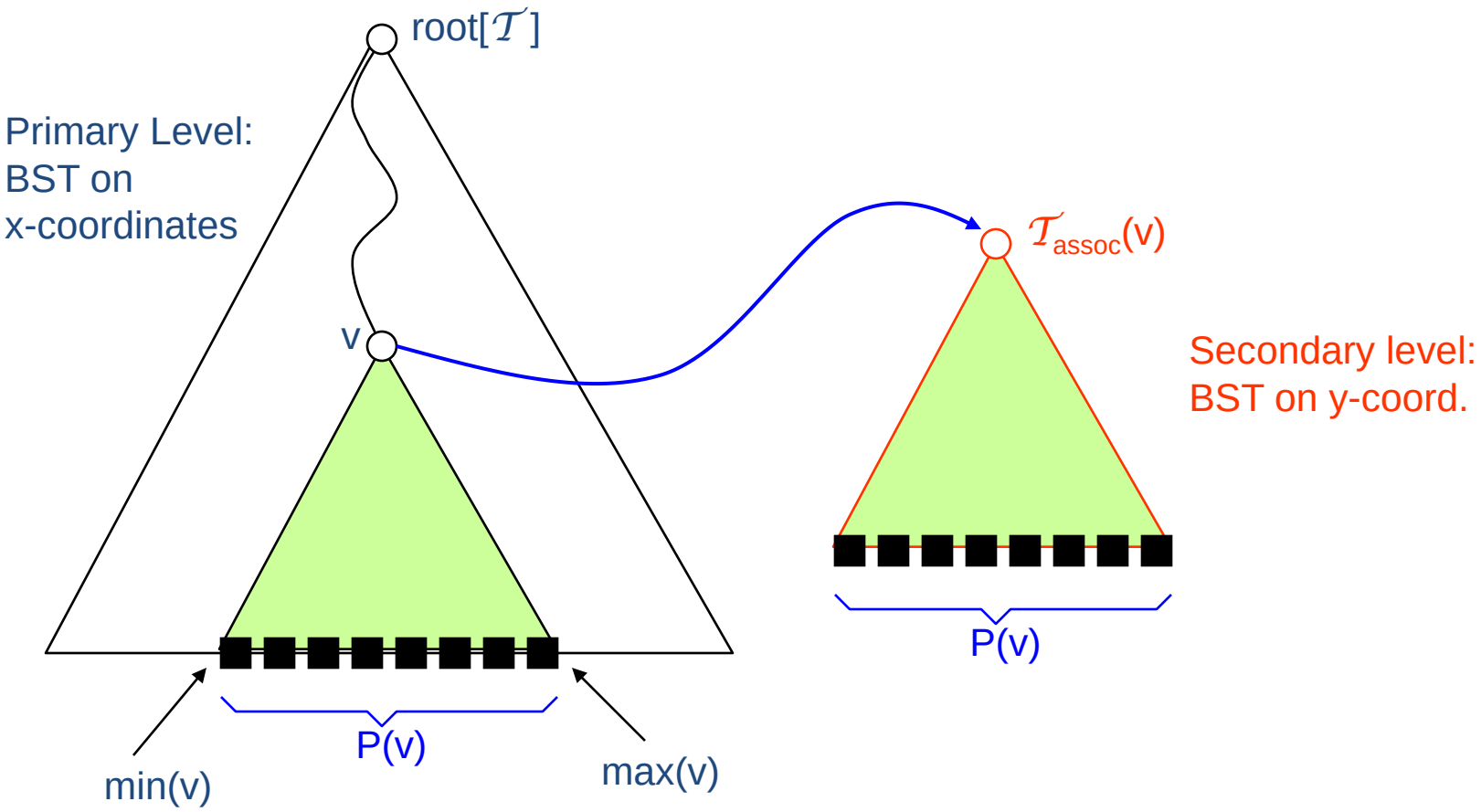
O(log n) canonical sub-trees



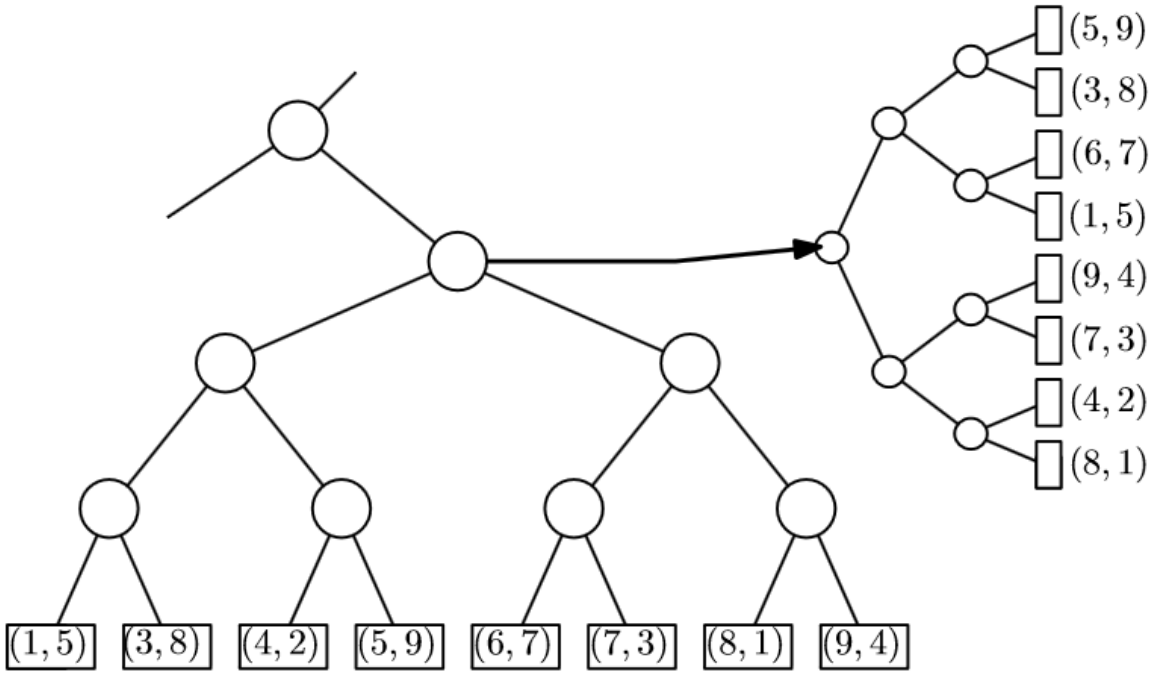
Each x-range [x : x'] can be expressed as the disjoint union of O(log n) canonical x-ranges.

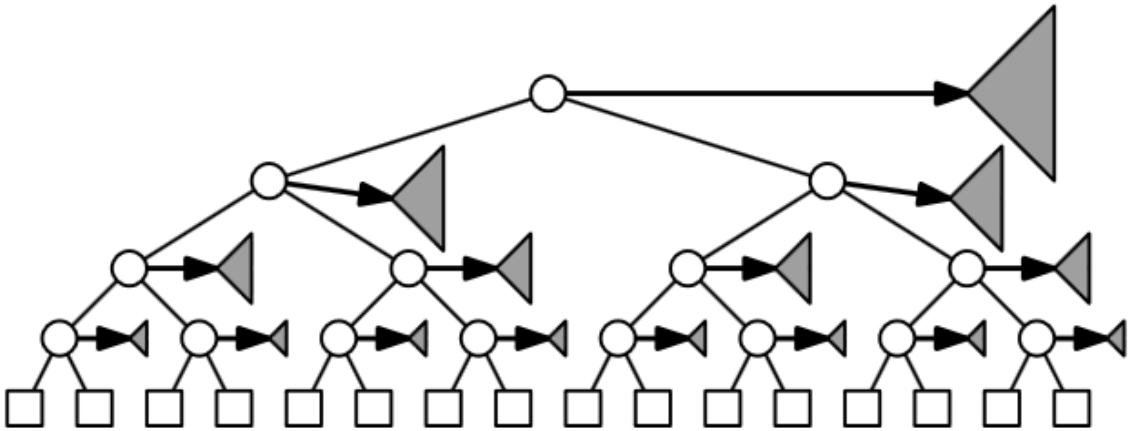
Range Trees

2-level data structure:



Εναλλακτικές Όψεις των Εκτασιακών Δέντρων





Range Tree Construction

ALGORITHM Build 2D Range Tree (P)

Input: $P = \{ p_1, p_2, \ldots, p_n \} \subseteq \Re^2$, $P = (P_x, P_y)$
represented by pre-sorted list on x (named P_x) and on y (named P_y).

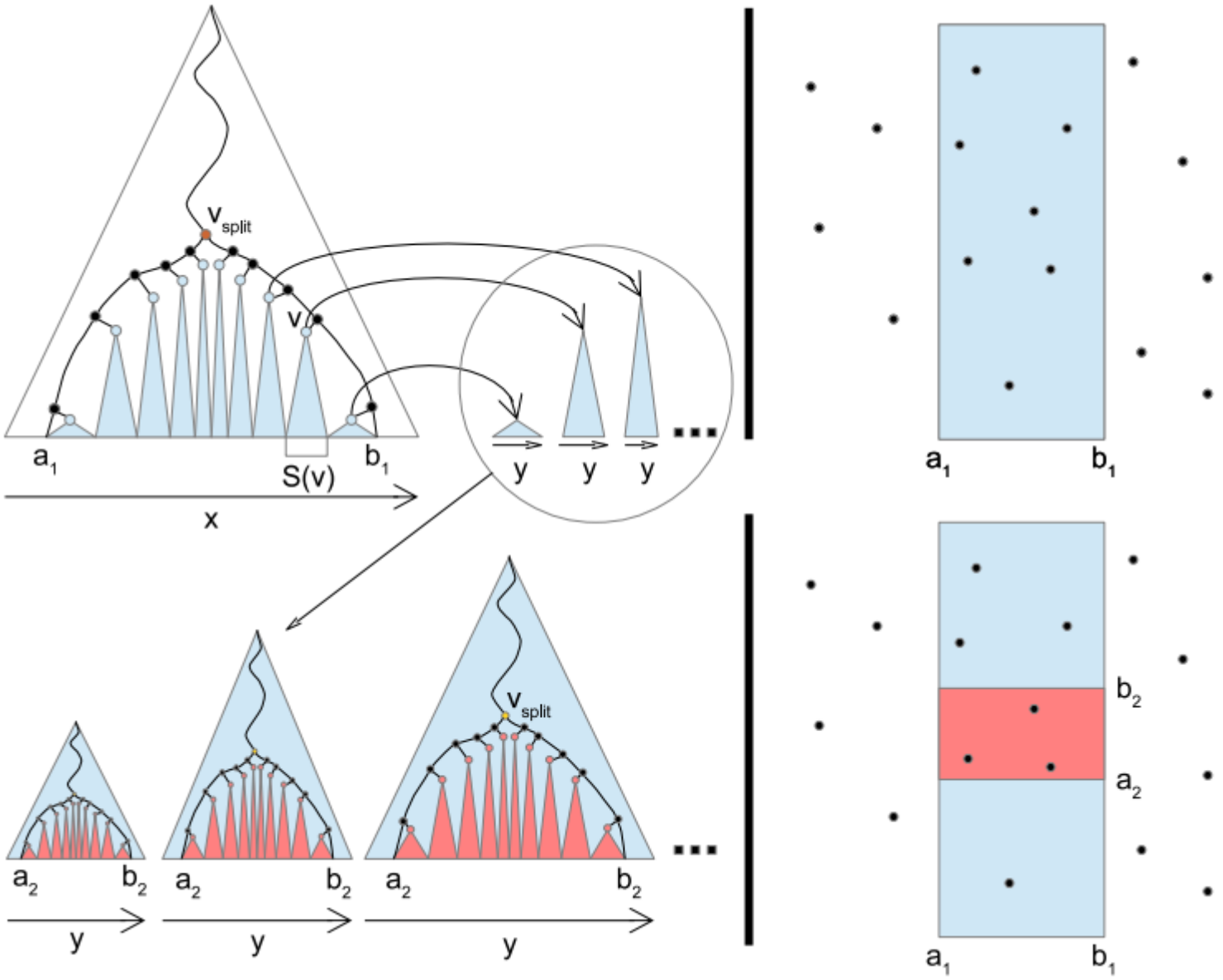
Output: pointer to the root of 2D range tree for P.

Construct $\mathcal{T}_{assoc}$, bottom up, based on P_y,
but store in each leaf the points, not just their y-coordinates.

if |P| > 1
then do
$P_{left} \leftarrow \{ p \in P \mid p_x \leq x_{med} \text{ of } P \}$ (* both lists P_x and P_y should split *)
$P_{right} \leftarrow \{ p \in P \mid p_x > x_{med} \text{ of } P \}$
lc(v) $\leftarrow$ Build 2D Range Tree (P_{left})
rc(v) $\leftarrow$ Build 2D Range Tree (P_{right})
od
min(v) $\leftarrow$ min (P_x); max(v) $\leftarrow$ max(P_x)
$\mathcal{T}_{assoc}(v) \leftarrow \mathcal{T}_{assoc}$
return v
end

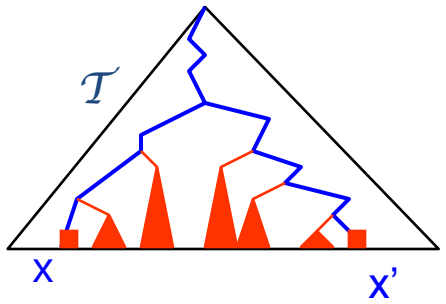
T(n) = 2 T(n/2) + O(n) = O(n log n) time.
This includes time for pre-sorting.

Ερώτημα σε 2D Εκτασιακό Δέντρο



2D Range Query

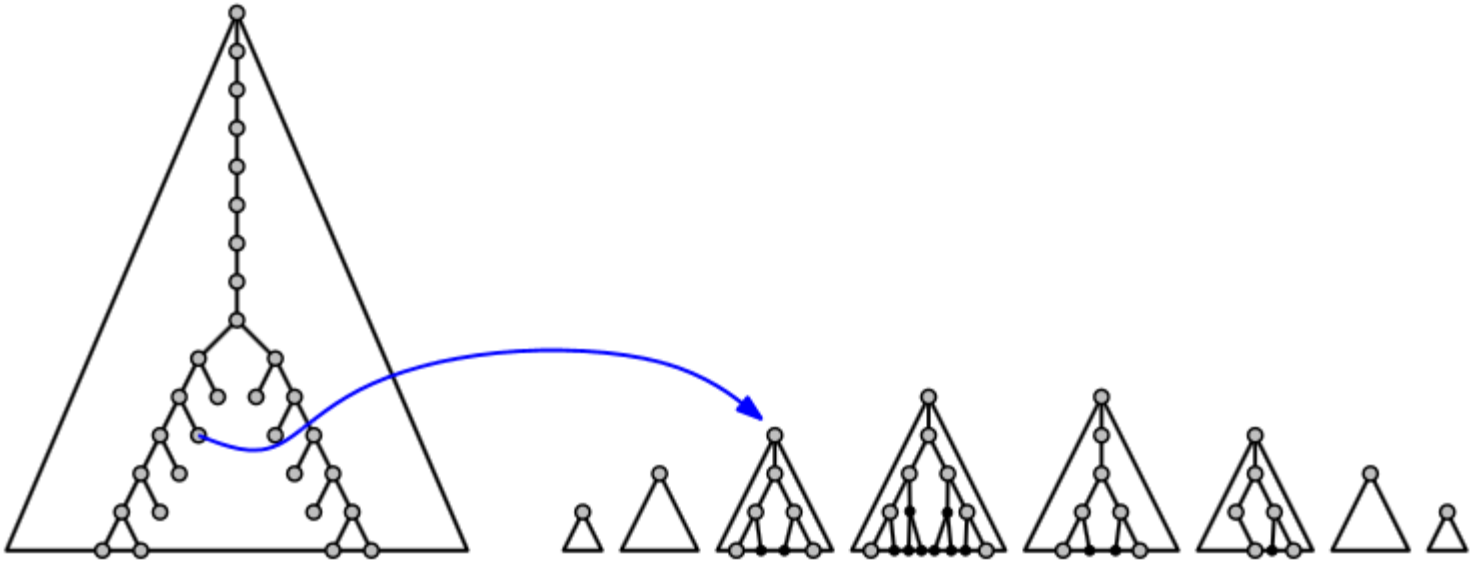
```
ALGORITHM  2DRangeQuery ( v, [x : x'] × [y : y'] )
1.    if   x ≤ min(v) & max(v) ≤ x'
2.       then  1DRangeQuery (T_assoc(v) , [y : y'] )
3.       else if  v is not a leaf  do
4.              if   x ≤ max(lc(v))
5.                 then  2DRangeQuery ( lc(v), [x : x'] × [y : y'] )
6.              if   min(rc(v)) ≤ x'
7.                 then  2DRangeQuery ( rc(v), [x : x'] × [y : y'] )
8.        od
end
```



- Line 2 called at roots of red canonical sub-trees, a total of O(log n) times.
  Each call takes $O(K_v + \log |\mathcal{T}_{assoc}(v)|) = O(K_v + \log n)$ time.
- Lines 5 & 7 called at blue shoulder paths. Total cost O(log n).
- Total Query Time = $O(\log n + \sum_v (K_v + \log n)) = O(\sum_v K_v + \log^2 n) = O(K + \log^2 n)$.

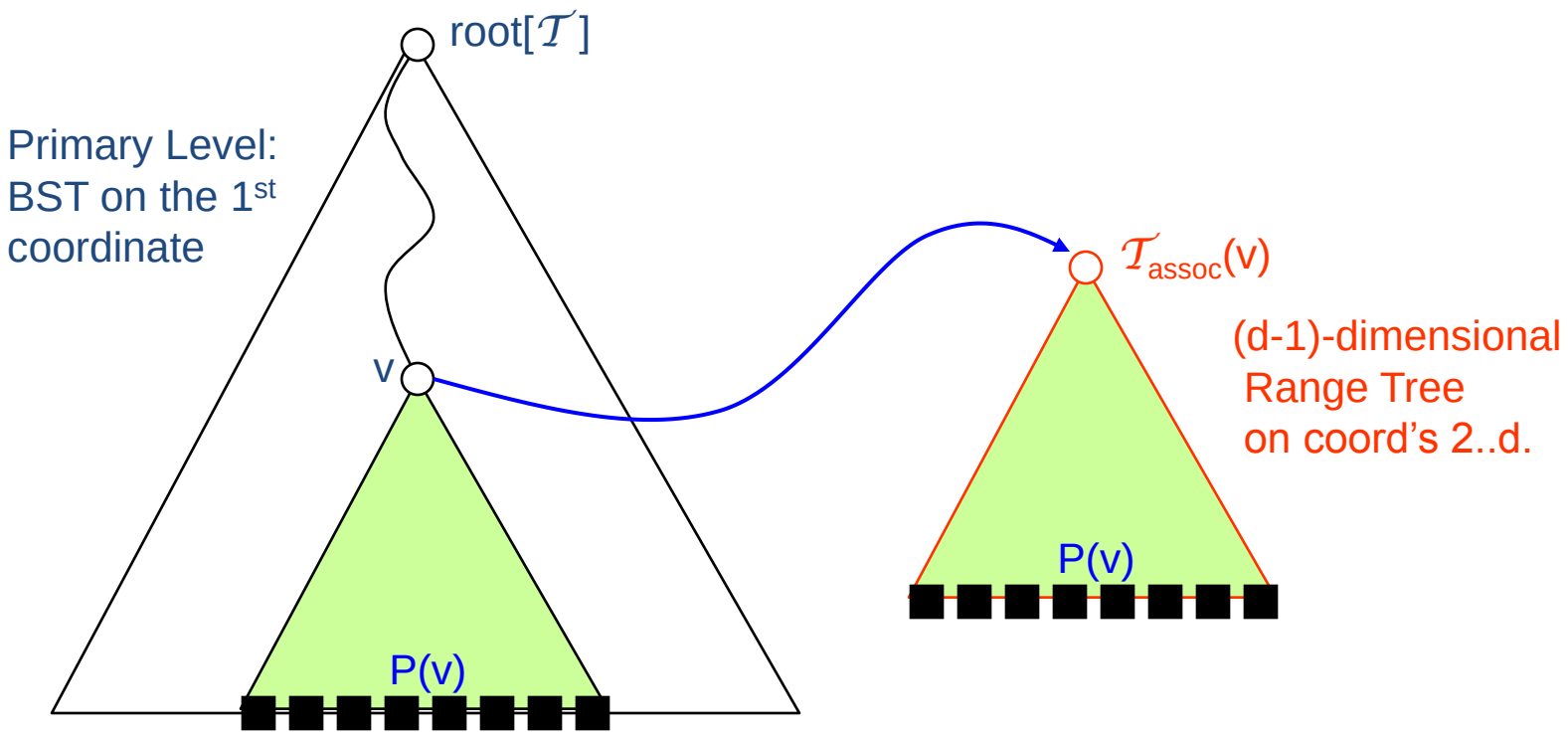
Query Time:	$O(K + \log^2 n)$	will be improved to O(K + log n) by Fractional Cascading
Construction Time:	O(n log n)	
Space:	O(n log n)	

Απεικόνιση Χρόνου Ερώτησης

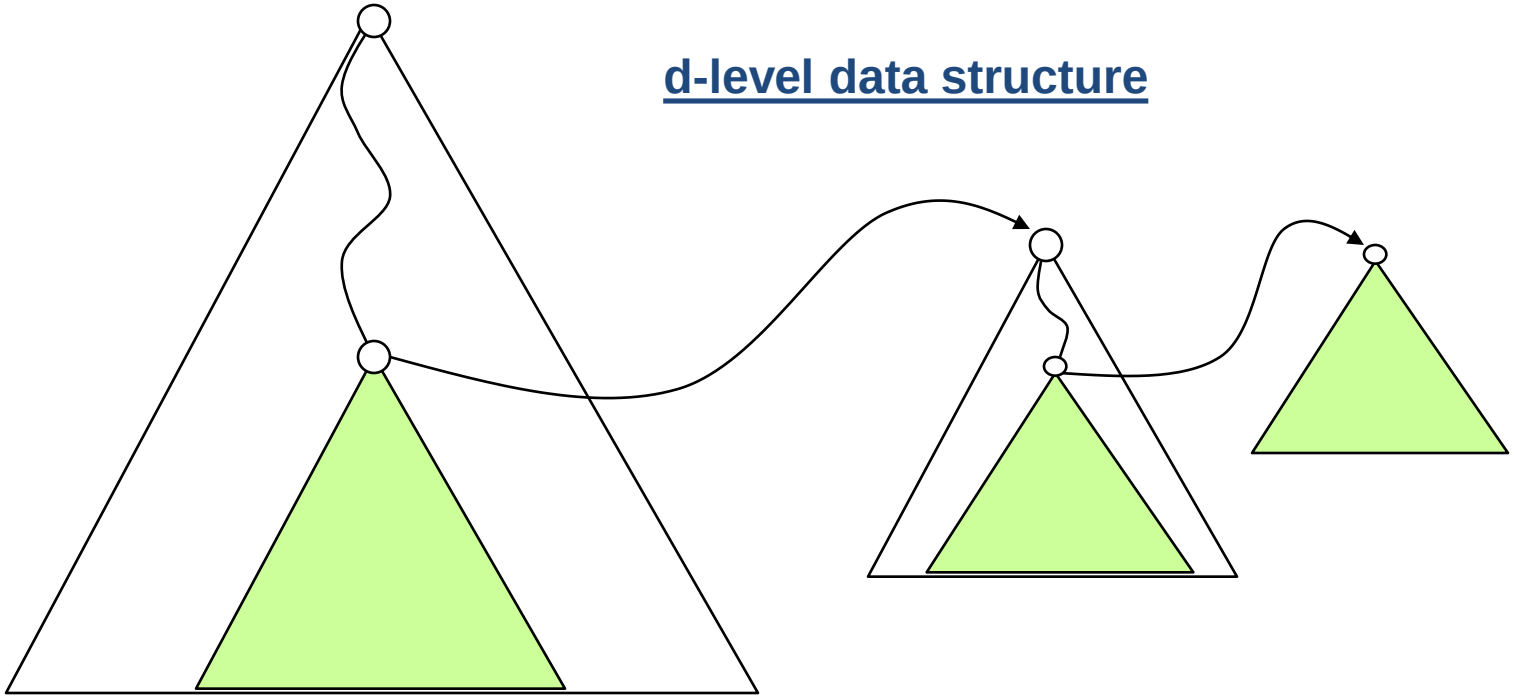


Higher Dimensional Range Trees

$P = \{ p_1, p_2, \ldots, p_n \} \subseteq \Re^d, \quad p_i = (x_{i1}, x_{i2}, \ldots, x_{id}), \; i=1..n.$



Higher Dimensional Range Trees



Higher Dimensional Range Trees

Query Time: $Q_d(n) = O(K + \log^d n)$ improved to $O(K + \log^{d-1} n)$ by Frac. Casc.

Construction Time: $T_d(n) = O(n \log^{d-1} n)$

Space: $S_d(n) = O(n \log^{d-1} n)$

$$\left\{\begin{array}{c} T_d(n) = 2T_d\left(\frac{n}{2}\right) + T_{d-1}(n) + O(n) \\ T_2(n) = O(n \log n) \end{array}\right\} \Rightarrow T_d(n) = O(n \log^{d-1} n)$$

$$\left\{\begin{array}{c} S_d(n) = 2S_d\left(\frac{n}{2}\right) + S_{d-1}(n) + O(1) \\ S_2(n) = O(n \log n) \end{array}\right\} \Rightarrow S_d(n) = O(n \log^{d-1} n)$$

$$\left\{\begin{array}{c} Q_d(n) = O(K) + \hat{Q}_d(n) \\ \hat{Q}_d(n) = O(\log n) + O(\log n) \cdot \hat{Q}_{d-1}(n) \\ \hat{Q}_2(n) = O(\log^2 n) \end{array}\right\} \Rightarrow \left\{\begin{array}{c} \hat{Q}_d(n) = O(\log^d n) \\ Q_d(n) = O(K + \log^d n) \end{array}\right.$$

General Sets of Points

What if 2 points have the same coordinate value at some coordinate axis?

- **Composite Numer Space:** (lexicographic order)

  $(a,b) \rightarrow (a \mid b)$

  $(a \mid b) < (a' \mid b') \Leftrightarrow a<a' \text{ or } (a=a' \ \& \ b<b')$

- $p = (p_x , p_y) \rightarrow p' = ((p_x \mid p_y) , (p_y \mid p_x))$

  $R=[x:x'] \times [y:y'] \rightarrow R' = [(x \mid -\infty) : (x' \mid +\infty)] \times [(y \mid -\infty) : (y' \mid +\infty)]$

- $p \in R \quad \Leftrightarrow \quad p' \in R'$

  $x \le p_x \le x' \qquad (x \mid -\infty) \le ((p_x \mid p_y) \le (x' \mid +\infty)$

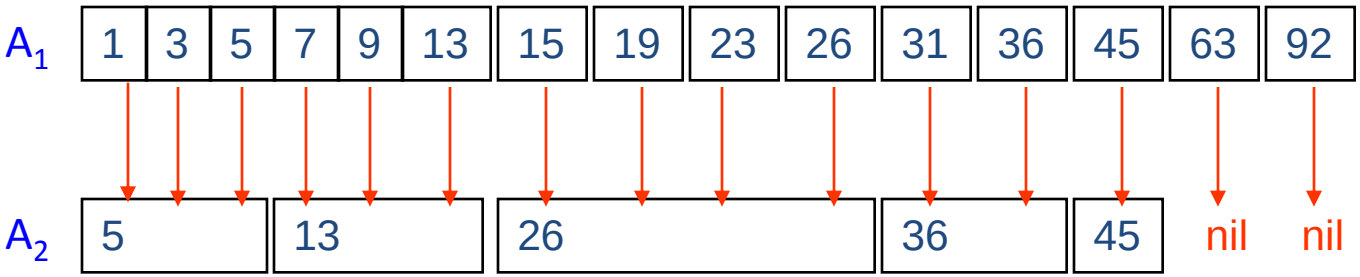
  $\& \ y \le p_y \le y' \qquad \& \ (y \mid -\infty) \le ((p_y \mid p_x) \le (y' \mid +\infty)$

- Note: no two points in the composite space have the same value at any coordinate (unless they are identical points).

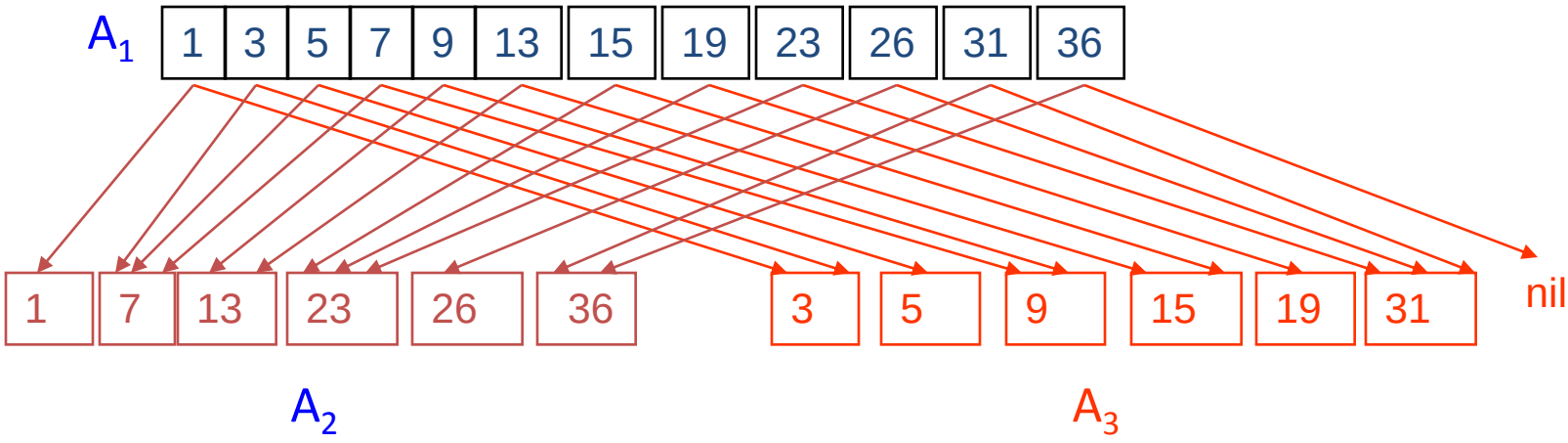
Fractional Cascading (Κλασματική Επαλληλία)

IDEA: Save repeated cost of binary search in many sorted lists for the same range [y : y'] if the list contents for one are a subset of the other.

- $A_2 \subseteq A_1$
- Binary search for y in A_1 to get to $A_1[i]$.
- Follow pointer to A_2 to get to $A_2[j]$.
- Now walk to the right in each list.

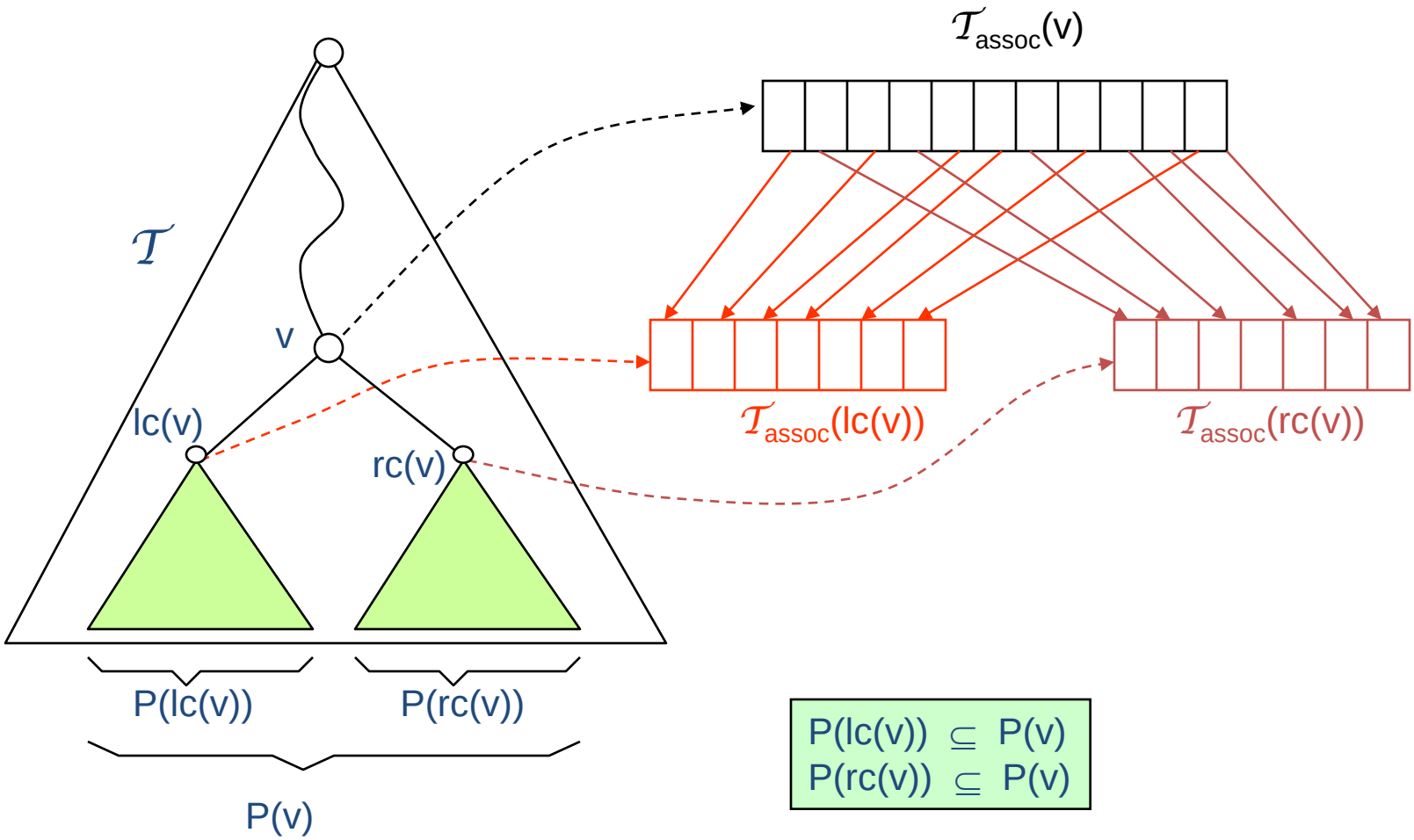


Fractional Cascading



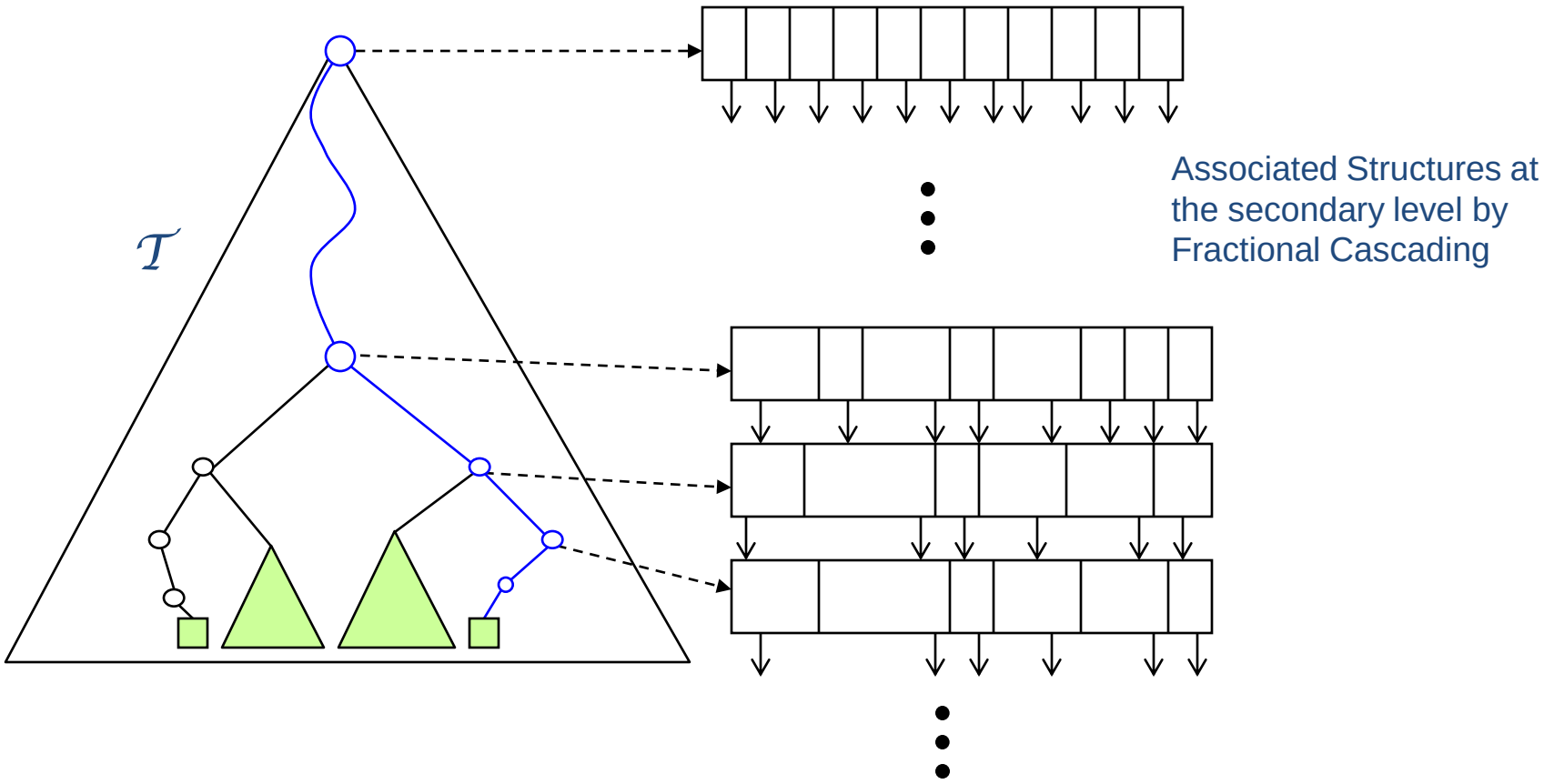
- $A_2 \subseteq A_1$, $A_3 \subseteq A_1$.
- No binary search in A_2 and A_3 is needed.
- Do binary search in A_1.
- Follow blue and red pointers from there to A_2 and A_3.
- Now we have the starting point in each sorted list. Walk to the right & report.

Layered 2D Range Tree



$P(lc(v)) \subseteq P(v)$
$P(rc(v)) \subseteq P(v)$

Layered 2D Range Tree



Layered 2D Range Tree (by Fractional Cascading)

Query Time:

$Q_2(n) = O(\log n + \sum_v (K_v + \log n)) = O(\sum_v K_v + \log^2 n) = O(K + \log^2 n)$

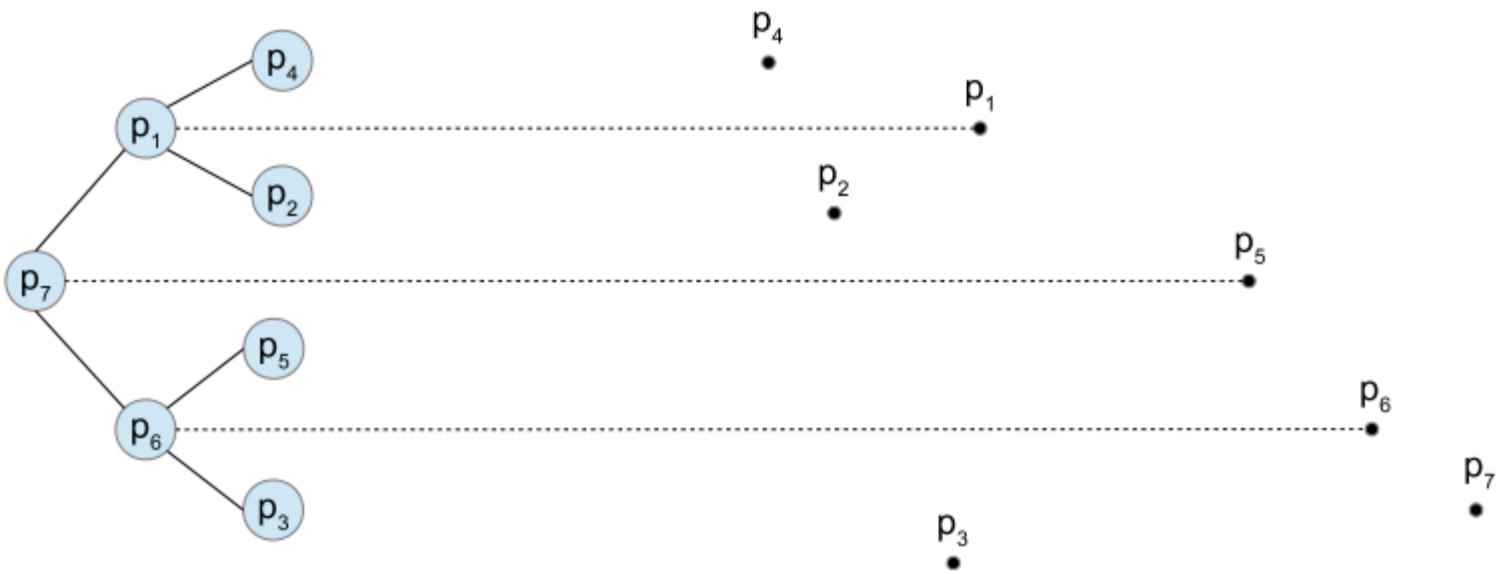
improves to:

$Q_2(n) = O(\log n + \sum_v (K_v + 1)) = O(\sum_v K_v + \log n) = O(K + \log n)$.

For d-dimensional range tree query time improves to:

$$\left\{ \begin{array}{c} Q_d(n) = O(K) + \hat{Q}_d(n) \\ \hat{Q}_d(n) = O(\log n) + O(\log n) \cdot \hat{Q}_{d-1}(n) \\ \hat{Q}_2(n) = O(\log n) \end{array} \right\} \Rightarrow Q_d(n) = O(K + \log^{d-1} n)$$

Priority Search Trees



$Q = [a_1, \infty) \times [a_2, b_2]$

$D_{left} = \{p = (x_i, y_i) \in D \setminus \{x_{\max}\} : y_i \leqslant y_{mid}\}$

$D_{right} = \{p = (x_i, y_i) \in D \setminus \{x_{\max}\} : y_i > y_{mid}\}$

Create a node v and store y_{mid} as the node's key and x_{max} in the node's additional information field.

$T(n) = 2T(n/2) + O(n) = O(n \log n)$.