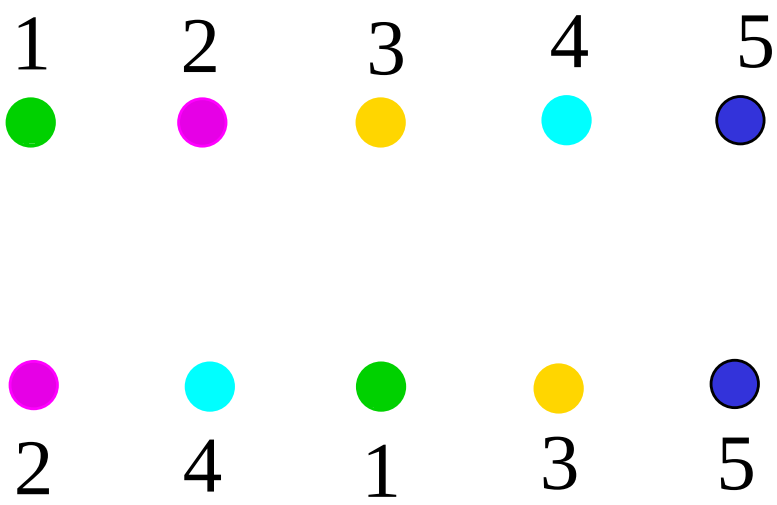
ΜΕΤΡΩΝΤΑΣ ΑΝΑΣΤΡΟΦΕΣ

Δίνονται: N αντικείμενα και δύο διαφορετικές κατατάξεις αυτών των αντικειμένων.

Στόχος: Να ορίσουμε ένα μέτρο σύγκρισης για αυτές τις κατατάξεις .



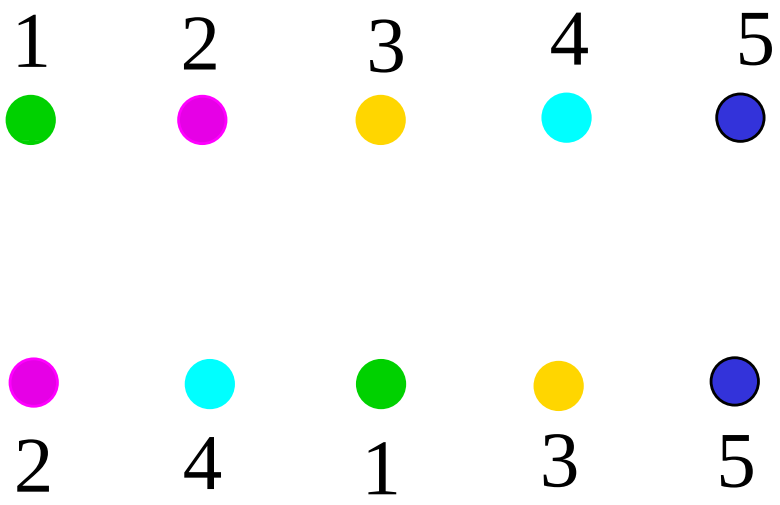
Εφαρμογές: Βαθμολόγηση ταινιών, βιβλίων, κ.ο.κ. από πολλούς χρήστες, Meta-search tools - εργαλεία αναζήτησης που συνδυάζουν αποτελέσματα από μερικές διαφορετικές μηχανές αναζήτησης.

ΔΙΑΤΥΠΩΣΗ ΜΕ ΑΡΙΘΜΟΥΣ

- Μας δίνεται μια μετάθεση $B = \{a_1, a_2, \ldots, a_n\}$ των αριθμών $A = \{1, 2, \ldots, n\}$
- Θέλουμε να υπολογίσουμε 'πόσο διαφέρει' η A από την B.

Ενα μέτρο σύγκρισης: Πλήθος των αναστροφών στην ακολουθία B.

Αναστροφή: Ζεύγος (a_i, a_j) τέτοιο ώστε $i < j$ και $a_i > a_j$.

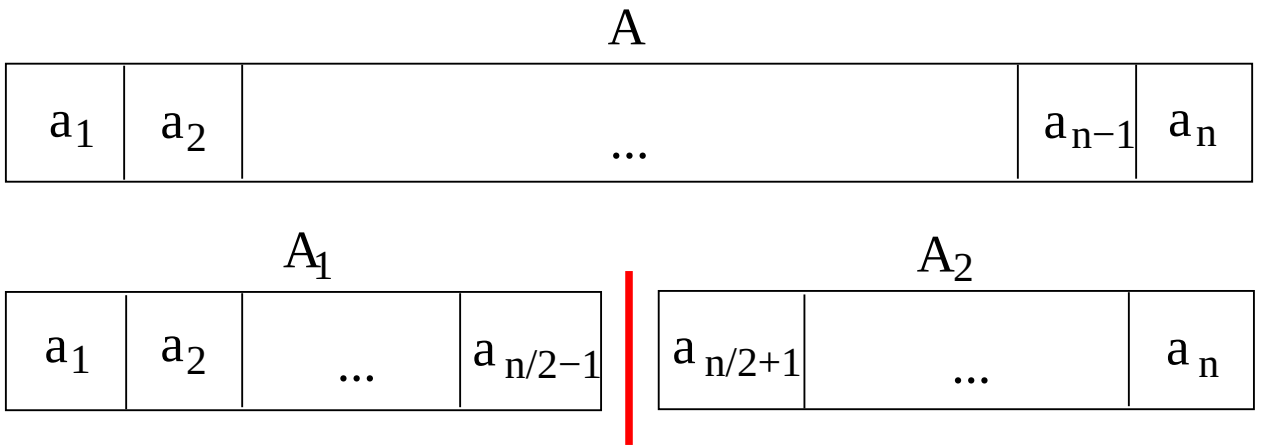


Παράδειγμα: $(2, 1),\ (4, 1),\ (4, 3)$

Απλός αλγόριθμος: Φτιάξε όλα τα δυνατά ζεύγη και μέτρησε πόσα από αυτά αποτελούν αναστροφές.

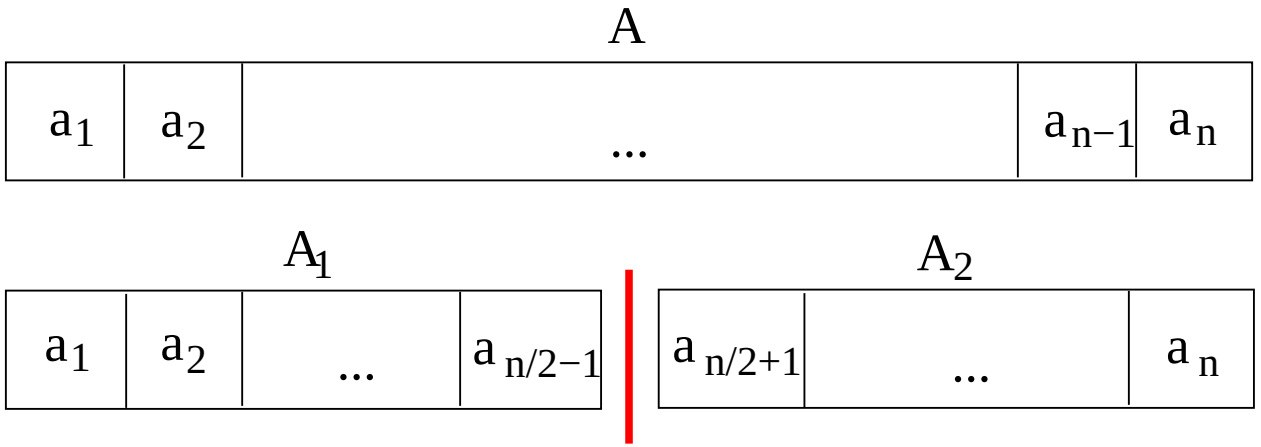
Πολυπλοκότητα: $O(n^2)$

Μπορούμε καλύτερα; ΝΑΙ - $O(nlogn)$



Αναδρομικος αλγόριθμος:

- Χώρισε της ακολουθία A στη μέση: A_1, A_2.
- Υπολόγησε αναδρομικά τον αριθμό των αναστροφών στο A_1 και A_2, έστω n_1, n_2
- Συνολικός αριθμός αναστροφών: $n_1 + n_2 + n_3$, όπου n_3=πλήθος των (a_i, a_j) τ.ω. $a_i \in A_1$, $a_j \in A_2$ και $a_i > a_j$



Αναδρομικος αλγόριθμος:

Προσοχή: Χρόνος που χρειάζεται για υπολογισμό του n_3 πρέπει να είναι $O(n)$ για να επιτύχουμε την συνολική πολυπλοκότητα $O(nlogn)$!

Ιδέα υλοποίησης: Ταξινόμησε αναδρομικά τις A_1 και A_2.

Συγχωνεύουμε τις λίστες A_1, A_2 και υπολογίζουμε το n_3:

Είσοδος: 2 ταξινομημένες λίστες A_1, A_2 με n_1, n_2 αναστροφές αντίστοιχα,

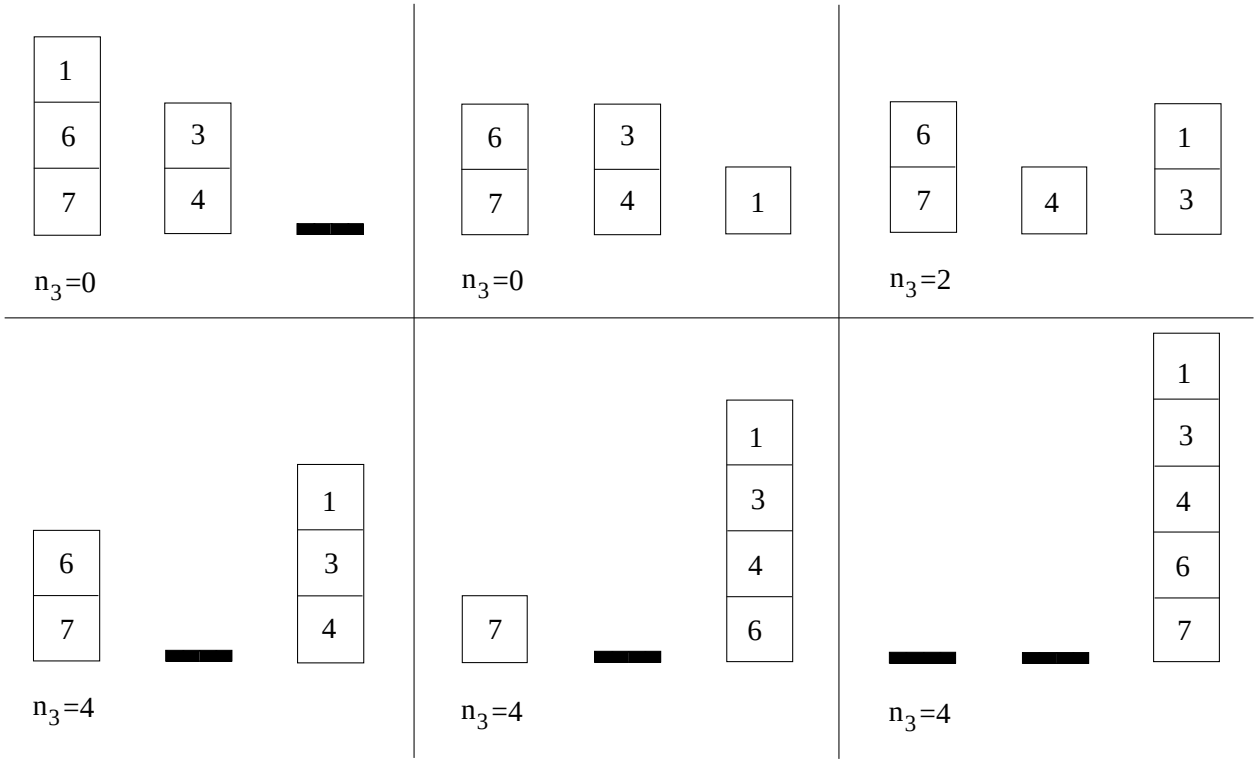
Έξοδος: Μια ταξινομημένη λίστα A η οποία περιέχει όλα τα στοιχεία των A_1 και A_2, και ο αριθμός n_3 των αναστροφών (a_i, a_j) τ.ω. $a_i \in A_1$, $a_j \in A_2$

Merge-CountInversions(A_1, A_2, A, n_3)

while υπάρχουν ακόμη στοιχεία στην A_1 ή στην A_2 **do**

- Σύγκρινε τα 'πρώτα' στοιχεία της A_1 και της A_2
- Τοποθέτησε το μικρότερο απο τα δύο στο τέλος της A
- Αν αυτό το στοιχείο ανήκει στην A_2, αύξησε n_3 κατά το πλήθος των στοιχείων της A_1
- Διέγραψε το στοιχείο απο την αρχική λίστα

Παράδειγμα:



Μετράμε της αναστροφές:

Count-Inversions$(A, p, r, newA, n)$
if $p < q$ **then**

- $q \leftarrow \lfloor (p+r)/2 \rfloor$
- Count-Inversions(A, p, q, A_1, n_1)
- Count-Inversions$(A, q+1, r, A_2, n_2)$
- Merge-CountInversions$(A_1, A_2, newA, n_3)$
- $n \leftarrow n_1 + n_2 + n_3$

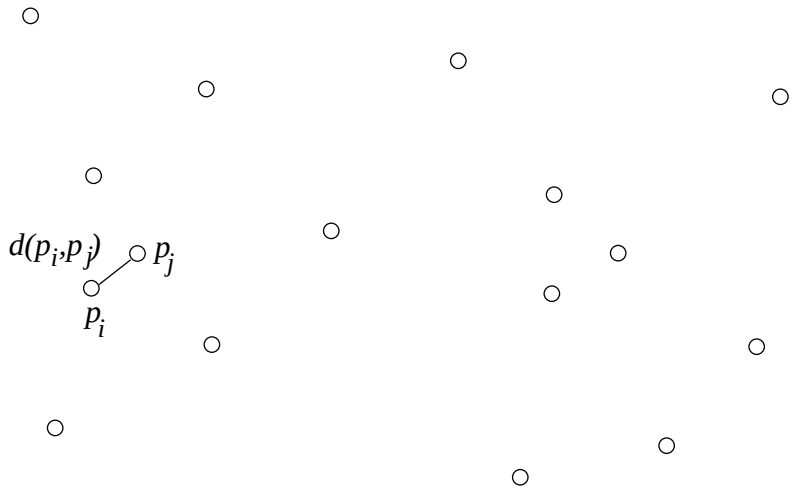
Πολυπλοκότητα: $T(n) = \begin{cases} \Theta(1) & \text{εάν } n \leq 1 \\ 2T(\frac{n}{2}) + \Theta(n) & \text{εάν } n > 1 \end{cases}$

Και άρα: $T(n) = O(nlogn)$

ΖΕΥΓΟΣ ΠΛΗΣΙΕΣΤΕΡΩΝ ΣΗΜΕΙΩΝ

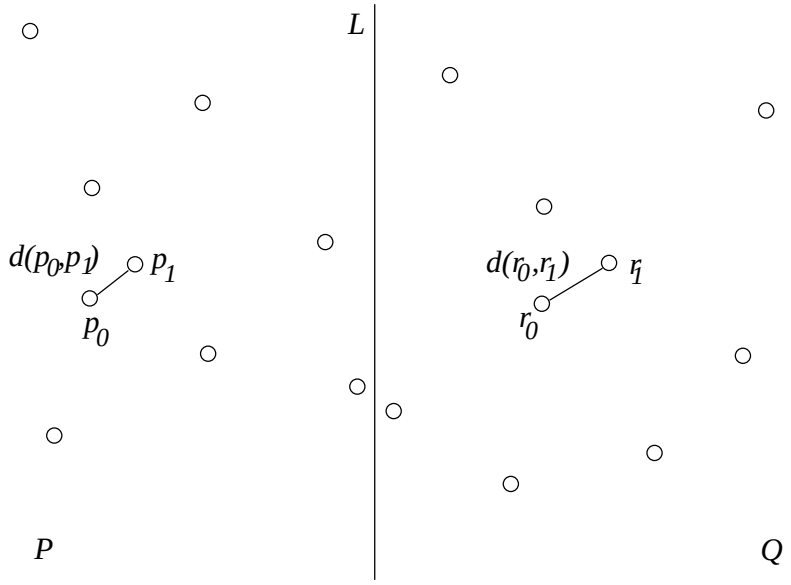
Δίνονται: n σημεία στο επίπεδο: $P = \{p_1, p_2, \ldots, p_n\}$, όπου το σημείο p_i έχει συντεταγμένες (x_i, y_i). Συμβολίζουμε με $d(p_i, p_j)$ την Ευκλείδια απόσταση μεταξύ των σημείων p_i και p_j.

Στόχος: Να βρεθούν δύο σημεία του P, p_i και p_j, που ελαχιστοποιούν την ποσότητα $d(p_i, p_j)$.



Εφαρμογές: Γραφικά, όραση υπολογιστών, γεωγραφικά συστήματα πληροφοριών, μοριακή μοντελοποίηση, κ.ο.κ.

- Τετριμμένος αλγόριθμος έχει πολυπλοκότητα $O(n^2)$.
- Μπορούμε καλύτερα;
- Ναι! $O(nlogn)$ με αναδρομή.
- Βασική ιδέα αλγορίθμου:
  - Χωρίζουμε το σύνολο με μία γραμμή L σε δύο υποσύνολα, Q και R.
  - Βρίσκουμε αναδρομικά τα ζεύγοι πλησιέστερων σημείων (q_0, q_1) (r_0, r_1) στα Q και R αντίστοιχα.
  - Υποθέτουμε οτι $d^* = min\{d(q_0, q_1), d(r_0, r_1)\}$.
  - Συνδιάζουμε τις λύσεις, αναζητώντας δύο σημεία $p \in P$, $q \in Q$ τ.ω. $d(p, q) < d^*$.



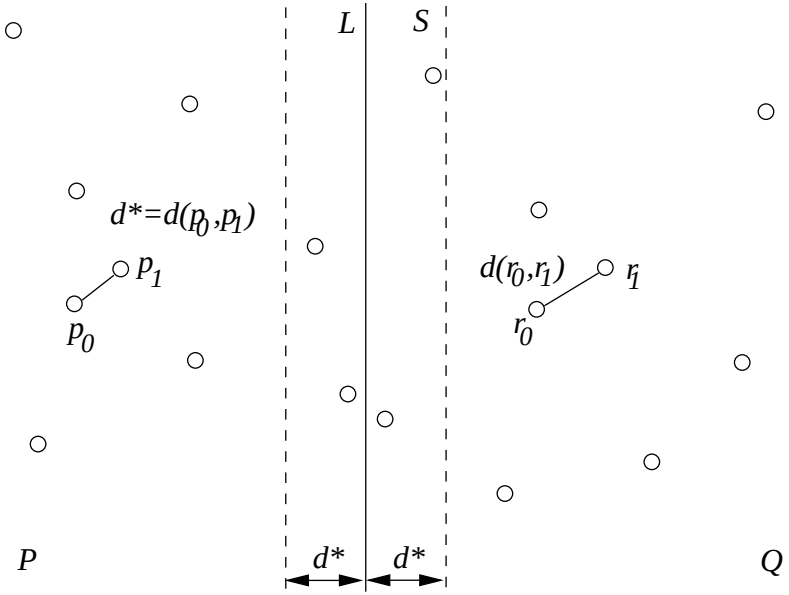
Πρόβλημα:

- Αν τα σημεία $r \in R$ και $q \in Q$ επιλέγονται πάνω σε ολόκληρα τα σύνολα R, Q, ο συνδιασμός των λύσεων χρειάζεται $\frac{n}{2} \cdot \frac{n}{2}$ πράξεις.
- Και η πολυπλοκότητα του αλγορίθμου γίνεται: $T(n) = 2T(\frac{n}{2}) + \frac{n^2}{4} \Rightarrow T = O(n^2)$.

Λήμμα 1 *Αν υπάρχουν δύο σημεία $r \in R$ και $q \in Q$ τέτοια ώστε $d(r,q) < d^*$, τότε και τα δύο σημεία απέχουν από την L λιγότερο από d^*.*

Απόδειξη: Υποθέτουμε οτι τα σημεία q, r απέχουν το ένα από το άλλο λιγότερο από d^* και ότι έχουν συντεταγμένες (q_x, q_y) και (r_x, r_y) αντίστοιχα. Έστω ότι η ευθεία L δίνεται από την εξίσωση $x = x^*$. Επειδή $q \in Q$ και $r \in R$ ισχύει: $q_x \leq x^* \leq r_x$. Αφαιρώντας q_x από τα δύο μέρη της ανισότητας $x^* \leq r_x$ έχουμε:
$x^* - q_x \leq r_x - q_x \leq d(q,r) \leq d^*$. Επίσης προσθέτοντας r_x και στα δύο μέρη τις ανισότητας $-x^* \leq -q_x$ έχουμε $r_x - x^* \leq r_x - q_x \leq d^*$. Άπό της ανισότητες $x^* - q_x \leq d^*$ και $r_x - x^* \leq d^*$ συμπεραίνουμε ότι τα σημεία q, r απέχουν από την L λιγότερο από d^*. □

Πόρισμα: Για να συνδιάσουμε τις λύσεις στο Q και R, αρκεί να κάνουμε αναζήτηση των σημείων $p \in P$, $q \in Q$ τ.ω. $d(p,q) < d^*$ σε μία λωρίδα πάχους $2d^*$ γύρω από την L.



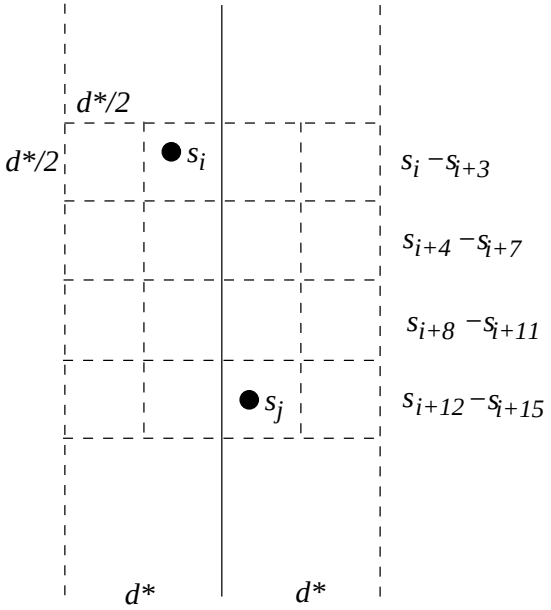
Ονομάζουμε S το σύνολο σημείων που περιέχει η λωρίδα πάχους $2d^*$ γύρω από την L.

Πρόβλημα! Αν S περιέχει όλα τα σημεία των $R,\ Q$.

Λύση:

- Χωρίζουμε την περιοχή S σε τετράγωνα πλευράς $\frac{d^*}{2}$.

- Παρατηρούμε ότι ένα κουτάκι δεν μπορεί να περιέχει δύο σημεία: αν s, s' άνηκαν στο ίδιο κουτάκι τότε $d(s, s') \leq \frac{d^*}{\sqrt{2}} \leq d^*$.
- Θεωρούμε οτι τα σημεία του S είναι ταξινομημένα ως προς y-συντεταγμένη: $S = \{s_1, s_2, \ldots, s_\}$
- Παρατηρούμε ότι τα δύο σημεία $s_i, s_j \in S$ τ.ω. $d(s_i, s_j) \leq d^*$ απέχουν το πολύ 15 θέσεις στο S. Υποθέστε ότι s_i, s_j απέχουν 16 θέσεις, τότε επειδή σε κάθε κουτάκι μπορεί να βρίσκεται μόνο ένα σημείο του S, συμπεραίνουμε ότι s_i, s_j χωρίζονται από τρείς λωρίδες πλάτους $\frac{d^*}{2}$, άρα $d(s_i, s_j) \geq 3\frac{d^*}{2}$, άτοπο.



Συμπέρασμα: Για κάθε $s_i \in S$ πρέπει να ψάξουμε μόνο 15 σημεία. Άρα $15 \cdot n$ πράξεις!

Αλγόριθμος εύρεσης δύο πλησιέστερων σημείων:

Είσοδος: 2 ταξινομημένες λίστες P_x, P_y με x, y-συντεταγμένες των n σημείων.

Έξοδος: Το ζεύγος πλησιέστερων σημείων.

Closest-Pair-Rec(P_x, P_y)

- **if** $\|P\| \leq 3$ **then** βρες τα πλησιέστερα σημεία δοκιμάζοντας όλα τα ζεύγοι
- **else**
  - Χόρισε P σε δύο σύνολα Q, R, κατασκευάζοντας Q_x, Q_y, R_x, R_y (Q περιέχει τα μισά σημεία με μικρότερη x-συντεταγμένη)
  - $(q_0, q_1) =$ *Closest-Pair-Rec*(Q_x, Q_y)
  - $(r_0, r_1) =$ *Closest-Pair-Rec*(R_x, R_y)
  - $d^* = min\{d(q_0, q_1), d(r_0, r_1)\}$
  - Υποθέτουμε ότι x^* είναι η μέγιστη x-συντεταγμένη του συνόλου Q (εκεί περνά η ευθεία L)
  - Κατασκευάζουμε σύνολο S που περιέχει σημεία $p_i = (x_i, y_i)$ των Q, R τ.ω. $|x_i - x^*| < d^*$ (έχουν απόσταση d^* από την ευθεία L)
  - Κατασκευάζουμε S_y (ταξινομημένη λίστα y-συντεταγμένων της S)
  - **For each** $s \in S_y$ υπολόγισε απόσταση του s από τα επόμενα 15 σημεία της S_y.
  - Έστω s, s' τα κοντινότερα μεταξύ όλων των σημείων του S_y που ελέγξαμε.
  - **if** $d(s, s') < d^*$ **return**(s, s^*)
  - **else if** $d(q_0, q_1) < d(r_0, r_1)$ **return**(q_0, q_1) **else return**(r_0, r_1)

Πολυπλοκότητα: $T(n) = \begin{cases} \Theta(1) & \text{εάν } n \leq 1 \\ 2T(\frac{n}{2}) + \Theta(n) & \text{εάν } n > 1 \end{cases}$

Και άρα: $T(n) = O(nlogn)$