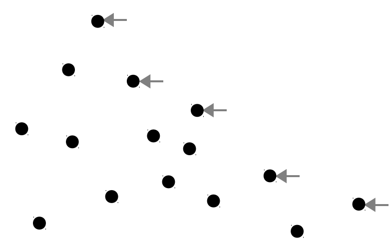
Υπολογιστική Γεωμετρία
Άσκηση 1η

Ερώτημα 1.

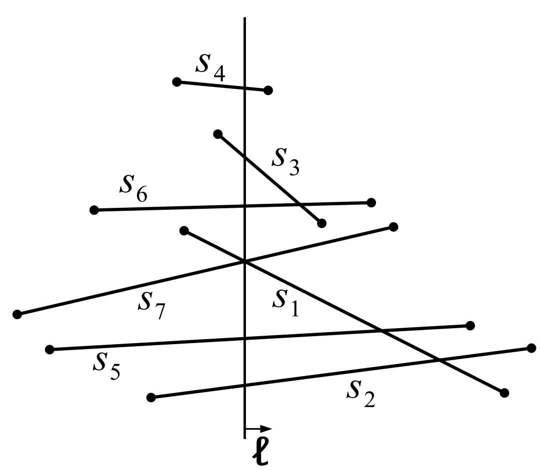
Δοθέντων δύο σημείων $p_i = (x_i, y_i)$ και $p_j = (x_j, y_j)$ στο επίπεδο, λέμε ότι το p_j *κυριαρχεί* του p_i εάν $x_j \geq x_i$ και $y_j \geq y_i$. Δοθέντος ενός συνόλου σημείων $P = \{p_1, p_2, \ldots, p_n\}$, ένα σημείο του συνόλου καλείται *μέγιστο* εάν δεν κυριαρχείται από κανένα άλλο σημείο του P. Σχεδιάστε έναν αλγόριθμο διαίρει-και-βασίλευε $O(n \log n)$ χρόνου ο οποίος να υπολογίζει όλα τα μέγιστα σημεία του P.

Υπόδειξη: Ταξινομήστε πρώτα τα σημεία του P ως προς αύξουσα x-συντεταγμένη. Παρατηρήστε ότι καθώς επισκεπτόμαστε τα μέγιστα σημεία με σειρά φθίνουσας x-συντεταγμένης οι y-συντεταγμένες των μεγίστων σημείων αυξάνουν. Στο σχήμα τα μέγιστα σημεία σημειώνονται με βέλη.



Ερώτημα 2.

Εκτελούμε τον αλγόριθμο εύρεσης τομών για τα ευθύγραμμα τμήματα του παρακάτω σχήματος. Δώστε την κατάσταση ευθείας σάρωσης και την κατάσταση της ουράς γεγονότων λίγο πριν και αμέσως μετά την επεξεργασία του τρέχοντος γεγονότος.



Ερώτημα 3.

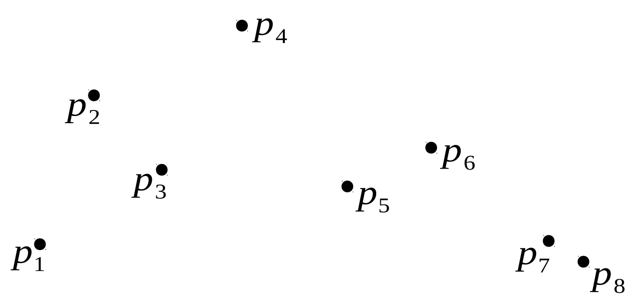
(α) Έστω δύο κυρτά σύνολα σημείων A και B. Να αποδείξετε ότι και η τομή τους $A \cap B$ είναι επίσης ένα κυρτό σύνολο.

(β) Σχεδιάστε ένα πολύγωνο με n κορυφές για το οποίο ο αλγόριθμος διαίρεσης ενός πολυγώνου σε μονότονα πολύγωνα προσθέτει $\Theta(n)$ διαγωνίους.

(γ) Σε τι μετασχηματίζεται ένα ευθύγραμμο τμήμα στο δυικό επίπεδο; Δώστε ένα σχηματικό παράδειγμα.

Ερώτημα 4.

(α) Σχηματίστε το kd-δέντρο για τα ακόλουθα σημεία. Επιλέξτε ένα ορθογώνιο ερώτησης R το οποίο να περιέχει τρία από τα σημεία και σημειώστε τους κόμβους του kd-δέντρου που επισκέπτεται ο αλγόριθμος αναζήτησης με ερώτηση περιοχής το R.



(β) Δώστε ένα παράδειγμα kd-δέντρου πάνω σε n σημεία και ενός ορθογωνίου ερώτησης R που οδηγούν ο χρόνος του kd-δέντρου για την απάντηση της ερώτησης καταμέτρησης με το R να είναι τουλάχιστον $\Omega(\sqrt{n})$. Με άλλα λόγια, δείξτε δηλαδή ότι το γνωστό άνω φράγμα $O(\sqrt{n})$ στο χρόνο ερώτησης του kd-δέντρου είναι σφιχτό.

Υπολογιστική Γεωμετρία

Άσκηση 1η

Αλέξανδρος Πλέσσιας
2025201100068
cst11068@uop.gr

Ερώτημα 1

Περιγραφή: Αρχικά, όπως αναφέρεται στην υπόδειξη σας ταξινομώ όλα τα σημεία P ως προς αύξουσα x συντεταγμένη. Έπειτα σαρώνω όλα τα σημεία ένα-ένα από το μεγαλύτερο προς το μικρότερο. Καθώς τα σαρώνω παρακολουθώ αυτά με την μεγαλύτερη y συντεταγμένη, εάν το τρέχων σημείο έχει y συντεγμένη μεγαλύτερη από την μεγαλύτερη y συντεταγμένη έως τώρα τότε είναι κυρίαρχο από το άλλο σημείο (δηλαδή το προηγούμενο με την μικρότερη y συντεταγμένη). Αλλιώς, δεν είναι κυρίαρχο και προχωράμε στο επόμενο (προς εξέταση).

Ακολουθεί αλγόριθμος σε ψευδογλώσσα τύπου Java ωστέ να γίνει ακόμα ποιο κατανοητή η παραπάνω περιγραφή (του αλγορίθμου):

Αλγόριθμος σε ψευδογλώσσα:

```
List<Points> findDominantPoints(P) { //Επιστρέφει λίστα με τα κυρίαρχα σημεία ως όρισμα παίρνει τα σημεία P.
    sortedPoints=mergeSort(P); //ταξινόμηση των P ως προς x συντεγμένη (αύξουσα)
    listWithDominantPoints=null; //αρχικοποίηση της λίστας αποτελεσμάτων.
    Point tmpPoint=sortedPoints[last-element]; //αρχικοποίηση βοηθητικής μεταβλητής με το σημείο με μέγιστο x.
    for(i=sortedPoints.lenght;i<0;i--) {
        //Εάν το tmpPoint (δηλαδή το προηγούμενο) έχει μεγαλύτερη ή ίση y συντεταγμένη* τότε είναι κυρίαρχο το tempPoint
        if(sortedPoints[i].getYcord()<=tmpPoint.getYcord())
            listWithDominantPoints.add(tmpPoint);
        else //Εάν το tmpPoint (δηλαδή το προηγούμενο σημείο) έχει μικρότερη y συντεταγμένη* τότε το tmpPoint δεν είναι κυρίαρχο.
            tmpPoint=sortedPoint[i];
    }
    return listWithDominantPoints;
}
```

* σε σχέση με το τωρινό σημείο

Ορθότητα: Αφού όλα τα σημεία είναι ταξινομημένα ως προς x σε αύξουσα σειρά εξασφαλίζουμε ότι όλα τα $x_{tmpPoint} > x_{currentPoint}$ δηλαδή ότι στο for-loop του αλγόριθμου πάντα όλα τα x της μεταβλητής tmpPoint είναι πάντα μεγαλύτερα από τα x της μεταβλητής sortedPoints[i] άρα εξασφαλίζω το $x_j \geq x_i$ που αναφέρεται στην εκφώνηση επίσης εξασφαλίζεται και το $y_j \geq y_i$ που αναφέρεται στην εκφώνηση μέσα από τον έλεγχο που κάνω στην if-statement όποτε όταν ισχύουν και τα 2 πάντα κάνω add το κυρίαρχο σημείο στην λίστα των αποτελεσμάτων.

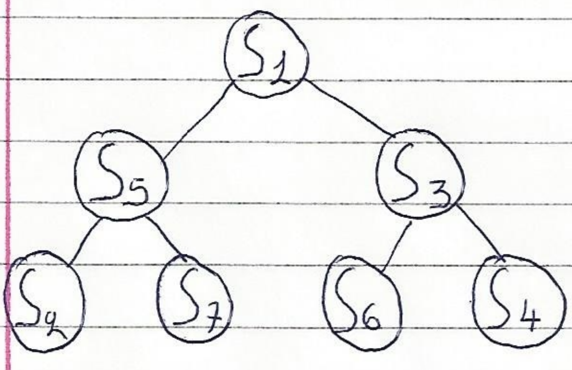
Χρόνος εκτέλεσης: Ο χρόνος της ταξινόμησης (MergeSort) είναι $O(n\log n)$ το οποίο προκύπτει από την αναδρομική σχέση $T(n) = 2T(n/2) + n$ αφού της έχουμε εφαρμόσει το Κεντρικό Θεώρημα (Master Theorem). Ενώ το for-loop εκτελείται $O(n)$ φορές και σε κάθε loop έχουμε $O(1)$ χρονική πολυπλοκότητα. Άρα συνολικός χρόνος εκτέλεσης είναι $O(n\log n) + (O(n) * O(1))$ που μας κάνει $O(n\log n)$ όπως ζητείται.

Παρατήρηση: Ο αλγόριθμος που έφτιαξα ακολουθεί το διαίρει-και-βασίλευε λόγω της mergeSort που χρησιμοποίησα για την ταξινόμηση των σημείων!

Ερώτημα 2 Αρχικά, θα ήθελα να αναφέρω ότι επιλέγω για την κατάσταση της ευθείας σάρωσης να διατηρώ ένα AVL δέντρο του οποίου τα στοιχεία είναι τα τμήματα που τέμνουν την ευθεία σάρωσης, αποθηκευμένα με αύξουσα σειρά y-συντεταγμένης κατά μήκος της τρέχουσας ευθείας σάρωσης. Ενώ για την ουρά γεγονότων κρατάω το σύνολο των γεγονότων που είναι να συμβούν στο μέλλον, ταξινομημένα κατά αύξουσα x-συντεταγμένη και επιλέγω για την αποθήκευση αυτών έναν σωρό. Οι επιλογές των παραπάνω δομών έγιναν βάση του pdf cg-notes-2015 και πιο συγκεκριμένα από την διαφάνεια 18.

Λίγο πριν

Κατάσταση ευθείας:

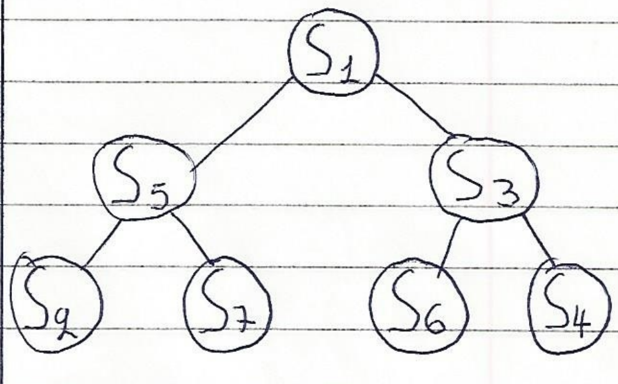


Ουρά γεγονότων:

push(τομή S_7, S_1)
push(διαγραφή S_4)
push(τομή S_6, S_3)
push(διαγραφή S_3)
push(» S_6)
push(τομή S_5, S_1)
push(διαγραφή S_7)
push(τομή S_2, S_1)
push(διαγραφή S_5)
push(» S_1)
push(» S_2)

Αμέσως μετά

Κατάσταση ευθείας:



Ουρά γεγονότων:

push(διαγραφή S_4)
push(τομή S_6, S_3)
push(διαγραφή S_3)
push(» S_6)
push(τομή S_5, S_1)
push(διαγραφή S_7)
push(τομή S_2, S_1)
push(διαγραφή S_5)
push(» S_1)
push(» S_2)

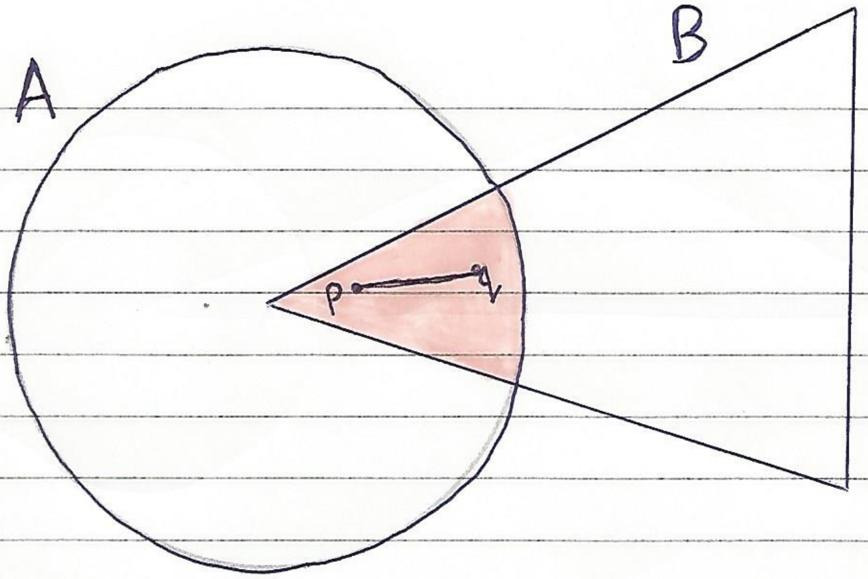
Άρα οι ουρές γεγονότων θα είναι οι εξής:

Λίγο πριν: (τομή S_7, S_1) → (διαγραφή S_4) → (τομή S_6, S_3) → (διαγραφή S_3) →
→ (διαγραφή S_6) → (τομή S_5, S_1) → (διαγραφή S_7) → (τομή S_2, S_1) →
→ (διαγραφή S_5) → (διαγραφή S_1) → (διαγραφή S_2) → nill.

Αμέσως μετά: (διαγραφή S_4) → (τομή S_6, S_3) → (διαγραφή S_3) → (διαγραφή S_6) →
→ (τομή S_5, S_1) → (διαγραφή S_7) → (τομή S_2, S_1) → (διαγραφή S_5) →
→ (διαγραφή S_1) → (διαγραφή S_2) → nill.

Παρατήρηση: Βλέπουμε ότι όντως λίγο πιο πριν ∃ η τομή των S_7, S_1 ενώ αμέσως μετά ∄. Διότι έχει γίνει pop η κεφαλή δηλαδή η (τομή S_7, S_1). Τέλος, μπορούμε να επιλέξουμε και άλλες δομές όπως κυκλική λίστα για τα γεγονότα ή κάτι άλλο, εγώ επέλεξα αυτά που αναφέρονται στο pdf cg-note-2015 και στο βιβλίο για να είμαι μέσα στην ύλη.

Ερώτημα 3α)

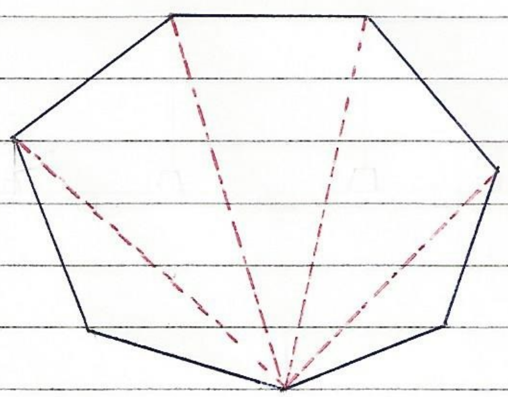


Περιγραφή σχήματος: Στα παραπάνω σχήμα φαίνεται το A που είναι κύκλος (ο κύκλος είναι κυρτό σχήμα) και το B που είναι τρίγωνο (το τρίγωνο είναι και αυτό κυρτό σχήμα). Η γραμμοσκιασμένη περιοχή με πορτοκαλί χρώμα είναι η τομή των δυο κυρτών συνόλων ενώ τα σημεία p, q είναι σημεία της τομής.

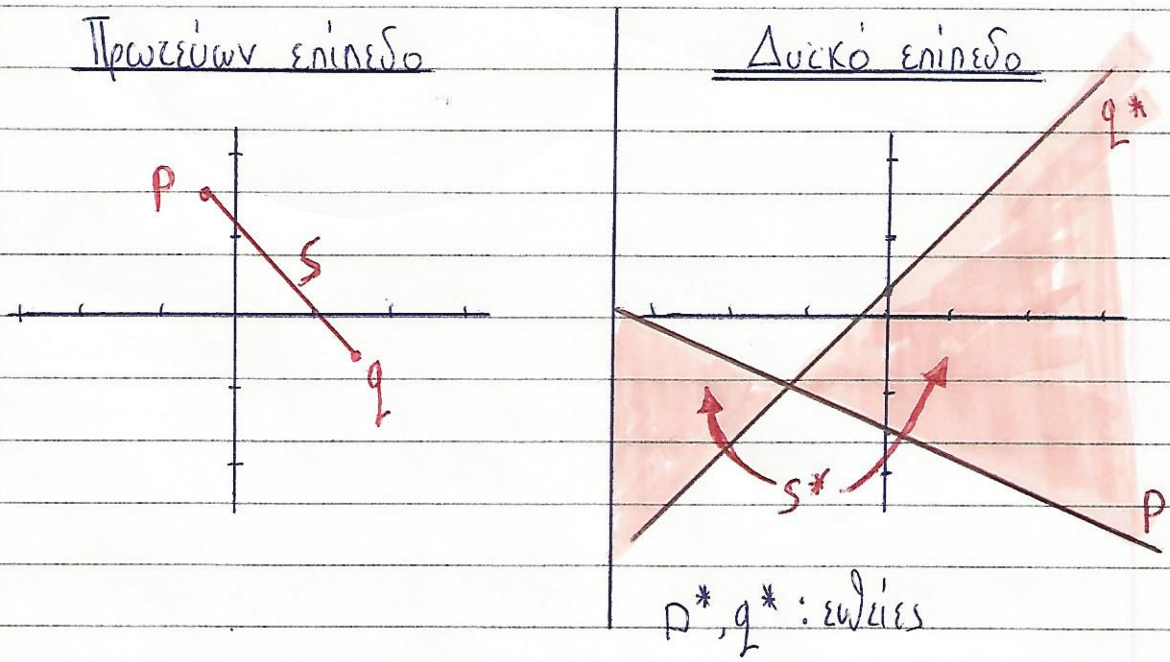
Πρόταση: Να δείξω ότι η τομή $A \cap B$ είναι επίσης κυρτό σύνολο.

Απόδειξη: Έχουμε A, B δύο κυρτά σύνολα και p, q δύο σημεία που ανήκουν στην τομή $A \cap B$. Τότε $p \in A, p \in B$ και $q \in A, q \in B$ οπότε αφού τα A, B είναι κυρτά θα έχουμε $pq \in A$ και $pq \in B$ δηλαδή $pq \in A \cap B$ κι έτσι η τομή $A \cap B$ είναι κυρτό σύνολο.

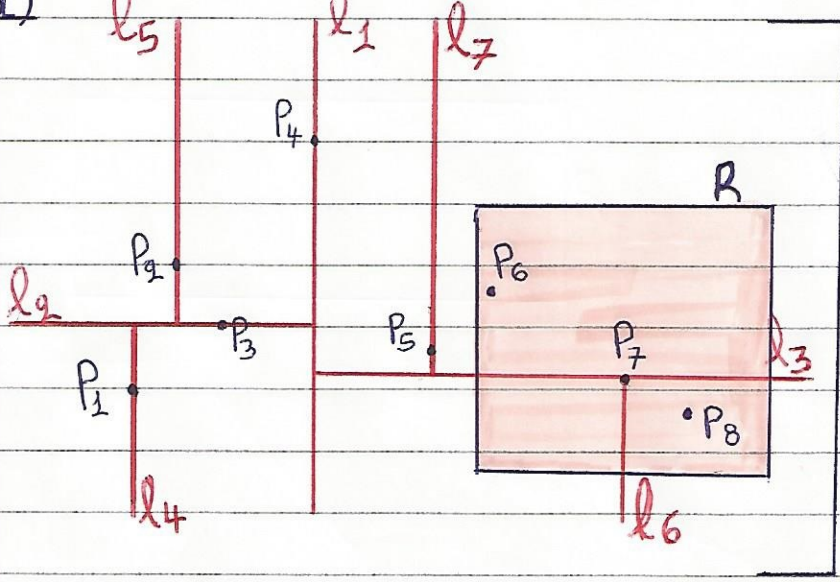
β) Ουσιαστικά μας ζητάται να κατασκευάσουμε ένα μονότονο πολύγωνο ώστε ο αλγόριθμος να προσθέτει $\Theta(n)$ διαγωνίους. Ακολουθεί το πολύγωνο:



γ) Επέλεξα να <u>μην</u> φτιάξω το παράδειγμα της σελίδος 178 του βιβλίου αλλά να φτιάξω ένα δικό μου (όπως αναφέρατε στην τάξη). Ακολουθεί το παράδειγμα:

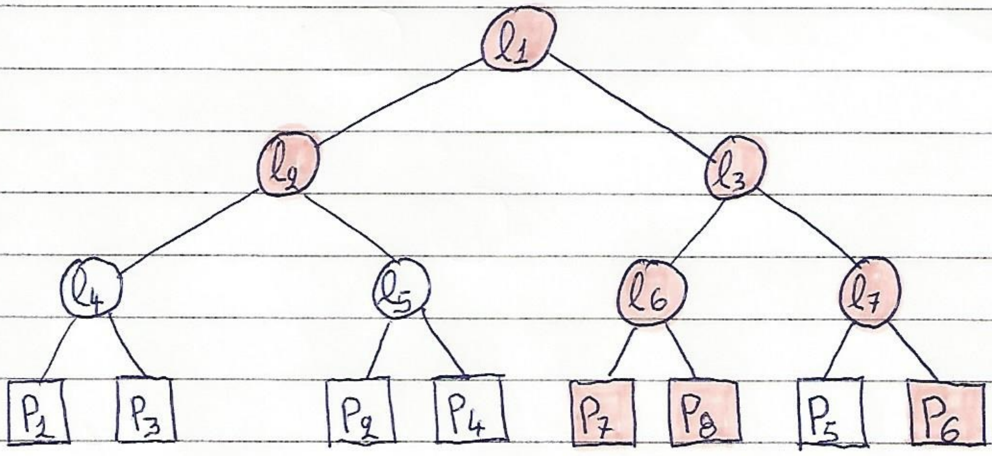


Ερώτημα 4α)



Το σχήμα της εκφώνησης, οι περιοχές, οι ευθείες και το ορθογώνιο R της ερώτησης.

Kd-δέντρο:



Κόμβοι Kd-δέντρου: Οι κόμβοι που επισκέπτεται ο αλγόριθμος για να απαντήσει στην ερώτηση περιοχής του R είναι οι κόμβοι l_1, l_3, l_6, P_7, P_8, l_7, P_6 αλλά και ο κόμβος l_2 οποίος μάλιστα στο pg.jar δεν εμφανίζεται επειδή έχει χαλαρή προσέγγιση. Δηλαδή οι πορτοκαλί κόμβοι του Kd-δέντρου.

B) Στο Λήμμα 5.4 της σελίδας 104 του βιβλίου Υπολογιστική Γεωμετρία αλγόριθμοι και εφαρμογές αναφέρεται: "Σε ένα Kd-δέντρο όπου βρίσκονται αποθηκευμένα n σημεία, κάθε ερώτημα που αφορά αξονοπαράλληλο ορθογώνιο μπορεί να απαντηθεί σε χρόνο $O(\sqrt{n}+K)$, όπου K το πλήθος των αναφερόμενων σημείων". Άρα το $O(\sqrt{n}+K)$ ισοδύναμη με το $O(\sqrt{n})$ διότι K σταθερά και όταν ισχύει το θεώρημα το $O(\sqrt{n})$ γίνεται $\Theta(\sqrt{n})$ διότι O και Θ συμπίπτουν. Η απόδειξη του λήμματος υπάρχει στην ίδια σελίδα (δηλαδή 104).

Παράδειγμα: Σαν παράδειγμα επιλέγω αυτό του Ερωτήματος 4α διότι η ερώτηση R που έκανα αφορά αξονοπαράλληλο ορθογώνιο άρα θα απαντηθεί σε χρόνο $\Theta(\sqrt{n})$. Όταν αναφέρω το παράδειγμα του Ερωτήματος 4α αναφέρομαι στα 8 σημεία p και στο ορθογώνιο R που περιέχει τα σημεία P_6, P_7 και P_8.

Εναλλακτική απόδειξη: Το Kd-δέντρο είναι ουσιαστικά ένα AVL δέντρο που σημαίνει πως χρειάζεται το πολύ $O(\log n)$ χρόνο για να το διανύσεις άρα αν δείξω ότι το $O(\sqrt{n}) > O(\log n)$, που ισχύει έχω απόδειξη αυτό που ζητάει ότι δηλαδή το $O\sqrt{n}$ είναι είναι σφιχτό και άνω φράγμα.

Καλές γιορτές!!!