Παράλληλοι Αλγόριθμοι: Ανάλυση Εικόνας και Υπολογιστική Γεωμετρία

Πέτρος Ποτίκας
CoReLab
4/5/2006

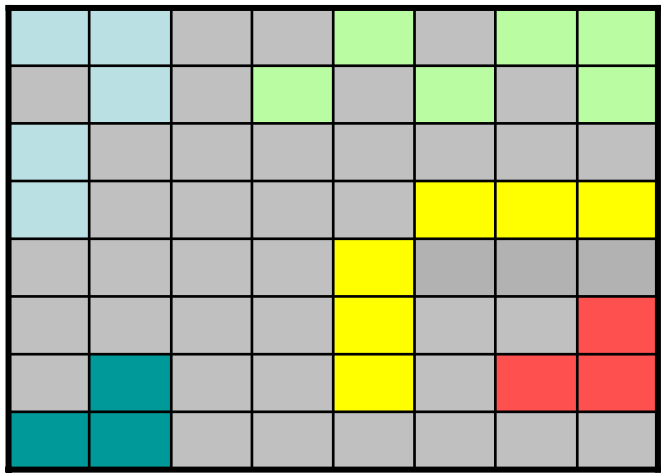
Επισκόπηση

- Ετικέτες σε συνιστώσες (Component labelling)
- Hough μετασχηματισμοί (transforms)
- Πλησιέστερος γείτονας (Nearest neighbor)
- Κυρτό περίβλημα (Convex hull)

Component labelling

Ορισμός: Δίνεται ένας διδιάστατος πίνακας από 0-1 pixels. Ζητείται να ανατεθεί μια ετικέτα σε κάθε 1-pixel έτσι ώστε κάθε δύο 1-pixels να έχουν την ίδια ετικέτα ανν τα δύο αυτά pixels είναι στην ίδια συνεκτική συνιστώσα.

1	1	0	0	1	0	1	1
0	1	0	1	0	1	0	1
1	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	1
0	1	0	0	1	0	1	1
1	1	0	0	0	0	0	0



Component labelling

- Αλγόριθμος δύο φάσεων:
- $1^{η}$ φάση (αρχικοποίηση): κάθε 1-pixel παίρνει ετικέτα τη θέση του στον πίνακα
- $2^{η}$ φάση (ενημερώσεις): ενημέρωσε την ετικέτα με την ελάχιστη ετικέτα ανάμεσα στην τρέχουσα και τις γειτονικές της

Παράδειγμα

11	12			15		17	18
	22		24		26		28
31							
41					46	47	48
				55			
				65			68
	72			75		77	78
81	82						

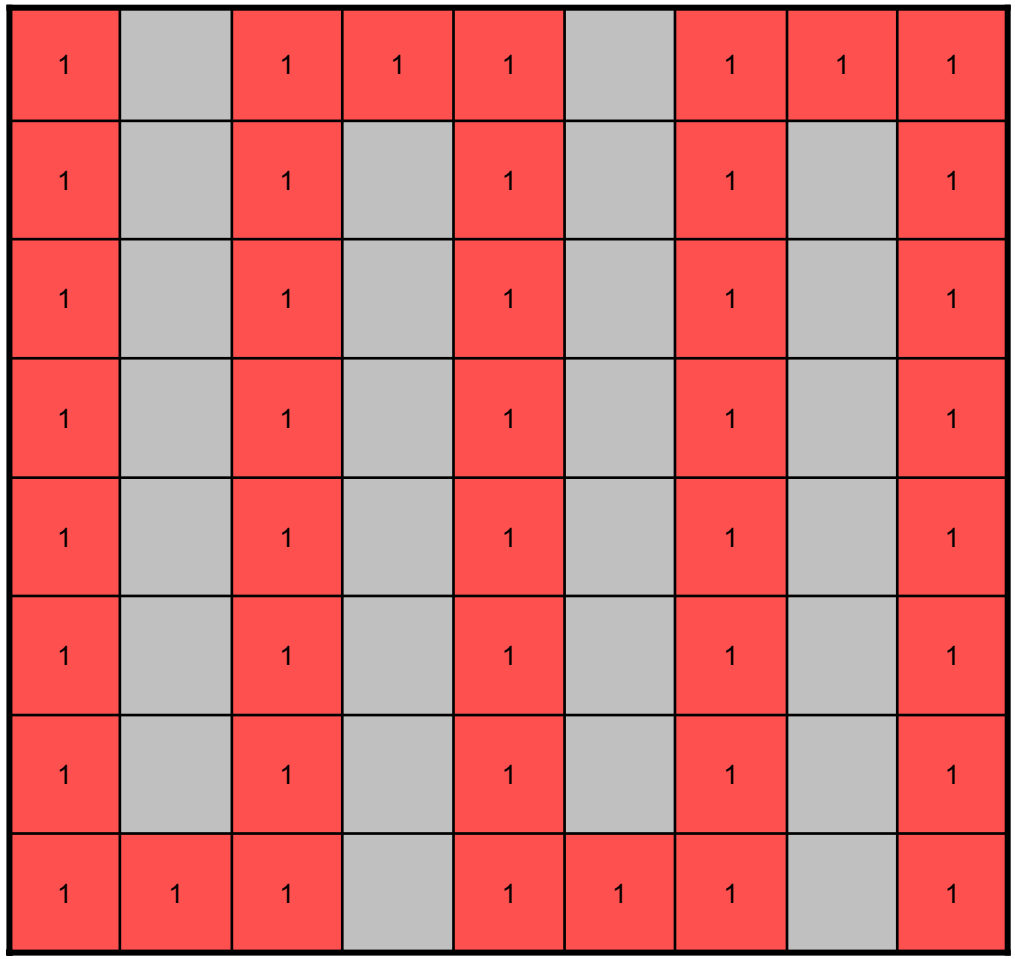
11	11			15		17	17
	11		15		15		17
22							
31					46	46	47
				46			
				55			68
	72			65		68	68
72	72						

11	11			15		15	17
	11		15		15		17
11							
22					46	46	46
				46			
				46			68
	72			55		68	68
72	72						

11	11			15		15	15
0	11		15		15		15
11							
11					46	46	46
				46			
				46			68
	72			46		68	68
72	72						

Χειρότερη περίπτωση

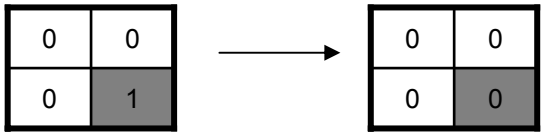
Θ(N): μέγιστη *εσωτερική διάμετρος* (μικρότερο μήκος μονοπατιού από 1-pixels, ώστε κάθε ζευγάρι από 1-pixels να ενώνεται)

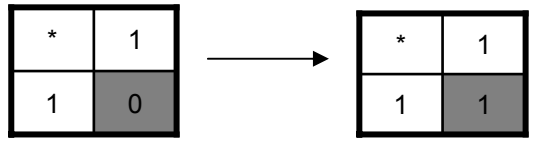


Αλγόριθμος Levialdi

- 1η φάση (συρρίκνωση): συρρικνώνουμε κάθε component μέχρι να μείνει μόνο ένα 1-pixel
- 2η φάση (επέκταση): δίνουμε ετικέτα και την επεκτείνουμε σε όλα τα 1-pixels της ίδιας συνιστώσας

Αλγόριθμος Levialdi (συν.)





0	0	1	0
0	1	1	0
1	1	0	0
0	0	0	1

0	0	0	0
0	0	1	0
0	1	1	0
0	0	0	0

0	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Ιδιότητες μετασχηματισμού

- Δεν συγχωνεύονται ποτέ δύο συνιστώσες
- Δεν διασπάται καμία συνιστώσα

Χρόνος

Με επαγωγή: στο βήμα t, κάθε (i,j) pixel θα είναι 0-pixel για $i+j<t+1$

$$2\sqrt{N}-1$$

Επέκταση

- Κάθε επεξεργαστής φυλάει τα 2√N-1 βήματα της φάσης συρρίκνωσης και τα αντιστρέφει (πολύ χώρο)
- Χρόνος: 2√N-1 βήματα

Συνολικός χρόνος

4√N-2 βήματα

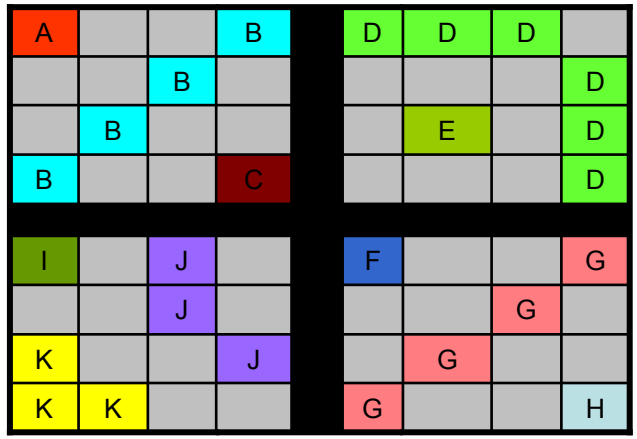
Αναδρομικός Αλγόριθμος

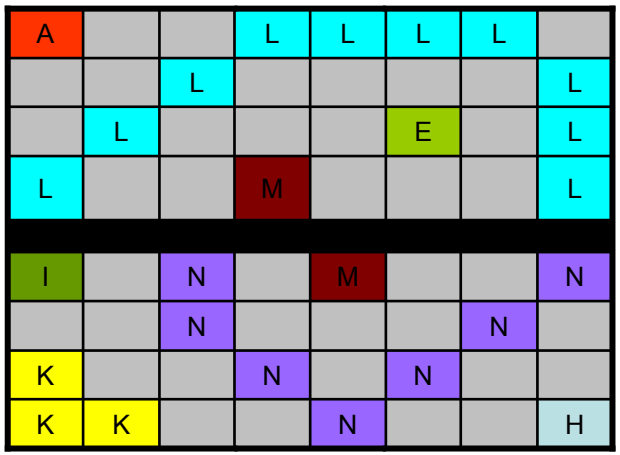
- Divide and conquer: διαίρεση του πίνακα σε 4 τεταρτημόρια και ανάθεση σε αυτούς ετικέτες
- Συγχώνευση συνιστωσών ως προς τον κάθετο άξονα
- Συγχώνευση συνιστωσών ως προς τον οριζόντιο άξονα

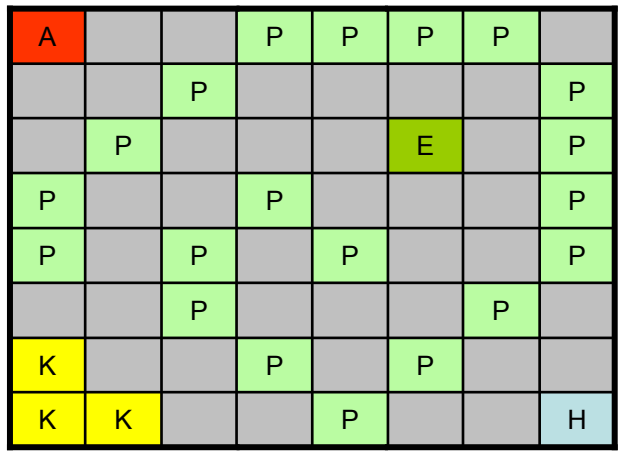
Χρόνος: $O(\sqrt{N})$

Παράδειγμα (αναδρομικού)

1	0	0	1	1	1	1	0
0	0	1	0	0	0	0	1
0	1	0	0	0	1	0	1
1	0	0	1	0	0	0	1
1	0	1	0	1	0	0	1
0	0	1	0	0	0	1	0
1	0	0	1	0	1	0	0
1	1	0	0	1	0	0	1







Ανάλυση

Για τις συγχωνεύσεις έχουμε O(√N) βήματα:

- Το πολύ √N διαφορετικές συνιστώσες μπορούν να βρίσκονται στα όρια
- Γράφος G παριστάνει τις συνιστώσες στα όρια (πρόβλημα εύρεσης συνιστωσών στο γράφο G)
- Πίνακας γειτνίασης σε κατάλληλη μορφή (sorting O(√N), routing O(√N))
- Pipeline τις √N αλλαγμένες ετικέτες

$$T(\sqrt{N}) \leq T(\sqrt{N}/2) + O(\sqrt{N}) = O(\sqrt{N})$$

Μετασχηματισμοί Hough

Ορισμός: Δίνεται μια διδιάστατη εικόνα και μια γωνία θ (-π/2≤θ≤π/2). Ο *μετασχηματισμός Hough* υπολογίζεται αν χωρίσουμε την εικόνα σε παράλληλες λωρίδες πάχους ενός pixel που σχηματίζουν γωνία θ με τον x-άξονα και στη συνέχεια αθροίσουμε τις τιμές των pixels σε κάθε λωρίδα.

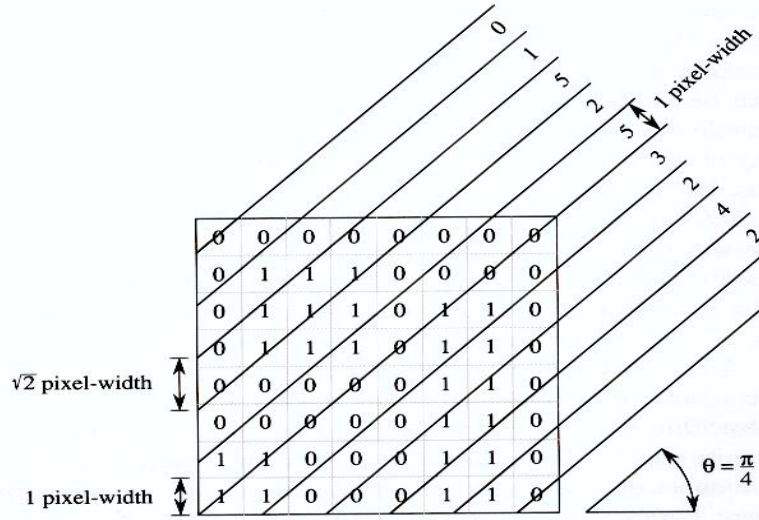


Figure 1-108 *The Hough transform of an 8 × 8 image with projection angle* $\theta = \frac{\pi}{4}$. *The number of 1-pixels contained in each band is displayed at the upper-right end of the band. Each 1-pixel is counted for only the band that contains its center. If the center of a pixel lies on the boundary between two bands, then the uppermost of the two bands gets credit for the pixel.*

Αλγόριθμος

- Χρήση μετρητή b_i για κάθε λωρίδα (≤ *√2N+1 μετρητές)*
- Μετακίνηση του μετρητή πάνω, δεξιά ή πάνω-δεξιά, ανάλογα με τη γωνία θ, το τρέχων κελί και το κελί εκκίνησης (τοπική απόφαση)
- Αυξάνει η τιμή του μετρητή όταν βρίσκει 1
- Έξοδος σε O(*√N) βήματα πάνω ή δεξιά*

Παράδειγμα

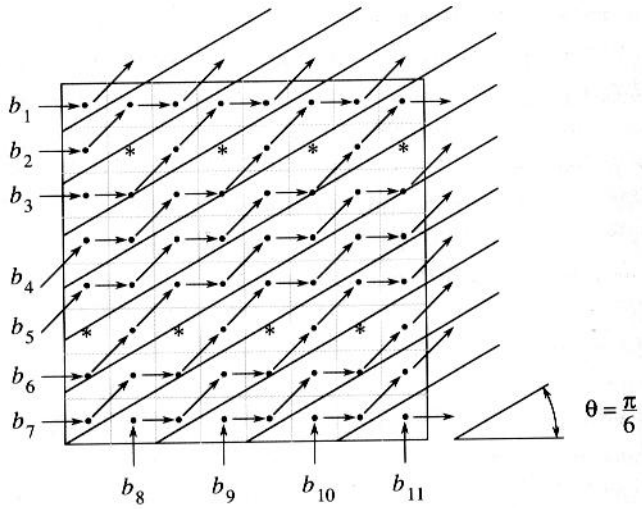


Figure 1-109 *Paths followed by variables summing the pixel values in each band of a* $\frac{\pi}{6}$*-angle Hough transform. Each* b_i *visits at most one processor in each column. Processors denoted by an asterisk are not visited, since their pixel values are also held by their neighboring cell just below.*

Εφαρμογές

- Με διαφορετικές γωνίες βρίσκουμε αντικείμενα σε μια εικόνα (π.χ. ύπαρξη δίσκου)
- Με pipeling M διαφορετικών γωνιών χρόνος $O(M+\sqrt{N})$
- Θέλουμε όλες τις γωνίες της μορφής

$$\pi j/2\sqrt{N}, \text{ με } j \in [-\sqrt{N}, \sqrt{N}]$$

$$\text{χρόνος } O(\sqrt{N})$$

Κοντινότερος γείτονας

- Εύρεση κοντινότερου αντικειμένου για κάθε 1-pixel (για κάθε pixel p βρίσκει το κοντινότερο 1-pixel με διαφορετική ετικέτα)
- Εύρεση κοντινότερου αντικειμένου για κάθε αντικείμενο

Κοντινότερο αντικείμενο για pixel

1. Βρίσκουμε το κοντινότερο pixel σε κάθε λωρίδα (τροποποίηση του μετασχηματισμού Hough, σήμα που βρίσκει το κοντινότερο αντί για άθροιση)
2. Εύρεση κοντινότερου από αριστερά και από δεξιά και επιλογή του πλησιέστερου
3. *σε όλη την εικόνα: $\pi j/2\sqrt{N}$, με $j \in [-\sqrt{N}, \sqrt{N}]$*

Κοντινότερο αντικείμενο για αντικείμενο

- Για κάθε ένα pixel δημιουργούμε ένα πακέτο πληροφοριών: ετικέτα του, απόσταση από πλησιέστερο αντικείμενο και ετικέτα αντικειμένου
- Ταξινομούμε τα πακέτα αυτά με βάση την ετικέτα τους και αν είναι ίδια με βάση την απόσταση τους (O($\sqrt{N}$) χρόνος)
- Το πρώτο πακέτο για κάθε αντικείμενο θα δίνει το κοντινότερο αντικείμενο

Χρόνος

- O($\sqrt{N}$) βήματα για να υπολογιστεί το συντομότερο αντικείμενο προς κάθε αντικείμενο
- Είναι βέλτιστο, αλλά όχι work efficient (καλύτερο με mesh of trees: *O(log^2N)* χρόνο με O(N) επεξεργαστές)

Κυρτό περίβλημα

Ορισμός: Δίνεται μια συλλογή από σημεία S στο επίπεδο. Το *κυρτό περίβλημα* των σημείων είναι το μικρότερο κυρτό πολύγωνο που περιλαμβάνει όλα τα σημεία του S.

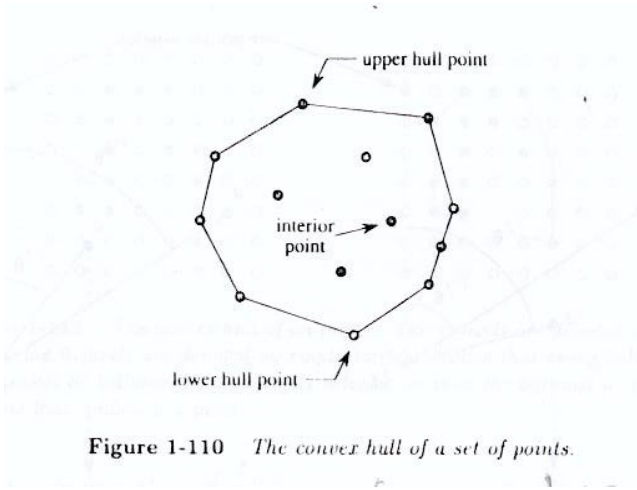


Figure 1-110 *The convex hull of a set of points.*

- Σε N τυχαία σημεία

(3N βήματα, N-κελιά)

- Σε διδιάστατη εικόνα

(3√N βήματα, √N-κελιά)

Κυρτό περίβλημα για Ν σημεία

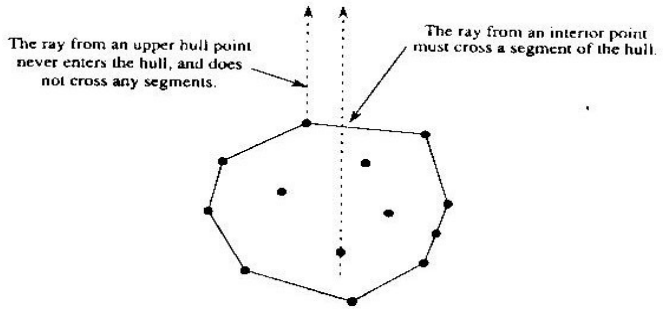
- Ταξινόμηση σημείων ως προς x-συντεταγμένη (Ν βήματα)
- Υπολογισμός των upper hull σημείων

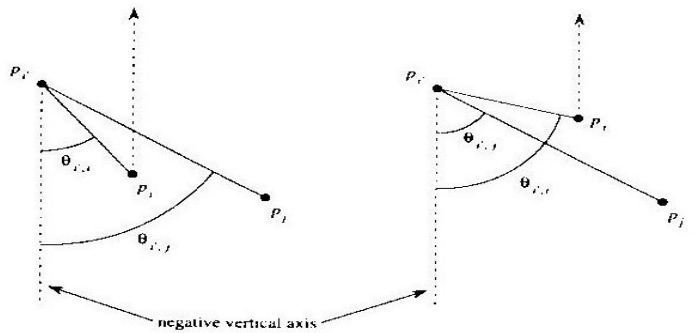
$\theta_{i,j}$: γωνία μεταξύ p_i,p_j και αρνητικού y-άξονα

Ζητούμενο: το κοντινότερο από δεξιά p_j για το οποίο το $\theta_{i,j}$ είναι μέγιστο (αν το p_i είναι σημείο στο περίβλημα, τότε το p_j είναι το επόμενο σημείο)

Λήμμα: Δίνεται μια συλλογή σημείων $p_1,\ldots,p_N$ ταξινομημένα ως προς το x-συντεταγμένη. Το σημείο p_i είναι σημείο στο περίβλημα ανν $x_1<x_i<x_N$ και $r(i')\leq i$, όπου το $r(i')$ είναι η μικρότερη τιμή του $j>i$, για την οποία η $\theta_{i,j}$ μεγιστοποιείται.

Διαισθητικά: Το p_j είναι αριστερά από το p_i, άρα το ray από το p_i δεν τέμνει κανένα ευθύγραμμο τμήμα του περιβλήματος.





Διαδικασία υπολογισμού:

- Υπολογισμός r(i) για κάθε i (σε N βήματα) στέλνοντας κάθε σημείο προς τα αριστερά. Ο i-επεξεργαστής υπολογίζει το $\theta_{i,j}$ και κρατάει το μεγαλύτερο που έχει βρει.
- Από αριστερά προς τα δεξιά μεταβιβάζεται στον i-επεξεργαστή το $max_{i'\leq i} r(i')$. Ο i-επεξεργαστής ελέγχει αν $max_{i'\leq i} r(i') \leq i$ (σε N βήματα).

Χρόνος : 3N (N ταξινόμηση, 2N για upper/lower hull points)

Κυρτό περίβλημα για εικόνα

- $\sqrt{N}*\sqrt{N}$ εικόνα με 0-1 pixels
- $4\sqrt{N}$ υποψήφια σημεία για το περίβλημα:

  το πιο πάνω και το πιο κάτω κάθε στήλης και τα 1-pixels της πιο αριστερής και πιο δεξιάς στήλης που περιέχει 1-pixels
- Upper hull σημεία: είσοδος στον πίνακα γραμμή-γραμμή, κάθε επεξεργαστής κρατάει το πρώτο και το τελευταίο 1-pixel που συναντά και εφαρμόζεται ο προηγούμενος αλγόριθμος (είναι ταξινομημένα κατά x-συντεταγμένη)

Κυρτό περίβλημα για εικόνα (συν.)

- Υπολογισμός των αριστερότερων και δεξιότερων στηλών που έχουν 1-pixels: κάθε επεξεργαστής όταν παίρνει 1-pixel από τη στήλη του το αποθηκεύει δεξιά (τόρος). Συνεχίζεται αυτό μέχρι να λάβει από αριστερά έναν 1, οπότε στέλνει δεξιά τον 1 με τη μικρότερη x-συντεταγμένη. Μετά από $\sqrt{N}$ βήματα κάθε pixel της αριστερότερης στήλης θα έχει εισαχθεί και θα φυλάσσεται σε κάποιον επεξεργαστή. Με ένα σήμα βρίσκουμε τη στήλη που είναι αριστερότερη και με ένα ακόμα το μαθαίνουν όλοι. Αυτό θέλει χρόνο $2\sqrt{N}$

Συμπεράσματα

- Είναι αρκετά work efficient (ελέγχουμε N pixels)
- Σε mesh of trees έχουμε O(logN) βήματα σε $\sqrt{N} * \sqrt{N}$ επεξεργαστές για *logn* εικόνα.