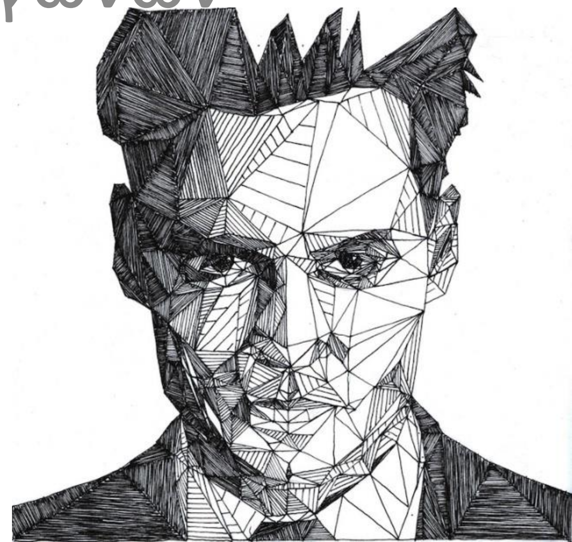
Υπολογιστική Γεωμετρία

Τριγωνοποίηση Απλών Πολυγώνων
(Art Gallery Problem)



Αναφορές:

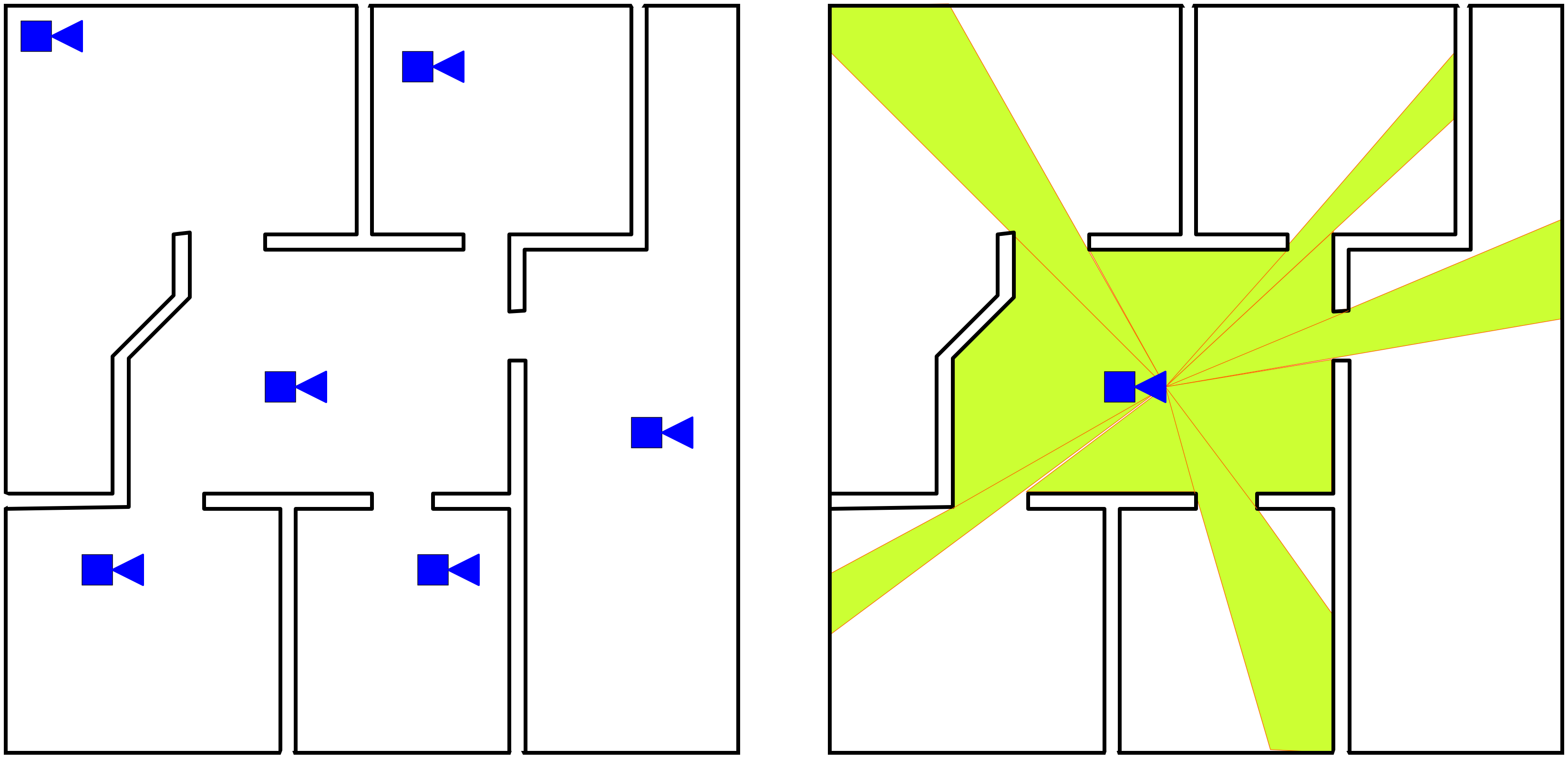
- [M. de Berg et al] Κεφάλαιο 3
- [Preparata-Shamos’85] Κεφάλαιο 6
- [O’Rourke’98] Κεφάλαιο 1

Εφαρμογές:

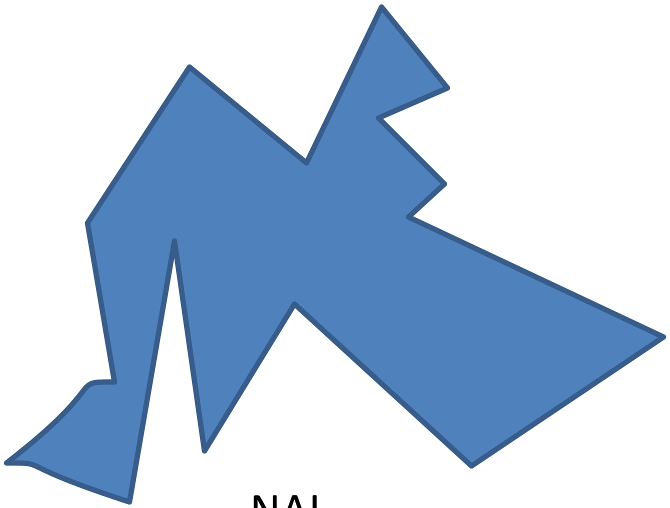
- ❑ Γραφικά: Ray Shooting
- ❑ Ρομποτική: Γεωδεσικά ελάχιστα μονοπάτια σε πολύγωνα, ορατότητα
- ❑ GIS: Αναζήτηση σημείου στο επίπεδο
- ❑ . . .

Τριγωνοποίηση Απλών Πολυγώνων

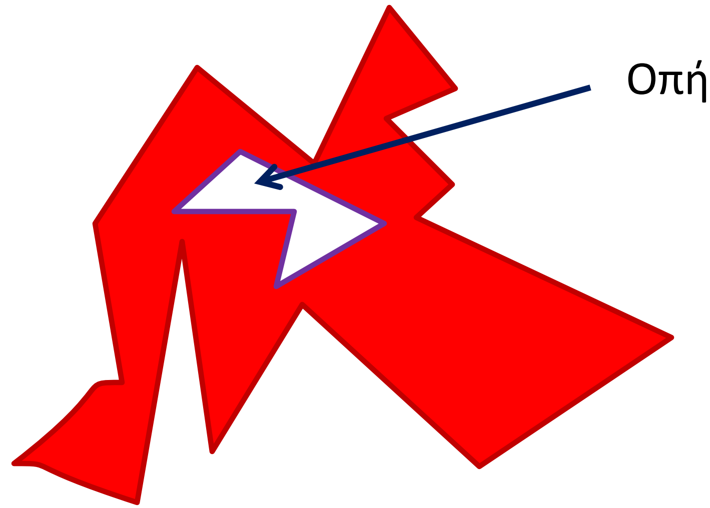
Φυλάσσοντας μια Πινακοθήκη (art gallery)



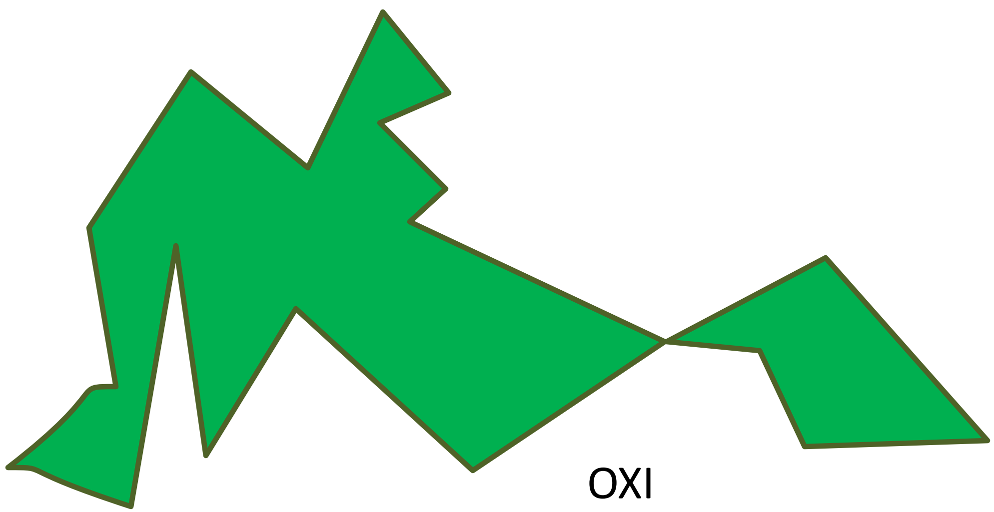
Απλά Πολύγωνα



ΝΑΙ



ΟΧΙ



ΟΧΙ

Περικλείονται από μία μοναδική κλειστή αλυσίδα που δεν περιέχει τον εαυτό της.

Το Πρόβλημα της Πινακοθήκης

Πόσες κάμερες χρειαζόμαστε για μία πινακοθήκη και πως αποφασίζουμε σχετικά με την τοποθέτησή τους;

- Είναι NP-δύσκολο να υπολογίσουμε το ελάχιστο πλήθος από κάμερες για ένα αυθαίρετο απλό πολύγωνο. [Aggarwal 1984].

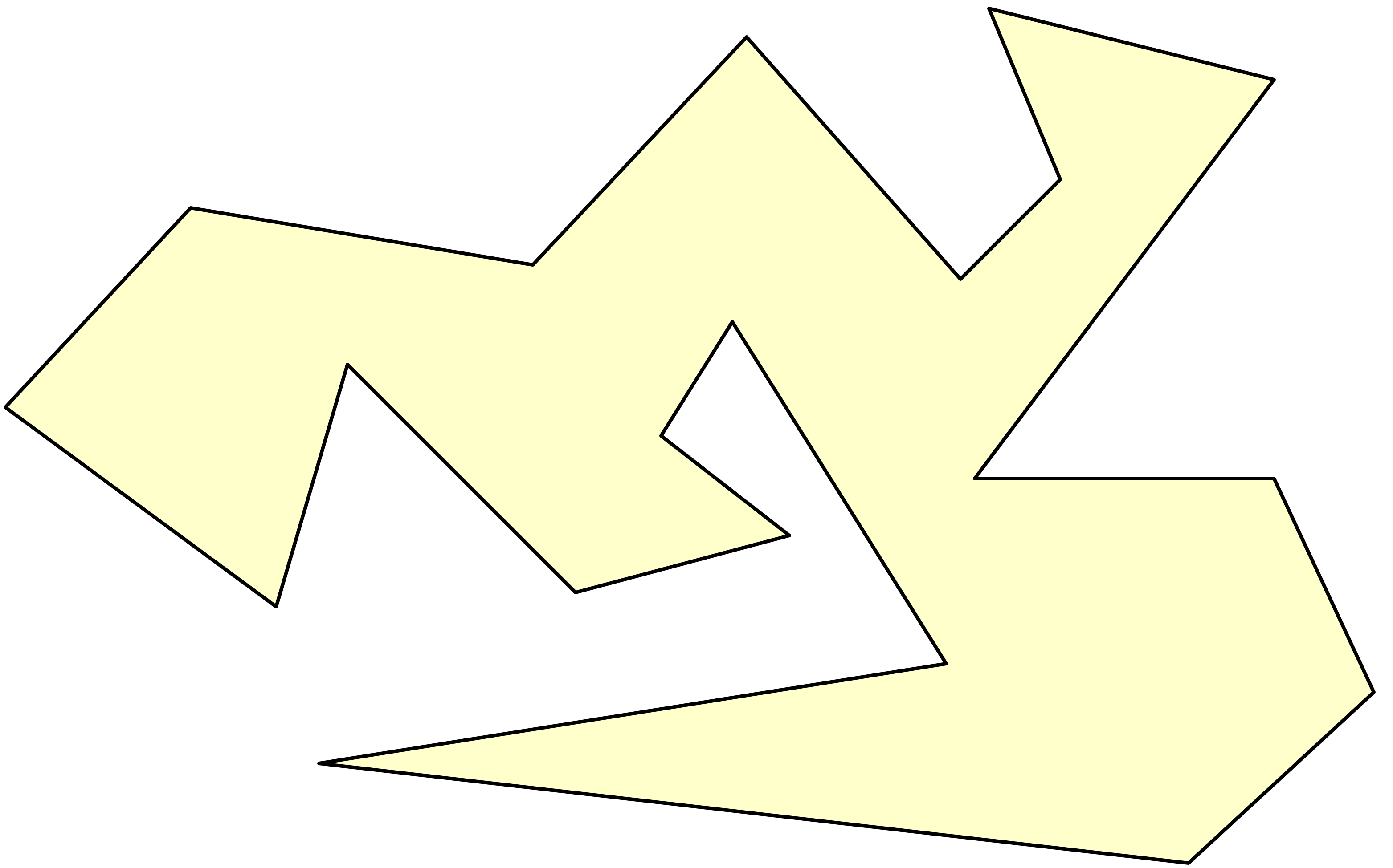
- Έστω P ένα απλό πολύγωνο με n κορυφές.

- Αν το P είναι κυρτό τότε μία κάμερα στο εσωτερικό του P είναι αρκετή.

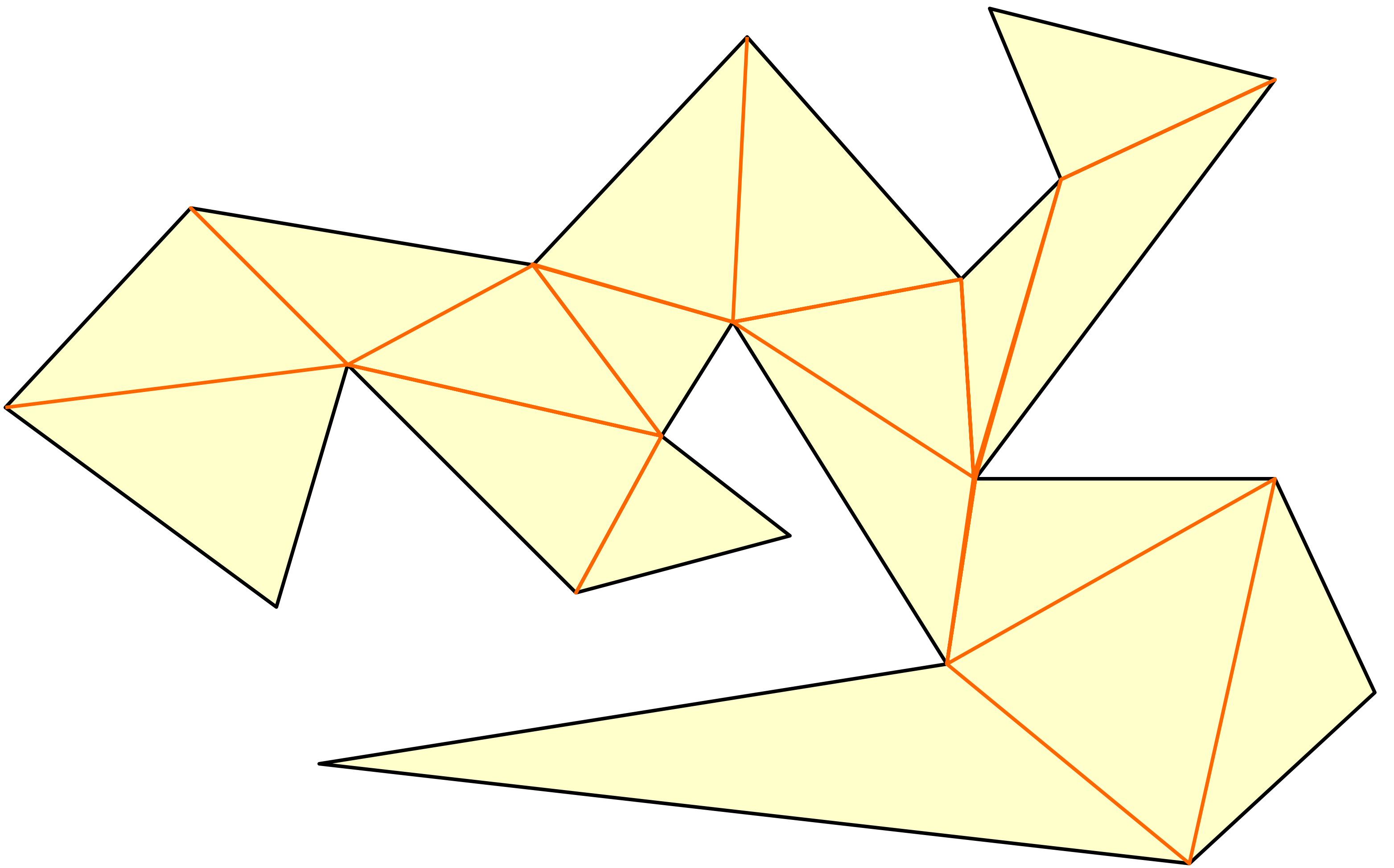
- n κάμερες για το P είναι πάντα αρκετές αφού μπορούμε να τοποθετήσουμε μία κάμερα σε κάθε κορυφή.

- Μπορούμε να χρησιμοποιήσουμε < n κάμερες; ΝΑΙ, με τριγωνοποίηση του P.

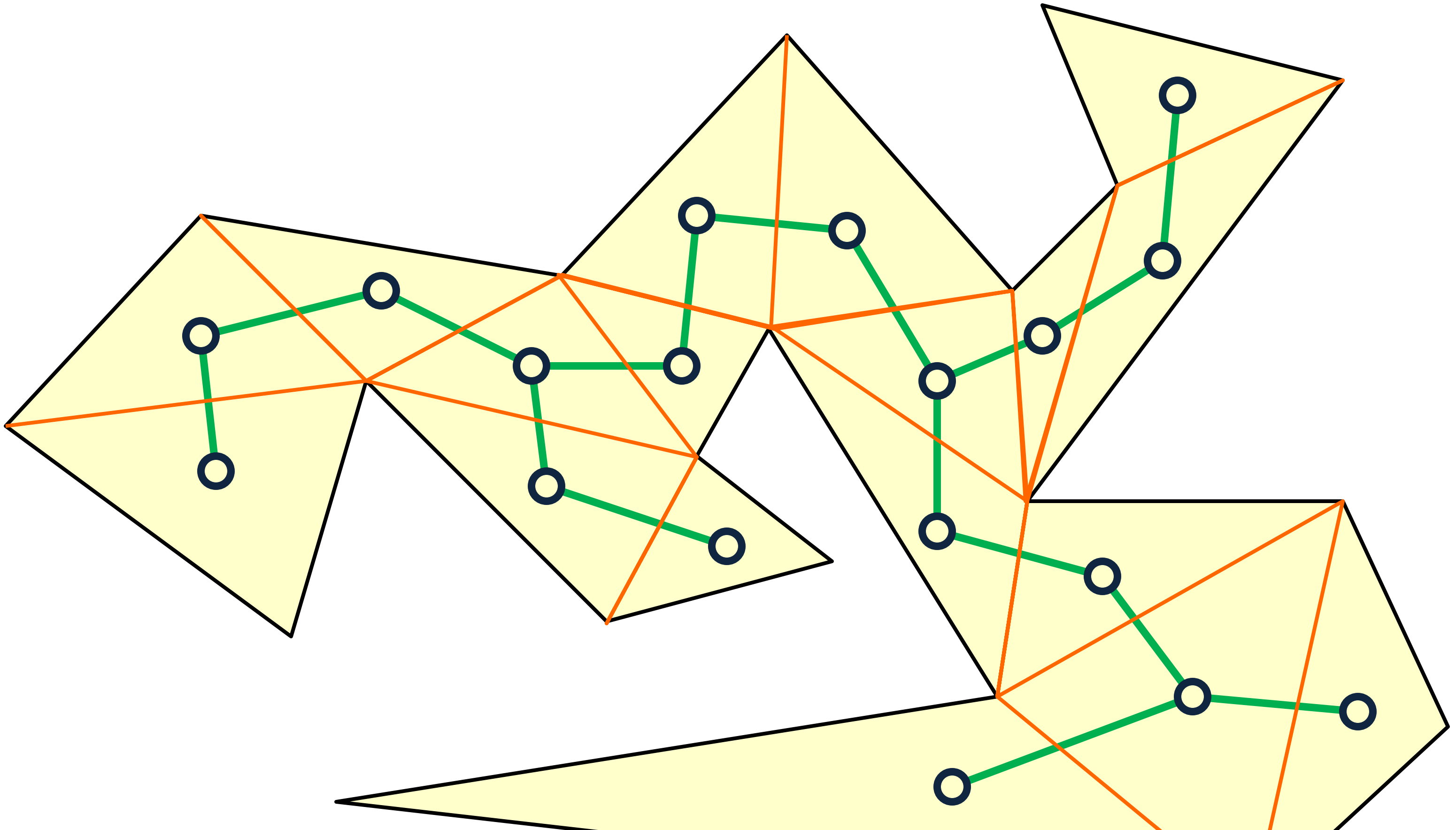
Ένα απλό πολύγωνο P



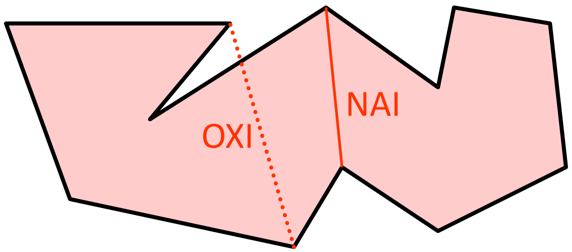
Μία τριγωνοποίηση του P



Δυϊκό δένδρο της τριγωνοποίησης



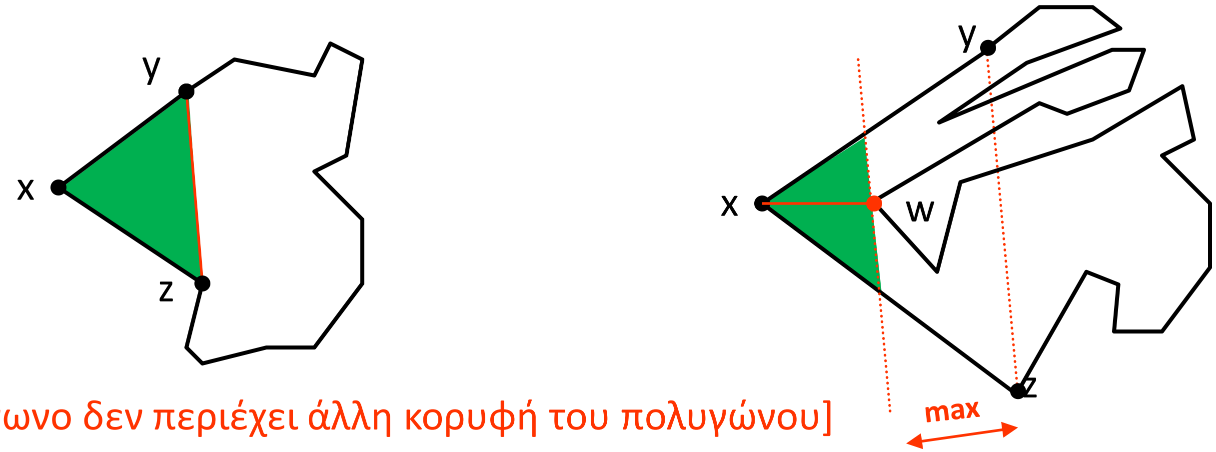
<u>Διαγώνιος</u> ενός απλού πολυγώνου P: ένα ευθύγραμμο τμήμα μεταξύ δύο μη γειτονικών κορυφών του P που κείται εντελώς εντός του P.



<u>ΛΗΜΜΑ 1</u> Κάθε απλό n-πολύγωνο με n>3 έχει τουλάχιστον μία διαγώνιο. Αυτή η διαγώνιος μπορεί να βρεθεί σε O(n) χρόνο.

<u>Απόδειξη</u>: Έστω x μία οποιαδήποτε κυρτή κορυφή (π.χ. , η χαμηλότερη και πιο αριστερά).

(a) το yz είναι μία διαγώνιος (b) το xw είναι μία διαγώνιος



[το σκιασμένο τρίγωνο δεν περιέχει άλλη κορυφή του πολυγώνου]

ΘΕΩΡΗΜΑ 2 Κάθε απλό n-πολύγωνο P επιδέχεται τουλάχιστον μία τριγωνοποίηση. Αυτή η τριγωνοποίηση T μπορεί να υπολογισθεί σε $O(n^2)$ χρόνο. Κάθε T έχει n-3 διαγώνιους και n-2 τρίγωνα.

Άρα, *n*-2 κάμερες είναι αρκετές. Μπορούμε καλύτερα;

Απόδειξη: επαγωγή στο n.

Βάση (n=3): Προφανές.

Επαγωγικό βήμα (n>3): Η διαγώνιος d χωρίζει το P σε απλά πολύγωνα P_1 & P_2 με n_1 & n_2 κορυφές, όπου η d είναι κοινή ακμή. $n = n_1+n_2 -2$.

Οι τριγωνοποιήσεις T_1 & T_2 των P_1 & P_2 υπολογίζονται αναδρομικά.
Έπειτα θέτουμε $T = T_1 \cup T_2$ με d ως επιπλέον διαγώνιο.

Συνολικός χρόνος: $\text{Time}(n) = \text{Time}(n_1) + \text{Time}(n_2) + O(n) = O(n^2)$.

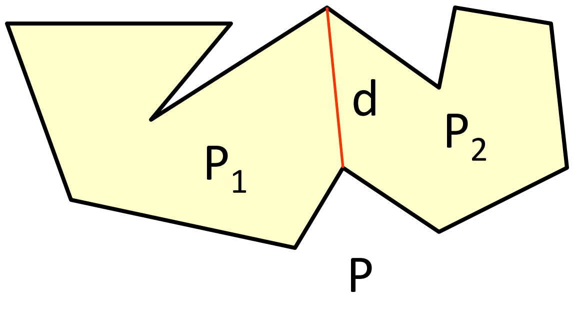
Από επαγωγική υπόθεση:

Το T_1 έχει n_1–3 διαγώνιους και n_1–2 τρίγωνα,

Το T_2 έχει n_2–3 διαγώνιους και n_2–2 τρίγωνα,

Αυτό σημαίνει:

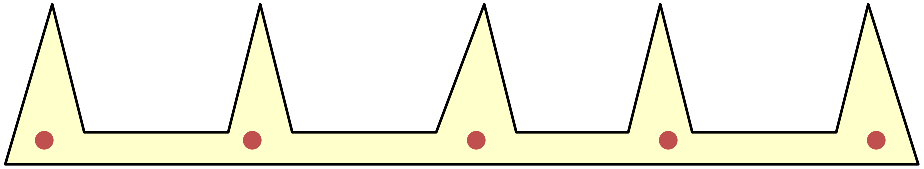
Το T έχει n–3 διαγώνιους και n–2 τρίγωνα.



Θεώρημα Πινακοθήκης [Chvătal 1975, Fisk 1978]

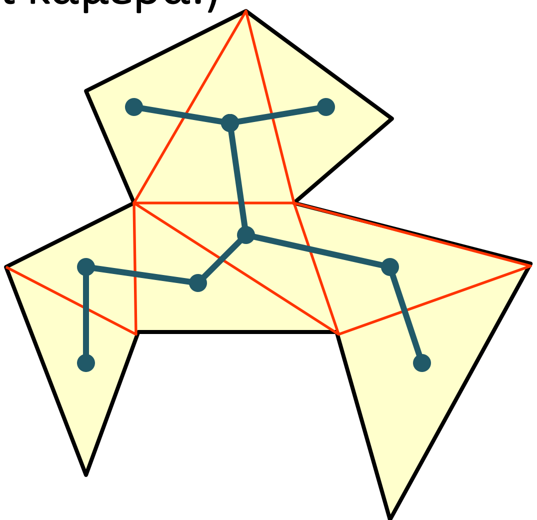
$\lfloor n/3 \rfloor$ κάμερες είναι πάντα αρκετές και μερικές φορές αναγκαίες για τη φύλαξη ενός n-πολυγώνου.

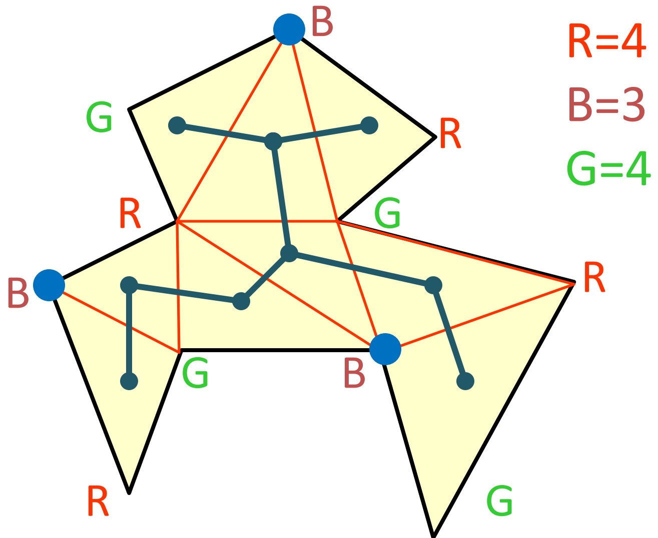
Απόδειξη: Αναγκαιότητα:



Ικάνη:

1. *T* = μία τριγωνοποίηση του n-πολυγώνου.
2. Χρωματισμός με 3 χρώματα των κορυφών του *T* (όλες οι κορυφές ενός τριγώνου έχουν διαφορετικά χρώματα). Ένα DFS στο δυϊκό δένδρο του *T* είναι αρκετό.
3. Επέλεξε το χρώμα που χρησιμοποιείται λιγότερο.
4. Τοποθέτησε κάμερα στην κορυφή με το επιλεγμένο χρώμα. (Κάθε τρίγωνο έχει κάμερα.)





Αλγόριθμοι Τριγωνοποίησης Απλών Πολυγώνων

- $O(n^2)$ Θεώρημα 2. Επιπλέον με “ear removal”, Lennes 1911.
- O(n log n) Garey-Johnson-Preparata-Tarjan (σάρωση επιπέδου) 1978.
- O(n log log n) Tarjan-van Wyk (ζυγισμένη τομή & Jordan-ταξινόμηση) 1986-88.
- O(n log* n) αναμενόμενος Clarkson-Tarjan-van Wyk 1989.
- O(n) Chazelle 1991. [πολύπλοκο. Απλοποίηση;]
- O(n) αναμενόμενος Amato-Goodrich-Ramos 2000. [LN15]

Ένας πιθανός υποψήφιος για απλούστερους και αποδοτικούς αλγορίθμους τριγωνοποίησης:

- ψευδοτριγωνοποιήσεις Mirzaian 1988 [LN14]

Αλγόριθμος

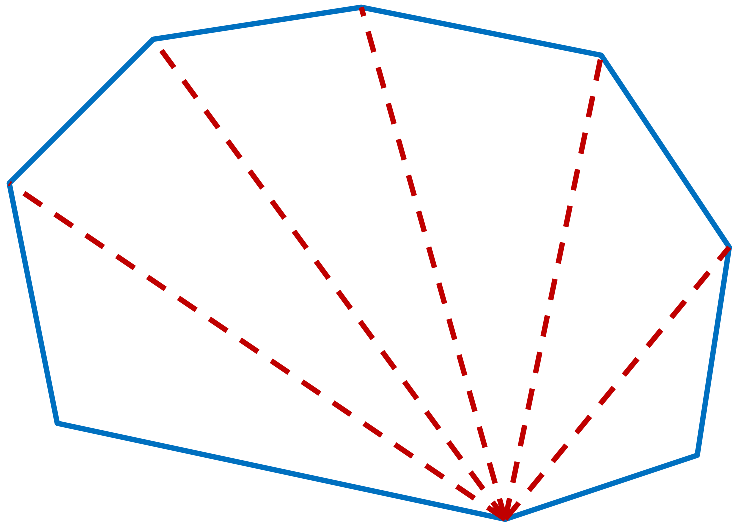
1. Τριγωνοποίηση απλού πολυγώνου με γρήγορο αλγόριθμο.

   Αναπαράσταση με διπλοσυνδεδεμένους καταλόγους ακμών ώστε να μπορούμε να μεταβαίνουμε σε γείτονα από ένα τρίγωνο σε σταθερό χρόνο.

2. Παραγωγή ενός 3-χρωματισμού με DFS.

3. Επιλέγουμε τη μικρότερη κλάση χρώματος και εκεί τοποθετούμε τις κάμερες.

Τριγωνοποίηση Κυρτού Πολυγώνου



$\Theta(n)$ χρόνος!

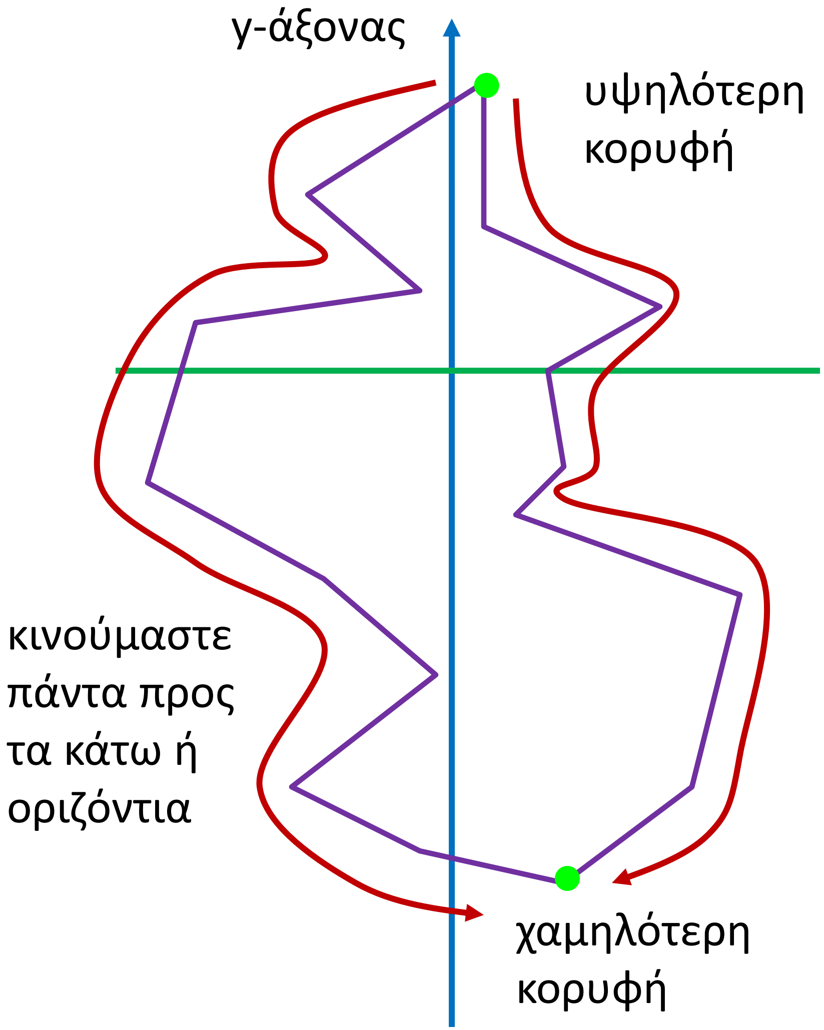
Ιδέα:

μονότονα

- Διαιρούμε ένα απλό πολύγωνο σε κυρτά τεμάχια.
- Τριγωνοποιούμε τα τεμάχια.

ίδιας δυσκολίας με την τριγωνοποίηση

y-μονότονα Τεμάχια

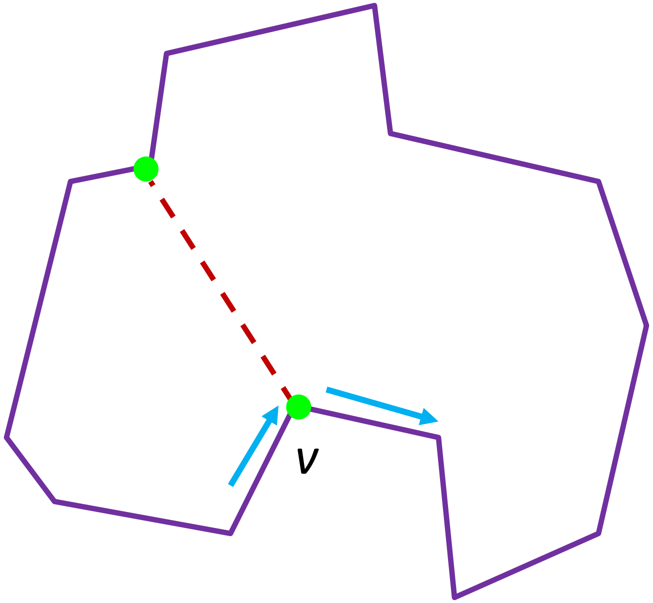


y-μονότονα αν κάθε οριζόντια γραμμή έχει τομή με το πολύγωνο ένα συνεκτικό χωρίο

Στρατηγική:

Διαίρεση του πολυγώνου σε μονότονα τεμάχια και μετά τριγωνοποίηση στο καθένα.

Κορυφή Στροφής



Η κορυφή στροφής είναι εκεί που η κίνηση από την υψηλότερη κορυφή προς τη χαμηλότερη αλλάζει κατεύθυνση (ανοδική σε καθοδική και αντίστροφα)

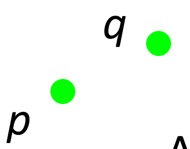
Στην κορυφή *v*

- Και οι δύο γειτονικές κορυφές είναι από κάτω
- Το εσωτερικό του πολυγώνου είναι από πάνω.

Επέλεξε μία διαγώνιο που πάει προς τα πάνω.

Πέντε Τύποι Κορυφών

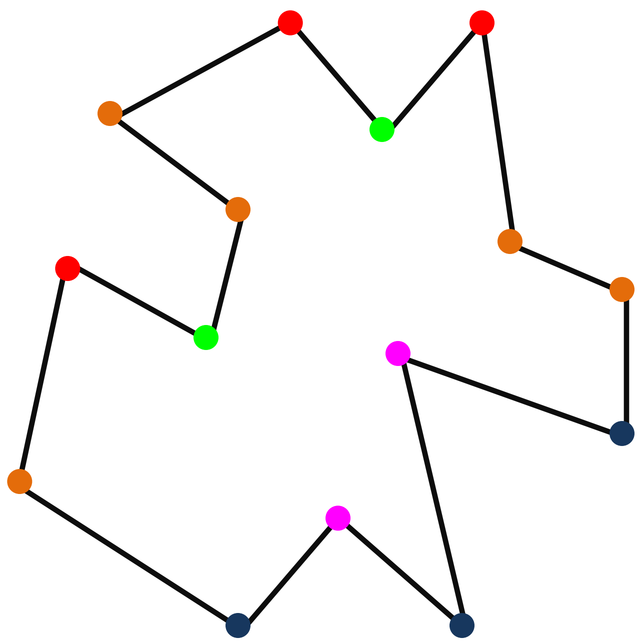
Το σημείο $p = (x, y)$ είναι “κάτω” από ένα διαφορετικό σημείο $q = (u, v)$ αν



$y < v$ ή $y = v$ και $x > u$.

Αλλιώς το p είναι “πάνω” από το q.

4 τύποι κορυφών στροφής



- *Εναρκτήρια κορυφή*: πάνω από τους γείτονες με εσωτερική γωνία < π.
- *Διχαστική κορυφή*: πάνω από τους γείτονες με εσωτερική γωνία > π.
- *Τερματική κορυφή*: κάτω από τους γείτονες και έχει εσωτερική γωνία < π.
- *Συγχωνευτική κορυφή*: κάτω από τους γείτονες με εσωτερική γωνία > π.
- *Κανονική κορυφή*: οι υπόλοιπες (όχι στροφή)

Τι συμβαίνει αν περιστρέψουμε το πολύγωνο κατά π;

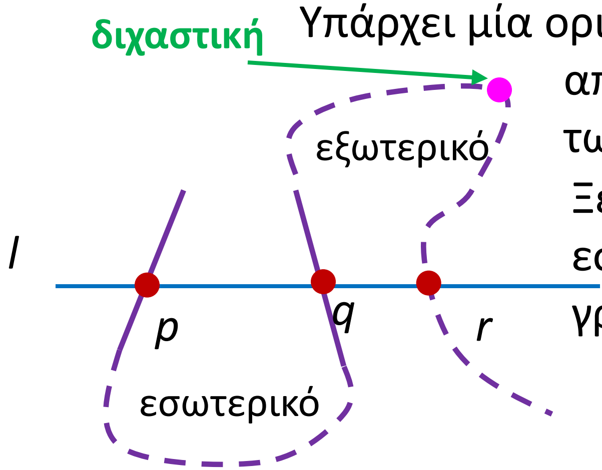
Εναρκτήριες κορυφές ⇔ τερματικές κορυφές   διχαστικές ⇔ συγχωνευτικές

Τοπική μη-μονοτονία

Λήμμα: Ένα πολύγωνο είναι *y*-μονότονο αν δεν έχει διχαστικές και συγχωνευτικές κορυφές.

Απόδειξη Έστω ότι δεν είναι *y*-μονότονο. Θα δείξουμε ότι περιέχει μία διχαστική ή συγχωνευτική κορυφή.

Υπάρχει μία οριζόντια γραμμή *l* που τέμνει το πολύγωνο σε >1 τμήματα, από τα οποία το αριστερότερο είναι ένα τμήμα μεταξύ των *p* (αριστερά) και *q* (δεξιά).



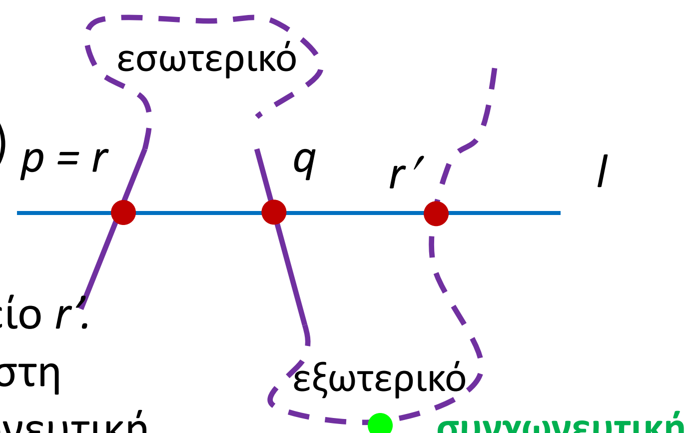
Ξεκίνα στο *q*, διαπέρασε το σύνορο με κατεύθυνση ώστε το εσωτερικό να είναι πάντα αριστερά, και έστω ότι τέμνει την γραμμή *l* στο σημείο *r*.

$r \neq p$

Η υψηλότερη κορυφή στην κίνηση από την *q* στην *r* πρέπει να είναι διχαστική κορυφή.

$r = p$

Διαπέραση κατά ($r' \neq p$) την αντίθετη φορά από *q* και τέμνει τη γραμμή *l* πάλι στο σημείο r'.
Η χαμηλότερη κορυφή στη διαπέραση είναι συγχωνευτική.



Διαχωρισμός σε Μονότονα Τεμάχια

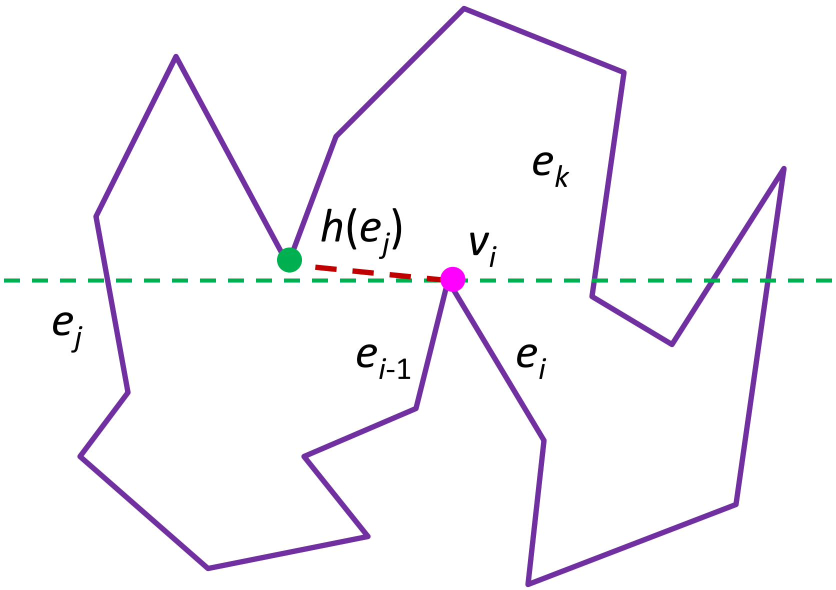
Αυτό το Λήμμα υποδηλώνει ότι το πολύγωνο θα έχει *y*-μονότονα τεμάχια αν οι διχαστικές και συγχωνευτικές κορυφές απομακρυνθούν.

- ♦ Πρόσθεσε μια διαγώνιο καθοδικής κατεύθυνσης από κάθε συγχωνευτική κορυφή.
- ♦ Πρόσθεσε μια διαγώνιο ανοδικής κατεύθυνσης από κάθε διχαστική κορυφή.

Καθοδική επίπεδη σάρωση:

- ★ Τα συμβάντα είναι μόνο οι υπάρχουσες κορυφές.
- ★ Η ουρά συμβάντων υλοποιείται με μία ουρά προτεραιότητας.
  $v_1, v_2, \ldots, v_n$ *(αντιωρολόγια φορά) (ομοίως και οι ακμές* e_i*)*
- ★ Το επόμενο συμβάν εντοπίζεται σε χρόνο $O(\log n)$.

Διαγραφή Διχαστικής Κορυφής



v_i : διχαστική κορυφή

e_j : ακμή στα αριστερά

e_k : ακμή στα δεξιά

$h(e_j)$: χαμηλότερη κορυφή πάνω του v_i και μεταξύ e_{j} και e_k.

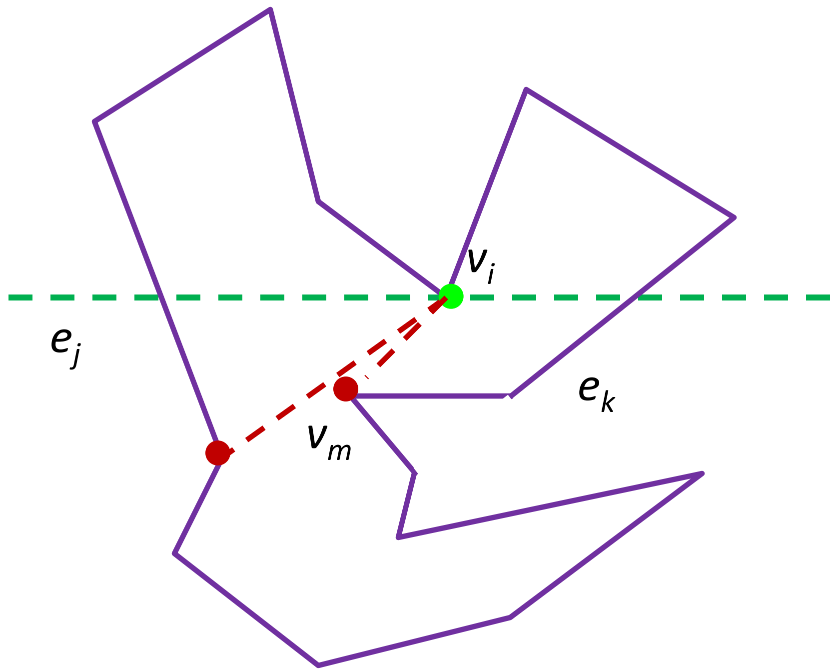
βοηθός(e_j)

ή την πάνω κορυφή της e_j αν τέτοια κορυφή δεν υπάρχει.

Σύνδεσε την v_i με την $h(e_j)$

Διαγραφή Συγχωνευτικής Κορυφής

- Οι συγχωνευτικές μπορούν να αντιμετωπισθούν με τον ίδιο σε μία προς τα πάνω σάρωση όπως οι διχαστικές σε μία προς τα κάτω σάρωση.



- Σε μία σάρωση;;;;

v_i : συγχωνευτική κορυφή

e_j : ακμή στα αριστερά

e_k : ακμή στα δεξιά

v_m : ψηλότερη κορυφή κάτω από τη γραμμή σάρωσης και μεταξύ των e_j και e_k.

Σύνδεσε την v_i με την v_m.

- Η v_m θα αντικαταστήσει την v_i ως βοηθός της e_j.
- Έλεγξε αν ο παλιός βοηθός είναι συγχωνευτική κορυφή και τότε πρόσθεσε μία διαγώνιο. (Η διαγώνιος πάντα προστίθεται αν ο νέος βοηθός είναι διχαστικός).
- Αν ο βοηθός της e_j δεν αντικαθίσταται τότε τη συνδέουμε με το χαμηλότερο άκρο της.

Κατάσταση Γραμμής Σάρωσης

Δυαδικό Δένδρο Αναζήτησης *T*

- Αποθηκεύονται μόνο ακμές στα αριστερά του εσωτερικού του πολυγώνου.

  Ενδιαφερόμαστε μόνο για ακμές αριστερά των διχαστικών και συγχωνευτικών κορυφών

- Οι ακμές αποθηκεύονται από αριστερά προς δεξιά.

- Με κάθε ακμή *e* αποθηκεύεται ο βοηθός του *h*(*e*).

Διπλοσυνδεδεμένος Κατάλογος

- Κατασκευή διπλοσυνδεδεμένου καταλόγου για την αναπαράσταση του πολυγώνου.

n κορυφές + *n* ακμές + 2 έδρες (αρχικά)

- Πρόσθεση στον κατάλογο των διαγωνίων για διχαστικές και συγχωνευτικές κορυφές.

- Οι ακμές στο δένδρο *T* και στον κατάλογο συνδέονται μέσω δεικτών.

Η πρόσθεση μίας ακμής στον κατάλογο κοστίζει $O(1)$ χρόνο.

Αλγόριθμος

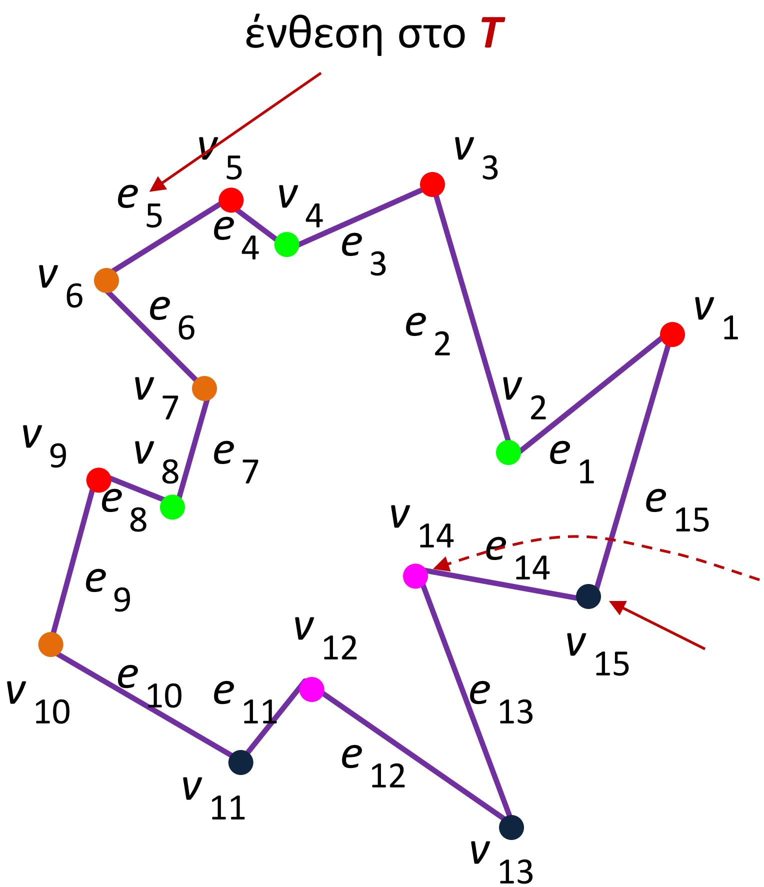
MakeMonotone(***P***)

Input: A simple polygon ***P*** stored in DCEL ***D***.

Output: A partitioning of P into monotone subpolygons stored in ***D***.

```
Q ← priority queue storing vertices of P
T ← ∅   // initialize the status as a binary search tree.
i ← 0
while Q ≠ ∅
   do v_i ← the highest priority vertex from Q  // removal from Q
       case v_i of
            start vertex:  HandleStartVertex(v_i)
            end vertex:   HandleEndVertex(v_i)
            split vertex:   HandleSplitVertx(v_i)
            merge vertex: HandleMergeVertex(v_i)
            regular vertex: HandleRegularVertex(v_i)
       i ← i + 1
```

Διαχείριση Εναρκτήριας και Τερματικής Κορυφής



HandleStartVertex(v_i)

1. $\boldsymbol{T} \leftarrow \boldsymbol{T} \cup \{e_i\}$
2. $h(e_i) \leftarrow v_i$ // βοηθός

HandleEndVertex(v_i)

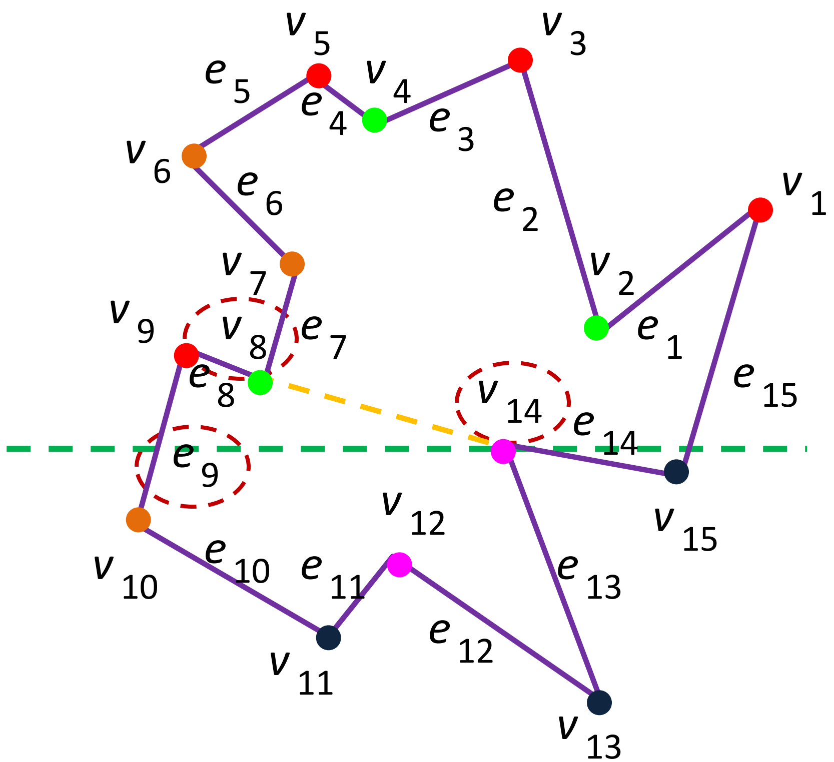
1. if $h(e_{i-1})$ είναι συγχωνευτική
2. then ένθεση διαγωνίου
   που συνδέει v_i με $h(e_{i-1})$
   στο ***D***
3. $\boldsymbol{T} \leftarrow \boldsymbol{T} - \{e_{i-1}\}$

- *Εναρκτήρια κορυφή*: πάνω από τους γείτονες με εσωτερική γωνία < π.
- *Διχαστική κορυφή*: πάνω από τους γείτονες με εσωτερική γωνία > π.
- *Τερματική κορυφή*: κάτω από τους γείτονες και έχει εσωτερική γωνία < π.
- *Συγχωνευτική κορυφή*: κάτω από τους γείτονες με εσωτερική γωνία > π.
- *Κανονική κορυφή*: οι υπόλοιπες (όχι στροφή)

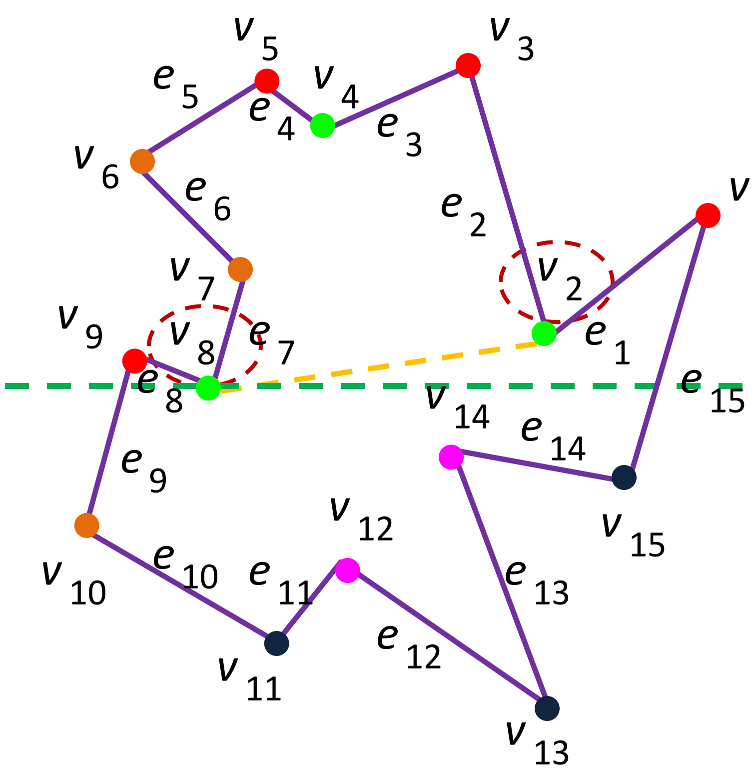
Διαχείριση Διχαστικής Κορυφής

HandleSplitVertex(v_i)

1. Αναζήτηση στο ***T*** για την e_j αμέσως αριστερά της v_i
2. Ένθεση στον ***D*** της διαγωνίου μεταξύ v_i και $h(e_j)$
3. $h(e_j) \leftarrow v_i$
4. $\boldsymbol{T} \leftarrow \boldsymbol{T} + \{e_i\}$
5. $h(e_i) \leftarrow v_i$



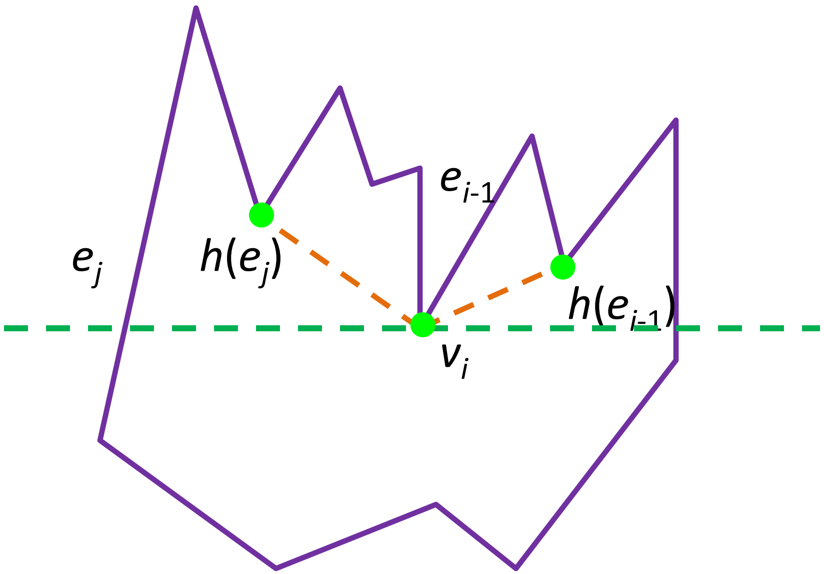
Διαχείριση Συγχωνευτικής Κορυφής



HandleMergeVertex(v_i)

1. if $h(e_{i-1})$ είναι συγχωνευτική
2. then ένθεση της διαγωνίου μεταξύ v_i και $h(e_{i-1})$ στο ***D***
3. ***T*** $\leftarrow$ ***T*** $- \{e_{i-1}\}$
4. Αναζήτηση στο ***T*** για εύρεση ακμής e_j αμέσως αριστερά της v_i.
5. if $h(e_j)$ είναι συγχωνευτική then ένθεση της διαγωνίου μεταξύ v_i και $h(e_j)$ στο ***D***
6. $h(e_j) \leftarrow v_i$

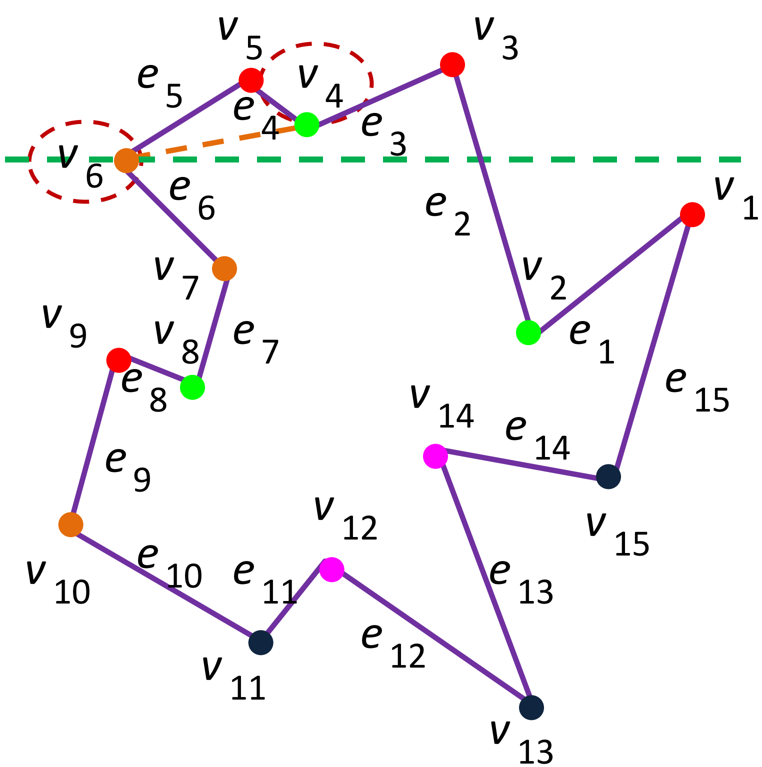
Διαχείριση Συγχωνευτικής Κορυφής



HandleMergeVertex(v_i)

1. if $h(e_{i-1})$ είναι συγχωνευτική
2. then ένθεση της διαγωνίου μεταξύ v_i και $h(e_{i-1})$ στο ***D***
3. ***T*** $\leftarrow$ ***T*** $- \{e_{i-1}\}$
4. Αναζήτηση στο ***T*** για εύρεση ακμής e_j αμέσως αριστερά της v_i.
5. if $h(e_j)$ είναι συγχωνευτική then ένθεση της διαγωνίου μεταξύ v_i και $h(e_j)$ στο ***D***
6. $h(e_j) \leftarrow v_i$

Διαχείριση Κανονικής Κορυφής



HandleRegularVertex(v_i)

if το εσωτερικό του πολυγώνου κείται στα δεξιά του v_i

then if $h(e_{i-1})$ είναι συγχωνευτική

then ένθεση της διαγωνίου που συνδέει την v_i με $h(e_{i-1})$ στον ***D***

$\boldsymbol{T} \leftarrow \boldsymbol{T} - \{e_{i-1}\}$

$\boldsymbol{T} \leftarrow \boldsymbol{T} + \{e_i\}$

$h(e_i) \leftarrow v_i$

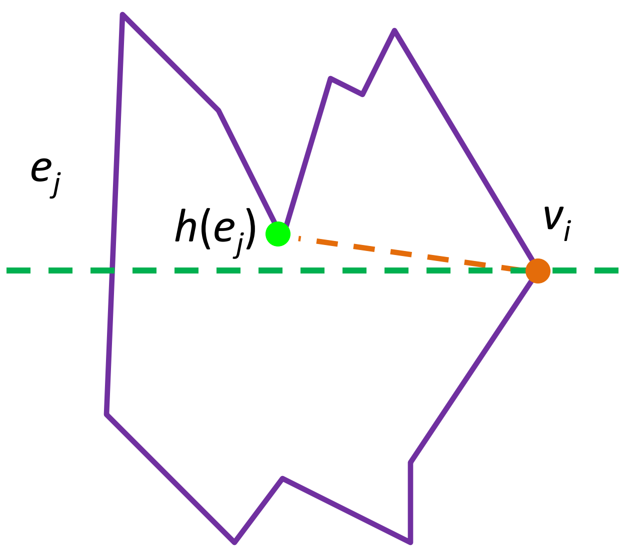
else αναζήτηση στο ***T*** για την ακμή e_j αμέσως αριστερά της v_i.

if $h(e_j)$ είναι συγχωνευτική

then ένθεση της διαγωνίου που συνδέει την v_i με $h(e_j)$ στον ***D***

$h(e_j) \leftarrow v_i$

Διαχείριση Κανονικής Κορυφής



HandleRegularVertex(v_i)

1. if το εσωτερικό του πολυγώνου κείται στα δεξιά του v_i
2. then if $h(e_{i-1})$ είναι συγχωνευτική
3. then ένθεση της διαγωνίου που συνδέει την v_i με $h(e_{i-1})$ στον ***D***
4. $\boldsymbol{T} \leftarrow \boldsymbol{T} - \{e_{i-1}\}$
5. $\boldsymbol{T} \leftarrow \boldsymbol{T} + \{e_i\}$
6. $h(e_i) \leftarrow v_i$
7. else αναζήτηση στο ***T*** για την ακμή e_j αμέσως αριστερά της v_i.
8. if $h(e_j)$ είναι συγχωνευτική then ένθεση της διαγωνίου που συνδέει την v_i με $h(e_j)$ στον ***D***
9. $h(e_j) \leftarrow v_i$

Ορθότητα

Θεώρημα: Ο αλγόριθμος προσθέτει ένα σύνολο μη τεμνόμενων διαγωνίων που διαμερίζει το πολύγωνο σε μονότονα τεμάχια

Απόδειξη: Τα τεμάχια δεν περιέχουν συγχωνευτικές/διχαστικές κορυφές και άρα είναι μονότονα.

Πρέπει να δείξουμε ότι *τα επιπλέον τμήματα είναι διαγώνιες που δεν τέμνονται με ακμές του πολυγώνου ή μεταξύ τους.*

Αποδεικνύουμε τον παραπάνω ισχυρισμό κοιτώντας κάθε τύπο κορυφής ξεχωριστά.

Χρόνος Εκτέλεσης

MakeMonotone(***P***)

Input: A simple polygon ***P*** stored in DCEL ***D***.

Output: A partitioning of P into monotone subpolygons stored in ***D***.

$Q \leftarrow$ priority queue storing vertices of P // $O(n)$

$T \leftarrow \varnothing$ // initialize the status as a binary search tree.

$i \leftarrow 0$

while $Q \neq \varnothing$

  do $v \leftarrow$ the highest priority vertex from Q // removal from Q // $O(\log n)$

    case v_i of // $O(\log n)$ σε κάθε περίπτωση

      start vertex: HandleStartVertex(v_i)

      end vertex: HandleEndVertex(v_i)

      split vertex: HandleSplitVertx(v_i)

      merge vertex: HandleMergeVertex(v_i)

      regular vertex: HandleRegularVertex(v_i)

    $i \leftarrow i + 1$

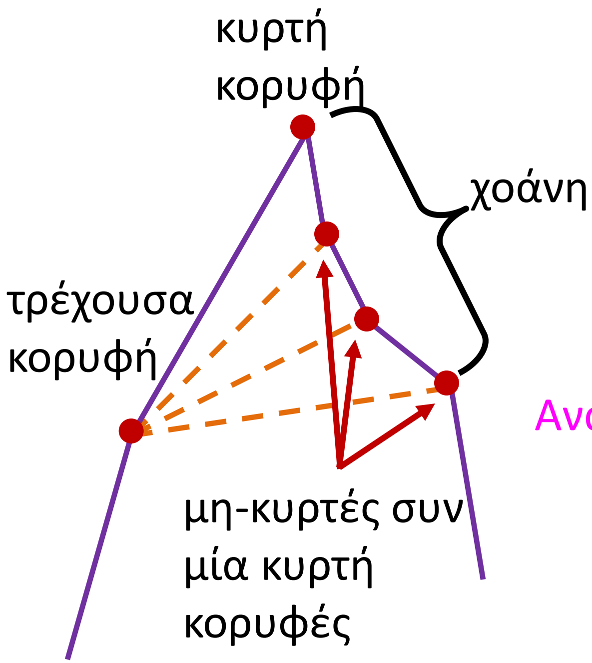
// ερωτήσεις και ενημερώσεις
// σε $O(\log n)$ χρόνο και
// ένθεση μίας διαγωνίου
// στο D σε $O(1)$ χρόνο.

Συνολικός χρόνος: $O(n \log n)$ Συνολικός χώρος: $O(n)$

Τριγωνοποίηση *y*-μονότονου Πολυγώνου

Υπόθεση: Το πολύγωνο είναι αυστηρά *y*-μονότονο (όχι οριζόντιες ακμές).

Επεξεργασία σε φθίνουσα σειρά ως προς *y*.



Μία στοίβα ***S***: κορυφές που έχουμε συναντήσει αλλά μπορεί ακόμα να χρειάζονται διαγώνιες.

Η χαμηλότερη κορυφή στην οροφή της στοίβας.

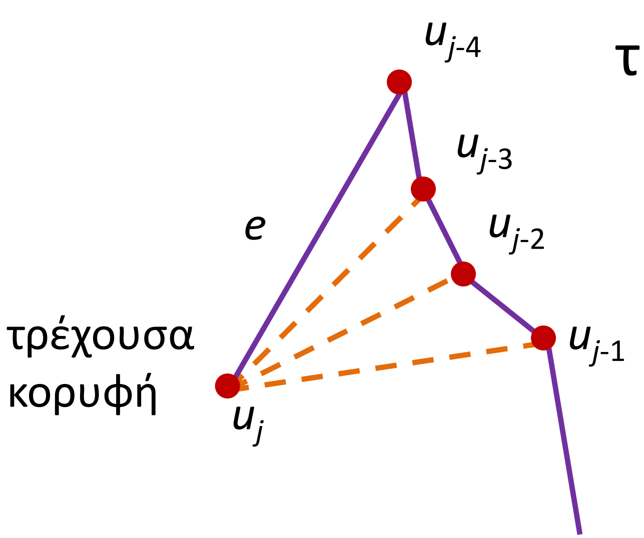
Ιδέα: πρόσθεσε όσες περισσότερες διαγώνιες από την τρέχουσα κορυφή προς αυτές στην στοίβα.

Αναλλοίωτες ιδιότητες στην επανάληψη:

- Ένα σύνορο (πλευρά) της χοάνης είναι μία ακμή του πολυγώνου
- Το άλλο σύνορο είναι μία ακολουθία από μη κυρτές κορυφές (με εσωτερικές γωνίες > π) συν μία κυρτή κορυφή (την υψηλότερη) στο κάτω μέρος της στοίβας.

Περίπτωση 1: Επόμενη Κορυφή στην Απέναντι Αλυσίδα

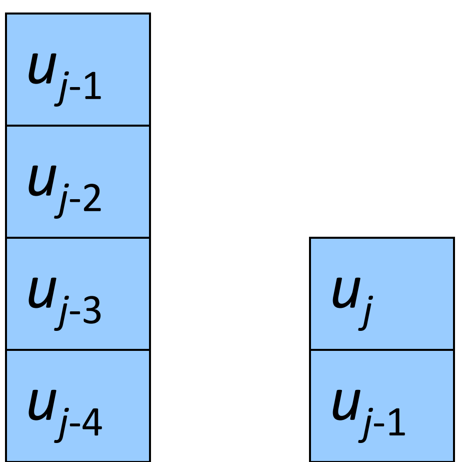
Αυτή η κορυφή θα είναι η χαμηλότερη κορυφή της ακμής *e* που φράσσει την αλυσίδα.



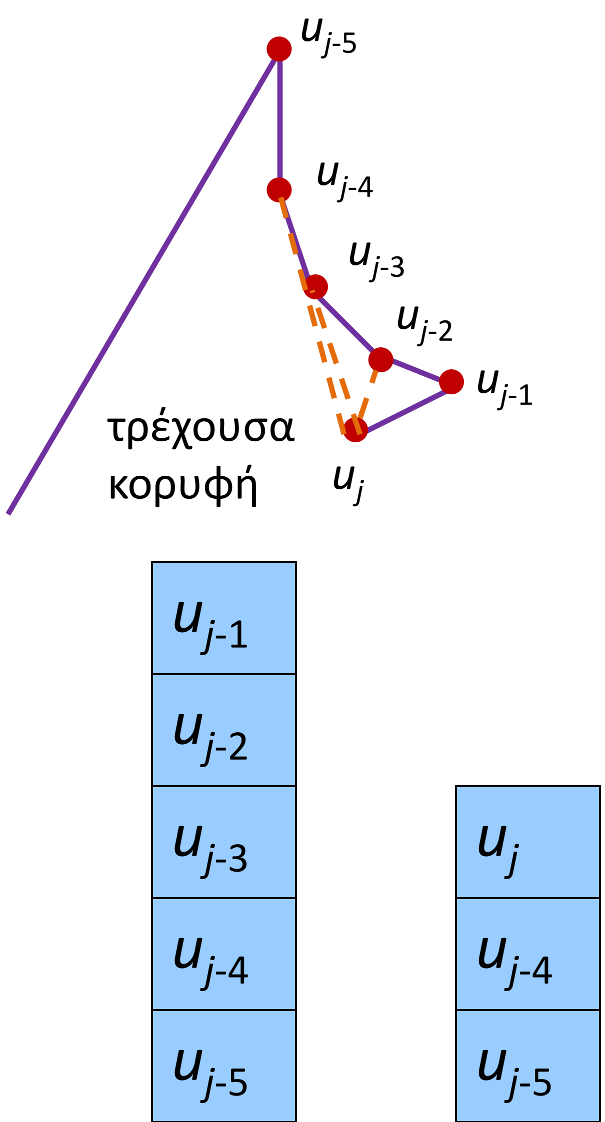
Εξαγωγή των κορυφών από τη στοίβα.

Πρόσθεση διαγωνίων από την τρέχουσα κορυφή σε αυτές (πλην της τελευταίας) με τη σειρά εξαγωγής.

Εισαγωγή της προηγούμενης οροφής της στοίβας και της τρέχουσας κορυφής πάλι στη στοίβα.



Περίπτωση 2: Επόμενη Κορυφή στην Ίδια Αλυσίδα



Οι κορυφές που μπορούν να συνδεθούν στην τρέχουσα κορυφή είναι εντός της στοίβας.

- Εξαγωγή μίας κορυφής από τη στοίβα.

  Υπάρχει μία ακμή με την τρέχουσα κορυφή.

- Εξαγωγή άλλων κορυφών όσο οι διαγώνιες από την τρέχουσα κορυφή είναι στο εσωτερικό.

- Εισαγωγή της τελευταίας εξαχθείσας κορυφής στη στοίβα και έπειτα της τρέχουσας κορυφής.

Ο Αλγόριθμος Τριγωνοποίησης

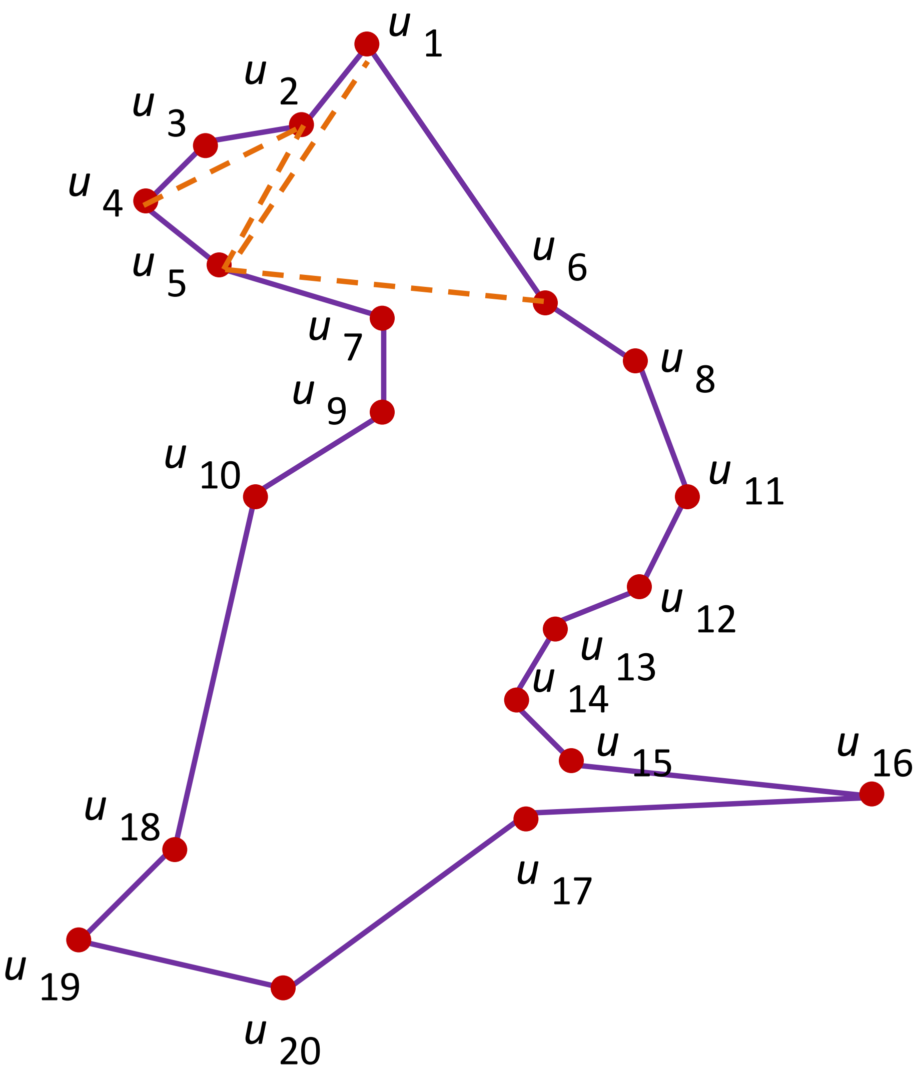
TriangulateMonotonePolygon(***P***)

Input: A *strictly y-monotone* polygon ***P*** stored in DCEL ***D***.

Output: A triangulation of ***P*** stored in ***D***.

```
Merge the vertices on the left and right chains into one sequence
  sorted in decreasing y-coordinate. (In case there is a tie, the one
  with smaller x-coordinate comes first.)
Push(u_1 , S)
Push(u_2 , S)
for j ← 3 to n – 1
    do if u_j and Top(S) are on different chains
         then while Next(Top(S)) ≠ NULL
                  v ← Top(S)
                  Pop(S)
                  insert a diagonal from u_j to v
              Pop(S)
              Push(u_j-1 , S)
              Push(u_j , S)
         else Pop(S)
              while Top(S) is visible from u_j inside the polygon
                  v ← Top(S)
                  insert a diagonal between u_j and v
              Push(v, S)
              Push(u_j , S)
```

Παράδειγμα



Αρχή:

u_2
u_1

j =3:

u_3
u_2
u_1

j =4:

u_4
u_2
u_1

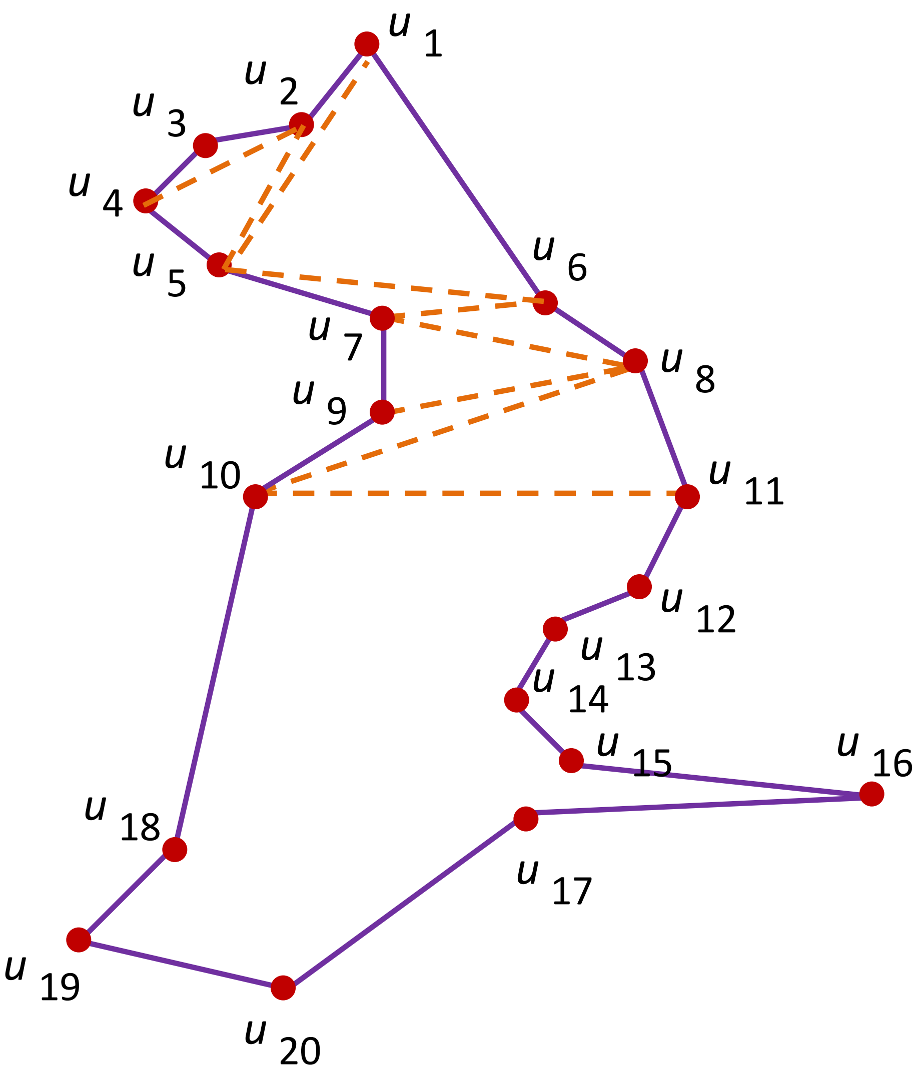
j =5:

u_4
u_5
u_1

j =6:

u_6
u_5

Παράδειγμα



j = 7: u_7, u_6

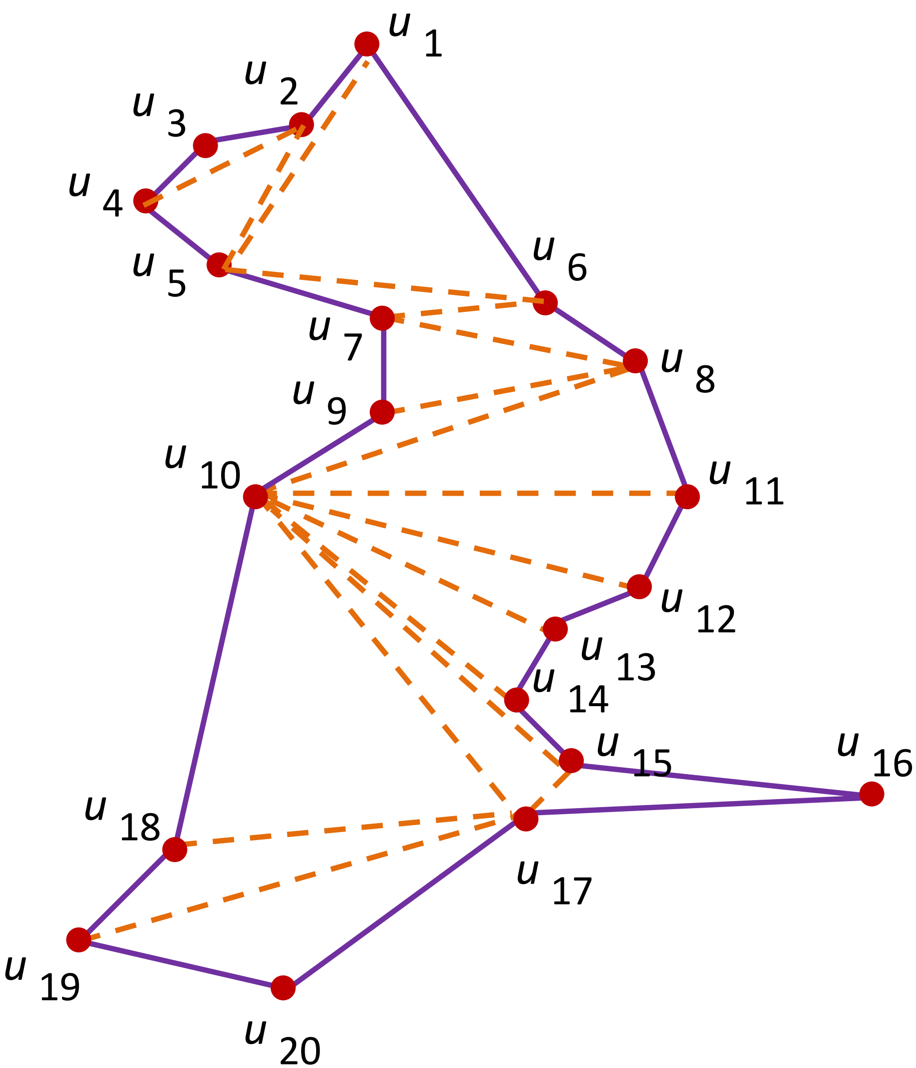
j =8: u_8, u_7

j =9: u_9, u_8

j =10: u_{10}, u_8

j =11: u_{11}, u_{10}

Παράδειγμα



j = 15:

u_{15}
u_{10}

j = 16:

u_{16}
u_{15}
u_{10}

j = 17:

u_{16}
u_{17}
u_{10}

j = 18:

u_{18}
u_{17}

j = 19:

u_{18}
u_{17}

Εκφυλισμός: Αποφυγή της Αυστηρής y-μονοτονικότητας

Η χρονική πολυπλοκότητα είναι $\Theta(n)$.

- #push $\leq$ 2n - 4
- #pop $\leq$ #push

Τί γίνεται αν κάποιες κορυφές έχουν ίδιες τεταγμένες;

- Τις χειριζόμαστε από αριστερά προς δεξιά.
- Είναι σαν να περιστρέφουμε το επίπεδο κατά την ωρολόγια φορά κατά λίγο ώστε κάθε κορυφή να έχει διαφορετική τεταγμένη.

Χρονική Πολυπλοκότητα Τριγωνοποίησης

1. Διαχωρισμός απλού πολυγώνου σε μονότονα τμήματα.

$O(n \log n)$

2. Τριγωνοποίηση κάθε μονότονου τμήματος.

$\Theta(n)$ για όλα τα μονότονα τμήματα μαζί

Θεώρημα: Ένα απλό πολύγωνο μπορεί να τριγωνοποιηθεί σε $O(n \log n)$ χρόνο και $O(n)$ χώρο.