Capitolo 4

Algoritmi di triangolazione

4.1 Introduzione

Per *triangolazione* si intende una suddivisione piana in cui le facce interne sono triangoli. La applicazioni della triangolazione sono principalmente:

- reti di comunicazione;
- discretizzazione per costruzione di modelli di superfici basati su insiemi discreti di punti;
- rapprentazione grafica;
- modellazione di superfici topografiche (digital Terrain Model).

Ad esempio:

- Triangolazione della metrica $l_1 \rightarrow$ routing di circuiti VLSI.
- Triangolazione in 3D $\rightarrow$ modelli volumetrici di oggetti solidi.

4.2 Triangolazione di un insieme di punti del piano

Sia V un insieme di n punti nel piano.

Definizione 23 *Una* triangolazione *di V è un grafo piano con vertici nei punti di V e i cui spigoli sono segmenti di linea retta (*straight-line plane graph*) in cui tutte le facce (eccetto al più* la faccia infinita*) sono triangoli.*

In alternativa, si può anche dare la seguente:

Definizione 24 *Siano:*

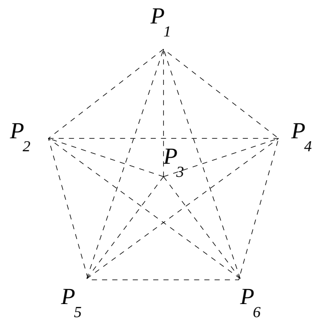
- $V = \{P_1, P_2, \ldots, P_n\}$ *l'insieme dei punti dati;*
- $E^* = \{\overline{P_iP_j} | P_i, P_j \in V, i \neq j\}$, *con* $\overline{P_iP_j}$ *il segmento che unisce* P_i *e* P_j*;*
- $G^* = (V, E^*)$ *il grafo completo definito da* V *e con segmenti di retta come spigoli.*

Allora

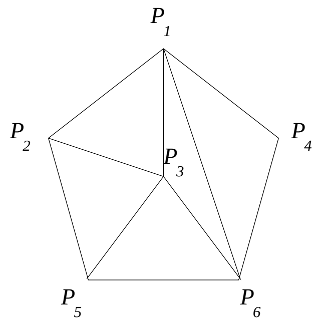
$$T = (V, E)$$

è una triangolazione *di* $V \quad \Leftrightarrow \quad E \subset E^*$*, ed* E *è un sottoinsieme* *<u>massimale</u>* *(cioè contenente il massimo numero di spigoli possibile) di* E^**, tale che due spigoli qualunque di* E *si intersecano al più negli estremi (ovvero* *<u>non</u>* *si intersecano in alcun punto interno).*

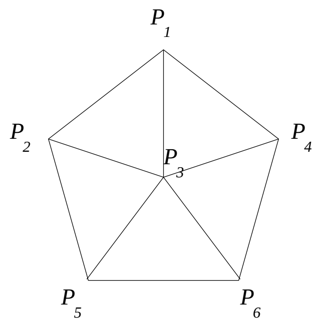
In generale un insieme V di punti ammette pi˘ di una triangolazione, come visibile in figura 4.1.



(a) *Grafo completo* $V = \{P_1, P_2, P_3, P_4, P_5, P_6\}$



(b) *Una possibile triangolazione*



(c) *Un'altra possibile triangolazione (*triangolazione di Delaunay*)*

Figura 4.1: *Esempio di triangolazione.*

4.3 Calcolo di una triangolazione di un insieme di punti

Problema: dato un insieme V finito di n punti nel piano, $V = \{P_1, P_2, \ldots, P_n\}$, costruire una triangolazione T di V.

4.3.1 Idea dell'algoritmo

Costruzione dei triangoli considerando i punti in "sequenza", senza lasciare buchi.

I passo:

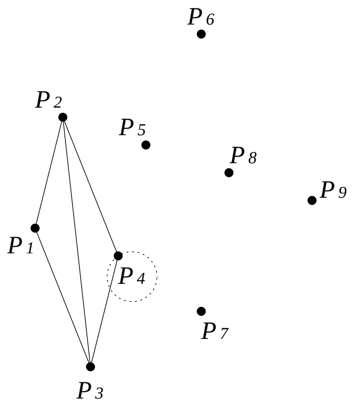
Ordinamento dei punti rispetto ad *una* delle coordinate (ad esempio l'ascissa) $\Rightarrow$ sia $\sigma = [P_1, P_2, \ldots, P_n]$ la sequenza ordinata per ascisse non decrescenti dei punti di V.

II passo:

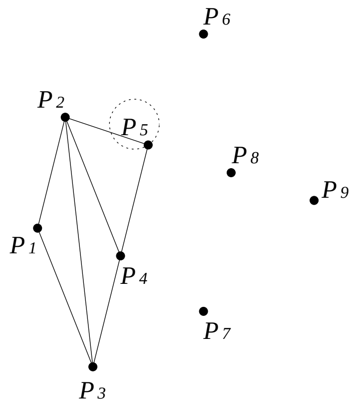
Costruzione della triangolazione

- Inizializzazione: costruire il triangolo t formato dai primi tre punti non allineati della sequenza ordinata.
- $\forall$ nuovo punto P in σ unire P_i alla trangolazione dei punti $\{P_1, P_2, \ldots, P_{i-1}\}$.
- $\forall P_i, i = 4, \ldots, n$, unire P_i a ciascun punto P_j, $j < i$ tale che P_j sia "visibile" da P_i.
  Dove con "visibile" si intende che il segmento $\overline{P_i P_j}$ *non* deve intersecare la triangolazione.

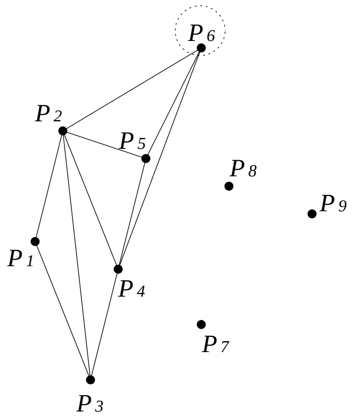
Esempio.



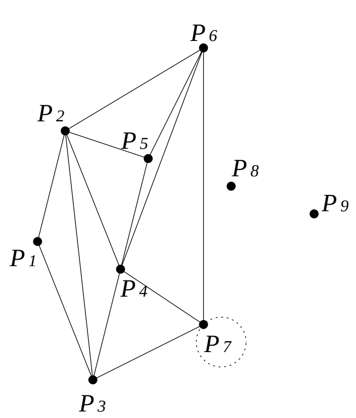
$P_i = P_4$



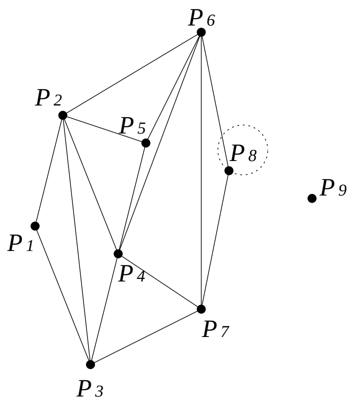
$P_i = P_5$



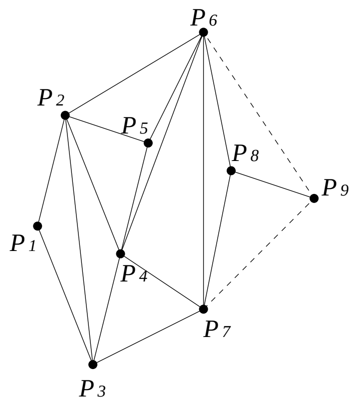
$P_i = P_6$



$P_i = P_7$



$P_i = P_8$



$P_i = P_8$

* * *

$\forall P_i \;\; \rightarrow \;\;$ consideriamo il segmento $\overline{P_iP_j}$, con $j = 1, 2, \ldots, i-1$, ed aggiungiamo $\overline{P_iP_j}$ alla triangolazione $\Leftrightarrow$ non interseca alcuno spigolo preesistente.
Valutiamo la *complessità* di questo approccio: è richiesto il calcolo di $O(e_i \cdot i)$ intersezioni di segmenti; con e_i =numero di spigoli al passo $i-1$ (caso peggiore). Quindi la **complessità** è $O(n^3)$.

4.3.2 Approccio più semplice

L'obiettivo è ridurre la complessità del *II* passo, evitando di esaminare, $\forall$ punto P_i, tutti gli $i-1$ punti precedenti. L'idea è di *muoversi*.

Osservazione n.1 P_j deve appartenere al guscio convesso dei primi $i-1$ punti $\{P_1, P_2, \ldots, P_{i-1}\}$ (nel caso peggiore $O(i)$ segmenti $\overline{P_iP_j}$ da considerare).

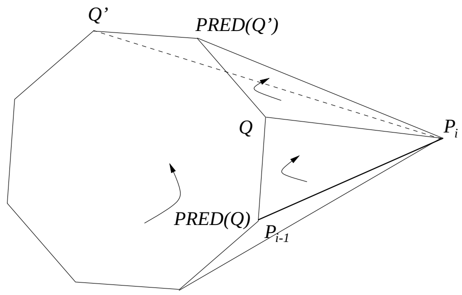
Osservazione n.2 P_{i-1} è sempre *visibile* da P_i (poiché P_{i-1} è il punto più a destra prima di P_i).

Le osservazioni 1 e 2 non riducono la complessità.

Soluzione Muoversi lungo il guscio convesso a partire da P_{i-1} in senso orario e antiorario e unire i punti visibili finché non si incontra (in entrambe le direzioni) un punto non visibile (regola di fermata).

Caso 1

Punti che seguono P_{i-1} nel convex hull (in senso antiorario).

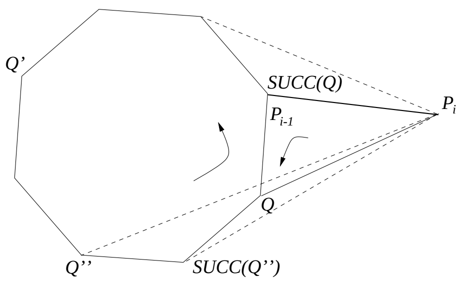


$[P_i, \mathrm{PRED}(Q), Q]$ $\rightarrow$ sequenza oraria (svolta a destra)
$\Rightarrow P_iQ =$ segmento della triangolazione.

$[P_i, \mathrm{PRED}(Q), Q]$ $\rightarrow$ sequenza antioraria (svolta a sinistra)
$\Rightarrow P_iQ$ non va inserito.

Caso 2

Punti sul convex hull che precedono P_{i-1}.



Complessità dell'algoritmo di trangolazione

Passo 1 Ordinamento: $O(n \log n)$ nel caso peggiore;

Passo 2

- ogni spigolo inserito è esaminato *una ed una sola volta:* $O(e)$ passi;
- per ogni punto P_j, $j = 4, \ldots, n$, si esaminano *due* sequenze, che non portano ad inserimento di segmenti: $O(n)$ passi;

Poiché $e \leq 3n - 6 \quad \Rightarrow$ la complessità dell'algoritmo è:

$$O(n)$$

(complessità del secondo passo).

4.4 Triangolazione di un poligono semplice

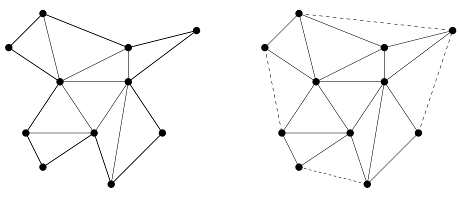
Triangolazione: insieme finito di triangoli tali che, per ogni coppia di triangoli distinti t_1 e t_2

- o t_1 e t_2 sono disgiunti;
- o condividono esattamente un lato e i suoi due vertici estremi;
- o condividono esattamente un vertice;

Triangolazione di un poligono semplice: triangolazione tale che

- l'insieme dei vertici coincide con l'insieme dei vertici del poligono;
- l'unione dei triangoli coincide con la regione poligonale delimitata dal poligono.

Notare la differenza con la triangolazione di un insieme di punti, dove l'unione dei triangoli copre sempre il guscio convesso.



Teorema 1 *Ogni poligono semplice ammette una triangolazione.*

Dimostrazione. Per induzione sul numero n di lati del poligono

- Se $n = 3$, allora la triangolazione ha un solo triangolo coincidente con il poligono stesso
- Se n è maggiore di 3, allora il poligono ammette sempre una diagonale interna. Tale diagonale divide il poligono in due parti, ciascuna delle quali ha un numero di lati strettamente minore di n e quindi (per ipotesi induttiva) ammette una triangolazione. L'unione delle due triangolazioni è una triangolazione del poligono iniziale.

* * *

Proprietà: In particolare ogni poligono con più di 3 lati ammette una diagonale interna tale che uno dei due sottopoligoni in cui è diviso da tale diagonale sia un triangolo.
Tale sottopoligono triangolare è chiamato un *orecchio*.
Tale proprietà è la base dell'algoritmo di triangolazione che vedremo.

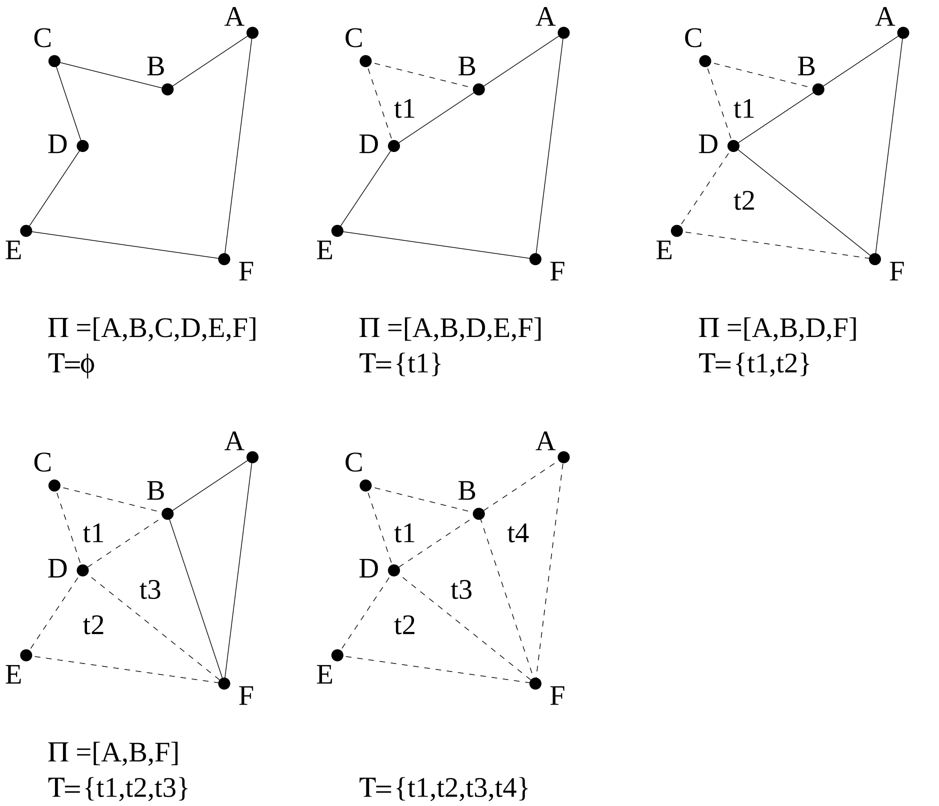
4.4.1 Algoritmo

INPUT: un poligono semplice `Pigreco` di n vertici $V_0, V_1, \ldots, V_{n-1}$.

OUTPUT: una triangolazione `Tau` di `Pigreco`.

Idea: restringo il poligono tagliando via un orecchio alla volta fino a rimanere con un solo triangolo. Ogni orecchio tagliato e il triangolo finale vanno a costituire la triangolazione.

Esempio. Passo per passo:



* * *

4.4.2 Schema generale

```
Tau := empty set;
finche' numero vertici in Pigreco e` maggiore di 3
  trova un orecchio, formato da tre vertici
        consecutivi v1,v2,v3 di Pigreco;
  aggiungi il triangolo v1,v2,v3 in Tau;
  aggiorna Pigreco rimuovendo v2;
aggiungi Pigreco (che e` un triangolo) in Tau;
```

Ricerca dell'orecchio

Esploro in contorno del poligono per esempio in senso antiorario. Per ogni terna di vertici consecutivi v_1, v_2, v_3 del poligono, controllo se il segmento v_1-v_3 è una diagonale interna:

- In caso negativo, proseguo la ricerca spostando la terna v_1, v_2, v_3 in avanti (ogni vi viene sostituito col suo successore).
- In caso positivo, v_1, v_2, v_3 formano un orecchio, e la ricerca termina.

Nota.

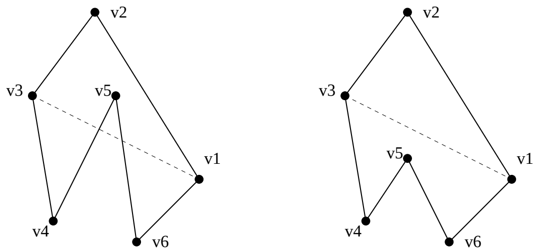
- Dopo avere trovato un primo orecchio, e avere aggiornato triangolazione e poligono, si intraprende una seconda ricerca e così via fino alla condizione di arresto.
- è importante che ogni nuova ricerca parta non dall'inizio del poligono, ma dal punto dove si era arrestata la ricerca precedente: deve partire dalla prima terna non ancora controllata. Se v_1, v_2, v_3 è l'orecchio appena ritagliato (v_2 è il vertice appena cancellato), allora la prima terna non controllata è quella di vertici v_0, v_1, v_3 dove v_0 è il predecessore di v_1.

Riscriviamo l'algoritmo

```
Tau := empty set;
v1 := primo vertice di Pigreco;
finche' numero vertici in Pigreco e' maggiore di 3
  v2 := successore di v1 in Pigreco;
  v3 := successore di v2 in Pigreco;
  if (la terna v1,v2,v3 svolta in senso antiorario) and
     (il segmento v1-v3 \'e diagonale di Pigreco)
  then
     aggiungi il triangolo v1,v2,v3 in Tau;
     aggiorna Pigreco rimuovendo v2;
     v1 := predecessore di v1 in Pigreco;
  else v1 := successore di v1 in Pigreco;
aggiungi Pigreco (che e' un triangolo) in Tau;
```

Sottocomponenti dell'algoritmo

Controllo se il segmento v1-v3 è diagonale Sappiamo già che v_1, v_2, v_3 sono consecutivi sul poligono e definiscono una svolta in senso antiorario. Quindi è sufficiente controllare che nessun vertice di `Pigreco` situato fra v_3 e v_1 (esclusi) sia interno al triangolo $v_1v_2v_3$.



Controllo della posizione di un punto rispetto ad un triangolo Siano A, B, C i tre vertici di un triangolo nel piano, ordinati in modo tale che la terna A, B, C svolti in senso antiorario. Sia P un altro punto.

- P giace dentro ABC se P sta a sinistra di tutte e tre le rette orientate definite da AB, BC, CA.
- P giace fuori ABC se P sta a destra di almeno una delle tre le rette orientate sopra dette.

- P giace sul contorno di ABC se P è allineato con almeno una delle tre rette sopra dette, e giace a sinistra delle rimanenti. In particolare:
  - se P è allineato con una e a sinistra delle altre due allora P giace all'interno di un lato
  - se P è allineato con due e a sinistra della terza allora P coincide con un vertice

4.4.3 Strutture dati

Struttura dati per il poligono

I vertici del poligono sono inizialmente caricati in una *lista circolare linkata doppia.*
Questa lista viene aggiornata man mano che, durante l'algoritmo, si tagliano orecchie del poligono, fino ad arrivare ad una lista con soli tre elementi.
L'uso di una lista circolare linkata doppia consente di avere cancellazione in tempo costante.

Struttura dati per la triangolazione

Utilizziamo una struttura dati per triangolazioni: la *struttura triangle-based.*

- memorizza vertici e triangoli (entita' V e T)
- per ogni vertice:
  - le coordinate x e y
  - il riferimento ad uno dei triangoli incidenti nel vertice (relazione $VT^* = VT$ parziale)
- per ogni triangolo:
  - i riferimenti ai tre vertici (relazione TV)
  - i riferimenti ai tre triangoli adiacenti (relazione TT)

Nota.L'ordine in cui sono memorizzati i tre vertici nella TV e i tre triangoli nella TT devono essere correlati da una opportuna convenzione.

Esempi di convenzioni:

- l'i-esimo triangolo della TT è l'adiacente opposto all'i-esimo vertice della TV;
- l'i-esimo triangolo della TT è l'adiacente lungo il lato avente come estremi l'i-esimo e l'$(i+1)$-esimo vertice della TV.

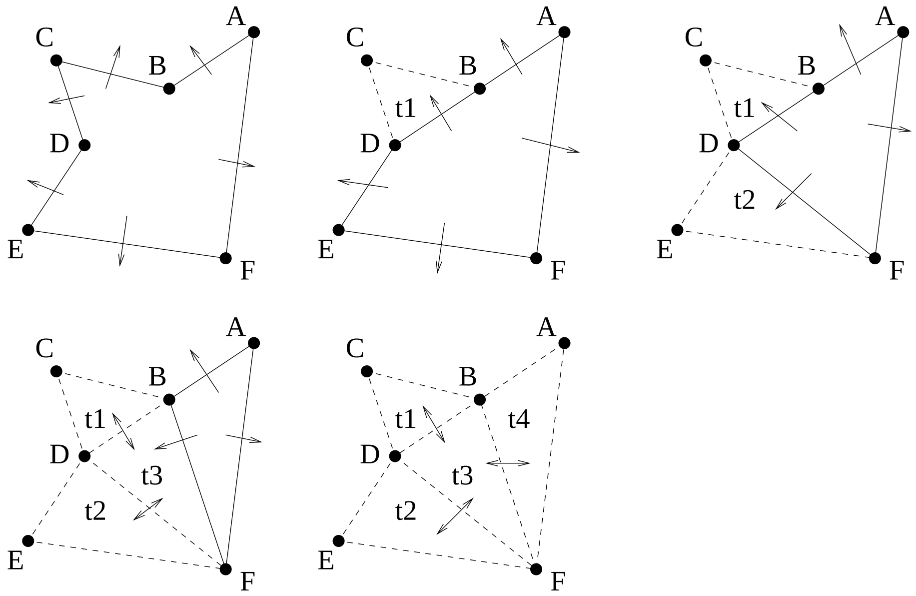
Durante l'algoritmo, ad ogni "taglio di orecchio" si crea un nuovo triangolo t:

- È facile determinare la relazione TV per tale triangolo, in quanto sappiamo i tre vertici.
- È anche possibile determinare la relazione TT per quanto riguarda le adiacenze del nuovo triangolo t con triangoli già creati.

  Il triangolo t condivide due lati con il poligono corrente. t diventa adiacente ai due triangoli (se esistono) che sono già stati tagliati lungo tali lati.

  Per recuperare i suddetti triangoli adiacenti è sufficiente mantenere, collegati ai lati del poligono corrente `Pigreco`, i triangoli creati esternamente adiacenti a tali lati.

Evoluzione dei legami di adiacenza durante l'esempio visto prima (le frecce che puntano a niente sono intese come puntatori nulli):



4.4.4 Complessità temporale

Costo della primitiva punto-in-poligono

Costante in quanto si riduce ad al piè tre chiamate alla primitiva che controlla l'orientamento di una terna di punti, ciascuna chiamata avente un costo costante.

Costo del controllo e'-diagonale interna

Tante chiamate alla punto-in-poligono quanti i vertici giacenti fra v_3 e v_1. Tali vertici sono k-3 se il poligono correntemente ha k lati. Quindi il costo è lineare in k.

Costo per creare un triangolo

Costante. Consiste nell'inizializzare 6 campi per TV e TT. I lavori con cui inizializzarli si determinano in tempo costante.

Costo di un passo del ciclo

Dominato dal costo del controllo diagonale. Ai fini della valutazione posso maggiorarlo con $O(n)$.

Quante volte viene eseguito il ciclo

Per triangolare un poligono di n lati è necessario tagliare n-3 orecchie. Nel caso migliore (poligono convesso) ad ogni passo del ciclo taglio un orecchio, il ciclo termina in n-3 passi. Nel caso generale posso notare che:

- sono eseguiti esattamente n-3 passi con taglio di un orecchio

- non posso avanzare a vuoto piu' di n volte altrimenti significherebbe che non esistono orecchie da tagliare, e questo viola il teorema
- siccome ad ogni taglio di orecchio torno indietro di una terna, devo riavanzare, ma questo non può succedere più di n-3 volte

In conclusione il ciclo viene eseguito un numero di volte maggiorabile con $3n = O(n)$.

4.4.5 Complessità totale

$O(n)$ passi del ciclo, ciascuno con un costo in $O(n)$. Pertanto la complessità totale è in $O(n^2)$. Puo' venire il dubbio che questa valutazione sia troppo pessimistica perché la maggiorazione del costo di un passo del ciclo con $O(n)$ è grossolana.

Facciamo i conti fini nel caso semplificato in cui il ciclo viene eseguito n-3 volte, ogni volta tagliando un orecchio. Il numero di vertici del poligono corrente al passo i-esimo è $n - i$, e pertanto il costo del controllo diagonale è in $O(n - i)$. Quindi il costo totale è dato da: somma per i da 1 a $n - 3$ di $O(n - i)$, il che è comunque in $O(n^2)$. Con questo abbiamo dimostrato che la maggiorazione $O(n)$ per costo di un passo del ciclo non influenza negativamente la valutazione totale.