Dispense di I.U.M. Modellazione Geometrica

© Leila De Floriani, Paola Magillo

Cap.3: Suddivisioni piane

GRAFI PIANI

Grafo (non orientato)

`G = (V,E)`

dove

- `V =` insieme dei vertici di `G`
- `E =` insieme degli spigoli di `G`

Indicheremo con

- `n = #V` il numero di vertici di `G`
- `e = #E` il numero di spigoli di `G`

Rappresentazione di un grafo

Disegno del grafo `G` nel piano in cui i vertici di `G` sono rappresentati da **punti** e gli di spigoli di `G` sono rappresentati da **segmenti di curva**.

Un grafo `G` puo' essere immerso nel piano se `G` puo' essere disegnato in modo tale che nessuna coppia di spigoli si intersechi eccetto che negli eventuali estremi comuni.

Un grafo `G` e' detto **planare** se esiste un'immersione di `G` nel piano.

Un **grafo piano** e' un grafo `G` planare gia' immerso nel piano.

Una **faccia** di un grafo piano `G` e' una regione minimale delimitata da spigoli di `G`.

Notare la differenza:

- Grafo planare = struttura "astratta"
- Grafo piano = disegno nel piano di un grafo planare

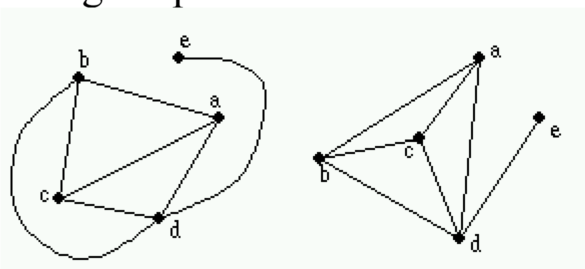
Un grafo planare puo' essere rappresentato da uno o piu' grafi plani.

Esempio di grafo planare

Grafo `G = (V,E)` con

- `V = {a,b,c,d,e}`
- `E = {(a,b), (a,c), (a,d), (b,c), (b,d), (c,d), (d,e)}`

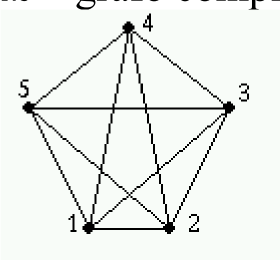
Due grafi piani che costituiscono due immersioni di `G` nel piano:



Nota: la seconda e' una rappresentazione del grafo `G` in cui gli spigoli sono segmenti di retta.

Esempio di grafo non planare

`K5 =` grafo completo con 5 vertici.



FORMULA DI EULERO

La formula di Eulero mette in relazione numero di vertici, spigoli e facce di un **grafo piano connesso**.

Sia `G` un grafo piano connesso. Siano `n`, `e`, `f` rispettivamente il numero di vertici, spigoli e facce di `G`. Allora:

$n - e + f = 2$

Dimostrazione

(Per induzione su e)

Base

La formula e' banalmente vera per un grafo piano connesso con un vertice e zero spigoli ($n=1$, $e=0$, $f=1$), ed e' banalmente vera per un grafo piano connesso con due vertici e uno spigolo ($n=2$, $e=1$, $f=1$).

Passo induttivo

Si suppone che la formula sia vera per un grafo con meno di e archi. Consideriamo due casi:

1. G e' un albero.
   Allora esiste almeno un vertice v in G con grado uguale a 1. Il grafo G' ottenuto da G eliminando v e lo spigolo in esso incidente ha $n-1$ vertici, $e-1$ spigoli ed f facce. Per l'ipotesi induttiva, $(n-1) - (e-1) + f = 2$ da cui $n - e + f = 2$. CVD
2. G non e' un albero.
   Allora esiste almeno un ciclo in G. Sia s uno spigolo appartenente a tale ciclo. Sia G' il grafo ottenuto da G eliminando lo spigolo s. G' e' ancora connesso, ha n vertici, $e-1$ spigoli ed $f-1$ facce. Per l'ipotesi induttiva, $n - (e-1) + (f-1) = 2$ da cui $n - e + f = 2$. CVD

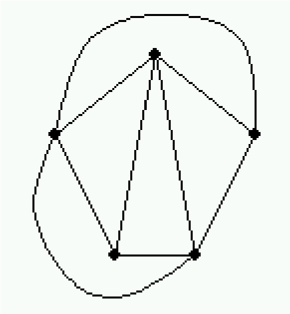
Nota

La formula di Eulero vale anche per poliedri con n vertici, e spigoli ed f facce, poiche' un poliedro e' omeomorfo (o topologicamente equivalente) ad una suddivisione piana.

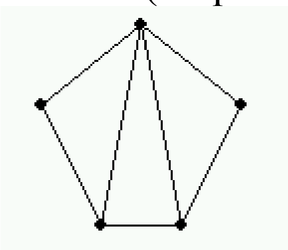
Proprieta' dei grafi piani

Un grafo piano (non necessariamente con spigoli rettilinei) e' detto **massimale** se non e' possibile aggiungere al grafo alcuno spigolo senza perdere la planarita'.

In qualunque grafo piano massimale **ogni faccia** (compresa quella esterna) e' contornata da esattamente **tre** spigoli.



Se imponiamo il vincolo degli spigoli rettilinei, ogni faccia **interna** e' contornata da esattamente tre spigoli. Invece, questo puo' non essere vero per la faccia **esterna** (in quanto puo' non essere possibile inserire spigoli sulla faccia esterna).



Proprieta'

Un grafo planare con n vertici ha O(n) spigoli e facce. Questa proprieta' e' conseguenza dei lemmi 1 e 2.

Lemma 1

Sia G un grafo piano connesso con n vertici ed e spigoli, con $n > 3$.
Allora $e <= 3n - 6$.
Inoltre $e = 3n - 6$ se G e' un grafo piano massimale.

Dimostrazione

Per prima cosa consideriamo il caso che G sia un grafo piano massimale:

- ogni faccia di G e' contornata da tre spigoli
- ogni spigolo di G e' comune a due facce

Percio' $3f = 2e$.

Sostituendo f nella formula di Eulero ($n-e+f=2$), otteniamo $n - e + (2/3)e = 2$, da cui $3n - 6 = e$ per un grafo piano massimale. CVD

Se G non e' massimale, possiamo aggiungere a G nuovi spigoli fino ad ottenere un grafo G' massimale. G' avra' ancora n vertici, ed un numero $e' > e$ di spigoli, e per esso varra' $e' = 3n - 6$. Da cui $e <= 3n - 6$ per un grafo piano qualunque. CVD

Lemma 2

Sia G un grafo piano connesso con n vertici, e spigoli ed f facce.
Allora $f <= 2n - 4$.
Inoltre $f = 2n - 4$ se G e' un grafo piano massimale.

Dimostrazione

(Simile alla dimostrazione del lemma 1)

Si osserva che $3f = 2e$ in un grafo piano massimale. Sostituendo e nella formula di Eulero si ha $f = 2n - 4$ per un grafo piano massimale. E quindi $f <= 2n -4$ per un grafo piano qualunque. CVD

Formula di Eulero per grafi piani non connessi

$$v - e + f = 1 + c$$

dove c e' il numero di componenti connesse del grafo, v,e,f sono il numero di vertici, spigoli e facce.

Traccia di dimostrazione

(Per induzione sul numero c di componenti connesse del grafo)

Base

L'abbiamo gia' dimostrata vera per $c=1$ (grafi piani connessi).

Passo induttivo

Basta osservare che per aggiungere una nuova componente e' necessario aggiungere un vertice.

OPERATORI DI EULERO

Operatori per la costruzione e l'aggiornamento di grafi piani (non connessi).

Permettono di mantenere valida ad ogni aggiornamento la relazione fra le entita' (vertici, spigoli, facce, componenti connesse) espressa dalla formula di Eulero.

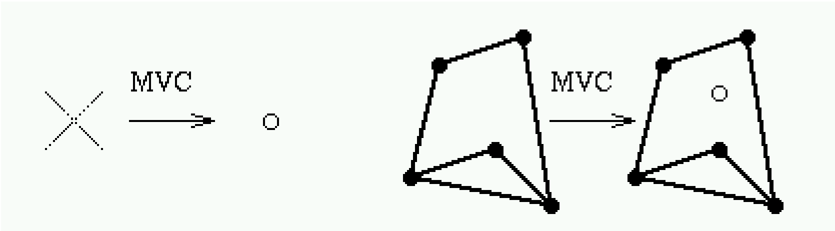
Poiche' la faccia infinita (faccia esterna) e' sempre presente anche in un grafo vuoto (e' data dal piano euclideo), consideriamo la seguente formulazione:

```
v - e + f* = c
```

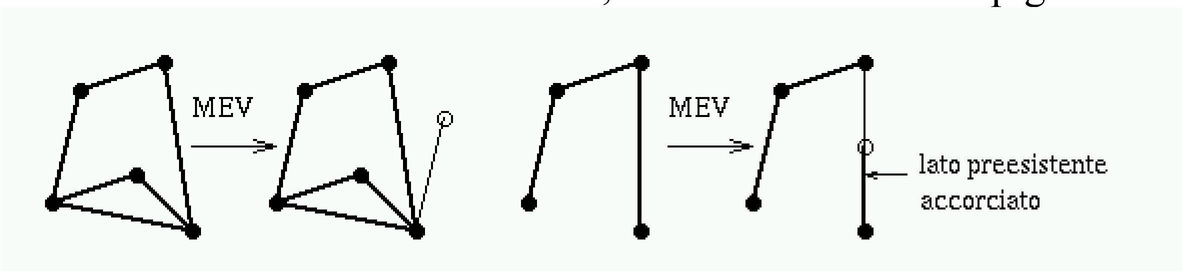
dove `f* = f-1` (il che significa contare solo il numero di facce interne al grafo piano).

Operatori costruttivi

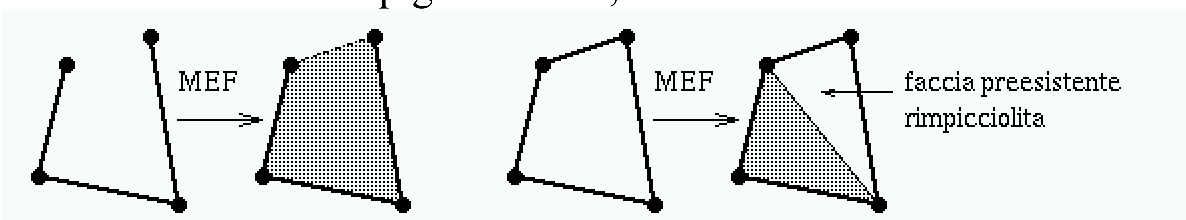
- MVC (Make-Vertex-Component)
  Inizializza una nuova componente connessa inserendo un nuovo vertice.
  Cambia il numero di vertici `v` in `v+1`, cambia il numero di componenti `c` in `c+1`.



- MEV (Make-Edge-Vertex)
  Inserisce un nuovo vertice ed un nuovo spigolo che connette tale vertice ad uno gia' esistente.
  Cambia il numero di vertici `v` in `v+1`, cambia il numero di spigoli `e` in `e+1`.



- MEF (Make-Edge-Face)
  Inserisce un nuovo spigolo che connette due vertici esistenti, ritagliando una nuova faccia da una gia' esistente.
  Cambia il numero di spigoli `e` in `e+1`, cambia il numero di facce `f` in `f+1`.



Operatori inversi (distruttivi)

- KVC (Kill-Vertex-Component)
  Cambia il numero di vertici `v` in `v-1`, cambia il numero di componenti `c` in `c-1`.

- KEV (Kill-Edge-Vertex)
  Cambia il numero di vertici `v` in `v-1`, cambia il numero di spigoli `e` in `e-1`.
- KEF (Kill-Edge-Face)
  Cambia il numero di spigoli `e` in `e-1`, cambia il numero di facce `f` in `f-1`.

SUDDIVISIONI PIANE

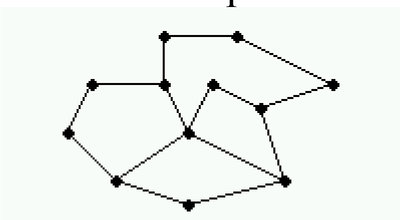
Caso particolare di grafi piani.

Definizione

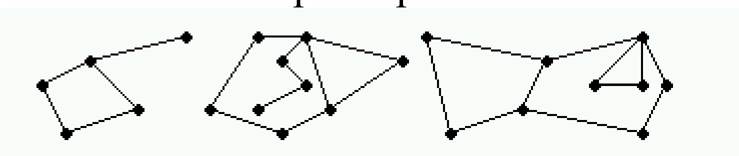
Una **suddivisione piana** e' un grafo piano `Sigma = (V,E,F)` con le seguenti proprieta':

- (A) e' connesso
- (B) in ciascun vertice incidono esattamente due spigoli appartenenti alla stessa faccia
- (C) gli spigoli sono segmenti di retta

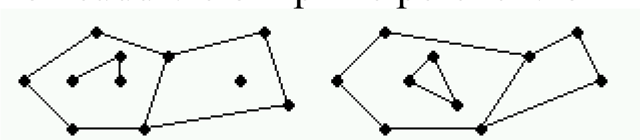
suddivisione piana:



non suddivisioni piane perche' violano condizione (B):



non suddivisioni piane perche' violano condizione (A)



Ne segue che le facce interne di una suddivisione piana sono **regioni poligonali semplicemente connesse**.

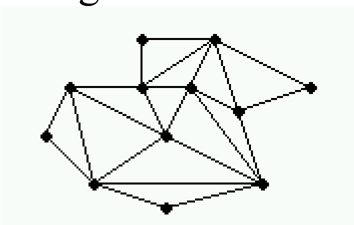
Le condizioni (A) e (B) semplificano le strutture dati per la codifica.
La condizione (C) riflette ipotesi fatte nelle applicazioni (curve sono approssimate con spezzate poligonali).

Triangolazioni

Caso particolare di suddivisioni piane.

Definizione

Una **triangolazione** e' una suddivisione piana in cui tutte le facce interne sono triangoli.



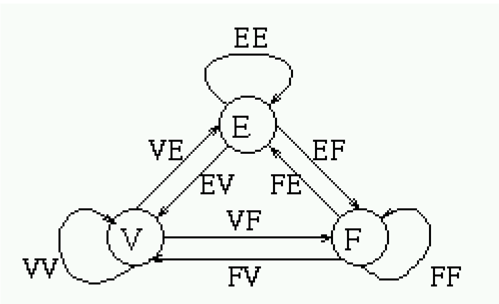
RAPPRESENTAZIONE DI UNA SUDDIVISIONE PIANA

Una **suddivisione piana** `Sigma = (V,E,F)` e' descritta da tre entita' (primitive):

- vertici
- spigoli
- facce

Relazioni di adiacenza fra entita'

Ogni entita' primitiva e' in relazione con se stessa e con ciascuna delle altre due. Si hanno in totale 9 relazioni di adiacenza, secondo il seguente schema:

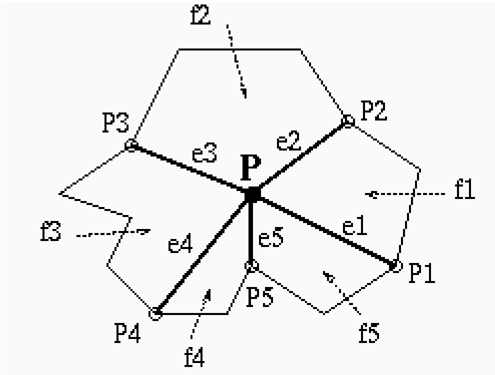


Relazioni basate sui vertici

- VE (Vertex-Edge)
  `P ---> (e1,e2,...er)`
  Ad ogni vertice `P` associa la lista degli `r` spigoli aventi `P` come estremo, ordinata (ad esempio in senso antiorario). Tali spigoli sono detti **spigoli incidenti** in `P`.
- VV (Vertex-Vertex)
  `P ---> (P1,P2,...Pr)`
  Ad ogni vertice `P` associa la lista degli `r` vertici degli spigoli incidenti in `P`,

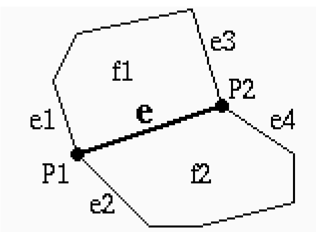
ordinata in modo consistente rispetto alla relazione VE. Tali vertici sono detti **vertici adiacenti** a `P`.

- VF (Vertex-Face)
  `P ---> (f1,f2,...fr)`
  Ad ogni vertice `P` associa la lista delle `r` facce aventi `P` come vertice, ordinata in modo consistente rispetto alla relazione VE. Tali facce sono dette **facce incidenti** in `P`.



Relazioni basate sugli spigoli

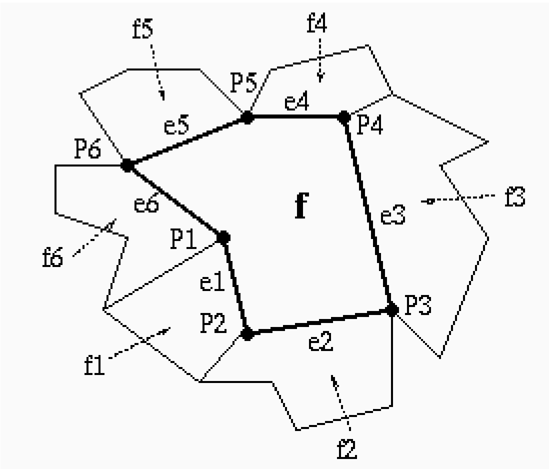
- EV (Edge-Vertex)
  `e ---> (P1,P2)`
  Ad ogni spigolo `e` associa i suoi due vertici estremi.
- EF (Edge-Face)
  `e ---> (f1,f2)`
  Ad ogni spigolo `e` associa le due facce aventi `e` in comune ordinate in modo consistente con la relazione EV: `f1` e' la faccia che giace alla sinistra dello spigolo `e` orientato da `P1` a `P2`, ed `f2` e' quella che giace a destra.
- EE (Edge-Edge)
  `e ---> ( (e1,e2), (e3, e4) )`
  Ad ogni spigolo `e` associa i quattro spigoli incidenti in esso e appartenenti alle due facce che hanno in comune `e`. Tali spigoli sono elencati in ordine consistente con le relazioni EV ed EF:
  - Sia `EV(e) = (P1,P2)` ed `EF(e) = (f1,f2)`
  - `e1,e2` sono gli spigoli incidenti in `P1` ed appartenenti alle facce `f1` ed `f2` rispettivamente
  - `e3,e4` sono gli spigoli incidenti in `P2` ed appartenenti alle facce `f1` ed `f2` rispettivamente



Si veda Preparata-Shamos, Cap.1, per una definizione alternativa della relazione EE.

Relazioni basate sulle facce

- FE (Face-Edge)
  `f ---> (e1,e2,...em)`
  Ad ogni faccia `f` associa la lista degli spigoli che ne costituiscono il contorno, ordinata (ad esempio in senso antiorario).
- FV (Face-Vertex)
  `f ---> (P1,P2,...Pm)`
  Ad ogni faccia `f` associa la lista dei vertici che appartengono al suo contorno, ordinata in modo consistente con la relazione FE.
- FF (Face-Face)
  `f ---> (f1,f2,...fm)`
  Ad ogni faccia `f` associa la lista delle facce adiacenti ad `f` secondo uno spigolo, ordinata in modo consistente con la relazione FE.



Relazioni costanti e relazioni variabili

Una relazione e' detta **costante** quando coinvolge un numero prefissato di elementi, e' detta **variabile** in caso contrario.

In una suddivisione piana **arbitraria**:

- EV, EE, EF sono relazioni costanti
- VV, VE, VF sono relazioni variabili
- FV, FE, FF sono relazioni variabili

In una **triangolazione** (in cui non rappresentiamo la faccia esterna):

- anche FV, FE, FF sono relazioni costanti

In quanto il numero di vertici, spigoli e facce adiacenti ad una qualunque faccia interna e' prefissato.

STRUTTURE DATI PER SUDDIVISIONI PIANE

Una struttura dati per una suddivisione piana deve memorizzare (un sottoinsieme delle)

- entita' topologiche
- relazioni di adiacenza.

Valutazione di una struttura dati per la descrizione di suddivisioni piane in termini di:

- complessita' spazio-temporale degli algoritmi di accesso (che ricavano le relazioni di adiacenza non memorizzate)
- occupazione di memoria

Proprieta' fondamentale

Le relazioni ed entita' memorizzate devono permettere una descrizione non ambigua della suddivisione piana. Vale a dire che deve essere possibile ricavare dalla struttura dati tutte le relazioni di adiacenza e le entita' (memorizzate e non).

Esempio: una struttura dati che codifica solo la relazione EV e' una rappresentazione non ambigua di una suddivisione piana.

Obiettivo

Definire una struttura dati sufficiente e da cui sia possibile

- estrarre in tempo costante le relazioni costanti
- estrarre in tempo O(k) ogni relazione variabile che coinvolge k elementi

Occorre evitare di memorizzare "troppe" relazioni (quelle non necessarie per ottenere l'obiettivo di cui sopra).

Strutture dati per suddivisioni piane

- Struttura Winged-Edge
- Struttura simmetrica (Woo, 1985)

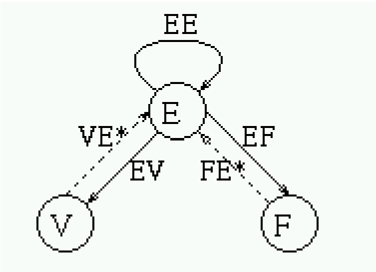
Struttura Winged-Edge

Memorizza:

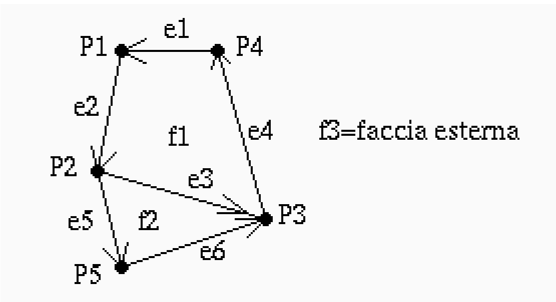
- entita': vertici, spigoli, facce
- relazioni: EV, EE, EF, FE*, VE*

dove FE* e VE* sono **relazioni parziali**:

- FE* associa ad ogni faccia `f` **uno** degli spigoli che la contornano (scelto arbitrariamente)
- VE* associa ad ogni vertice `P` **uno** degli spigoli incidenti in esso (scelto arbitrariamente)



Esempio di struttura Winged-Edge



Entita':

```
V = {P1,P2,P3,P4,P5}
E = {e1,e2,e3,e4,e5,e6}
F = {f1,f2,f3}
```

Relazioni:

```
     EV       EF       EE
e1   P4 P1    f1 f3    e4 e4 e2 e2
e2   P1 P2    f1 f3    e1 e1 e3 e5
e3   P2 P3    f1 f2    e2 e5 e4 e6
e4   P3 P4    f1 f3    e3 e6 e1 e1
e5   P2 P5    f2 f3    e3 e2 e6 e6
e6   P5 P3    f2 f3    e5 e5 e3 e4

     VE*                FE*
P1   e2           f1    e1
P2   e3           f2    e5
P3   e4           f3    e1
P4   e1
P5   e6
```

Complessita' spaziale della struttura Winged-Edge

- per ogni spigolo: tre relazioni costanti, totale 8 informazioni
- per ogni faccia: 1 informazione per FE*
- per ogni vertice: 1 informazione per VE* + le coordinate cartesiane

Spazio richiesto per la memorizzazione delle informazioni topologiche (= le relazioni): `8e + f + n < 27n`.

Spazio richiesto per la memorizzazione delle informazioni geometriche (= coordinate dei vertici): `2n`.

Complessita' spaziale della struttura: O(`n`).

Estrazione delle relazioni non memorizzate dalla struttura Winged-Edge

Relazione FE completa

La otteniamo come combinazione delle relazioni FE* ed EE.

Data una faccia `f`, si vuole ottenere `FE(f)` = lista degli spigoli che contornano la faccia `f`, ordinata in senso antiorario.

Algorimo:

1. Partiamo dallo spigolo `e`, dove `FE*(f) = e`.
2. Troviamo lo spigolo `e'` che segue `e` in senso antiorario sul contorno di `f`. Sia `EE(e) = ((e1,e2), (e3,e4))`. Sono possibili due casi:
   - `EF(e) = (f,f')`: allora `e' = e3`
   - `EF(e) = (f',f)`: allora `e' = e2`
3. Ripetiamo il procedimento dal punto 2 ponendo `e := e'`, fino a ritornare nuovamente allo spigolo iniziale `FE*(f)`.

L'algoritmo esegue `k` volte il punto 2, se `k` e' il numero di spigoli di contorno di `f`. La complessita' temporale e' dunque in O(`k`).

Relazioni FV ed FF

Otteniamo FV come combinazione delle relazioni FE ed EV. Otteniamo FF come combinazione delle relazioni FE ed EF. In entrambi i casi la complessita' temporale e' in O(`k`).

Relazione VE completa

La otteniamo come combinazione delle relazioni VE* ed EE.

Dato un vertice `P`, si vuole ottenere `VE(P)` = lista degli spigoli incidenti in `P`, ordinata in senso antiorario.

Algorimo:

1. Partiamo dallo spigolo `e`, dove `VE*(P) = e`.
2. Troviamo lo spigolo `e'` che segue `e` in senso antiorario girando attorno a `P`. Sia `EE(e) = ((e1,e2), (e3,e4))`. Sono possibili due casi:
   - (a) `EV(e) = (P,P')`: allora `e' = e1`
   - (a) `EV(e) = (P',P)`: allora `e' = e4`
3. Ripetiamo il procedimento dal punto 2 ponendo `e := e'`, fino a ritornare nuovamente allo spigolo iniziale `VE*(P)`.

L'algoritmo esegue `r` volte il punto 2, se `r` e' il numero di spigoli incidenti in `P`. La complessita' temporale e' dunque in O(`r`).

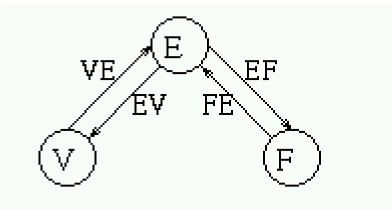
Relazioni VV ed VF

Otteniamo VV come combinazione delle relazioni VE ed EV. Otteniamo VF come combinazione delle relazioni VE ed EF. In entrambi i casi la complessita' temporale e' in O(`r`).

Stuttura simmetrica

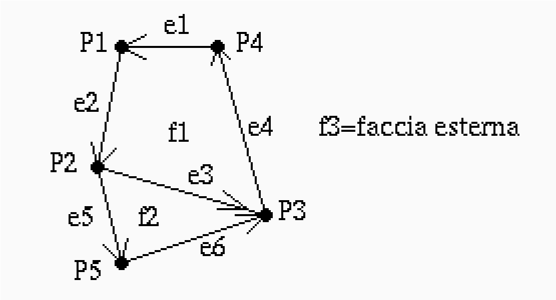
Memorizza:

- entita': vertici, spigoli, facce
- relazioni: VE, EV, FE, EF



E' detta simmetrica in quanto memorizza insieme ad ogni relazione la sua inversa.

Esempio di struttura simmetrica



Entita':

```
V = {P1,P2,P3,P4,P5}
E = {e1,e2,e3,e4,e5,e6}
F = {f1,f2,f3}
```

Relazioni:

```
     EV       EF
e1   P4 P1    f1 f3
e2   P1 P2    f1 f3
e3   P2 P3    f1 f2
e4   P3 P4    f1 f3
e5   P2 P5    f2 f3
e6   P5 P3    f2 f3

     VE                    FE
P1   e2 e1            f1   e1 e2 e3 e4
P2   e5 e3 e2         f2   e5 e6 e3
P3   e4 e3 e6         f3   e1 e2 e5 e6 e4
P4   e4 e1
P5   e6 e5
```

Complessita' spaziale della struttura simmetrica

- per ogni spigolo: due relazioni costanti, totale 4 informazioni
- per ogni faccia: relazione variabile FE
- per ogni vertice: relazione variabile VE + le coordinate cartesiane

Nota:

- ogni spigolo compare nella relazione FE di due facce (le due facce che hanno in comune lo spigolo stesso), percio' il numero totale di elementi in tutte le relazioni FE e' pari a `2e`
- ogni spigolo compare nella relazione VE di due vertici (i due vertici che sono gli estremi dello spigolo stesso), percio' il numero totale di elementi in tutte le relazioni VE e' pari a `2e`

Spazio richiesto per la memorizzazione delle informazioni topologiche (= le relazioni): `4e + 2e + 2e < 24n`.

Spazio richiesto per la memorizzazione delle informazioni geometriche (= coordinate dei vertici): `2n`.

NOTA: se si usa un'implementazione mediante liste linkate delle relazioni variabili FE e VE, l'occupazione di memoria della struttura simmetrica risulta superiore a quella della struttura Winged-Edge.

Complessita' spaziale della struttura: O(`n`).

Estrazione delle relazioni non memorizzate dalla struttura simmetrica

Relazioni basate sulle facce

La relazione FV si ottiene come combinazione delle relazioni FE ed EV. La relazione FF si ottiene come combinazione delle relazioni FE ed EF. Entrambe con complessita' temporale in O(`k`) dove `k` e' il numero di spigoli (ovvero il numero di vertici) sul contorno della faccia in esame. Valgono le stesse considerazioni fatte per la struttura Winged-Edge.

Relazioni basate sui vertici

La relazione VV si ottiene come combinazione delle relazioni VE ed EV. La relazione VF si ottiene come combinazione delle relazioni VE ed EF. Entrambe con complessita' temporale in O(`r`) dove `r` e' il numero di spigoli incidenti nel vertice in esame. Valgono le stesse considerazioni fatte per la struttura Winged-Edge.

Relazione EE

Puo' essere ottenuta come combinazione delle relazioni EV e VE, oppure come combinazione delle relazioni EF ed FE.

Consideriamo, ad esempio, EE come combinazione di EV e VE.

Vogliamo ottenere `EE(e) = ( (e1,e2), (e3,e4) ).`

Algoritmo:

1. Sia `EV(e) = (P1,P2)`
2. Scegliamo `e1` come lo spigolo che segue `e` in `VE(P1)`
3. Scegliamo `e2` come lo spigolo che precede `e` in `VE(P1)`
4. Scegliamo `e3` come lo spigolo che precede `e` in `VE(P2)`
5. Scegliamo `e4` come lo spigolo che segue `e` in `VE(P2)`

Occorre compiere una scansione completa sia della lista `VE(P1)` che della lista

`VE(P2)`.

La complessita' temporale e' quindi in O(`#VE(P1) + #VE(P2)`), e NON e' costante (nonostante EE sia una relazione costante).

Quindi la struttura simmetrica NON e' ottimale.

Struttura dati a liste di adiacenza

E' possibile codificare una suddivisione piana con una struttura dati per grafi (piani), la struttura a liste di adiacenza. Vedremo che tale struttura e' sufficiente a descrivere la suddivisione in modo non ambiguo, ma gli algoritmi di accesso non sono efficienti.

Per ogni vertice `P`, si mantiene la lista ordinata (ad esempio in senso antiorario) dei vertici adiaventi a `P`.

In termini della suddivisione piana, si memorizzano:

- entita': vertici
- relazioni: VV

Complessita' spaziale

O(`e`) = O(`n`) dove `e` e' il numero di spigoli ed `n` e' il numero di vertici. Effettiva occupazione di memoria legata all'implementazione.

Algoritmi di accesso

E' necessario sviluppare algoritmi per

- Generazione degli spigoli, descritti mediante la relazione EV (vale a dire, ogni spigolo come coppia dei suoi vertici estremi)
- Generazione delle facce, descritte mediante la relazione FV (vale a dire, ogni faccia come lista ordinata dei suoi vertici di contorno)
- Estrazione delle relazioni non codificate: VE, VF, EE, EF, FE, FF

Generazione degli spigoli

La generazione degli spigoli richiede un esame dell'intera struttura. L'algoritmo richiede un tempo in O(`e`)=O(`n`).

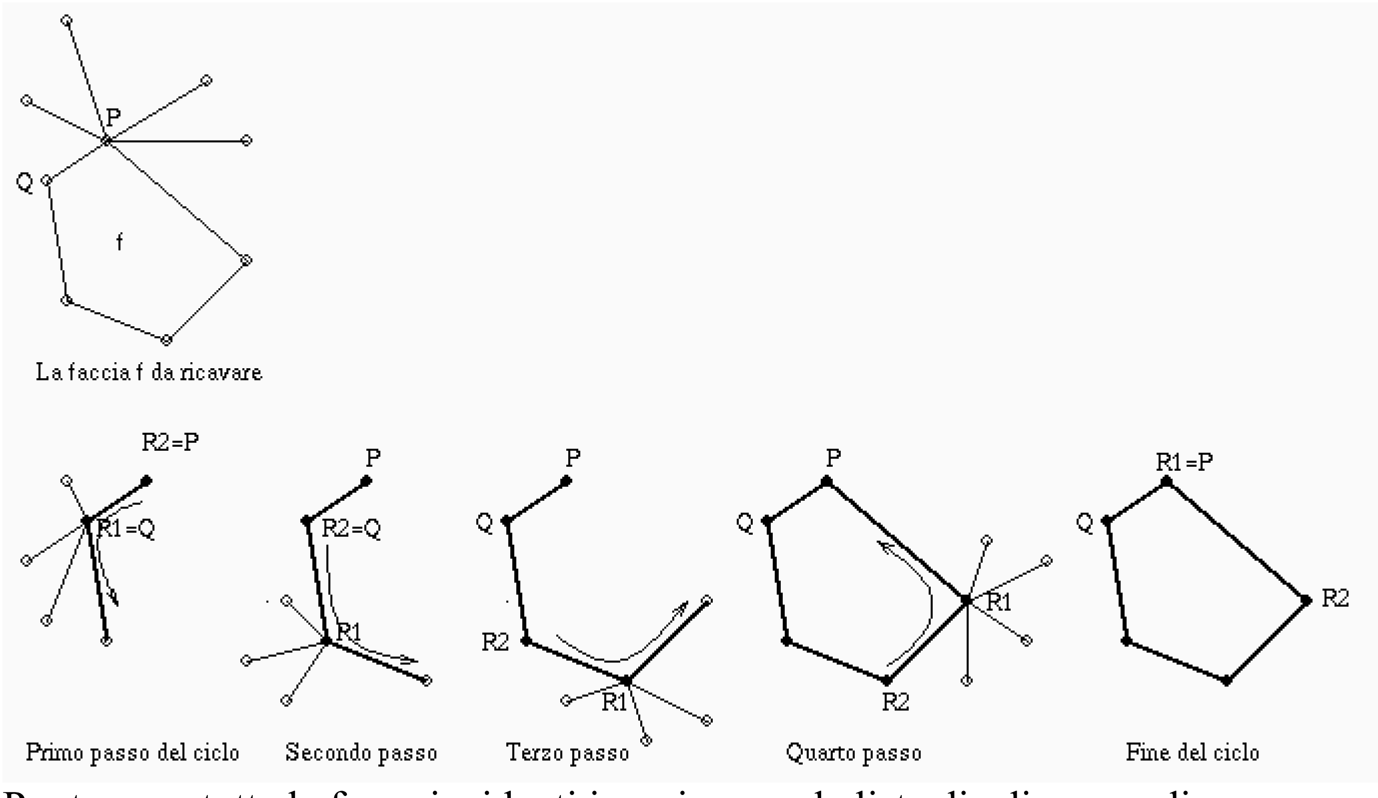
Generazione delle facce

La generazione delle facce richiede di risolvere il seguente sotto-problema:

> Dati un vertice `P` ed un altro vertice `Q` facente parte della lista di adiacenza di `P`, determinare la faccia `f` incidente in `P` e giacente alla sinistra dello spigolo `PQ`, rappresentandola come lista ordinata di vertici `FV(f)` (ad esempio in senso antiorario).

Algoritmo per il sotto-problema:

```
Algorithm FIND_FACE (P,Q)
  print P /* stampa primo vertice di f */
  R2 := P /* vertice appena stampato */
  R1 := Q /* prossimo vertice da stampare */
  while R1 <> P do
    print R1 /* stampa prossimo vertice di f */
    R = predecessore di R2 nella lista di adiacenza di R1
    R2 = R1  /* vertice appena stampato */
    R1 = R   /* prossimo vertice da stampare */
  end while
end Algorithm
```



Per trovare tutte le facce incidenti in `P` si scorre la lista di adiacenza di `P` chiamando l'algoritmo FIND_FACE per ogni vertice `Q` di tale lista.

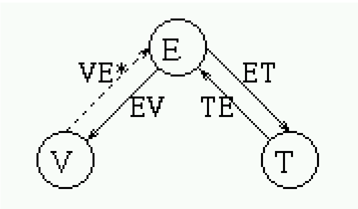
La ricerca di `R2` nella lista di adiacenza di `R1` richiede l'esame di tutta la lista di adiacenza, con una complessita' temporale pari a O(`#VV(R1)`) e questo viene fatto per ogni punto `R1` sul contorno della faccia `f` che stiamo costruendo.

Quindi la complessita' temporale di FIND_FACE e' lineare nella somma delle cardinalita' di `VV(R)`, per tutti i vertici `R` in `FV(f)`). Tale complessita' NON e' ottimale.

Struttura simmetrica "semplificata" per triangolazioni

Memorizza:

- entita': vertici (V), spigoli (E), triangoli (T)
- relazioni: Spigolo-Vertice (EV), Spigolo-Triangolo (ET), Triangolo-Spigolo (TE), Vertice-Spigolo parziale (VE*)



Complessita' spaziale della struttura simmetrica per triangolazioni

- per ogni spigolo: due relazioni costanti, totale 4 informazioni
- per ogni triangolo: relazione costante TE, 3 informazioni
- per ogni vertice: relazione parziale VE* (costante), 1 informazione + le coordinate cartesiane

Spazio richiesto per la memorizzazione delle informazioni topologiche (= le relazioni): `4e + 3t + n <= 12n + 6n + n = 19n`.

Spazio richiesto per la memorizzazione delle informazioni geometriche (= coordinate dei vertici): `2n`.

Complessita' spaziale della struttura: O(`n`). Nota: solo nel caso di triangolazioni, la struttura simmetrica (cosi' semplificata) e' piu' conveniente della Winged-Edge.

Estrazione delle relazioni non memorizzate dalla struttura simmetrica per triangolazioni

Sappiamo che e' possibile ricavare dalla struttura simmetrica generale tutte le relazioni VV, VF, FV ed FF in tempo ottimale.

Vediamo che, nella struttura simmetrica semplificata per triangolazioni:

- VE puo' essere ottenuta da VE* in tempo ottimale (= lineare nel numero di

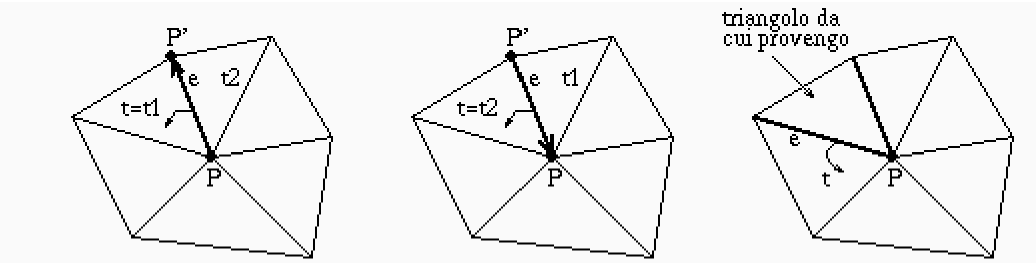
spigoli incidenti nel vertice in esame)
- EE puo' essere ottenuta in tempo ottimale

Estrazione della relazione VE da VE*

Dato un vertice `P`, si vuole ottenere `VE(P) =` lista degli spigoli incidenti in `P` ordinati (ad esempio, in senso antiorario).

Algoritmo:

1. Partiamo con `e = VE*(P).`
2. Sia `ET(e) = (t1,t2).`
   Scegliamo tra questi due il triangolo `t` che ha `e` come lato e che giace alla sinistra di `e` considerando `e` come orientato da `P` verso l'altro suo estremo. Nota bene: non necessariamente `e` e' orientato in questo modo nella struttura dati. Sono possibili due casi:
   - `EV(e) = (P,P')`: allora `t = t1`
   - `EV(e) = (P',P)`: allora `t = t2`
3. Ricaviamo lo spigolo `e'` che segue `e` girando attorno a `P`:
   - `e'` e' lo spigolo che precede `e` in `TE(t).`
4. Iteriamo il procedimento a partire dal passo 2 ponendo `e := e'`, fino a ritornare nuovamente allo spigolo iniziale `VE*(P).`



Primo passo, caso EV(e)=(P,P') Primo passo, caso EV(e)=(P',P) Passi successivi al primo

Nota: nelle iterazioni successive alla prima, la selezione del triangolo `t` al passo 2 si puo' fare in modo piu' semplice: uno fra i triangoli `t1` e `t2` e' stato appena esaminato nell'iterazione precedente, e il nuovo triangolo `t` sara' quello non ancora esaminato.

Se `r` e' il numero di spigoli incidenti in `P`, la complessita' temporale dell'algoritmo e' in O(`r`).

Estrazione della relazione EE

La otteniamo come combinazione delle relazioni ET ed TE (entrambe relazioni costanti).

Dato uno spigolo `e`, vogliamo ottenere `EE(e) = ( (e1,e2), (e3,e4) ).`

Algoritmo:

1. Sia `ET(e) = (t1,t2)`
2. Scegliamo `e1` come quello spigolo che precede `e` in `TE(t1)`
3. Scegliamo `e2` come quello spigolo che segue `e` in `TE(t2)`.
4. Scegliamo `e3` come quello spigolo che segue `e` in `TE(t1)`.
5. Scegliamo `e4` come quello spigolo che precede `e` in `TE(t2)`.

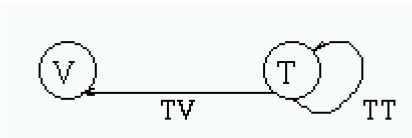
La relazione EE viene ricavata in tempo costante.

Pertanto la struttura simmetrica semplificata e' una struttura dati ottimale per triangolazioni.

Una struttura dati per triangolazioni basata su triangoli

Memorizza:

- entita': vertici (V), triangoli (T)
- relazioni: Triangolo-Vertice (TV), Triangolo-Triangolo (TT)



Nota: gli spigoli NON sono memorizzati esplicitamente.

Per ogni triangolo `t`:

- la lista `TV(t) = (P1,P2,P3)` dei vertici del triangolo `t` e' ordinata in senso antiorario
- la lista `TT(t) = (t1,t2,t3)` dei triangoli adiacenti a `t` e' ordinata in modo compatibile con la lista TV, per esempio:
  - `ti` e' il triangolo adiacente a `t` secondo lo spigolo di estremi `Pi` e `Pj`, dove `j = i mod 3 +1`.

  Un'altra possibile convenzione e':
  - `ti` e' il triangolo adiacente a `t` secondo lo spigolo opposto al vertice `Pi`.

Estrazione delle entita' non memorizzate dalla struttura basata su triangoli

La relazione TE puo' essere estratta dalla struttura in tempo costante:
Per ogni triangolo `t`, si considera la relazione TV e si generano i tre spigoli di `t`, ciascuno come coppia di vertici (ovvero si genera la relazione EV dello spigolo).

Gli spigoli possono essere estratti mediante l'esame di tutti i triangoli.

Ricavare una qualsiasi delle rimanenti 6 relazioni richiede l'esame dell'intera struttura, quindi O(`n`) operazioni nel caso peggiore.

ESERCIZIO
Ricavare una delle relazioni basate sui vertici o sugli spigoli e verificare che la complessita' e' O(`n`) nel caso peggiore.

Complessita' spaziale della struttura basata su triangoli

- per ogni triangolo: due relazioni costanti TV e TT, 6 informazioni
- per ogni vertice: solo le coordinate cartesiane

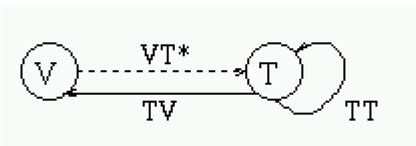
Spazio richiesto per la memorizzazione delle informazioni topologiche (= le relazioni): `6t <= 12n`.

Spazio richiesto per la memorizzazione delle informazioni geometriche (= coordinate dei vertici): `2n`.

Struttura molto economica.

Estensione della struttura basata su triangoli

Memorizza tutte le entita' e relazioni della struttura basata su triangoli, ed in piu' la relazione VT* (= Vertice-Triangolo parziale).



La relazione VT* fornisce, per ogni vertice, **uno** dei triangoli in esso incidenti (scelto in modo arbitrario).

E' possibile ricavare le relazioni VT, VV, VE in tempo ottimale.

Complessita' spaziale della struttura basata su triangoli estesa

- per ogni triangolo: due relazioni costanti TV e TT, 6 informazioni
- per ogni vertice: 1 informazione per VT* + le coordinate cartesiane

Spazio richiesto per la memorizzazione delle informazioni topologiche (= le relazioni): `6t + n <= 13n`.

Spazio richiesto per la memorizzazione delle informazioni geometriche (= coordinate dei vertici): `2n`.

Estrazione della relazione VT da VT*

1. Partiamo con `t = VT*(v)`.
2. Troviamo il triangolo `t'` seguente `t` girando attorno a `P` in senso antiorario. `t'` sara', tra gli adiacenti di `t` (ovvero, fra i triangoli in `TT(t)`), quello che condivide con `t` lo spigolo `PQ` dove `Q` e' il vertice precedente `P` in `TV(t)`. Siano `TV(t) = (P1,P2,P3)` e `TT(t) = (t1,t2,t3)`:
   - Se `P=P1` allora `t'=t3`
   - Se `P=P2` allora `t'=t1`
   - Se `P=P3` allora `t'=t2`
3. Ripetiamo il passo 2 ponendo `t := t'` fino a ritornare al triangolo iniziale `VT*(v)`.

La complessita' per ottenere VT e' lineare nel numero di triangoli incidenti nel vertice `P` in esame.

Estrazione delle ralazioni VV e VE

Si ottengono in modo analogo a VT*. Per la relazione VE, ogni spigolo e' generato come coppia di vertici.

Spigoli e relazioni basate sugli spigoli

L'insieme degli spigoli della triangolazione e' ricavabile in tempo O(`n`) esaminando tutti i triangoli. Ogni spigolo e' restituito come coppia di vertici (cioe' espresso tramite la sua relazione EV). Le relazioni ET ed EE possono essere costruite durante lo stesso processo di estrazione degli spigoli.

Supponendo invece che uno spigolo `e` sia dato come coppia di vertici `e=(v1,v2)`, la relazione ET(`e`) si puo' ricavare prendendo quei triangoli di VT(`v1`) che contengono il vertice `v2` nella loro TV. La complessita' e' in O(#VT(`v1`)). La relazione EE puo' ricavarsi in modo del tutto simile.