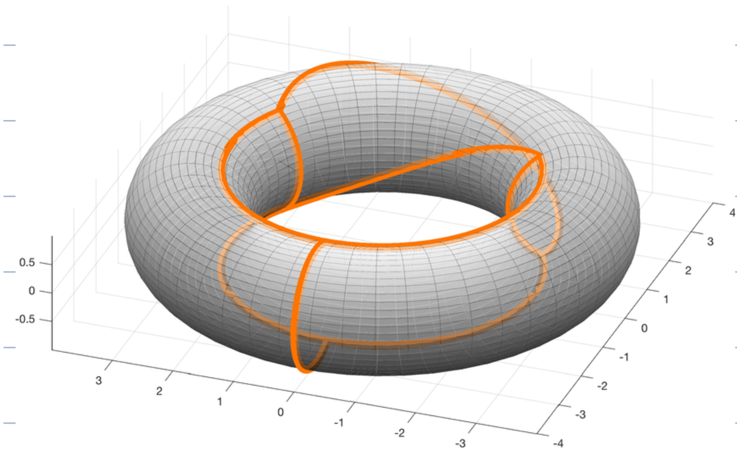
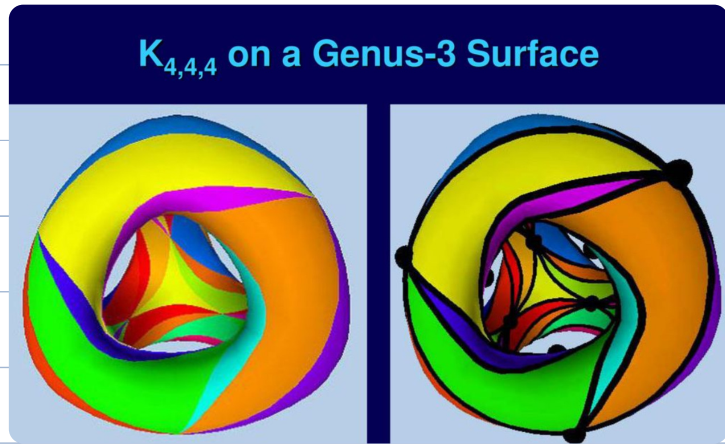
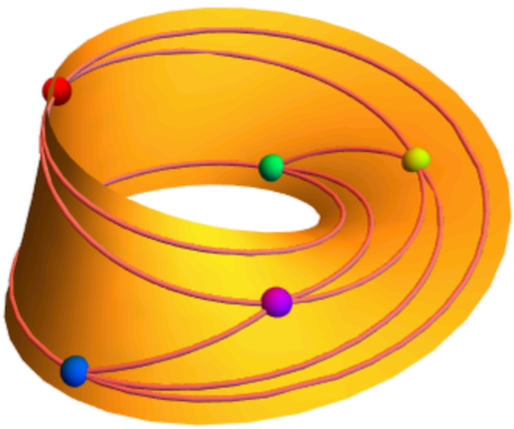
Hoy:

- Gráficas planas.

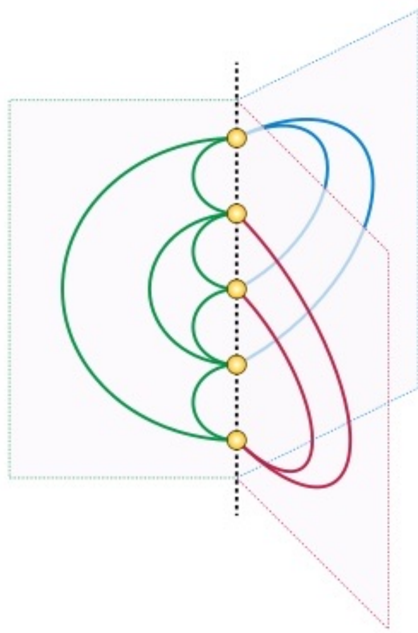
K_{33}

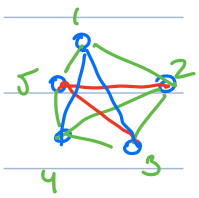




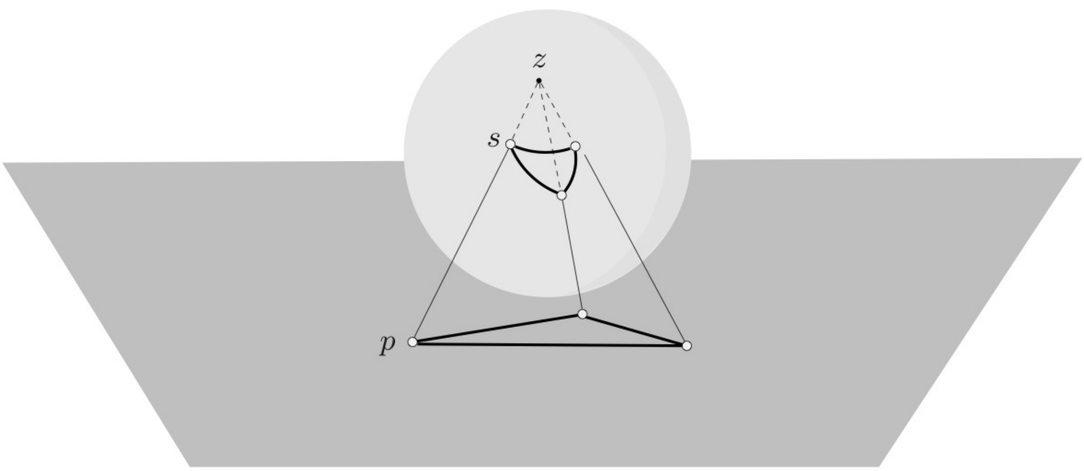


K_5





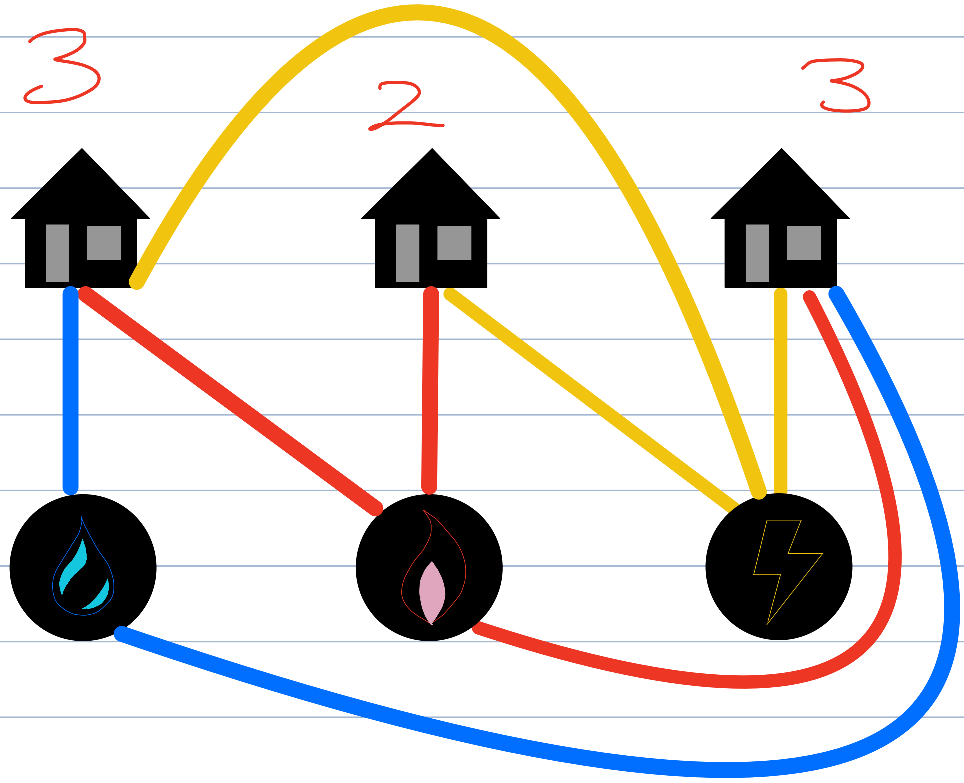
K_5



K_3

El problema de los <u>tres</u> servicios.

Dadas tres casas y tres servicios se desea conectar a cada casa con cada uno de los servicios, de tal forma que las líneas (curvas o rectas) que se usen para conectarlos no se auto-intersecten.



En la gráfica de arriba notamos que es imposible insertar la arista faltante sin generar un cruce.

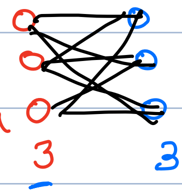
¿Cómo podemos demostrarlo?

$K_{3,3,3}$ $K_{n,n}$

$V = V_1 \cup V_2$ tq $V_1 \cap V_2 = \emptyset$ $|V_i| = 3$

Teorema: $K_{3,3}$ no es aplanable.

no la puedo dibujar en el plano sin que haya cruces entre sus aristas

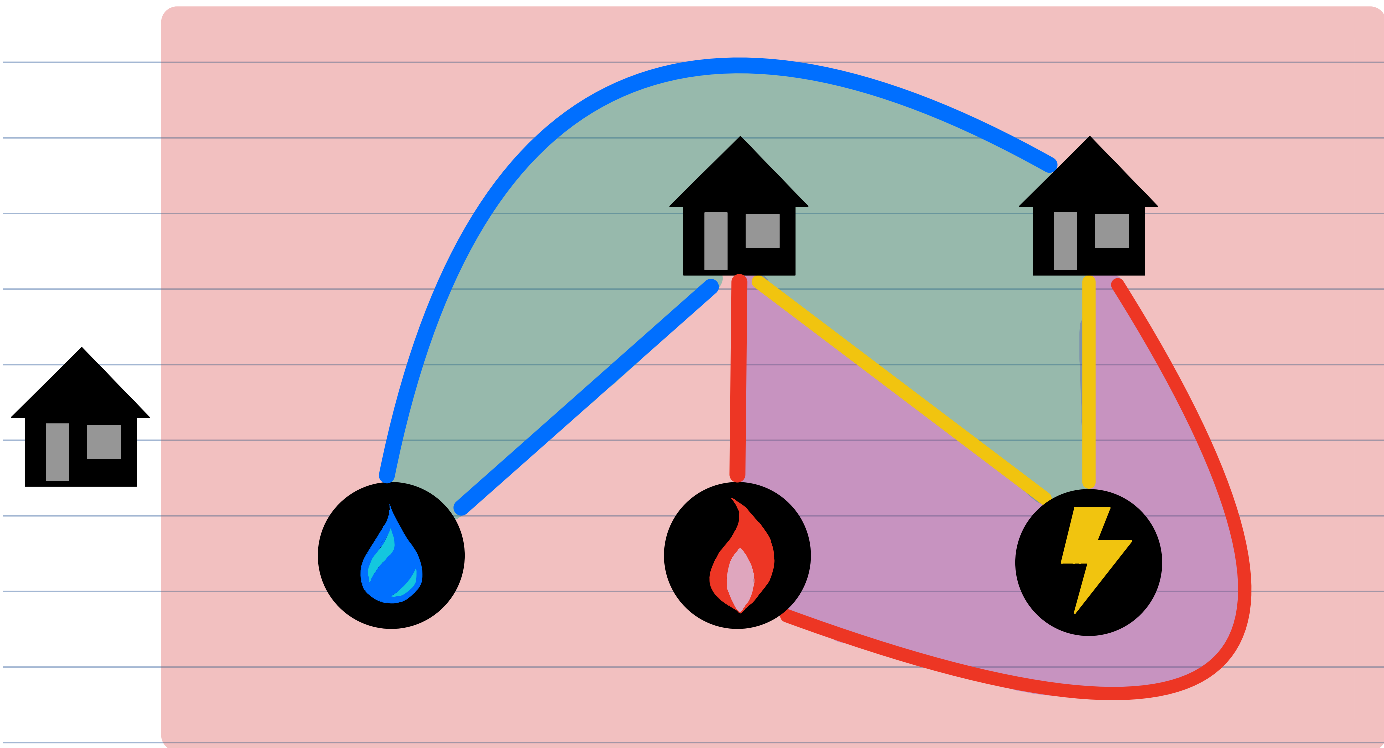


Bipartita

$E = \{(e_1, e_2) \mid e_1 \in V_1,\ e_2 \in V_2\}$

Demostración:

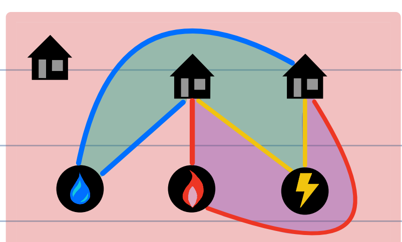
Consideremos la partición del plano generada por el siguiente dibujo de $K_{2,3}$.



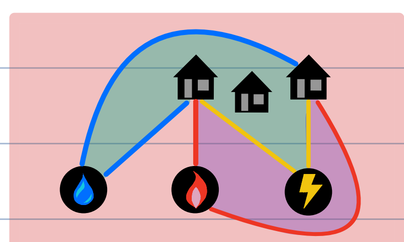
Dicha partición divide al plano en tres regiones, en el dibujo se muestran de tres colores distintos.

Para completar $K_{2,3}$ a $K_{3,3}$ necesitamos agregar la tercera casita, llamemosle C_3.

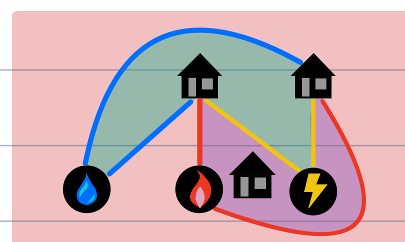
Hay tres opciones para colocar C_3: en la región rosa, morada o verde:



Para conectarla a la electricidad hay que cruzar una arista roja o una arista azul.

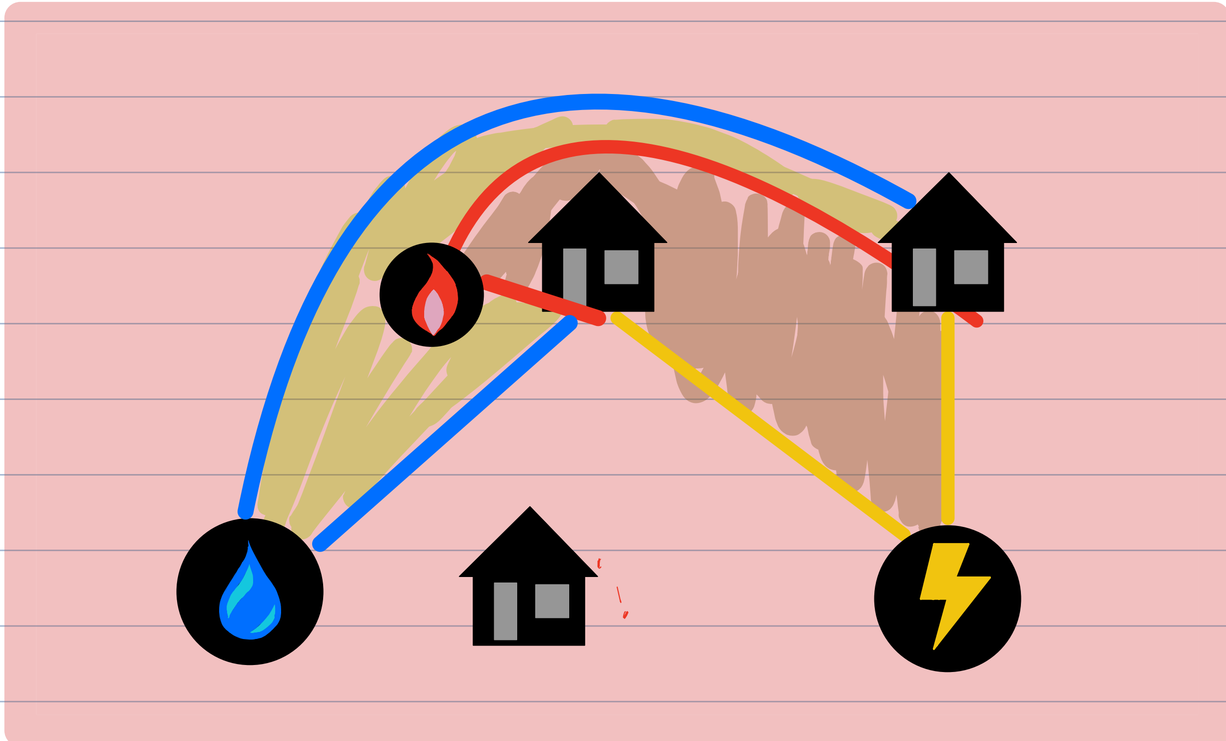


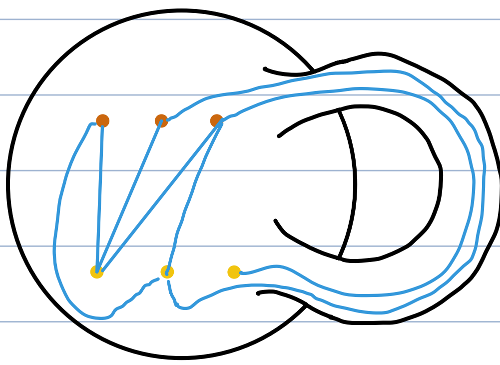
Para conectarla al gas hay que cruzar una arista amarilla o una arista azul.



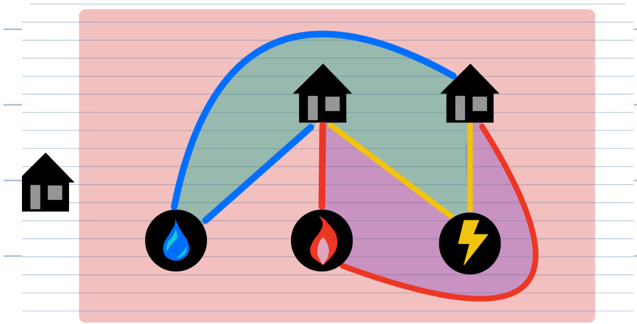
Para conectarla al agua hay que cruzar una arista amarilla o una arista roja.

∎

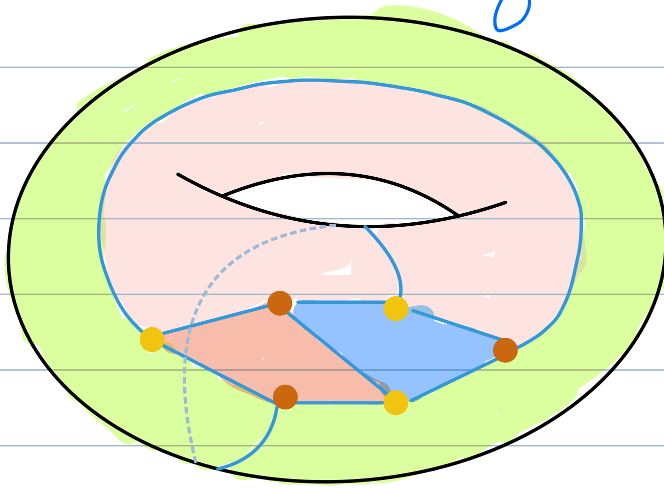




$K_{3,3}$ se puede encajar en el toro = taza.



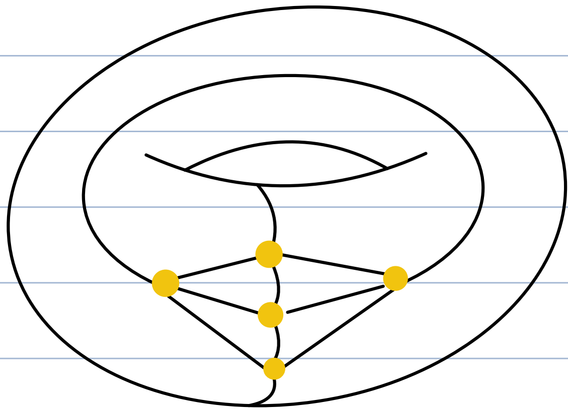
3 regions.



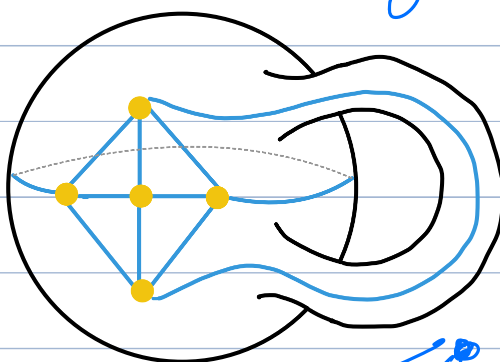
4 regiones.

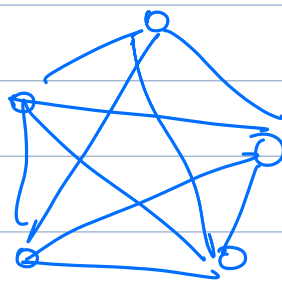
K_5 se puede encajar en el toro.

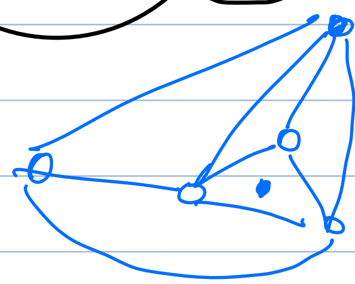
taza.



isomorfas
=







Encajes en el plano

Definición. Decimos que una gráfica es encajable en el plano, o aplanable (planar) si puede dibujarse en el plano de tal forma que sus aristas únicamente se intersecten en los vértices.

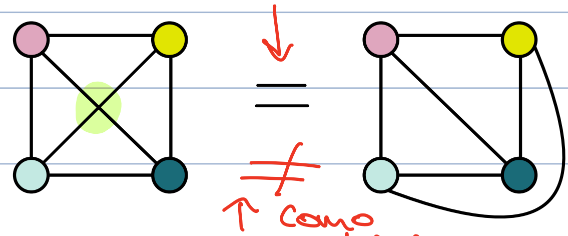
$K_{3,3}$ no es aplanable.
K_5 no es aplanable.

A un dibujo con tales características le llamamos encaje plano.

$G=(V,E)$ ← gráfica abstracta. K_4 $\quad V=\{v_1,v_2,v_3,v_4\}$
$E=\{\{v_i,v_j\} \mid i\neq j\}$.

como gráfica

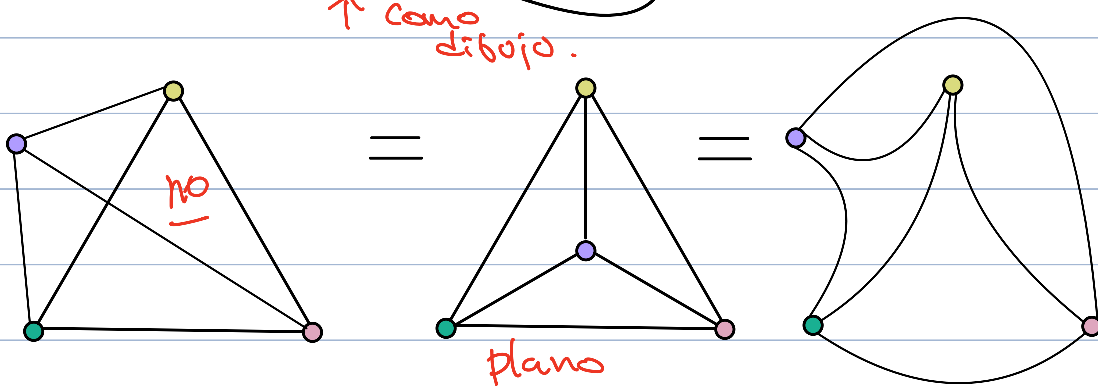
no es un encaje plano



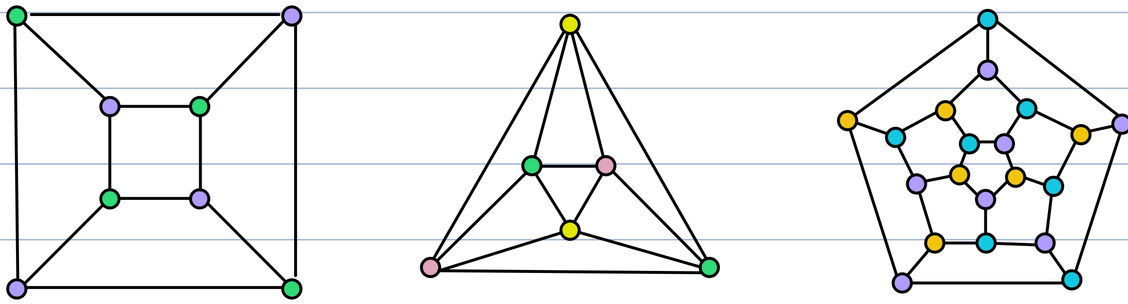
como dibujo.

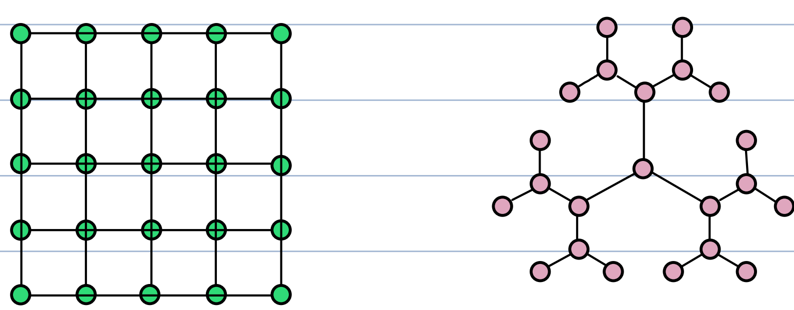
es un encaje plano.

plano

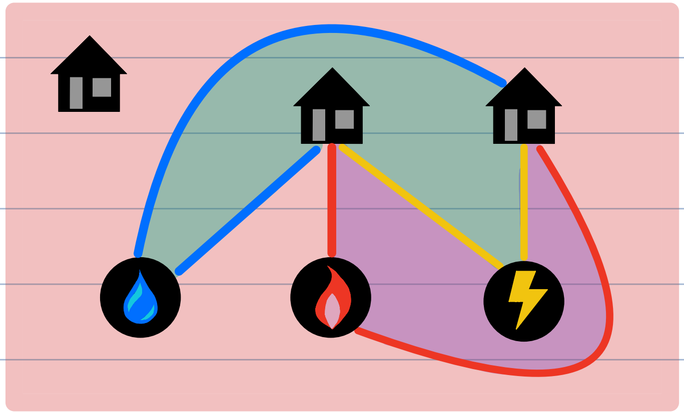


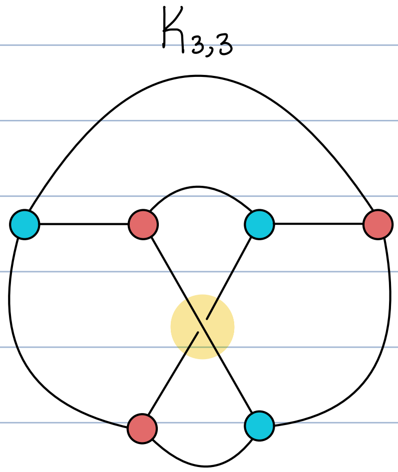
plano



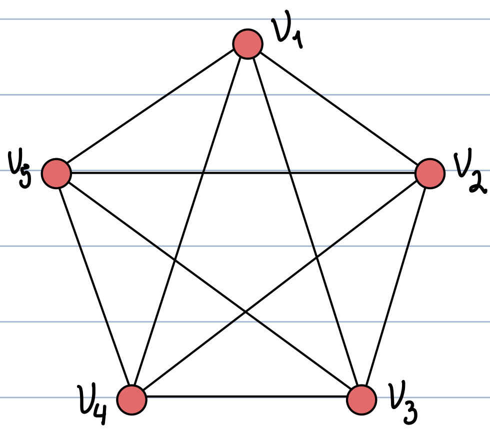


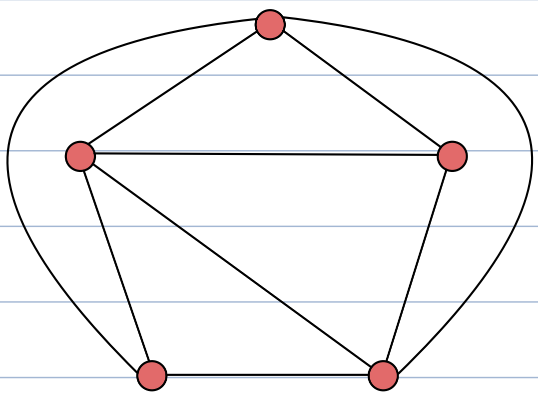
No todas las gráficas son aplanables.

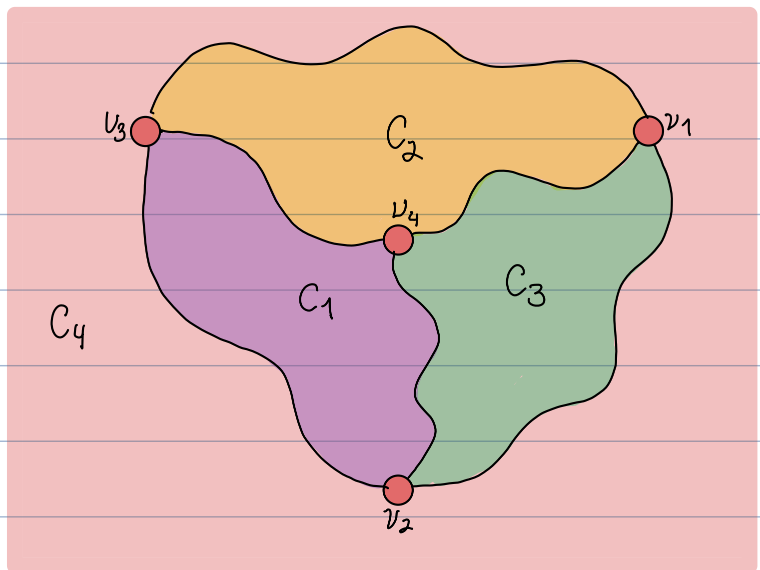




Teorema. K_5 no es aplanable.



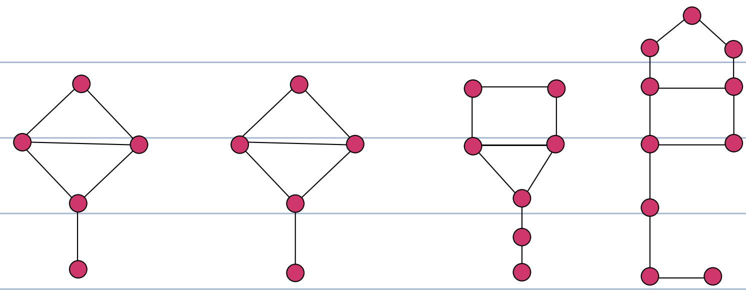




Caracterización de las gráficas aplanables

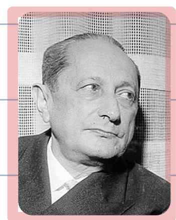
Definición. Una gráfica H es una subdivisión de una gráfica G si:

- H = G o
- H puede obtenerse insertando vértices de grado 2 en las aristas de G.



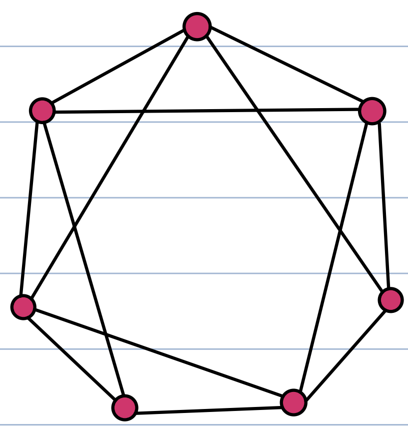
G

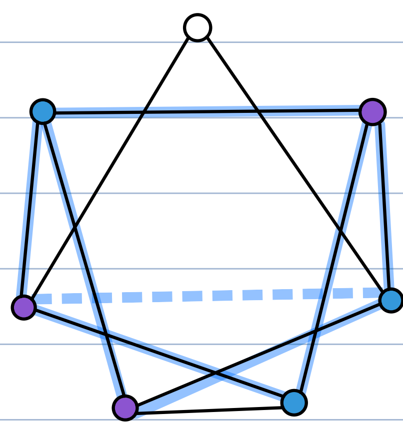
Teorema (Kuratowski)



Matemático polaco.
1896 – 1980

Una gráfica G es aplanable si y sólo si G no contiene ninguna subgráfica que sea una subdivisión de K_5 o de $K_{3,3}$. ∎





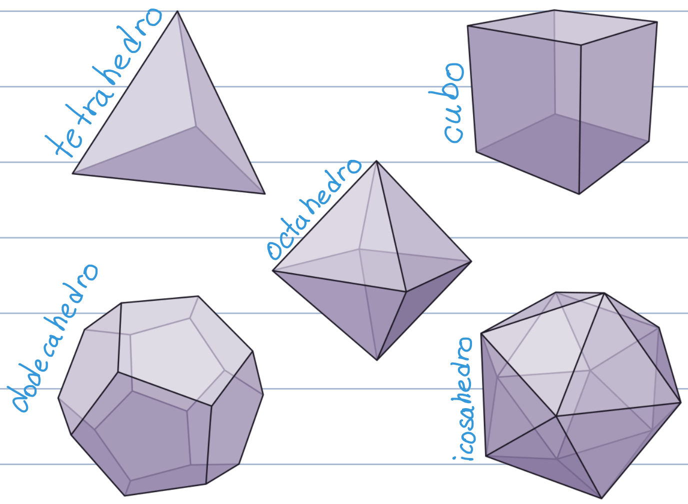
Una subdivisión de $K_{3,3}$.

Identidad de Euler

Un polihedro es un objeto 3-dimensional cuya frontera consiste de superficies planas poligonales.

Estas superficies poligonales se conocen como caras.

La frontera de una cara consiste de las aristas y los vértices del polígono.

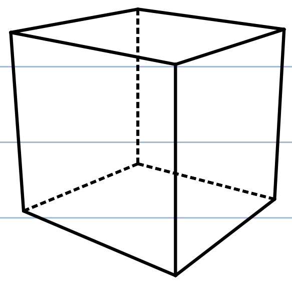


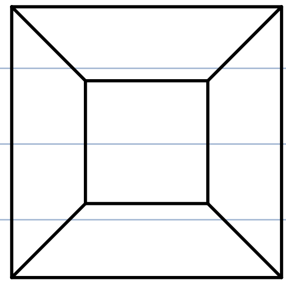
V → # de vértices
E → # de aristas
F → # de caras

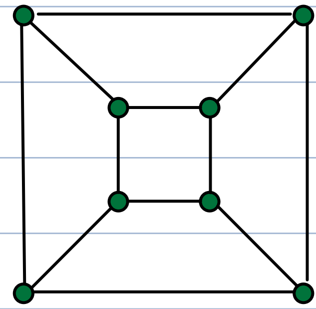
Sólido platónico	V	E	F	
tetrahedro	4	6	4	→ $4-6+4=2$
cubo	8	12	6	→ $8-12+6=2$
octahedro	6	12	8	
dodecahedro	20	30	12	→ $20-30+12=2$
icosahedro	12	30	20	→ $12-30+20=2$

La identidad polihedral de Euler:

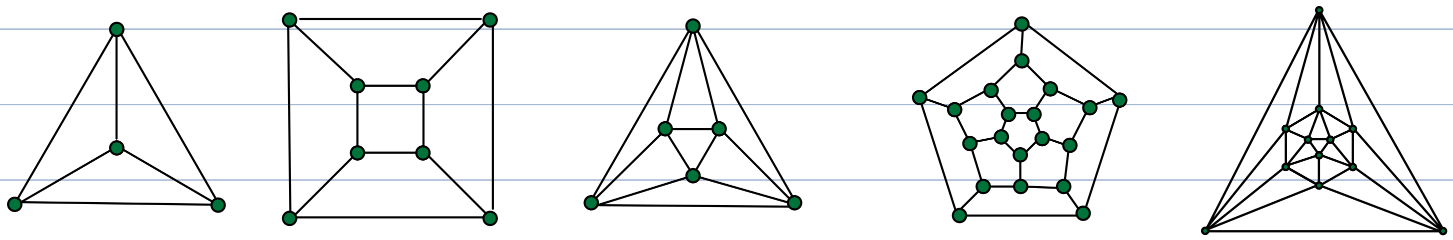
$$\boxed{V-E+F=2}$$







Gráficas platónicas.

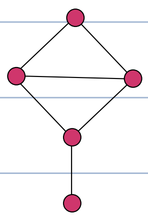


La identidad polihedral de Euler:

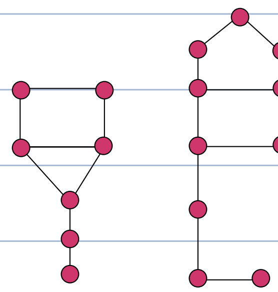
$$\boxed{V-E+F=2}$$

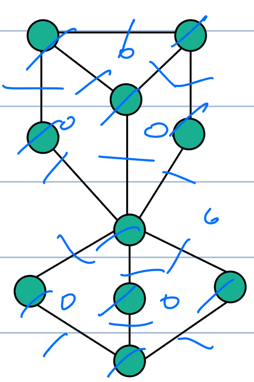
$5-6+3=2$ ✓

$V=5,\ E=6,\ F=3$



G





$V=10\quad E=14,\ F=6$

$10-14+6=2$ ✓

https://ics.uci.edu/~eppstein/junkyard/euler/

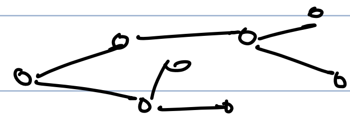
Teorema. (Identidad de Euler).

Para cada gráfica conexa con V vértices, E aristas y F caras ocurre que:

$$V-E+F=2.$$ y aplanable.

Demostración (por inducción sobre E).

Caso base: n vértices, $n-1$ aristas

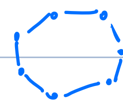


un árbol

$|F|=1$, $|V|=n$, $|E|=n-1$

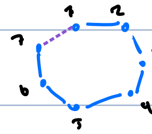
$$n-(n-1)+1=n-n+1+1=2 \checkmark$$

Consideremos una gráfica con más de $n-1$ aristas, necesariamente contiene un ciclo, y por lo tanto al menos una cara acotada.



Elegimos una arista de la cara acotada y la quitamos. La gráfica resultante sigue siendo conexa pero tiene una arista menos y una cara menos. Entonces, por hipótesis de inducción tenemos que se cumple que:

$$n-e'+f'=2 \qquad e'=e-1,\ f'=f-1$$



$n=7$
$f=1$
$e=6$

$7-6+1$
$1+1=2$

n
$7-7$

$f=2$
$e=7$

pd. $n-e+f=2$

$7-(6+1)+(1+1)$
$7-6-1+1+1$
$(7-6+1)$
$2-1+1$

Al regresar la arista, tenemos:

$$n-(e'+1)+(f'+1)=$$

$$\underbrace{n-e'+f'}_{=2}-1+1=$$

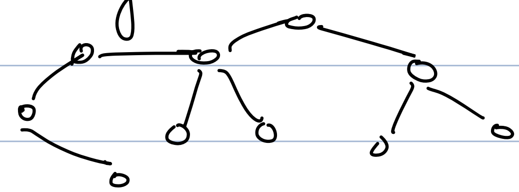
$$=2-1+1=2$$

$V-E+F=2.$

Demostracción por inducción sobre E.
Digamos que $|V|=n$.

Caso base: n vértices y $n-1$ aristas.

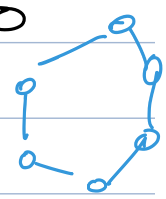
$\Rightarrow$ un árbol.



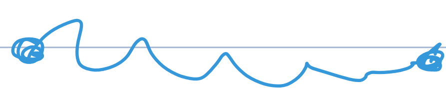
$F=1$.

$$n-(n-1)+1 = n-n+1+1 = 2 \checkmark$$

Consideremos una gráfica con al menos n aristas. Necesariamente tiene al menos un ciclo, y por lo tanto al menos una cara no acotada.



conexa:



Tomamos una arista de esa cara no acotada y la quitamos. La gráfica sigue siendo conexa, pero con una arista menos. Por mi hipótesis de inducción se cumple

$$n-E'+F'=2.$$

Al regresar la arista, tenemos:

$$n-(E'+1)+(F'+1)$$

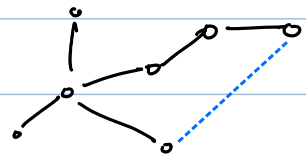
$$n-E'+F'-1+1$$

$$=2-\underline{1}+1=\underline{2}$$ ✓

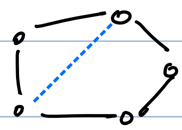
Teorema: Si G es una gráfica aplanable (conexa) con $V \geq 3$ y E aristas, entonces: $E \leq 3V-6$.

Demostración: Vamos a construir una gráfica G' a partir de G:

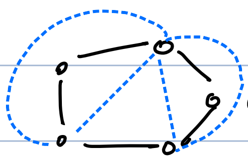
G' es la gráfica que tiene el mismo conjunto de vértices que G y el mismo conjunto de aristas más las que se obtienen en el siguiente proceso:



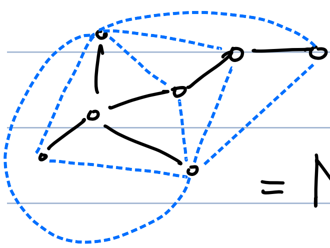
- Si G' no tiene ciclos, agregar cualquier arista entre dos vértices que no sean vecinos



- Si G' tiene un ciclo que no es de tamaño 3, agregar una arista uniendo vértices que no sean vecinos.



- Nos detenemos cuando todos los ciclos de G' sean de tamaño tres.

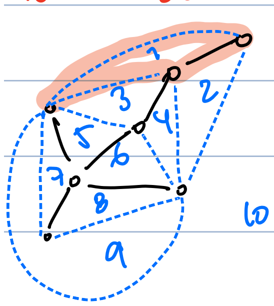


= Notemos que G' sigue siendo aplanable =

En cualquier dibujo de G':

Cara — arista

10 — 30



- Cada cara está acotada por exactamente 3 aristas
- Cada arista está en exactamente 2 caras.

$E' = 3F'$

$E' = \frac{3F'}{2} \Rightarrow F' = \frac{2}{3}E'$

Como G' es una gráfica aplanable y conexa, podemos usar la identidad de Euler:

$$\frac{2}{3} - \frac{3}{3} = -\frac{1}{3}$$

$$V - E' + F' = 2.$$

$$n - E' + \frac{2}{3}E' = 2$$

$$n - \frac{1}{3}E' = 2$$

$$-E' = 3(2-n)$$

$$E' = -3(2-n) = 3(n-2) = \underline{3n-6} \checkmark$$

Por lo tanto $\underline{E \leq 3n-6}$ ■

Corolario: Toda gráfica aplanable contiene un vértice de grado 5 o menos.

Consideremos una gráfica aplanable maximal.

Por el hand-shaking lema sabemos que la suma de los grados de los vértices es $2E$, es decir $6n-12$. El grado promedio es

$$\frac{6n-12}{n} = 6 - \frac{12}{n}$$

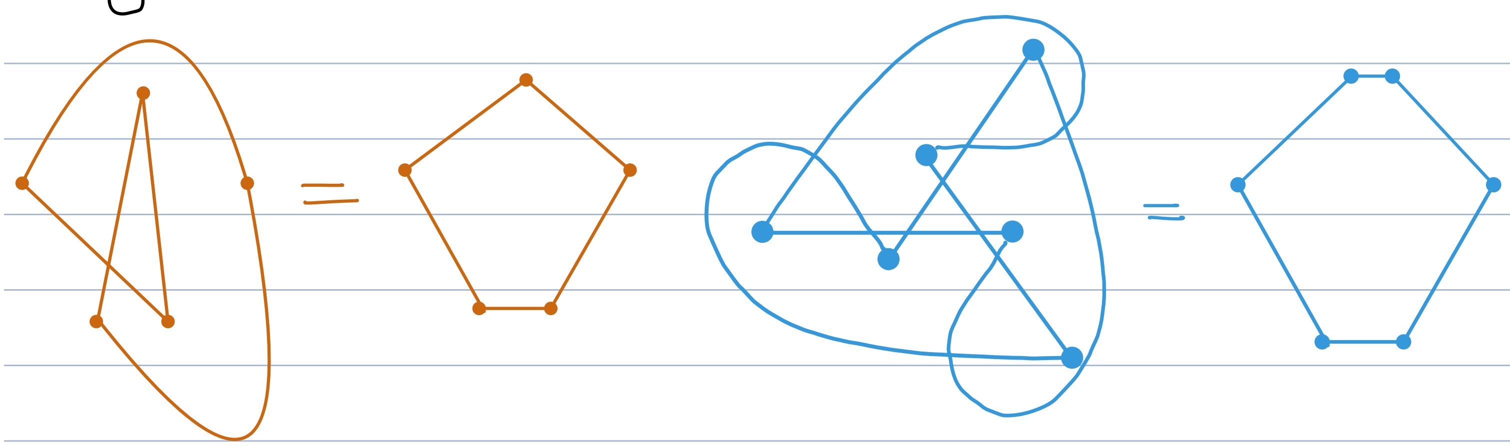
Por lo tanto la gráfica contiene al menos un vértice de grado estrictamente menor que 6. ■

¿Podemos construir un ejemplo de una triangulación que tenga exactamente un vértice de grado < 6?

Corolario: Toda gráfica completa K_n, con $n \geq 5$, es no aplanable.

Teorema de Fary.

Cualquier gráfica aplanable (simple puede dibujarse en el plano sin cruces y de tal forma que las aristas sean segmentos de recta.



DCEL. Doubly-connected edge list.

Las gráficas se pueden representar en la computadora usando distintas estructuras de datos, principalmente se usan dos:

- Matriz de adyacencia.
- Lista de adyacencia.

Matriz de adyacencia.

- Una matriz de tamaño $n \times n$ $(|V| \times |V|)$, en la que cada entrada vale 1 o 0, dependiendo de si la arista está o no en la gráfica. Es decir:

$$A[i,j] = \begin{cases} 1 & \text{si } \{i,j\} \in E \\ 0 & \text{si } \{i,j\} \notin E \end{cases}$$

Para gráficas no dirigidas siempre es simétrica

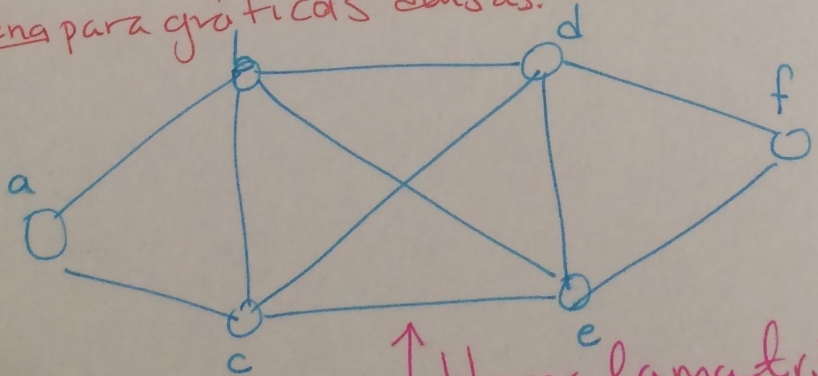
$$A[i,j] = A[j,i].$$

$$A[i,i] = \emptyset, \ \forall i.$$

$\Theta(1)$: dos vértices son disjuntos.

$\Theta(n^2)$ espacio

Buena para gráficas densas.



↑ Hacer la matriz para esta gráfica.

	a	b	c	d	e	f	
a	0	1	1	0	0	0	0
b		0	1	1	1	0	0
c			0	1	1	0	0
d				0	1	1	0
e					0	1	0
f						0	

Lista de adyacencia.

- Arreglo de tamaño n: almacena los vértices y una lista ligada.

- Para gráficas no dirigidas cada arista se almacena 2 veces.

  - Espacio $O(V+E) = O(n+m)$.
  - Listar vecinos de un vértice: $O(1+deg(v))$
  - Determinar si $(u,v) \in E$: $O(1+deg(u))$.

    Para gráficas no dirigidas:

    $O(1+\min\{deg(u), deg(v)\})$

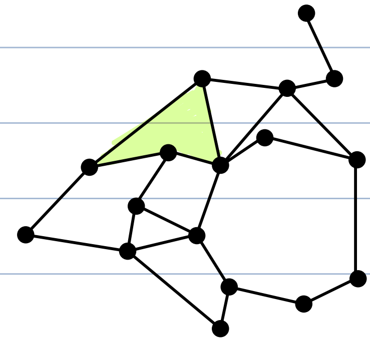
    Escanear ambas listas simultáneamente.

Sin embargo, a nosotrxs lo que nos interesa es almacenar los dibujos en el plano (encajes) de una gráfica.

Para esto vamos a usar una estructura llamada Doubly-connected edge list (DCEL).

¿Qué operaciones nos gustaría que soporte esta estructura?

Operaciones:

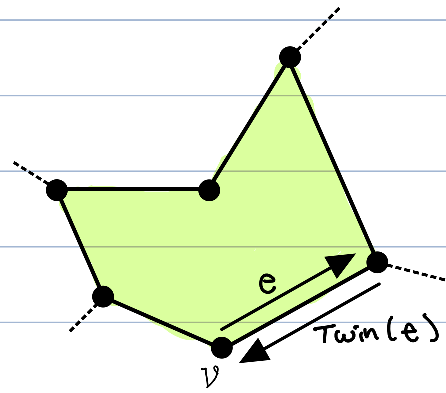


- Recorrer una cara
- Cambiar de cara
- Visitar todas las aristas incidentes en un vértice.

Una DCEL es una estructura que consiste de 3 listas doblemente ligadas:

1. Vértices
2. Caras
3. Aristas

Aristas



- Las llamaremos half-edges (semi-aristas)
- Contiene:
  - Apuntador a su origen Origin(e)
  - Apuntador Twin(e)
  - Apuntador IncidentFace(e), a la cara que acota. → Siempre a su izquierda
  - Next
  - Prev

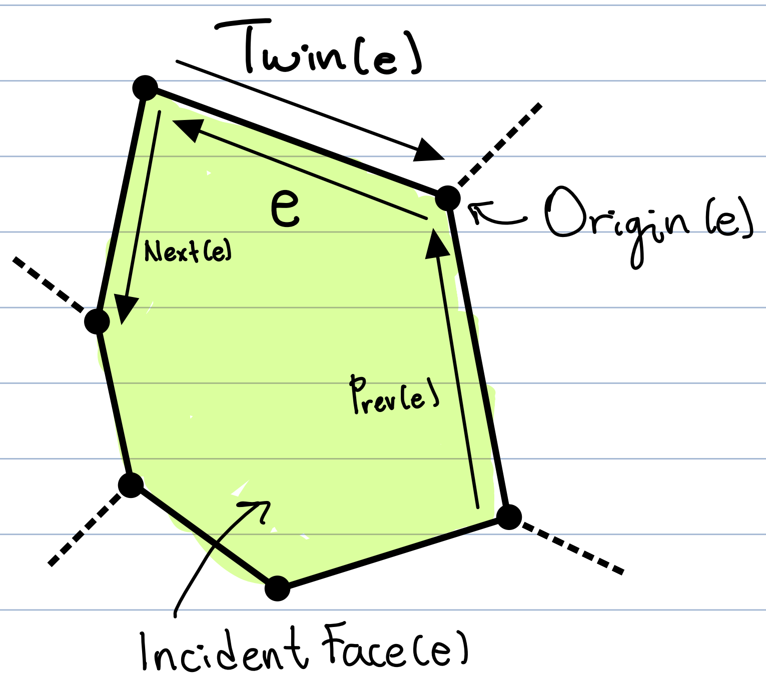
Vértices.

- Coordenadas del vértice: Coordinates (v).
- Apuntador Incident Edge(v) a una arista arbitraria que tenga a v como origen.

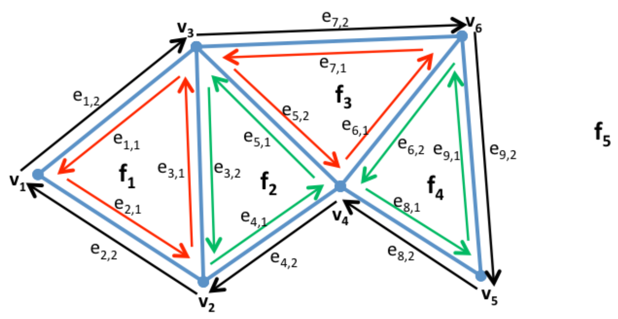
Caras.

- Apuntador OuterComponent (f) a algún half-edge en su frontera. Si la cara es no acotada entonces el apuntador es nulo.

- Lista InnerComponents(f) contiene un apuntador hacia algún half-edge en la frontera de cada componente conexa de la gráfica.



Ejemplo

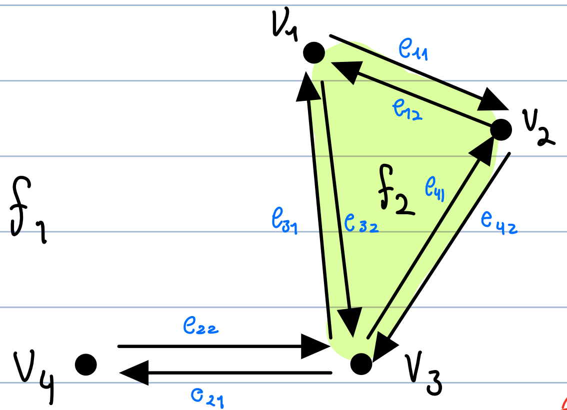


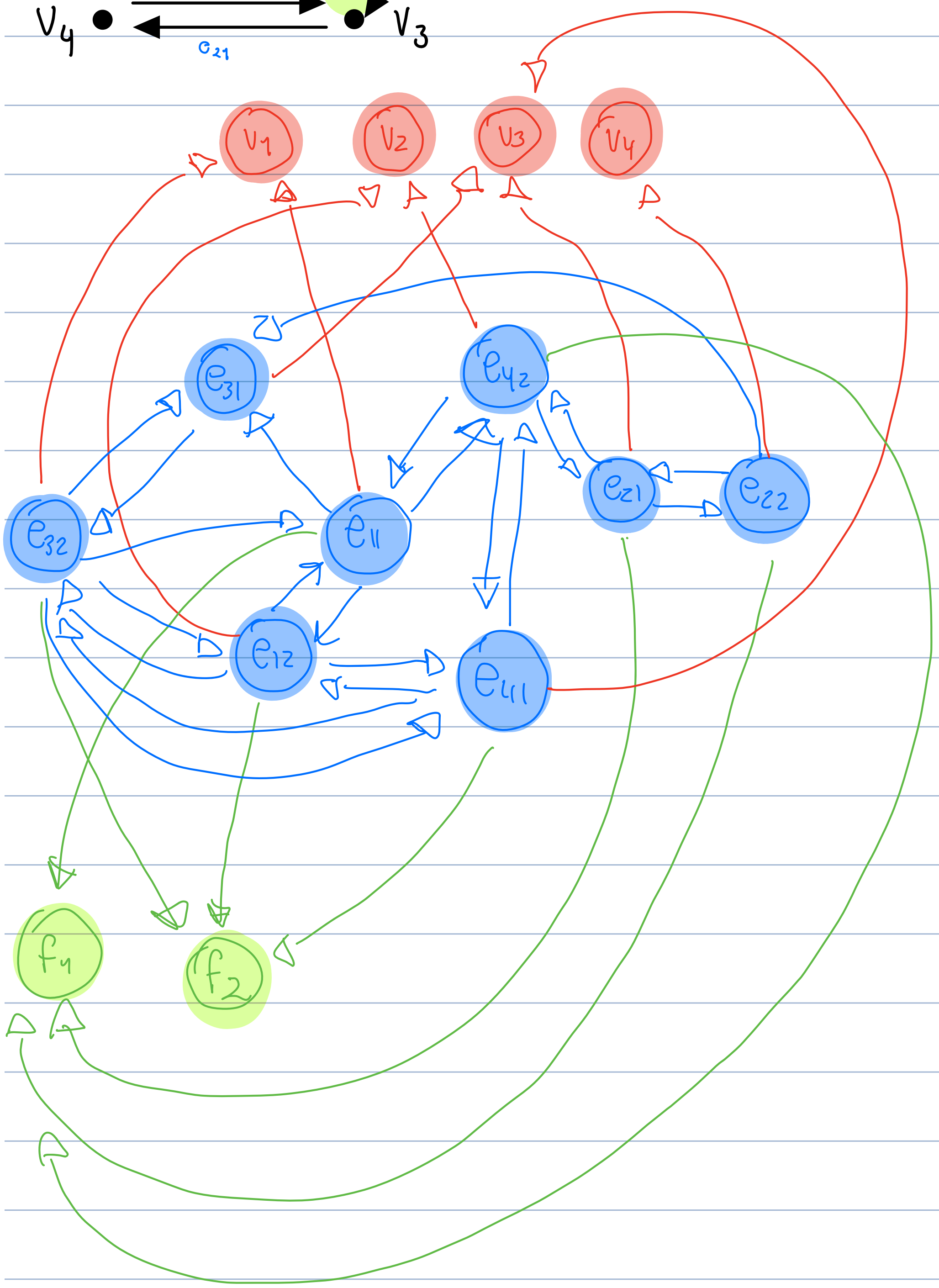
Vertex	Coordinates	IncidentEdge
v_1	(x_1, y_1)	$e_{2,1}$
v_2	(x_2, y_2)	$e_{4,1}$
v_3	(x_3, y_3)	$e_{3,2}$
v_4	(x_4, y_4)	$e_{6,1}$
v_5	(x_5, y_5)	$e_{9,1}$
v_6	(x_6, y_6)	$e_{7,1}$

Half-edge	Origin	Twin	IncidentFace	Next	Previous
$e_{3,1}$	v_2	$e_{3,2}$	f_1	$e_{1,1}$	$e_{2,1}$
$e_{3,2}$	v_3	$e_{3,1}$	f_2	$e_{4,1}$	$e_{5,1}$
$e_{4,1}$	v_2	$e_{4,2}$	f_2	$e_{5,1}$	$e_{3,2}$
$e_{4,2}$	v_4	$e_{4,1}$	f_5	$e_{2,2}$	$e_{8,2}$
…	…	…	…	…	…

Face	~~Edge~~ Outer Component	Inner Component
f_1	$e_{1,1}$	nil
f_2	$e_{5,1}$	nil
f_3	$e_{5,2}$	nil
f_4	$e_{8,1}$	nil
f_5	$\cancel{e_{9,2}}$ nil	$e_{9,2}$.

Otro ejemplo.





Notemos que la estructura almacena una cantidad constante de información por nodo.

Teorema:

Dado un encaje de una gráfica aplanable con n vértices, m aristas y f caras, la estructura DCEL tiene complejidad espacial

$$O(n+m+f) = O(n).$$

Demostración:

Para cada arista, vértice y cara se almacena una cantidad constante de información, por lo cual se requiere espacio $O(1)$ para cada elemento. De esto se sigue el tamaño $O(n+m+f)$. Como la gráfica es aplanable, por la identidad de Euler tenemos que $n+e-f=2$. Por lo cual

$$O(n+m+f) = O(n). \blacksquare$$

Manipulación de DCEL.

- Recorrer una cara

```
s ← halfedge(f)
h ← s
do
    something with h
    h ← next(h)
while h ≠ s
```

- Agregar un vértice.

  - Deseamos agregar un vértice **v** en alguna cara dada **f**.

<u>Entrada:</u> nuevo vértice v, y una semiarista h (f = IncidentFace(h)).

Sea u = Origin(Twin(h)), i.e. Target de h.
- u y v son vecinos

Requerimos que se cumpla la condición:

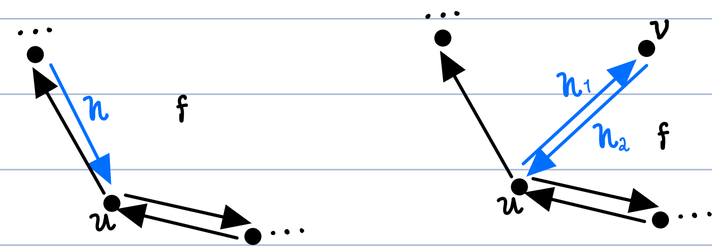
∘ Pre-condición (antes de ejecutar el alg.)

El segmento de recta

Coordinates(u), Coordinates(v)

debe estar completamente contenido en la cara f.

```
add-vertex-at(v, h) {
    h1 ← a new halfedge
    h2 ← a new halfedge
    halfedge(v) ← h2
    twin(h1) ← h2
    twin(h2) ← h1
    target(h1) ← v
    target(h2) ← u
    face(h1) ← f
    face(h2) ← f
    next(h1) ← h2
    next(h2) ← next(h)
    prev(h1) ← h
    prev(h2) ← h1
    next(h) ← h1
    prev(next(h2)) ← h2
}
```

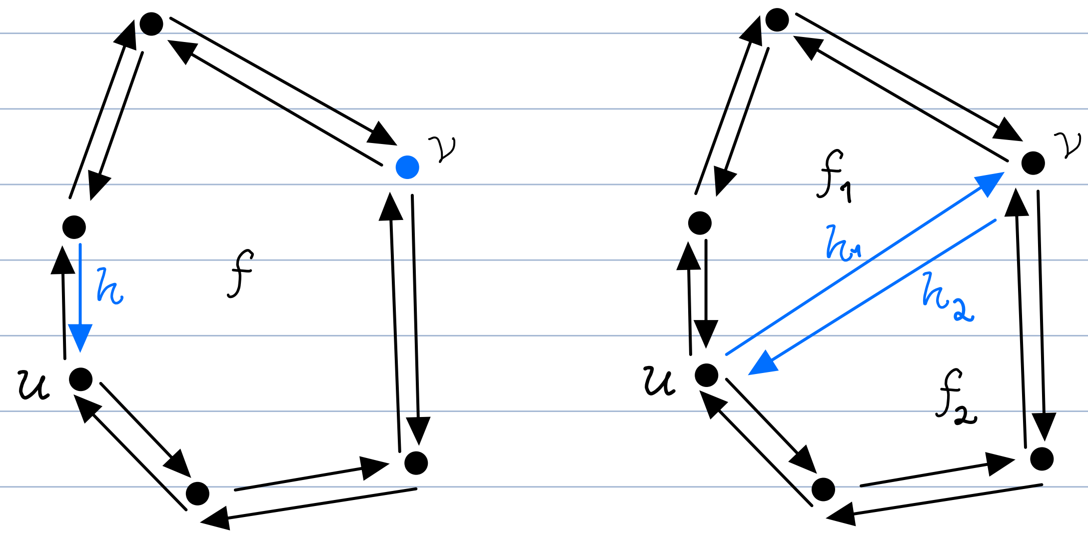


antes. después

O(1)

- Agregar una arista entre dos vértices dados.

- Igual que antes esta operación es posible únicamente si el segmento de recta uv está completamente contenido dentro de la cara f.



- Además supondremos que v está en f y que no es adyacente a u.

- target(e) = Origin(Twin(e)) ✓

```
split-face(h, v) {
    f1 ← a new face
    f2 ← a new face
    h1 ← a new halfedge
    h2 ← a new halfedge
    halfedge(f1) ← h1
    halfedge(f2) ← h2
    twin(h1) ← h2
    twin(h2) ← h1
    target(h1) ← v
    target(h2) ← u
    next(h2) ← next(h)
    prev(next(h2)) ← h2
    prev(h1) ← h
    next(h) ← h1
    i ← h2
    loop
        face(i) ← f2
        if target(i) = v break the loop
        i ← next(i)
    endloop
    next(h1) ← next(i)
    prev(next(h1)) ← h1
    next(i) ← h2
    prev(h2) ← i
    i ← h1
    do
        face(i) ← f1
        i ← next(i)
    until target(i) = u
    delete the face f
}
```

$O(|f|)$

- Algunas otras operaciones básicas se dejan como ejercicio.