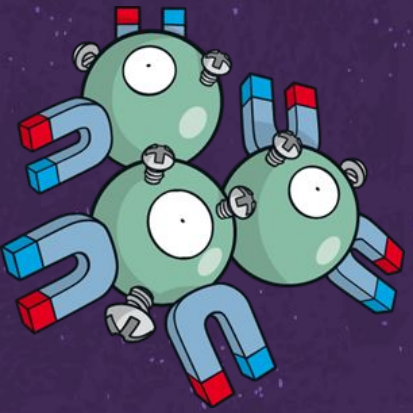
TDA Grafos

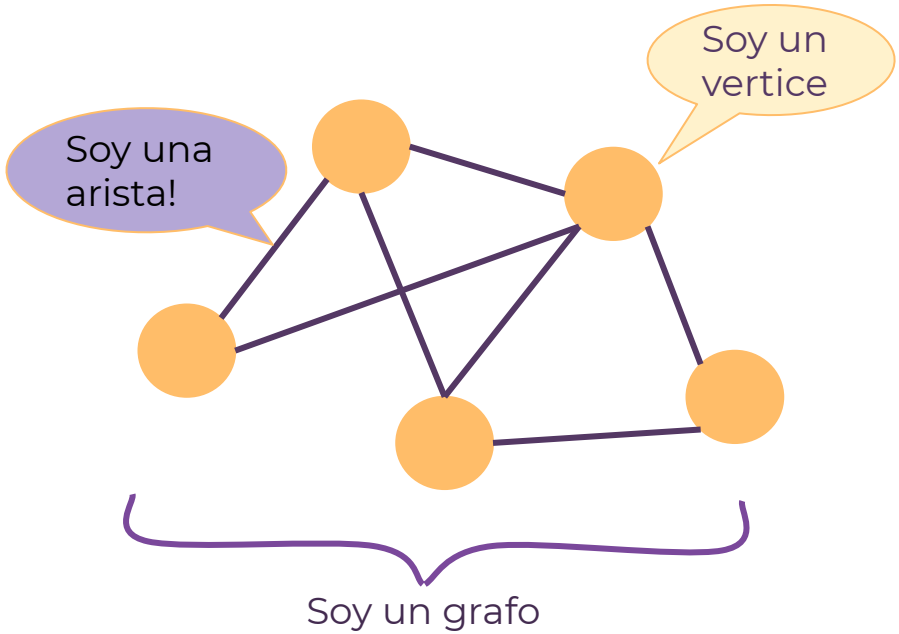


Algo2Mendez
1C 2021

Izquierdo Stephanie
Sibikowski Nicolas

Qué es un grafo?

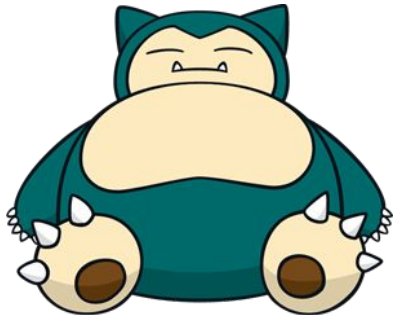
Es el par ordenado G = (V,E) donde V es el conjunto de vértices y E el conjunto de aristas:

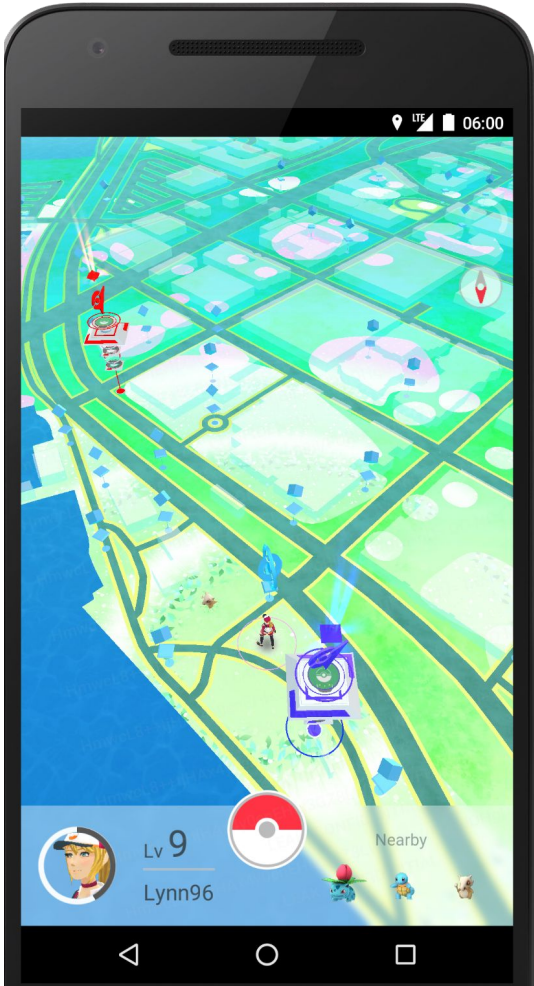


Para que nos sirve?

Supongamos que queremos ir a un gym pokemon y mientras tanto pasar por las pokeparadas que tenemos cerca

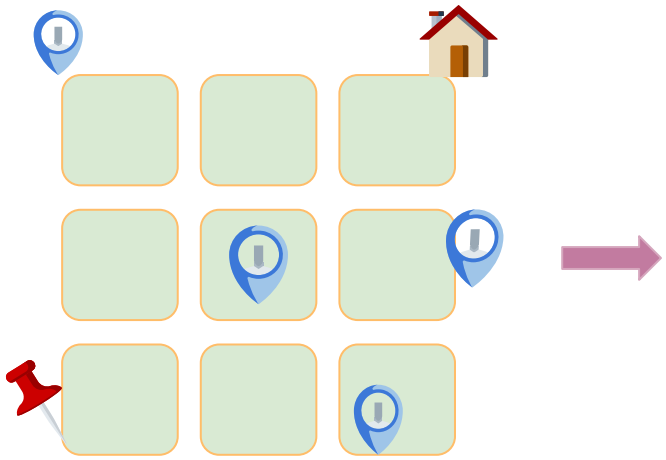
Pero tenemos mucha fiaca y queremos caminar lo menos posible sin desviarnos mucho...

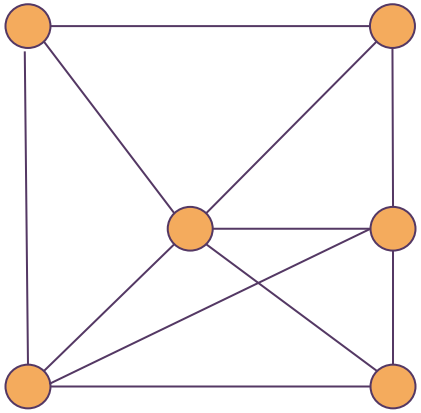


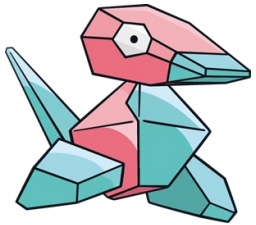


Hagamos un grafo!

1 Pensemos las pokeparadas como vértices y los caminos son las aristas



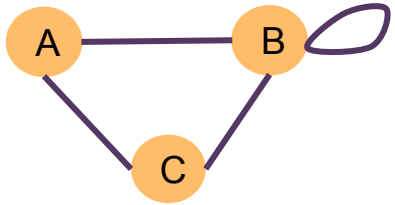




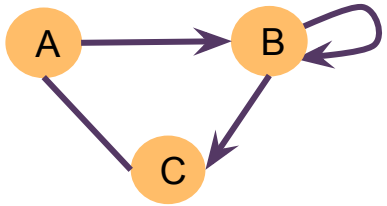
Definiciones!

Dirección

Grafo no dirigido: Todas sus aristas son bidireccionales, es decir que estas se pueden recorrer en dos direcciones



Grafo dirigido: Al menos una de sus aristas es unidireccional (pueden haber aristas bidireccionales)



Pesos

Grafo con peso: Todas sus aristas poseen un valor numérico que implica un “costo” a la hora de tener que transitar de un vértice hacia otro.

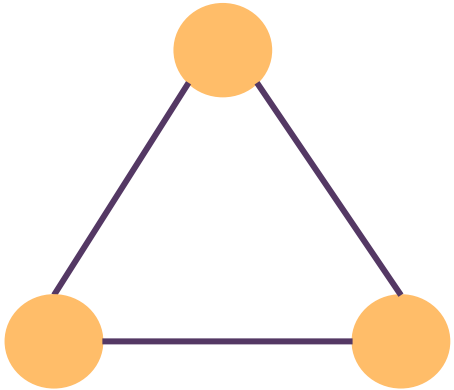
Clasificación

Con todas estas propiedades, los grafos tienen las siguientes clasificaciones:

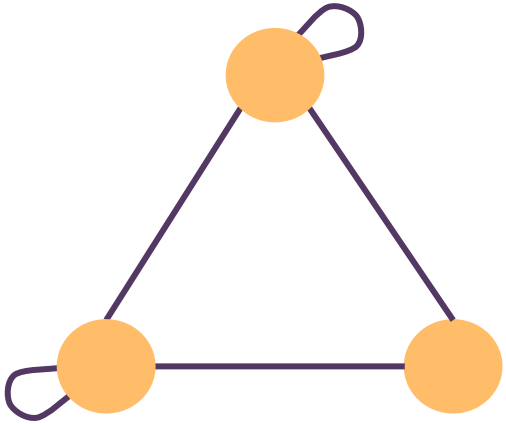
- **Grafos dirigidos con peso.**
- **Grafos dirigidos sin peso.**
- **Grafos no dirigidos con peso.**
- **Grafos no dirigidos sin peso.**

Grafo Simple

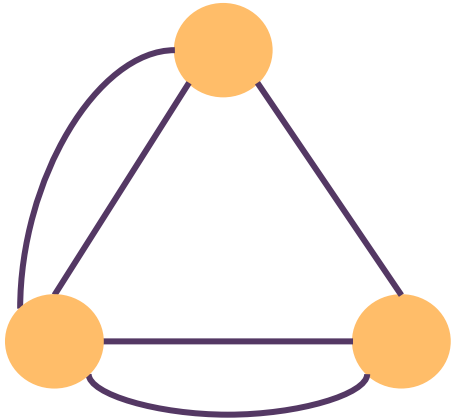
Un grafo simple es aquel que no posee aristas múltiples ni lazos.
Es decir, que en un par de vértices hay solo una arista.



Grafo Simple



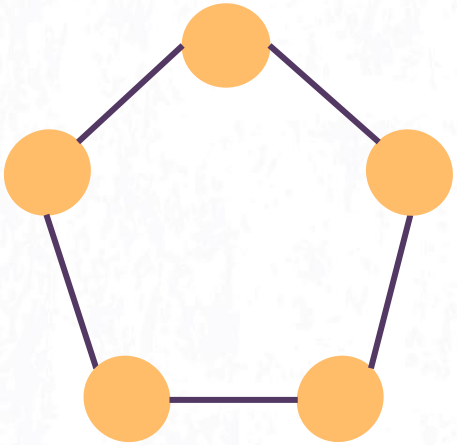
No simple



No simple

Densidad

Un grafo denso es un grafo cuyo número de aristas está muy cerca del valor máximo de aristas que este puede tener.

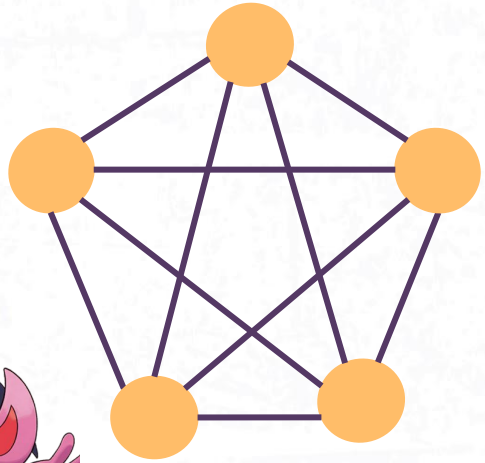


D-> 0: Grafo disperso

Para un grafo simple:

$$D = \frac{2\,|E|}{|V|\,(|V| - 1)}$$

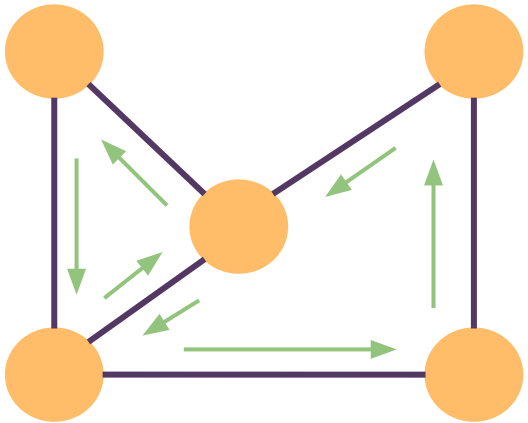
Índice de densidad

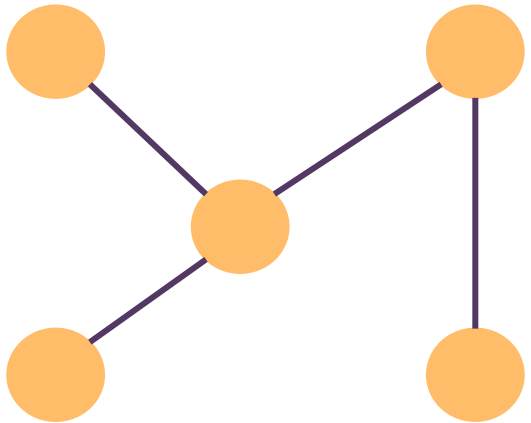


D -> 1 : Grafo denso

Ciclico - Aciclico

Ciclo: Recorrido de aristas adyacentes que empieza y termina en el mismo lugar (pasando una sola vez por cada arista)

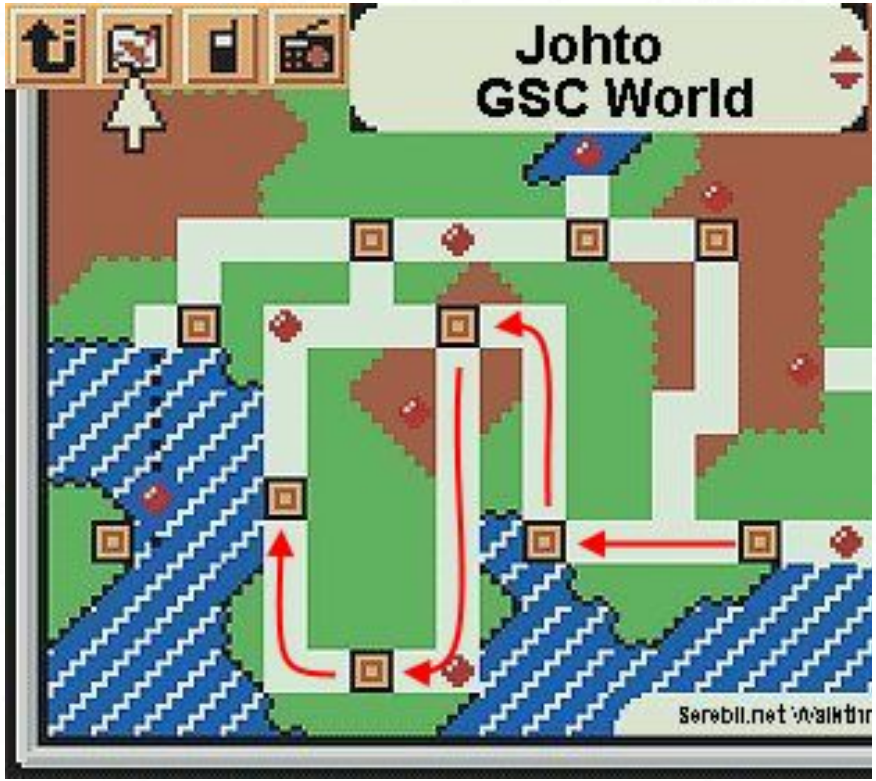




Más definiciones...

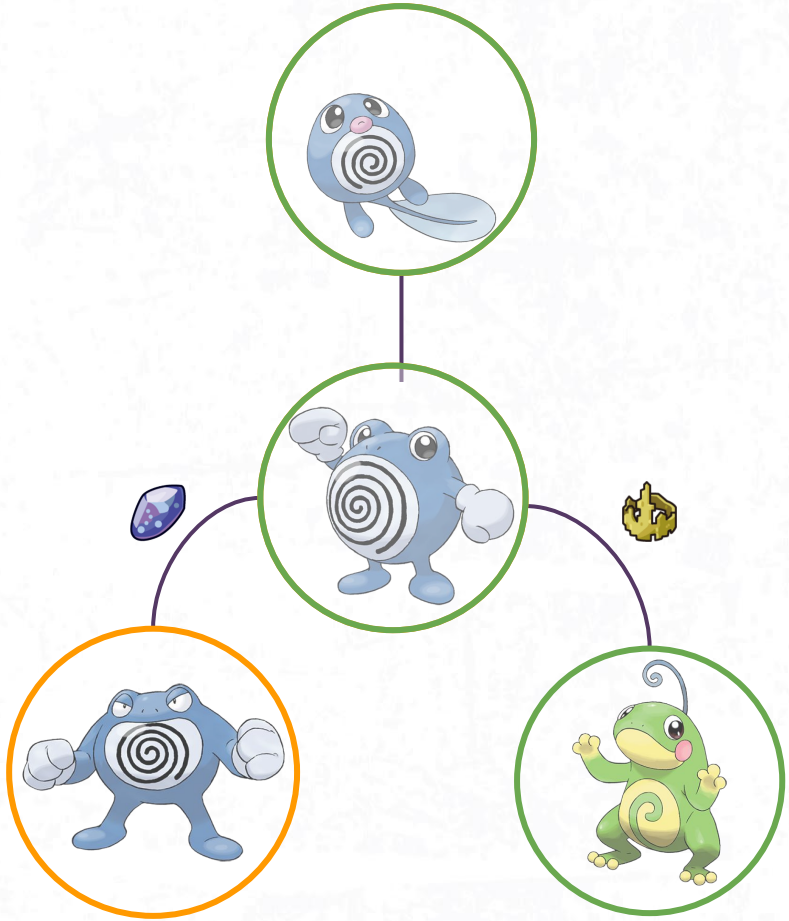
Camino: Un recorrido a través de un grafo.

Camino Simple: Un camino, pero que pasa **a lo sumo** una vez por cada vertice

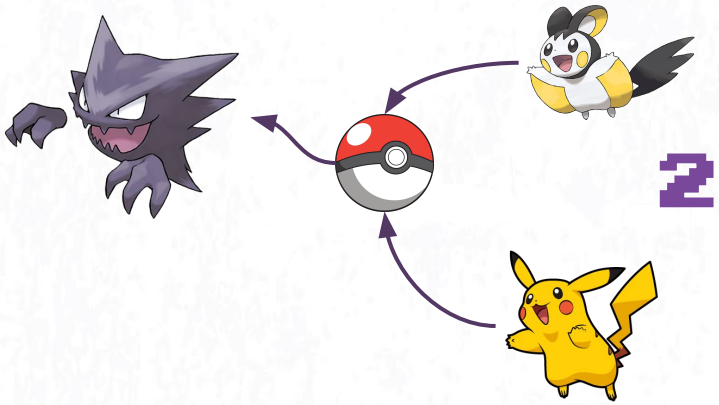


Componente Conexa:

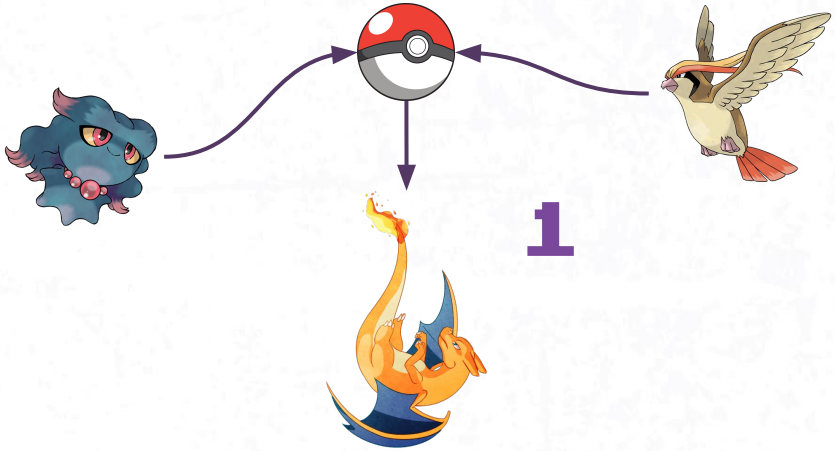
Conjunto de vértices donde existe un camino de un vértice a otro. (**Solo** para grafos no dirigidos)



Grado de entrada: Cantidad de aristas que entran al vértice.



Grado de salida: Cantidad de aristas que salen del vértice.



En grafos no dirigidos ambos grados son iguales! -> **Grado del vertice**

Primitivas del tda!

grafo_crear()

grafo_insertar(grafo, elemento)

grafo_insertar_arista(grafo, arista)

grafo_borrar(grafo, elemento)

grafo_borrar_arista(grafo, arista)

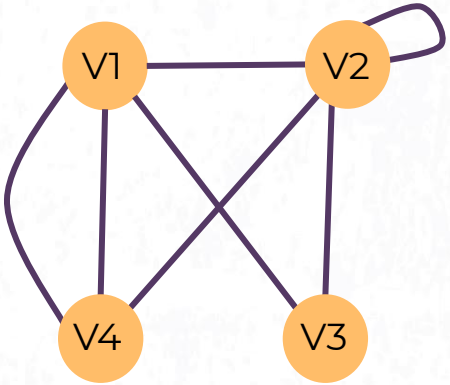
grafo_destruir(grafo)

Ideas de representación

Representaciones

- Matriz de adyacencia
- Lista de adyacencia
- Matriz de incidencias

Matriz de adyacencia



	V1	V2	V3	V4
V1 →	0	1	1	2
V2 →				
V3 →				
V4 →				

Me paro en un vertice

01

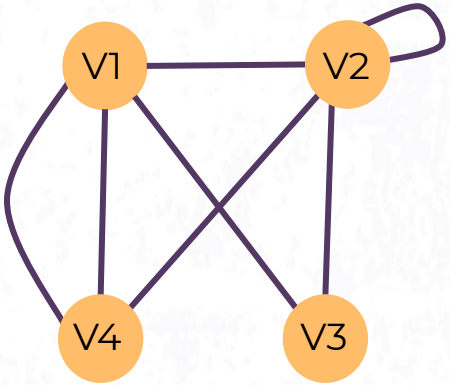
02

Me fijo desde ese cuántas aristas de long 1 conecta con otro vértice

03

Completo matriz!

Matriz de adyacencia



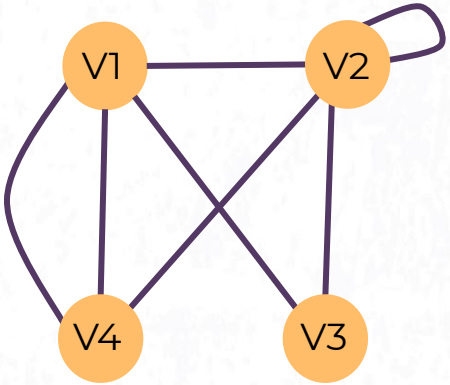
	V1	V2	V3	V4
V1 →	0	1	1	2
V2 →	1	1	1	1
V3 →				
V4 →				

01 Me paro en un vertice

02 Me fijo desde ese cuántas aristas de long 1 conecta con otro vértice

03 Completo matriz!

Matriz de adyacencia



	V1 ↓	V2 ↓	V3 ↓	V4 ↓
V1 →	0	1	1	2
V2 →	1	1	1	1
V3 →	1	1	0	0
V4 →				

Me paro en un vertice

01

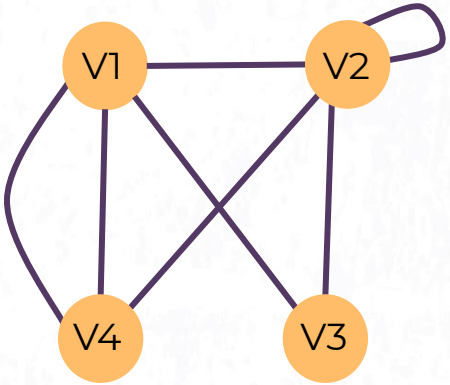
02

Me fijo desde ese cuántas aristas de long 1 conecta con otro vértice

03

Completo matriz!

Matriz de adyacencia



	V1 ↓	V2 ↓	V3 ↓	V4 ↓
V1 →	0	1	1	2
V2 →	1	1	1	1
V3 →	1	1	0	0
V4 →	2	1	0	0

Me paro en un vertice

01

02

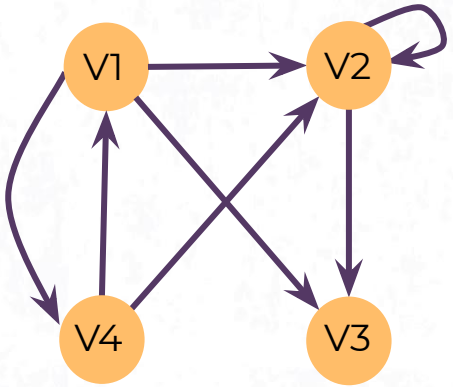
Me fijo desde ese cuántas aristas de long 1 conecta con otro vértice

03

Completo matriz!

Matriz de adyacencia

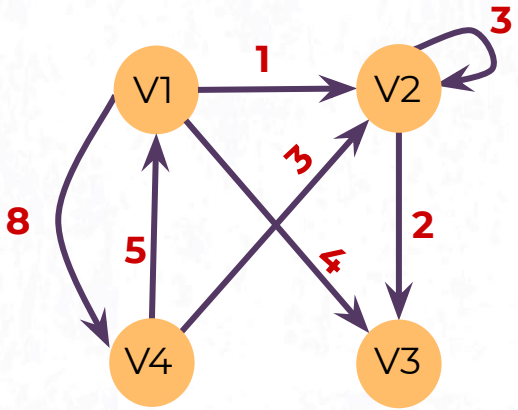
En un un grafo dirigido la matriz de adyacencia solo registra el vértice de inicio y al vértice final, A[i][j]= 1, si y solo si la arista parte del vertice i hacia el vértice j:



	V1	V2	V3	V4
V1 →	0	1	1	1
V2 →	0	1	1	0
V3 →	0	0	0	0
V4 →	1	1	0	0

Matriz de adyacencia

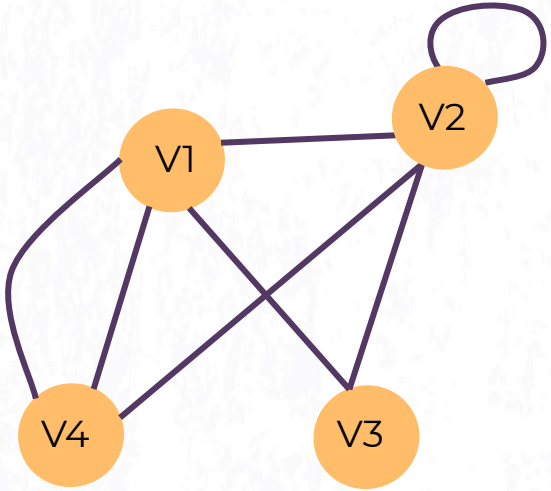
Y si el grafo tiene pesos? En vez de completar con 1, ponemos el peso

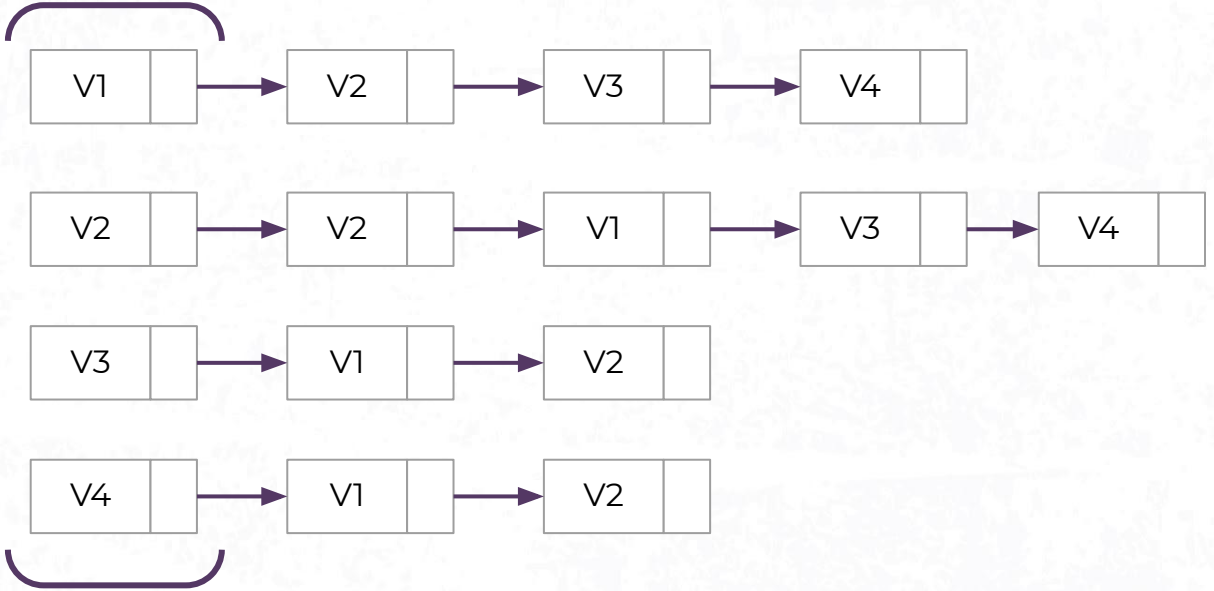


	V1	V2	V3	V4
V1 →	0	1	4	8
V2 →	0	3	2	0
V3 →	0	0	0	0
V4 →	5	3	0	0

Lista de adyacencia

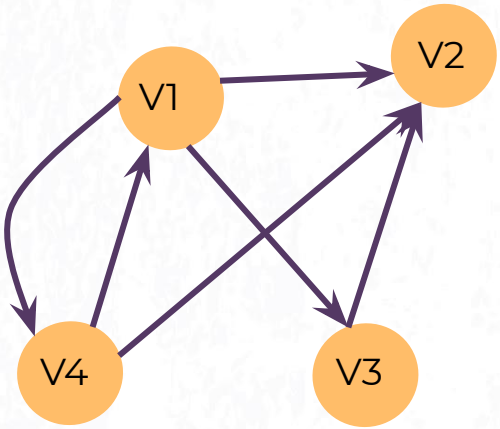
Cada arista posee una lista simplemente enlazada de a qué vértices está unida.

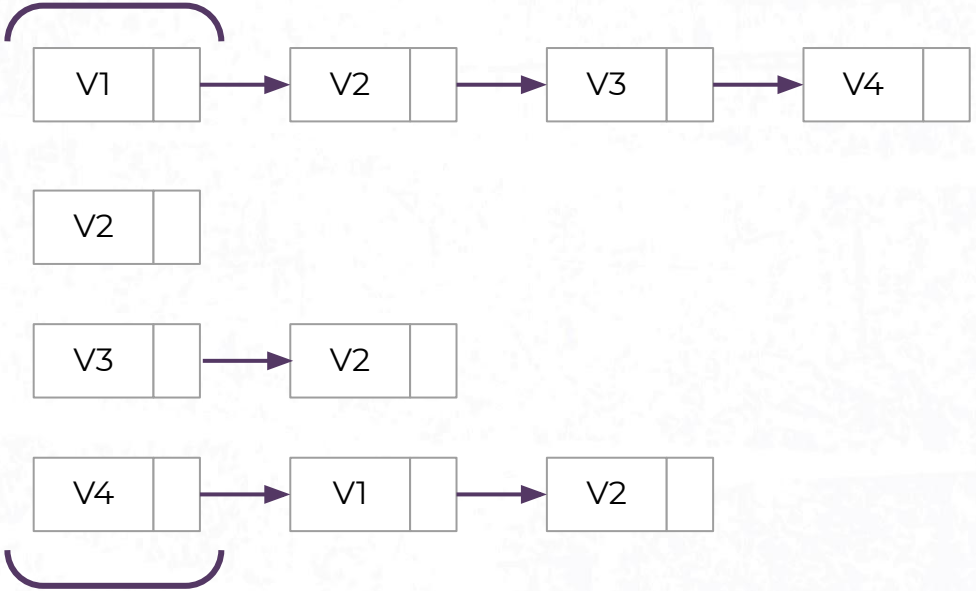




Lista de adyacencia

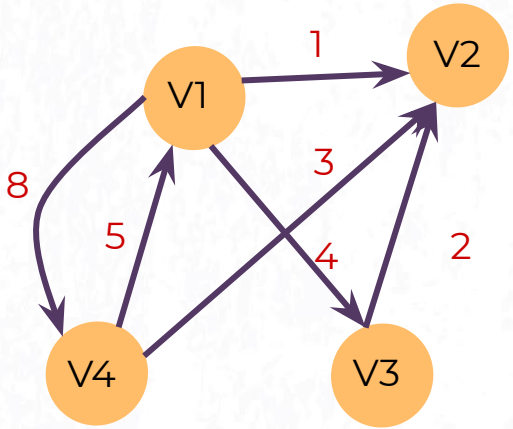
Y en grafos dirigidos? Respetamos las direcciones!

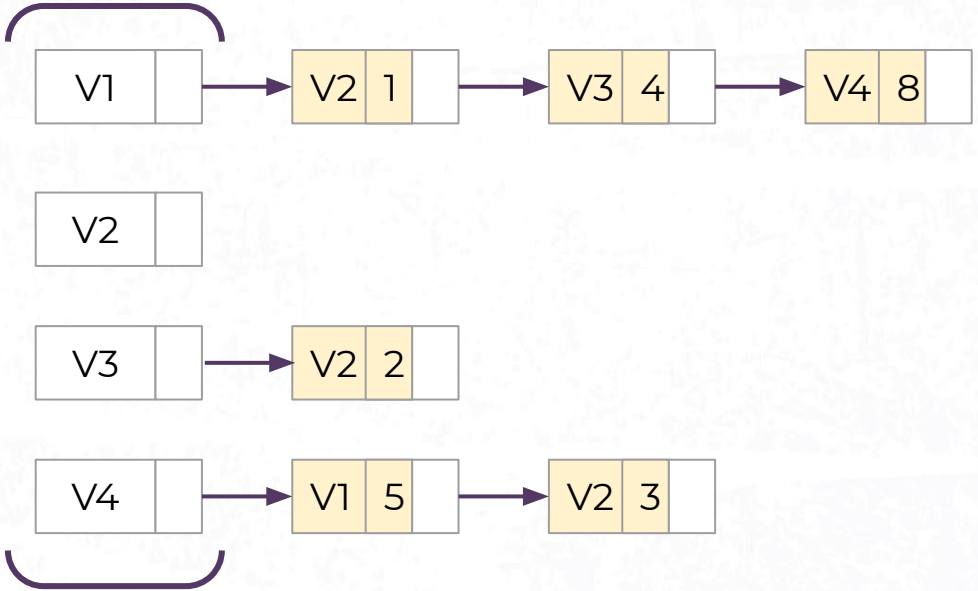




Lista de adyacencia

Y en grafos dirigidos? Respetamos las direcciones!





Matriz de incidencia

Para esto necesitamos identificar las aristas

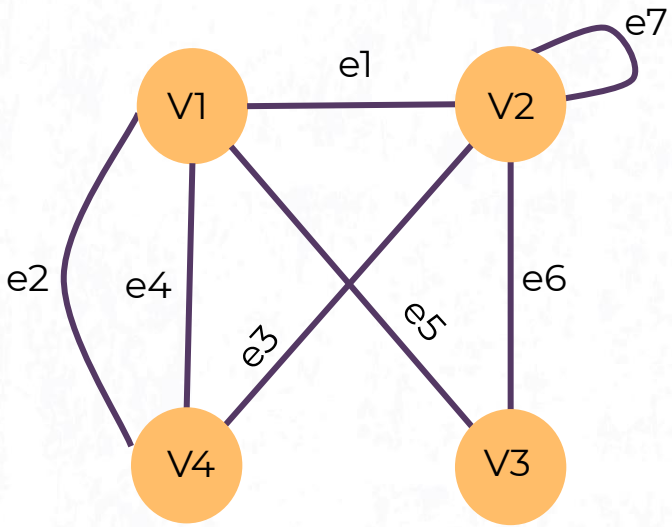
Esta matriz es Z^{mxn}, donde m es la cantidad de aristas y n es la cantidad de vértices.

Por cada arista identificamos que nodos inciden con él y lo marcamos en un grafo no dirigido con 1.

Si el grafo a representar es dirigido: se le asigna -1 al vértice de salida y 1 al vértice de entrada

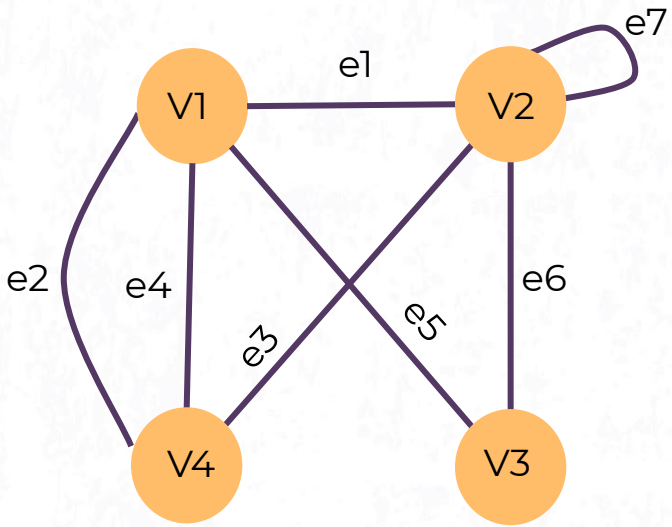
Si el grafo fuera dirigido y con peso, en vez de utilizar el número 1 se utiliza el número del peso de cada arista.

Matriz de incidencia



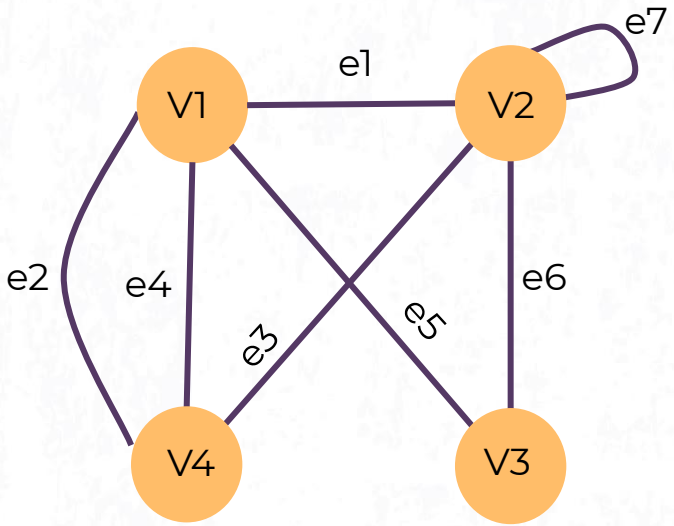
	V1	V2	V3	V4
e1 →				
e2 →				
e3 →				
e4 →				
e5 →				
e6 →				
e7 →				

Matriz de incidencia



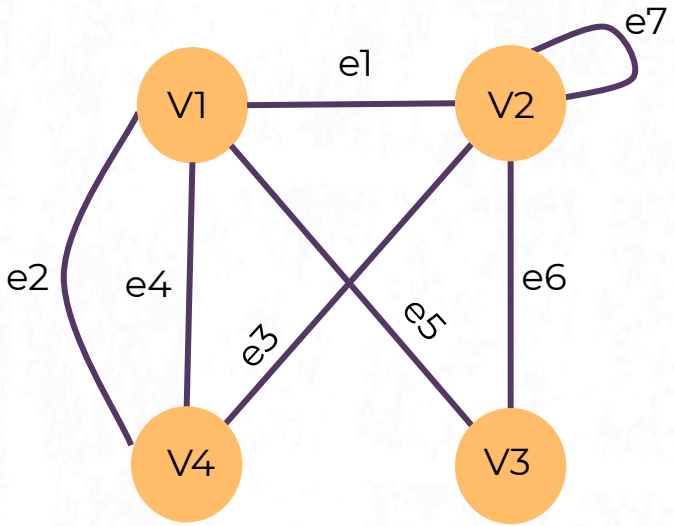
	V1 ↓	V2 ↓	V3 ↓	V4 ↓
e1 →	1	1	0	0
e2 →				
e3 →				
e4 →				
e5 →				
e6 →				
e7 →				

Matriz de incidencia



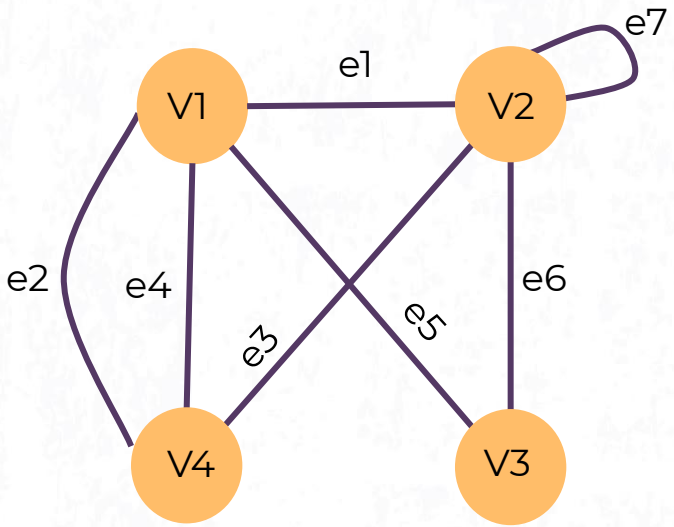
	V1 ↓	V2 ↓	V3 ↓	V4 ↓
e1 →	1	1	0	0
e2 →	1	0	0	1
e3 →				
e4 →				
e5 →				
e6 →				
e7 →				

Matriz de incidencia



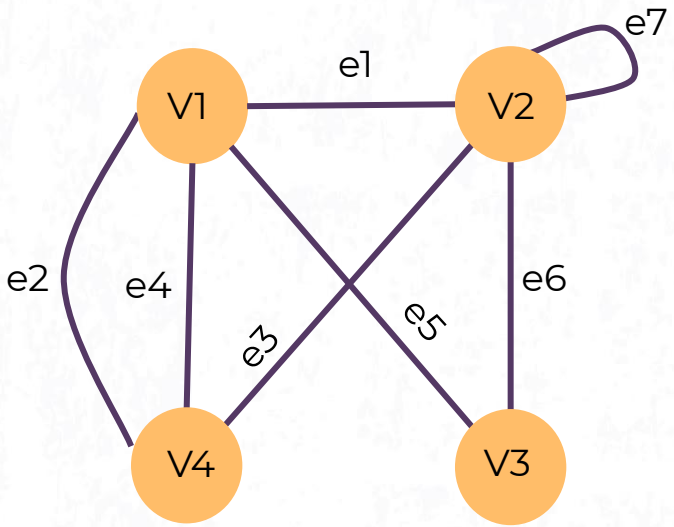
	V1 ↓	V2 ↓	V3 ↓	V4 ↓
e1 →	1	1	0	0
e2 →	1	0	0	1
e3 →	0	1	0	1
e4 →				
e5 →				
e6 →				
e7 →				

Matriz de incidencia



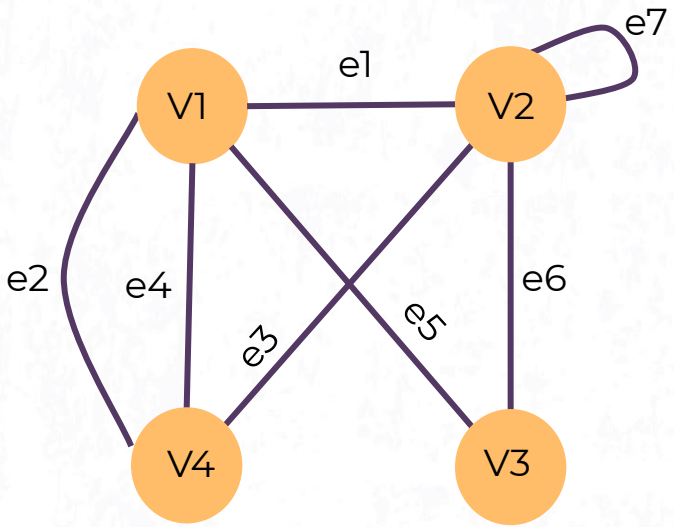
	V1 ↓	V2 ↓	V3 ↓	V4 ↓
e1 →	1	1	0	0
e2 →	1	0	0	1
e3 →	0	1	0	1
e4 →	1	0	0	1
e5 →				
e6 →				
e7 →				

Matriz de incidencia



	V1 ↓	V2 ↓	V3 ↓	V4 ↓
e1 →	1	1	0	0
e2 →	1	0	0	1
e3 →	0	1	0	1
e4 →	1	0	0	1
e5 →	1	0	1	0
e6 →	0	1	1	0
e7 →				

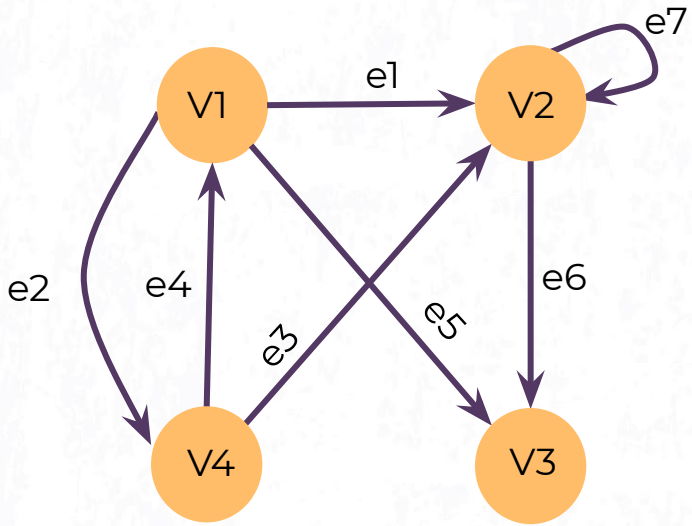
Matriz de incidencia



	V1 ↓	V2 ↓	V3 ↓	V4 ↓
e1 →	1	1	0	0
e2 →	1	0	0	1
e3 →	0	1	0	1
e4 →	1	0	0	1
e5 →	1	0	1	0
e6 →	0	1	1	0
e7 →	0	1	0	0

Matriz de incidencia

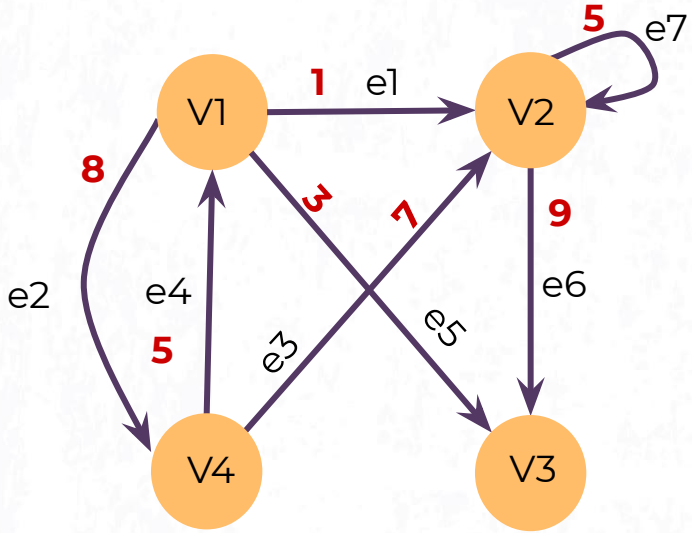
Grafo **dirigido**:



	V1 ↓	V2 ↓	V3 ↓	V4 ↓
e1 →	-1	1	0	0
e2 →	-1	0	0	1
e3 →	0	1	0	-1
e4 →	1	0	0	-1
e5 →	-1	0	1	0
e6 →	0	-1	1	0
e7 →	0	±1	0	0

Matriz de incidencia

Grafo dirigido y con **pesos**:



	V1 ↓	V2 ↓	V3 ↓	V4 ↓
e1 →	-1	**1**	0	0
e2 →	-1	0	0	**8**
e3 →	0	**7**	0	-1
e4 →	**5**	0	0	-1
e5 →	-1	0	**3**	0
e6 →	0	-1	**9**	0
e7 →	0	**±5**	0	0

Recorridos

Proceso de búsqueda o forma de atravesar la estructura de datos de un grafo que involucra la visita de cada vértice en el grafo.

El orden en el cual los vértices son visitados es la forma en que se clasifica el tipo de recorrido:

Dos acciones:

- Visitar: Marcar como visitado al nodo.
- Explorar: Política en la que vamos a definir cómo se comportará el algoritmo (explorar los vecinos, los hijos, etc.)

Tipos de Recorridos

- En anchura: BFS (Breath First Search)

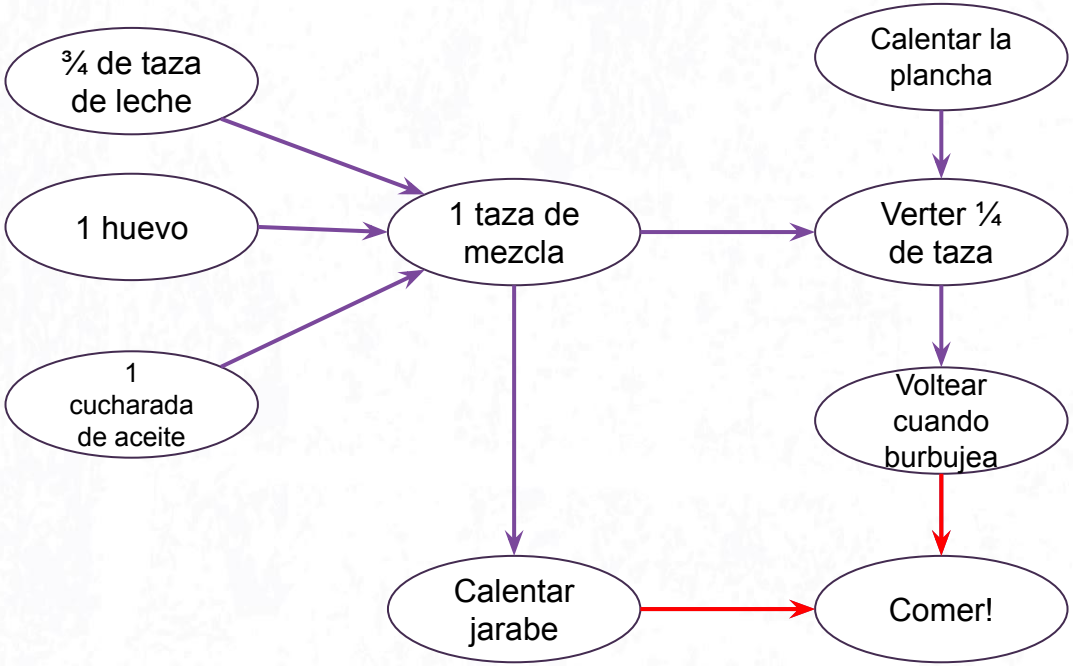
  Se encara el recorrido del grafo en anchura, es decir, primero visita nodos hermanos/vecinos antes de visitar nodos hijos. Para su implementación se utiliza una cola

- En profundidad: DFS (Depth First Search)

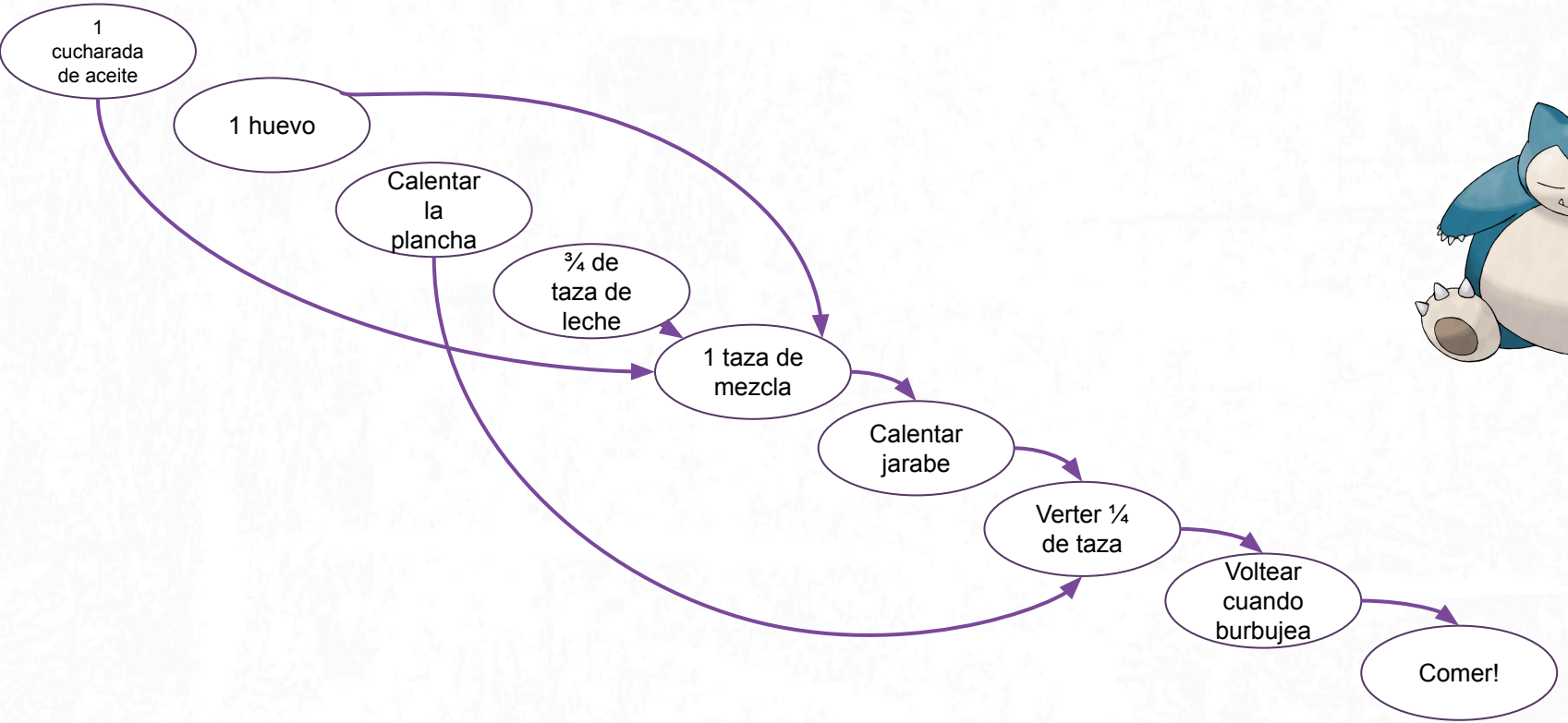
  En este caso se van a visitar los nodos hijos primero, avanzando hasta que no se pueda continuar. Luego se “vuelve” hasta un hijo donde se tenían más caminos, y se vuelve a realizar la misma lógica.

Orden topológico - Comida pokémon!

La idea del orden topológico es la de procesar los vértices de un grafo **acíclico** de forma tal que si el grafo contiene la arista dirigida uv entonces el nodo u aparece antes del nodo v.



Orden topológico - Comida pokémon!



Dijkstra

El algoritmo de Dijkstra es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista.

La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices.

Cuando se obtiene el camino más corto desde el vértice origen hasta el resto de los vértices que componen el grafo, el algoritmo se detiene.

No funciona en grafos con aristas de peso negativo.

Algoritmo dijkstra

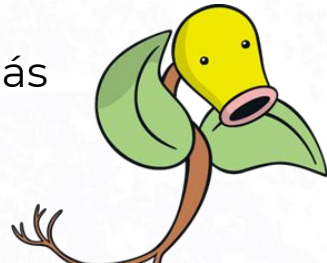
1. Se elige el vértice V sobre el cual se quiera aplicar el algoritmo

2. Se crean dos lista de nodos, una lista de nodos Visitados y otra listas de nodos NO Visitados, que contiene a todos los nodos del grafo.

3. Se crea una tabla con 3 columnas, Vértice, Distancia mínima V y el nodo anterior por el cual se llegó.

4. Se toma el vértice V como vértice inicial y se calcula su distancia a sí mismo, que es 0.

5. Se actualiza la tabla, en la cual todas las distancias de los demás vértices a V se marcan como infinito.



6. Se visita el vértice NO VISITADO con menor distancia conocida desde el primer vértice V, que es el vértice con el que comenzamos ya que la distancia a ese es 0 y las demás infinito.

7. Se calcula la distancia entre los vértices sumando los pesos de cada uno con la distancia de V.

8. Si la distancia calculada de los vértices conocidos es menor a la que está en la tabla se actualiza y también los vértices desde donde se llegó..

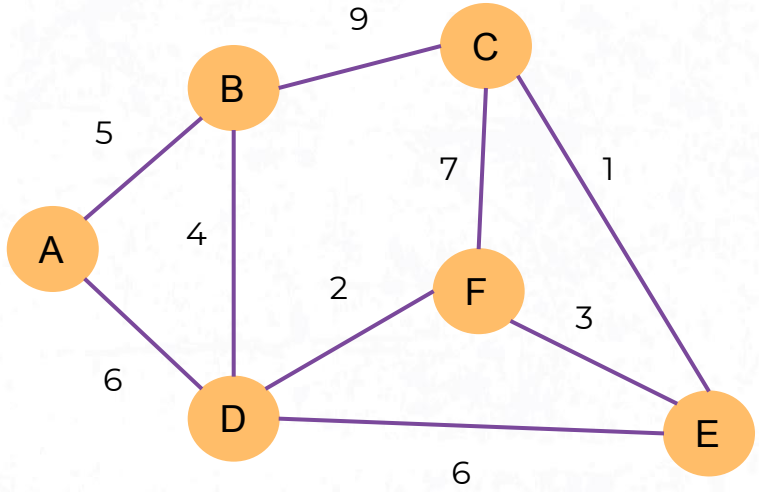
9. Se pasa el vértice V a la lista de Vértices visitados.

10. Se continua con el vértice no visitado con menor distancia desde ese.

11. Y así sucesivamente

Veámoslo con un ejemplo!

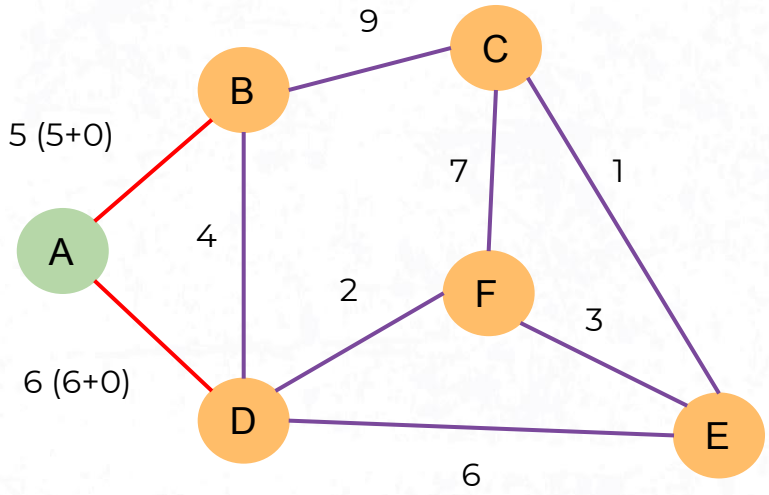
Vértice	Distancia	V. Anterior
A	0	-
B	∞	-
C	∞	-
D	∞	-
E	∞	-
F	∞	-



Visitados: [ ]

No visitados: [A,B,C,D,E,F]

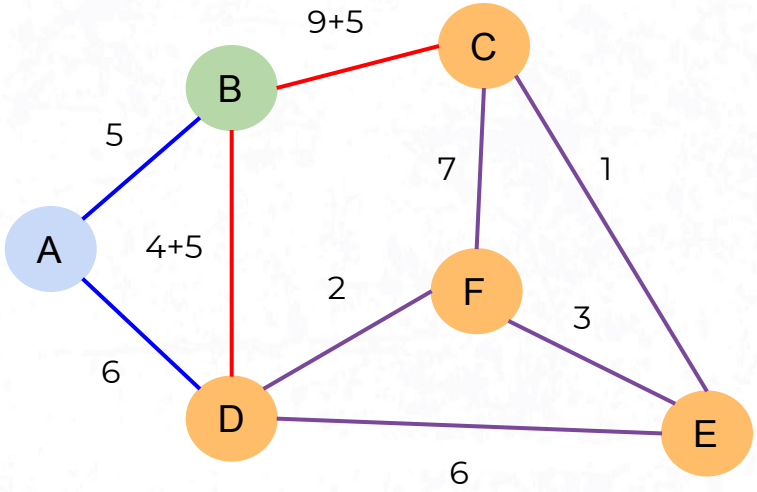
Vértice	Distancia	V. Anterior
A	0	-
B	5	A
C	∞	-
D	6	A
E	∞	-
F	∞	-



Visitados: [A]

No visitados: [B,C,D,E,F]

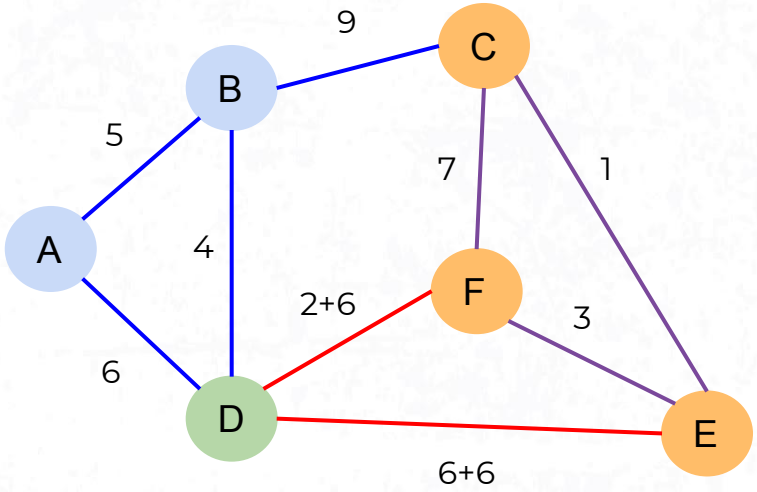
Vértice	Distancia	V. Anterior
A	0	-
B	5	A
C	14	B
D	6	A
E	∞	-
F	∞	-



Visitados: [A,B]

No visitados: [C,D,E,F]

Vértice	Distancia	V. Anterior
A	0	-
B	5	A
C	14	B
D	6	A
E	12	D
F	8	D



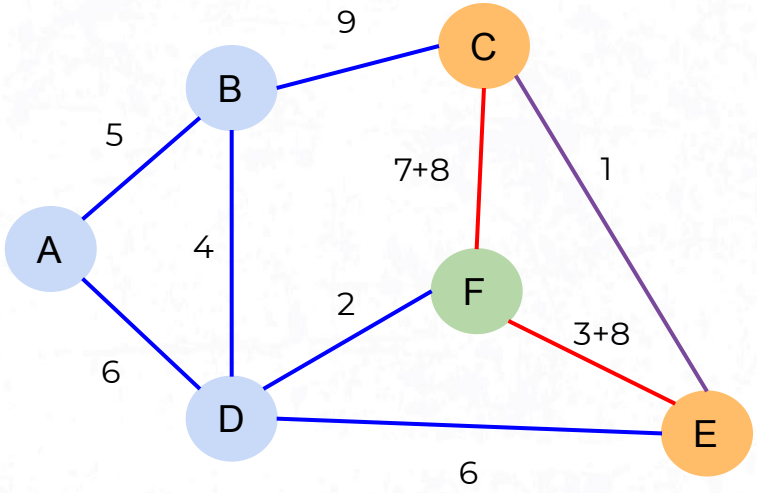
Visitados: [A,B,D]

No visitados: [C,E,F]

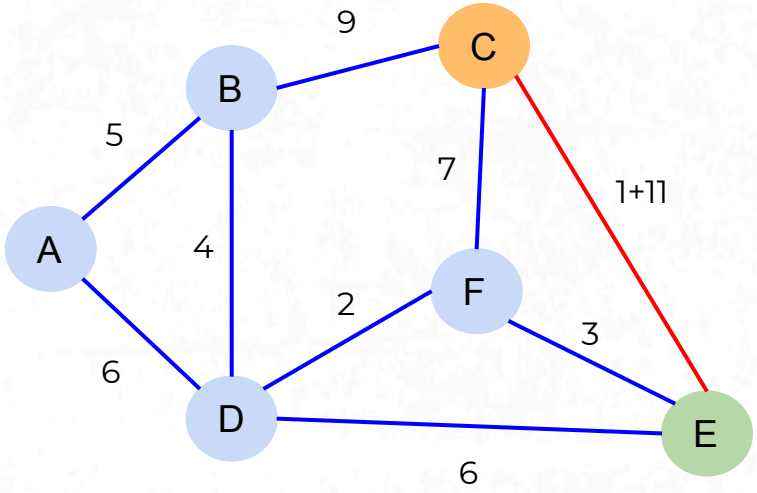
Vértice	Distancia	V. Anterior
A	0	-
B	5	A
C	14	B
D	6	A
E	**11**	**F**
F	8	D

Visitados: [A,B,D,F]

No visitados: [C,E]



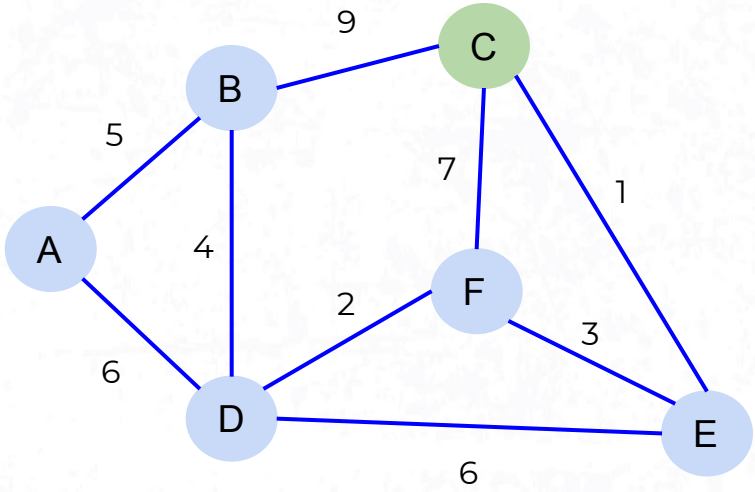
Vértice	Distancia	V. Anterior
A	0	-
B	5	A
C	**12**	**E**
D	6	A
E	11	F
F	8	D



Visitados: [A,B,D,F,E]

No visitados: [C]

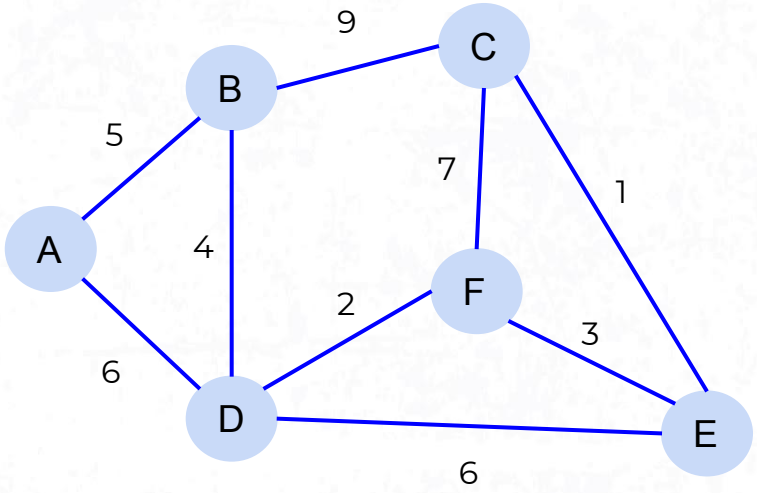
Vértice	Distancia	V. Anterior
A	0	-
B	5	A
C	12	E
D	6	A
E	11	F
F	8	D



Visitados: [A,B,D,F,E,C]

No visitados: []

Vértice	Camino Minimo	Costo
A → B	A → B	5
A → C	A → D → F → E → C	12
A → D	A → D	6
A → E	A → D → F → E	11
A → F	A → D → F	8



Visitados: [A,B,D,F,E,C] → **¿Y esto?**

Aplicaciones

Aplicaciones

- Redes sociales
- GPS
- Compiladores/Interpretes
- Redes de computadoras (servidores)
- Rutas

Preguntas?

Gracias

Vuelva prontos!