Descripción de la asignatura

- Tema 1: Introducción
- Tema 2: Árboles genéricos
- Tema 3: Mapas y Diccionarios
- Tema 4: Mapas y Diccionarios Ordenados
- **Tema 5: Grafos**
- Tema 6: EEDD en Memoria Secundaria

Bloque 1

Bloque 2

Bloque 3

V 1.2

Tema I.5 Grafos

angel.sanchez@urjc.es
jose.velez@urjc.es

abraham.duarte@urjc.es
raul.cabido@urjc.es

Índice

- Tipos de grafos y definiciones
- Implementaciones de los grafos
- Algoritmos sobre grafos

Objetivos

- Conocer la **terminología** relativa a los grafos y su tipología
- Dado un problema, **identificar** si la estructura de grafo se adapta a su solución
- **Implementación** eficiente de los grafos
- Implementación de **algoritmos** sobre grafos

Tipos de grafos y definiciones

Definición de grafo

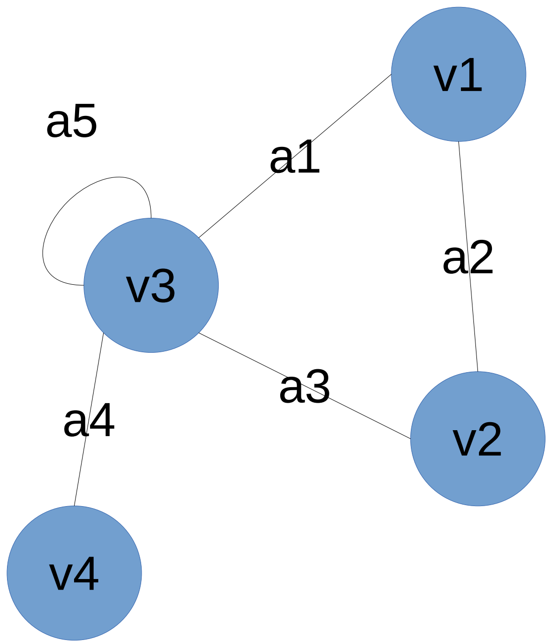
Un grafo ***G*** es un par ordenado ***G=(V,A)*** donde *V* es un conjunto de vértices o nodos y *A* es un conjunto de aristas o arcos que relacionan esos vértices.

Aunque ***V*** podría ser infinito, muchos resultados importantes para grafos solo se aplican a grafos finitos.

Grafo no dirigido o grafo

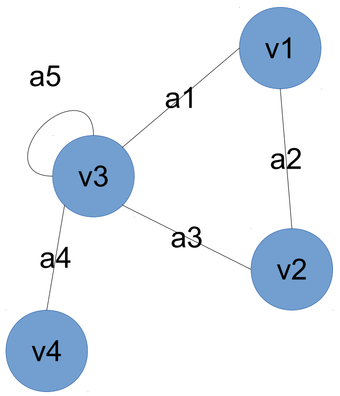
Es un grafo donde ***V*** no es vacío y ***A*** es un conjunto de pares no ordenados de elementos de ***V***.

V={v1,v2,v3,v4}
A={a1,a2,a3,a4,a5}
a1={v1,v3}
a2={v1,v2}
a3={v2,v3}
a4={v3,v4}
a5={v3,v3}



Definiciones

- Dos grafos ***G=(V,A)*** y ***G'=(V',A')*** se dicen isomorfos si existe una biyección ***f*** tal que **{*u*,*v*}∈*A*** si y solo si **{*f*(*u*),*f*(*v*)}∈*A*'**
- En un grafo se dice que una arista incide en un vértice si el vértice forma parte del par que define la arista
- En un grafo dos aristas son adyacentes si tienen un vértice en común
- Dos vértices son adyacentes o vecinos si existe una arista que los une
- Un bucle es una arista que conecta un vértice consigo mismo

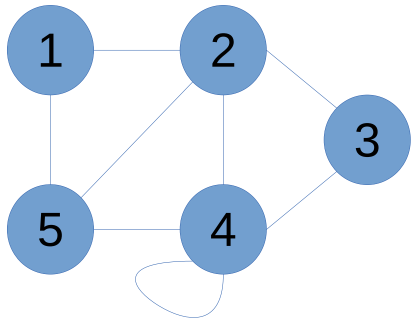


Son adyacentes las aristas: a1 y a5, a4 y a5, a1 y a4, a4 y a3, a3 y a2, a4 y a1.
No lo son, por ejemplo a2 y a4.
También, son adyacentes v4 y v3, v3 y v1, v3 y v2, v1 y v2.
Las arista a5 es un bucle.

Matriz de adyacencia de un grafo

Es una matriz en la que filas y columnas representan a los nodos y cualquier elemento de la matriz a_{ij} representa el número de arcos cuyos extremos son $\{v_i, v_j\}$

En los grafos la relación de adyacencia es simétrica



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	1	1
5	1	1	0	1	0

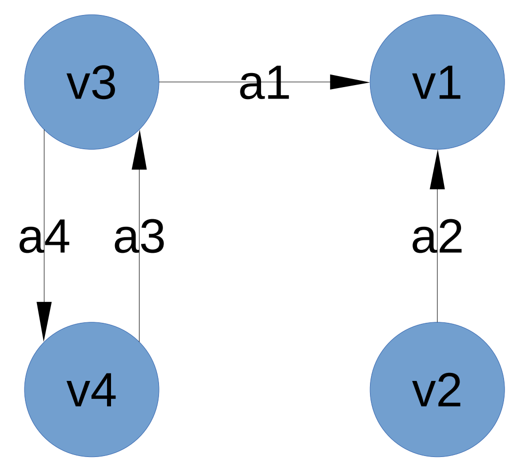
Grafo dirigido o digrafo

Es un grafo donde ***V*** no es vacío y A es un conjunto de pares ordenados de elementos de ***V***.

Dada un arista **(*a*,*b*)** se dice que *a* es su vértice inicial y *b* su vértice final.

Algunos autores proponen que los digrafos no pueden contener bucles y otros llaman digrafos simples a los digrafos sin bucles.

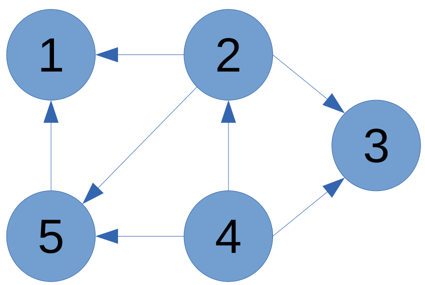
V={v1,v2,v3,v4}
A={a1,a2,a3,a4,a5}
a1=(v3,v1)
a2=(v2,v1)
a3=(v4,v3)
a4=(v3,v4)
a5=(v3,v3)



Matriz de adyacencia en digrafos

Es una matriz en la que filas y columnas representan a los nodos y cualquier elemento de la matriz a_{ij} representa el número de arcos cuyos extremos son $\boldsymbol{(v_i,v_j)}$

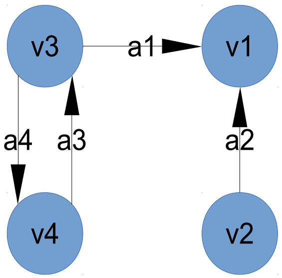
En los digrafos la relación de adyacencia no tiene porqué ser simétrica



	1	2	3	4	5
1	0	0	0	0	0
2	1	0	1	0	1
3	0	0	0	0	0
4	0	1	1	0	1
5	1	0	0	0	0

Adyacencia

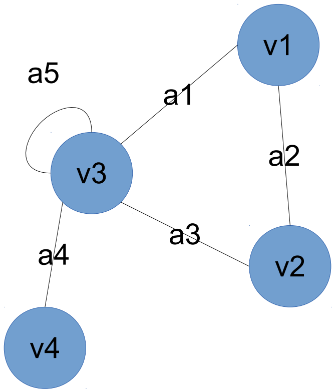
- En un digrafo se dice que una arista incide en un vértice si ese vértice es vértice final de la arista
- En un digrafo, el nodo v_j es adyacente al nodo v_i si el par ordenado $(v_i,v_j)\in V$
- En un grafo dirigido dos aristas son adyacentes si tienen un vértice en común que es final para una y origen para la otra
- Dos grafos $G=(V,A)$ y $G'=(V',A')$ se dicen isomorfos si existe una biyección f tal que $(u,v)\in A$ si y solo si $(f(u),f(v))\in A'$



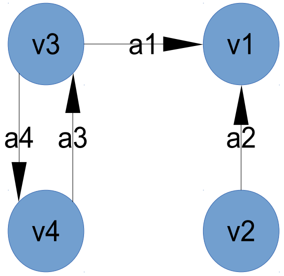
La arista a1 incide en v1 pero no en v3.
Las aristas a3 y a1 son adyacentes, pero a1 y a2 no lo son.
El vértice v1 es adyacente al v3, pero v3 no es adyacente a v1.
También se suele hablar de vecinos para referirse a los nodos adyacentes.

Caminos

- Un camino es una secuencia ordenada de nodos adyacentes y las aristas que los unen
- Al número de aristas dentro de un camino se le denomina longitud del camino
- Un ciclo de un grafo es un camino en el que no se repite ningún vértice, a excepción del primero que se repite al final
- Si hay un camino de ***u*** a ***v***, decimos que ***v*** es accesible desde ***u***



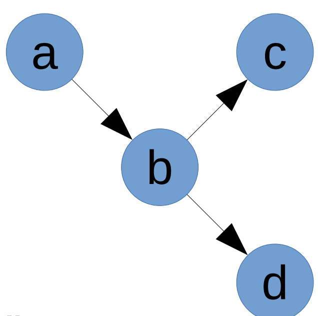
Entre v4 y v1 hay infinitos caminos y el mas corto tiene de longitud 2.



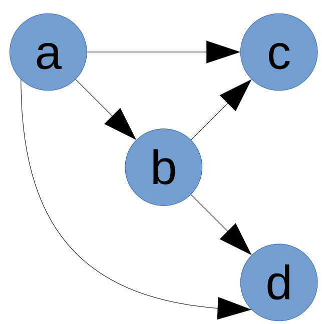
Entre v3 y v2 no hay ningún camino (v2 no es accesible desde v3).
Entre v4 y v1 hay un camino de longitud 2.

Cierre transitivo

Dado un digrafo ***G*(*V*,*A*)**, su clausura o cierre transitivo es el grafo $\boldsymbol{G^+(V,A^+)}$ tal que para todo par **(*v*,*w*)** en ***V*** hay un arco en $\boldsymbol{A^+}$ si y solo si hay un camino no nulo en ***G***.



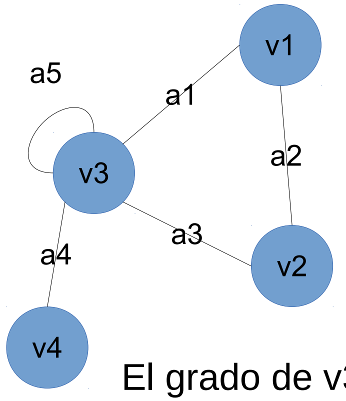
G(V,A)
V={a,b,c,d}
A={(a,b),(b,c),(b,d)}



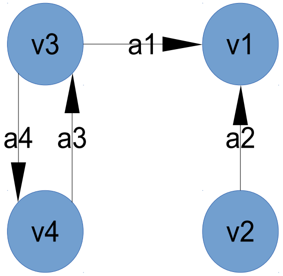
$G^+(V,A^+)$
V={a,b,c,d}
A^+={(a,b),(b,c),(b,d),(a,d),(a,c)}

Grado de un nodo

- En un grafo no dirigido el grado de un vértice es igual al número de vértices adyacentes, contando los bucles como 2
- En un grafo dirigido se define el grado de entrada a un vértice como el número de aristas que tienen al vértice como final
- En un grafo dirigido se define el grado de salida a un vértice como el número de aristas que tienen al vértice como inicial



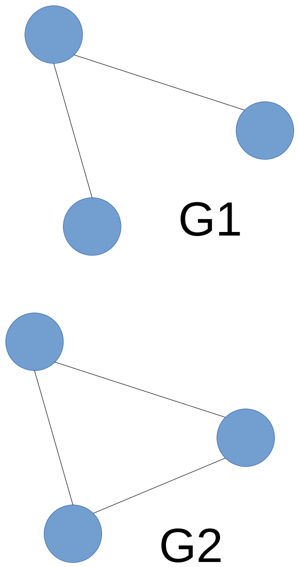
El grado de v3 es 5



El grado de entrada de v3 es 1 y el de salida 2

Tipos de grafos

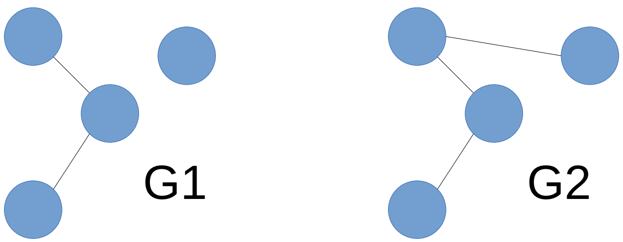
- Se llama grafo mixto a aquel que tiene aristas dirigidas y no dirigidas
- Se llama grafo nulo al que no tiene vértices ni aristas
- Se llama grafo vacío al que no tiene aristas
- Se llama grafo trivial al que tiene un solo vértice y ninguna arista
- Se llama grafo simple al que no tiene bucles
- Se llama grafo completo al grafo simple en el que cada par de vértices están unidos por una arista



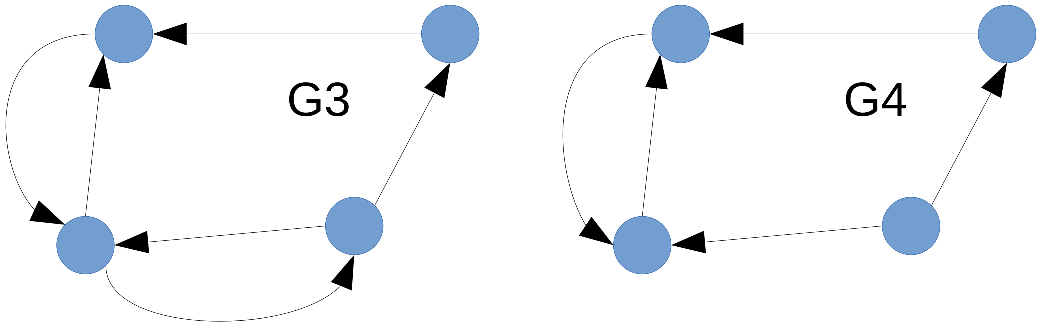
G1 no es completo, pero G2 sí.
Ambos son simples.

Conectividad

- Un grafo no dirigido se dice conexo si entre cada par de vértices existe un camino que los une
- Un grafo dirigido es fuertemente conexo si para cada par de vértices ***u*** y ***v*** existe un camino de ***u*** hacia ***v*** y un camino de ***v*** hacia ***u***



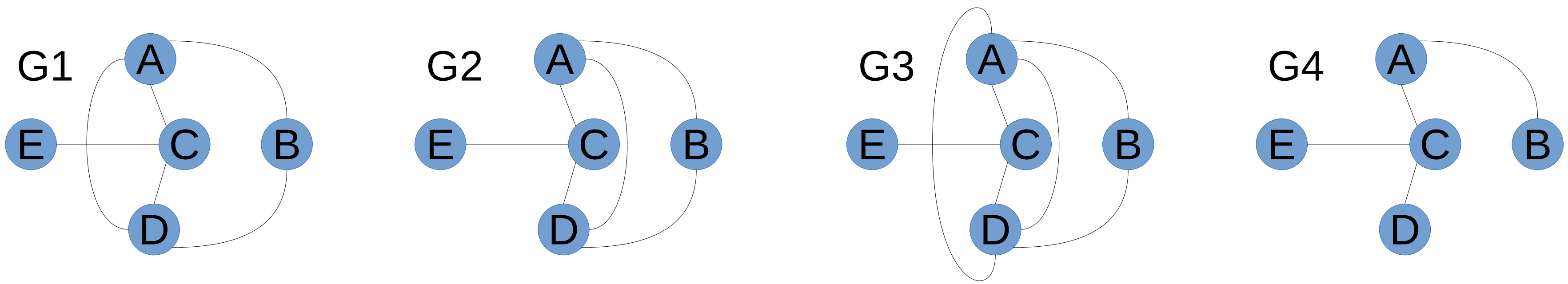
G1 no es conexo pero G2 sí



G3 es fuertemente conexo pero G4 no

Tipos de grafos

- Es un grafo plano aquel que se puede dibujar en una plano cartesiano sin cruzar aristas
- Un árbol es un grafo conexo y sin ciclos
- Un multigrafo (o pseudografo) es un grafo en el que se permiten múltiples aristas entre cada par de vértices



G1 no parece plano, pero G1 es isomorfo a G2 y como G2 es plano, G1 también lo es.

G3 es un multigrafo y G4 es un árbol

Tipos de grafos y definiciones

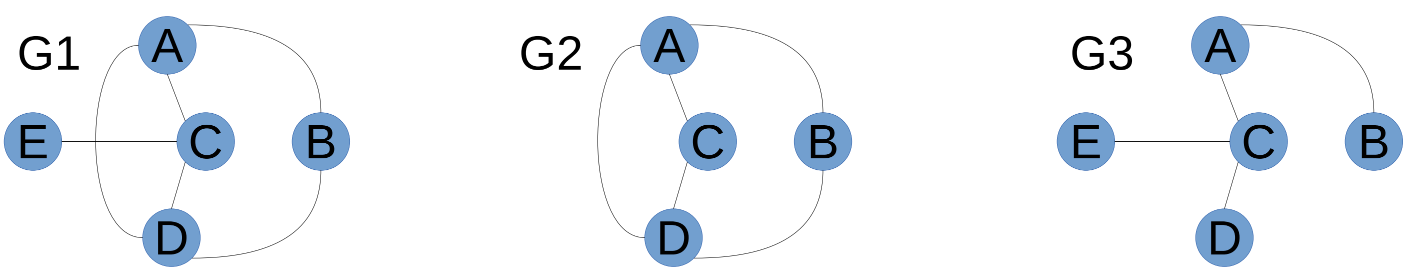
Subgrafo

Un subgrafo es una parte de un grafo que por si mismo es un grafo.

Un grafo ***G'=(V',A')*** es un subgrafo de ***G=(V,A)*** si y solo si se verifica que:

- $A' \subseteq A$
- $V' \subseteq V$

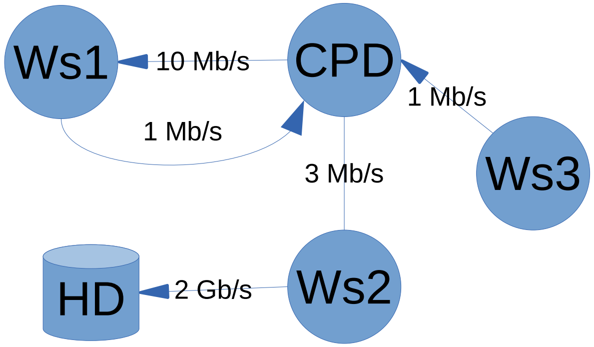
Un árbol se dice que es árbol de expasión o generador del grafo ***G*** si es un subgrafo de ***G*** que contiene todos los vértices de ***G***

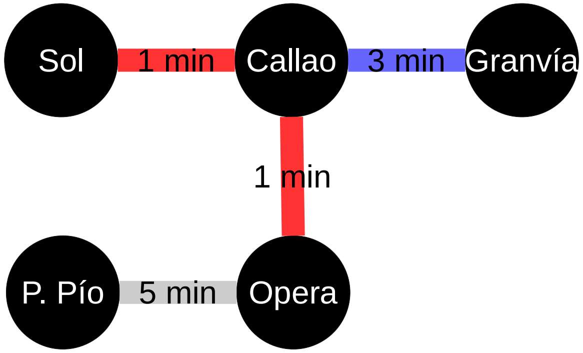


G2 y G3 son subgrafos de G1. Además, G3 es árbol de expasión de G1.

Grafos ponderados

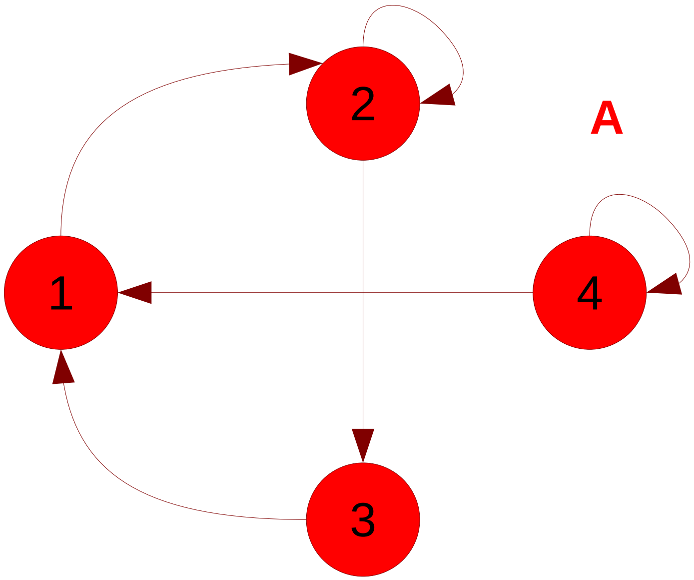
- Los grafos ponderados (también conocidos como valorados o con pesos) adjuntan a cada arista un valor numérico (peso o coste)
- El peso de una arista se suele ver como el coste asociado a transitar entre los vértices que une
- Estos grafos tienen múltiples aplicaciones en modelización, como por ejemplo: mapas, redes de ordenadores...



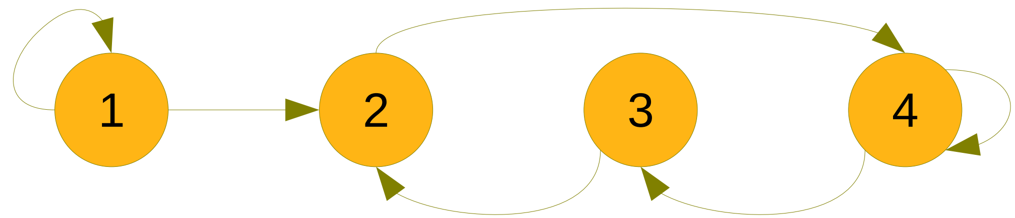


Tipos de grafos y definiciones

Repaso

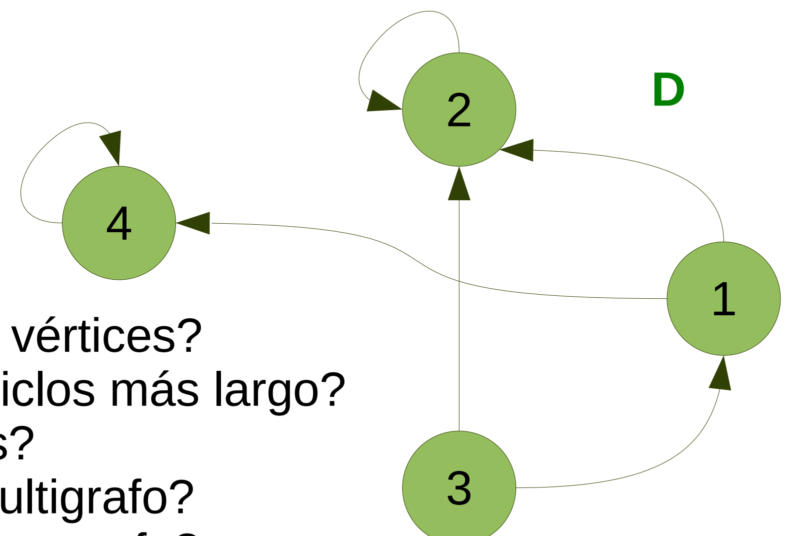


A

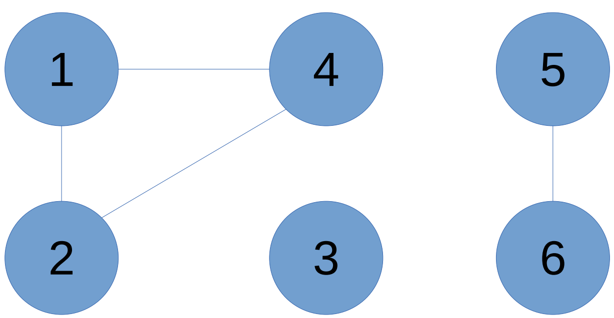


C

D



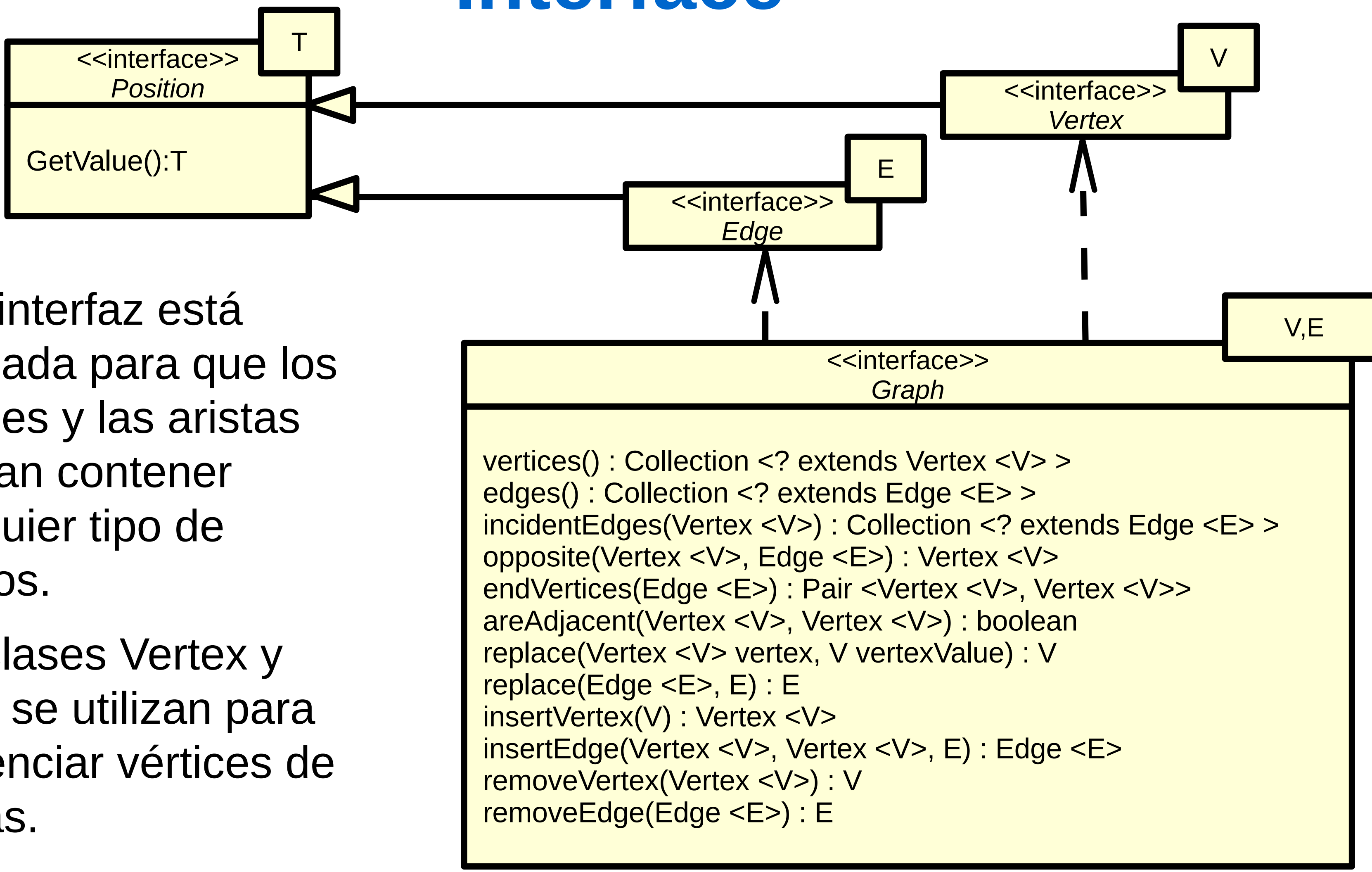
B



¿Grado de los vértices?
¿Camino sin ciclos más largo?
¿Isomorfismos?
¿Qué es un multigrafo?
¿Qué es un hipergrafo?
¿Vértices aislados?

Implementación

Interface



Esta interfaz está diseñada para que los vértices y las aristas puedan contener cualquier tipo de objetos.

Las clases Vertex y Edge se utilizan para diferenciar vértices de aristas.

Interface

```
public interface Graph <V,E> {

    /**
     * @return all vertices of the graph.
     */
    Collection <? extends Vertex <V> > vertices();

    /**
     * @return all the edges of the graph.
     */
    Collection <? extends Edge <E> > edges();

    /**
     * @return the end vertex of the edge e distinct of e.
     */
    Collection <? extends Edge <E> > incidentEdges(Vertex <V> v);

...
```

Interface

```
...

    /**
     * @return the end vertex of the edge e distinct of e.
     */
    Vertex <V> opposite(Vertex <V> v, Edge <E> e);
    /**
     * @return an array storing the end vertices of edge.
     */
    Pair <Vertex <V>, Vertex <V>> endVertices(Edge <E> edge);

    /**
     * Test whether vertices v1 and v2 are adjacent.
     * @return true if are adjacent
     */
    boolean areAdjacent(Vertex <V> v1, Vertex <V> v2);

...
```

Interface

```
...

    /**
     * Insert and return a new vertex storing element value.
     */
    Vertex <V> insertVertex(V value);

    /**
     * Insert and return a new undirected edge with end vertices
     * v1 and v2 and storing element vertexValue.
     */
    Edge <E> insertEdge(Vertex <V> v1, Vertex <V> v2, E edgeValue);

    /**
     * Replace the element stored at vertex with vertexValue.
     * @return the old element stored at vertex.
     */
    V replace(Vertex <V> vertex, V vertexValue);

...
```

Interface

```
...

    /**
     * Replace the element stored at edge with edgeValue.
     * @return the old element stored at edge.
     */
    E replace(Edge <E> edge, E edgeValue);

    /**
     * Remove vertex v and all its incident edges.
     * @return the element stored at vertex
     */
    V removeVertex(Vertex <V> vertex);

    /**
     * Remove edge
     * @return the element stored at e.
     */
    E removeEdge(Edge <E> edge);
}
```

Tipos de implementaciones

Se proponen 3 implementaciones de la interfaz grafo:

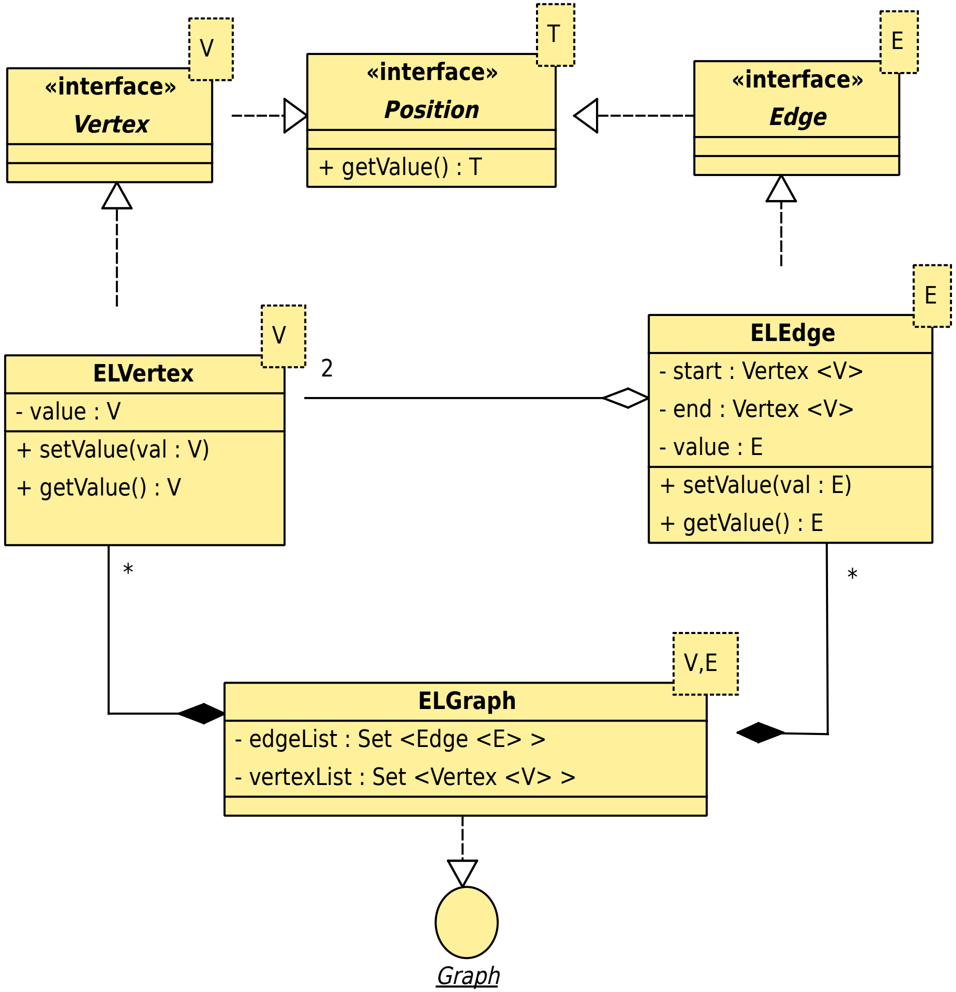
- Lista de aristas.- El grafo tiene un conjunto de aristas y otro de vértices Además, cada arista conoce los vértices que une.
- Lista de adyacencia.- Misma implementación que lista de aristas añadiendo que cada vértice conoce las aristas en que incide.
- Matriz de adyacencia.- Misma implementación que lista de aristas añadiendo una matriz de adyacencia.

Lista de aristas

La lista de aristas constituye la implementación más sencilla.

El grafo contiene conjuntos de aristas y de vértices, que se implementan utilizando tablas hash para optimizar las búsquedas.

Las aristas conocen los nodos que tienen en sus extremos, pero los nodos no conocen las aristas que inciden en ellos.



Lista de aristas

Operación	Coste
vertices	O(1)
edges	O(1)
endVertices	O(1)
opposite	O(1)
incidentEdges	O(m)
areAdjacent	O(m)
replace	O(1)
insertVertex	O(1)
insertEdge	O(1)
removeEdge	O(1)
removeVertex	O(m)

insertEdge precisa comprobar que la arista a insetar no exista y esto implica recorrer la lista de aristas.

incidentEdges requiere recorrer la lista de aristas para encontrar las incidentes a un vértice. areAdjacent también necesita recorrela para saber si dos vértices son adyacentes. Finalmente removeEdge y removeVertex necesita recorrerla para eliminar las aristas adyacentes.

El resto de complejidades son constantes porque la información necesaria es directamente accesible.

En vertices y edges se devuelve un iterador (de solo consulta) a la lista de vértices o de aristas. Si se requiriese una copia la complejidad sería O(n) y O(m) respectivamente.

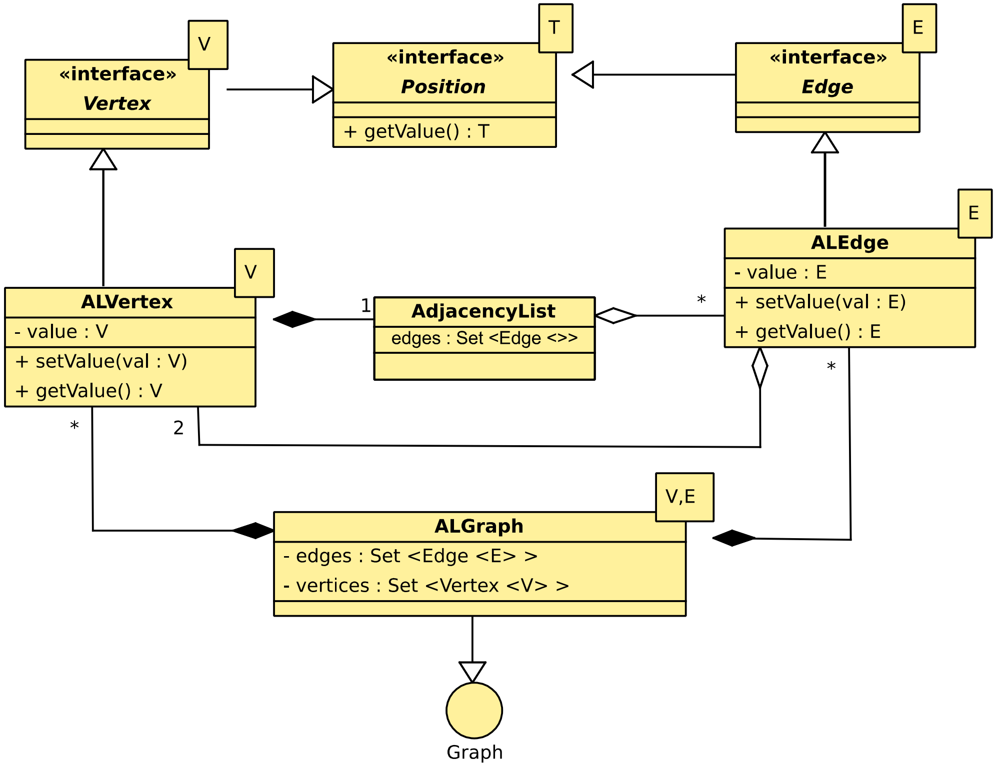
n=número de vértices, m=número de aristas

Lista de adyacencia

Esta implementación se caracteriza porque cada vértice tiene una lista de las aristas que inciden en él.

Para encontrar las aristas que inciden en un nodo solo hay que recorrer su lista de adyacencia asociada.

Suele ser ideal cuando los grafos están lejos de ser completos.



Lista de adyacencia

Operación	Coste
vertices	O(1)
edges	O(1)
endVertices	O(1)
opposite	O(1)
incidentEdges(v)	O(1)
areAdjacent(v,w)	O(min(grado(v),grado(w))
replace	O(1)
insertVertex	O(1)
insertEdge	O(1)
removeEdge	O(1)
removeVertex(v)	O(grado(v))

n=número de vértices, m=número de aristas

En areAdjacent se mirá el tamaño de las listas de adyacencia de los dos nodos involucrados, eligiendo la menor. Luego se recorre dicha lista intentando encontrar una arista que una ambos nodos.

En removeVertex hay que recorrer su lista de adyacencia para eliminar las aristas en las que era extremo.

El resto de complejidades son constantes porque la información necesaria es directamente accesible.

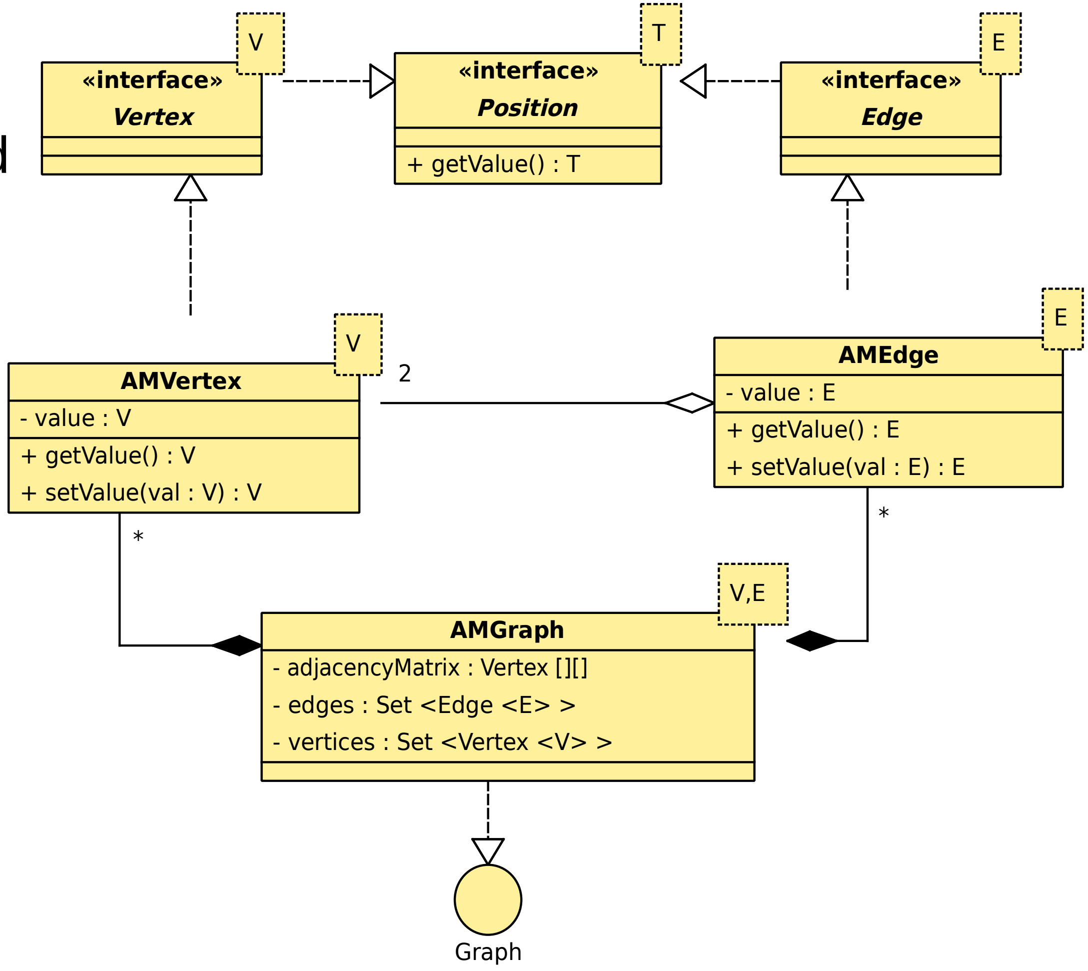
Matriz de adyacencia

Esta implementación añade una matriz de aristas como propiedad a la clase AMGraph.

Añadir la matriz de adyacencia permite saber si dos nodos son adyacentes simplemente consultando la matriz.

Si hay muchos nodos y pocas aristas, la matriz desperdicia mucho espacio.

Suele ser adecuado en grafos pequeños o cercanos a ser grafos completos.



Matriz de adyacencia

Operación	Coste
vertices	O(1)
edges	O(1)
endVertices	O(1)
opposite	O(1)
incidentEdges(v)	O(n)
areAdjacent	O(1)
replace	O(1)
insertVertex	$O(n^2)$
insertEdge(v,w)	O(1)
removeEdge	O(1)
removeVertex	$O(n^2)$

El coste de insertVertex y removeVertex es **$O(n^2)$** pues requiere copiar la matriz a una más grande (o más pequeña) cada vez que se añade (o se borra) un vértice

Una implementación basada en ArrayList de ArrayList de Aristas no crería una nueva matriz al añadir un elemento, salvo cuando el ArrayList se llena. Usando ArrayList, en media la complejidad de insertVertex y de removeVertex sería O(1). El peor caso segiría siendo $O(n^2)$, pero esto ocurriría raramente.

incidentEdges precisa recorrer la fila de la matriz correspondiente al nodo consultado.

El resto de complejidades son constantes porque la información necesaria es directamente accesible.

n=número de vértices, m=número de aristas

Algoritmos

Índice de algoritmos

- Algoritmos de búsqueda
  - En profundidad
  - En anchura
- Camino más corto
- Árbol de expansión
- Algoritmos sobre digrafos
  - Cierre transitivo

Recorrido en profundidad

Búsqueda en profundidad (DFS)

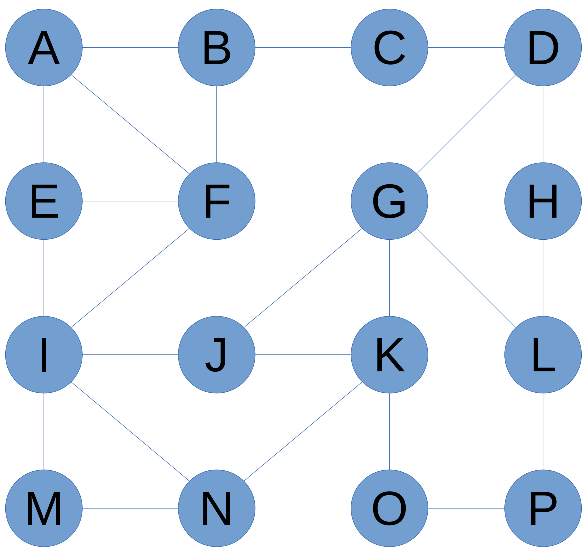
- Permite visitar todos los nodos del grafo conexo
- Procede profundizando por un camino hasta llegar a un punto en que no es posible continuar y luego explora las últimas posibilidades de la misma forma
- Se obtiene un recorrido no homogéneo, ya que los caminos de una longitud igual no se exploran en instantes de tiempo consecutivos

Recorrido en profundidad (DFS)

1. Se inserta el nodo desde el que se inicia la exploración en una pila y se etiqueta como visitado
2. Si la pila está vacía el algoritmo termina
3. Si el nodo de la cabeza de la pila no tiene ningún nodo adyacente sin etiqueta de visitado, se elimina la cabeza de la pila y se vuelve al paso 2
4. Se toma el primer nodo adyacente al nodo de la cabeza de la pila que no esté etiquetado como visitado, se etiqueta como visitado, se inserta en la pila y se vuelve al paso 2

Durante el proceso, los arcos que llevan a un nodo no visitado se etiquetan como aristas de exploración y los arcos que llevan a nodos visitados se etiquetan como aristas de vuelta (back).

Recorrido en profundidad (DFS)



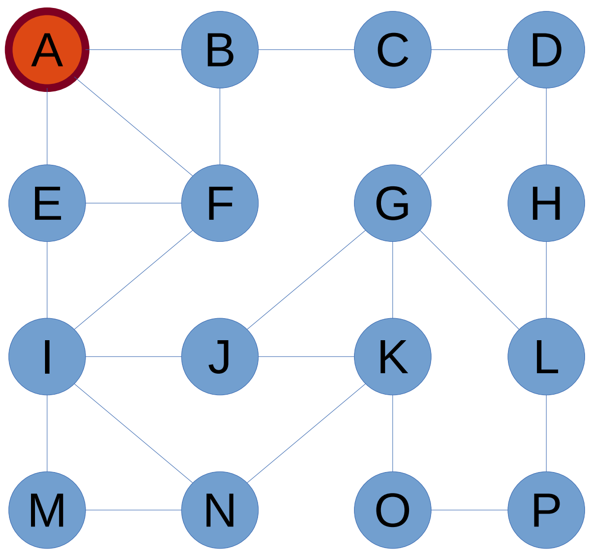
Pila

En la siguiente secuencia de transparencias vamos a explorar el grafo de la figura partiendo del nodo A

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



Pila

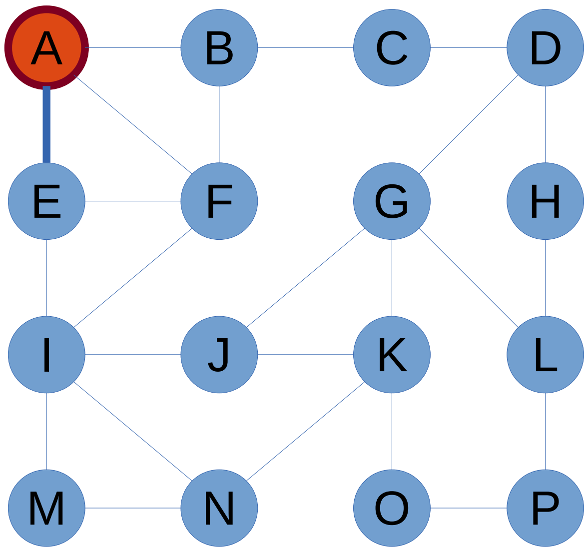
A

Se comienza insertando el nodo de inicio en la pila

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



Pila

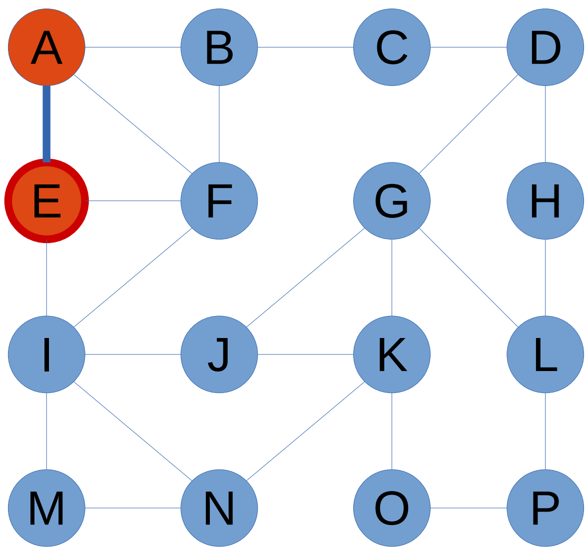
A

Ahora se revisan los nodos adyacentes al nodo A. En este ejemplo, supondremos la existencia de la siguiente lista de adyacencia.

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



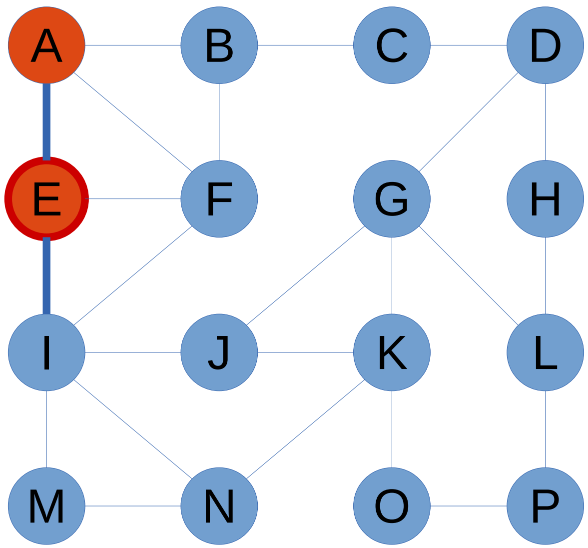
Pila

A, E

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



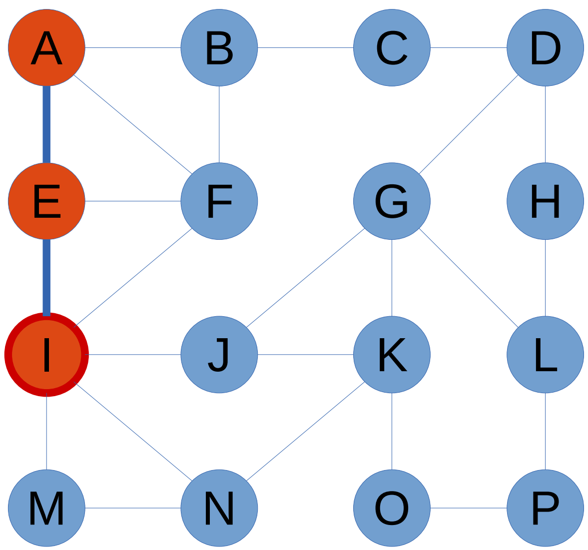
Pila

A, E

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



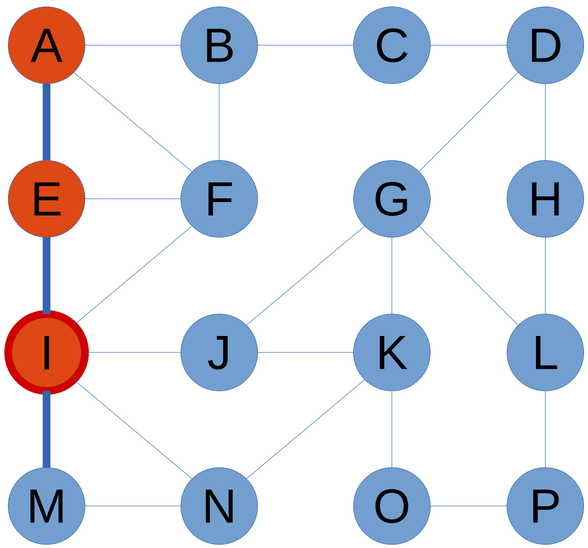
Pila

A, E, I

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	BA		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



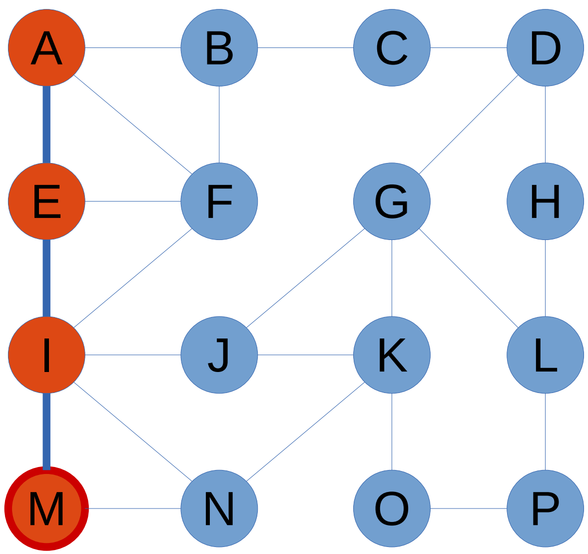
Pila

A, E, I

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



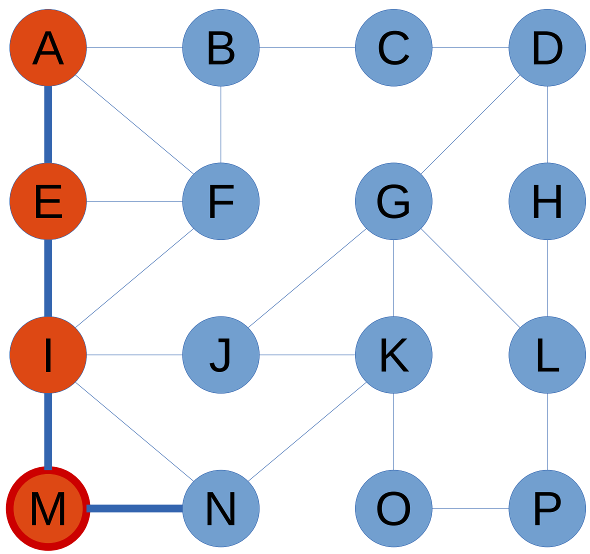
Pila

A, E, I, M

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	BA		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



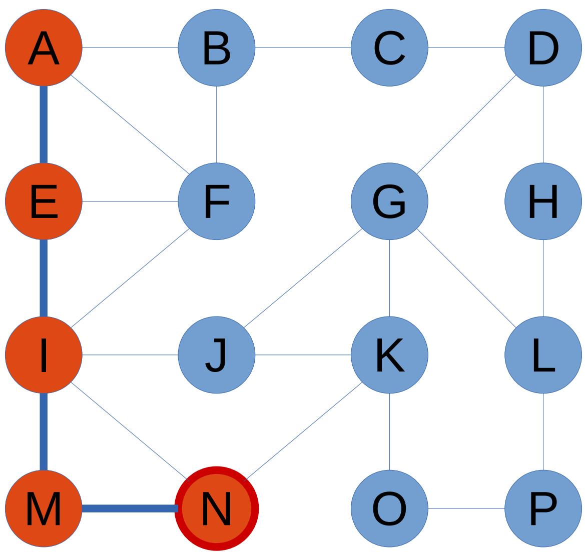
Pila

A, E, I, M

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	KO	LP	**MN**	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



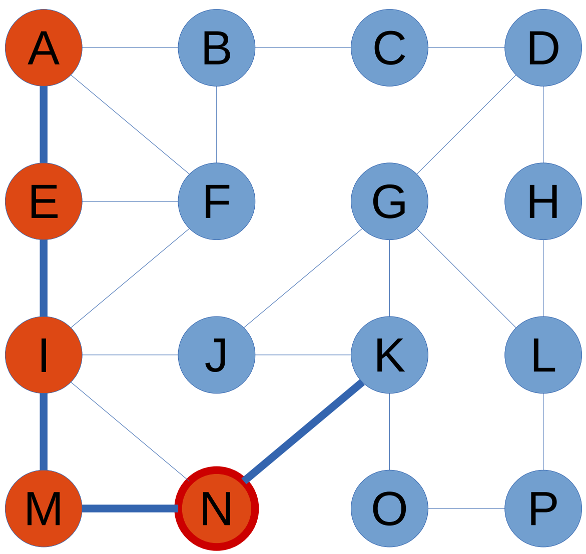
Pila

A, E, I, M, N

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	KO	LP	**MN**	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



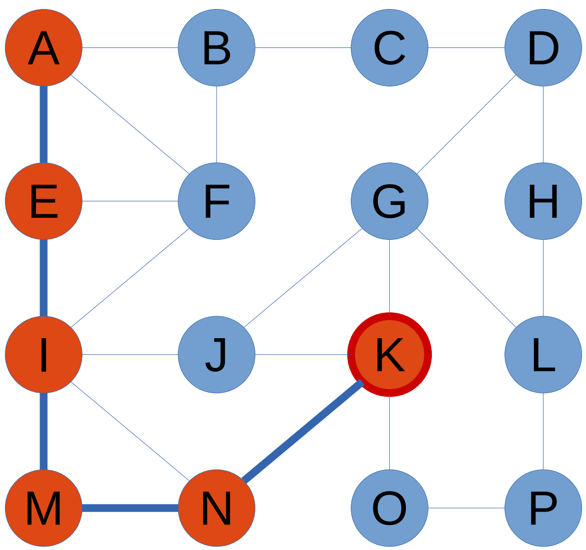
Pila

A, E, I, M, N

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	KO	LP	**MN**	**KN**	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



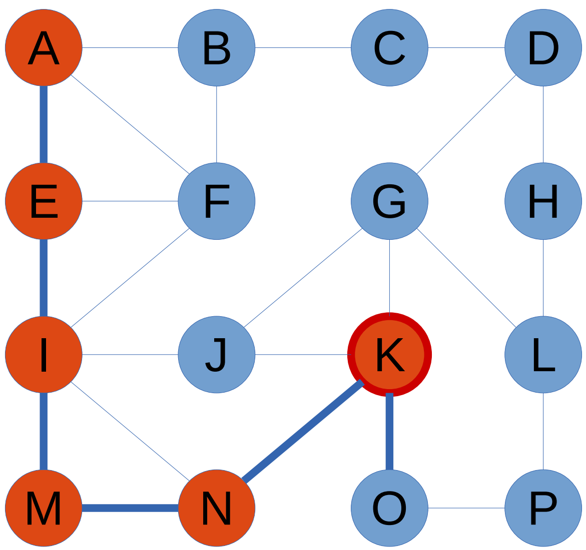
Pila

A, E, I, M, N, K

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	KO	LP	**MN**	**KN**	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



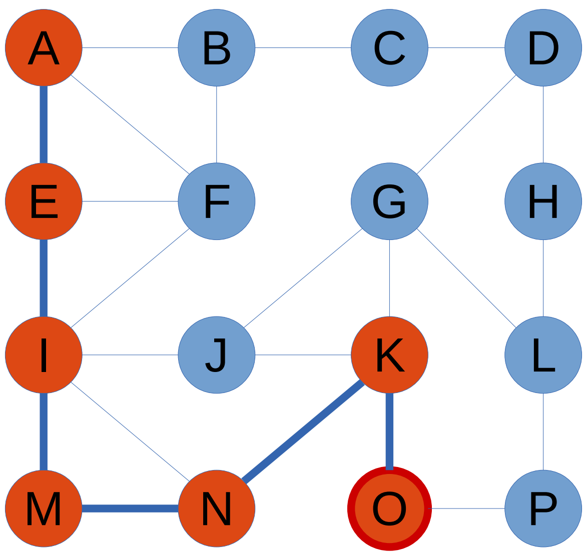
Pila

A, E, I, M, N, K

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	LP	**MN**	**KN**	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



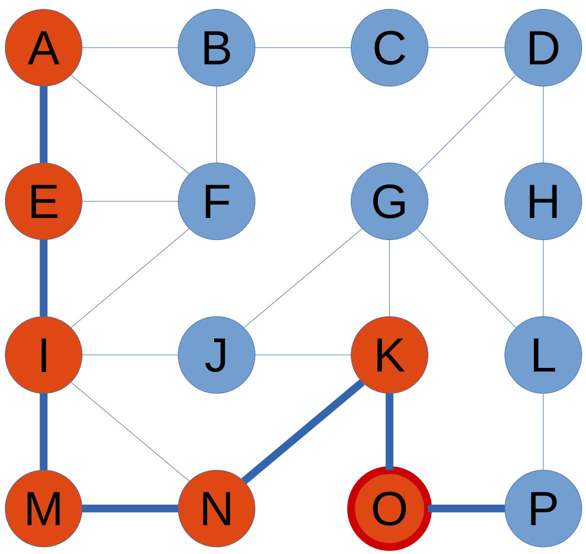
Pila

A, E, I, M, N, K, O

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	LP	**MN**	**KN**	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



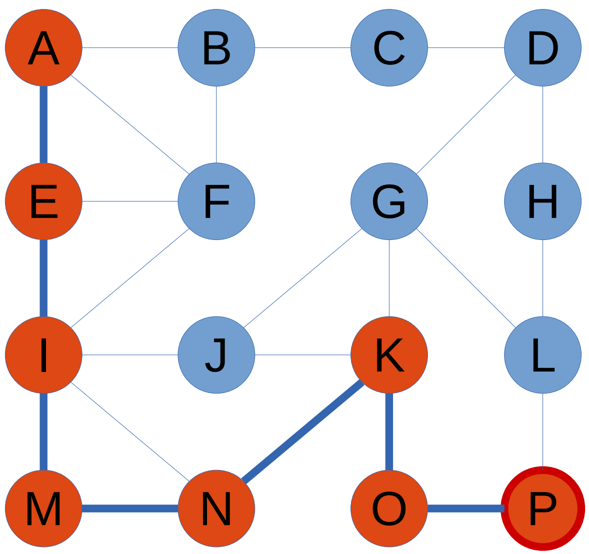
Pila

A, E, I, M, N, K, O

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	LP	**MN**	**KN**	**OP**	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



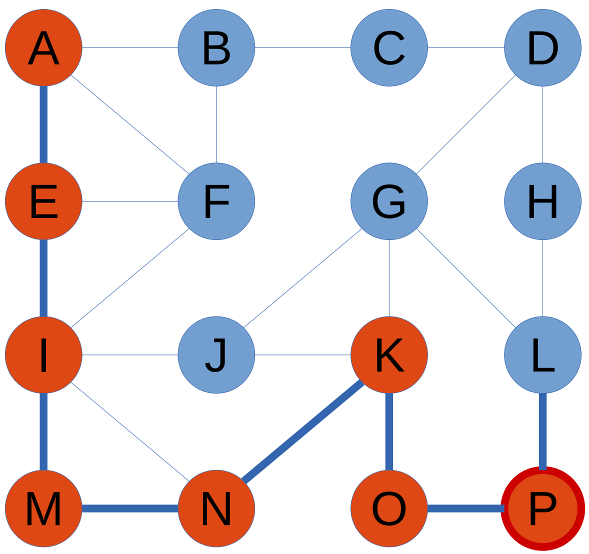
Pila

A, E, I, M, N, K, O, P

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	LP	**MN**	**KN**	**OP**	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



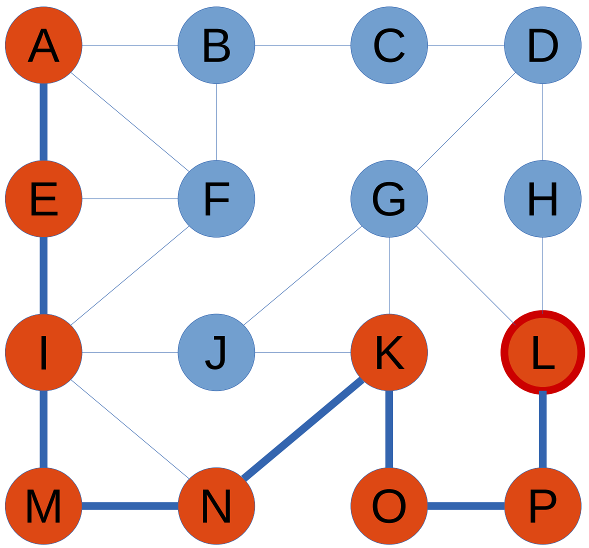
Pila

A, E, I, M, N, K, O, P

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	LP	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



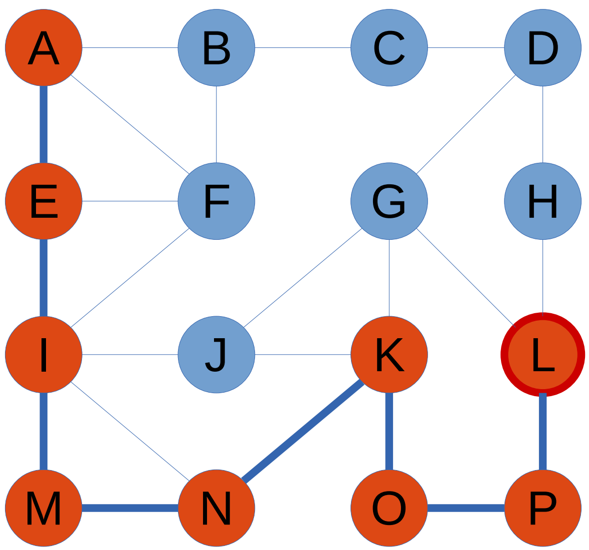
Pila

A, E, I, M, N, K, O, P, L

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	LP	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



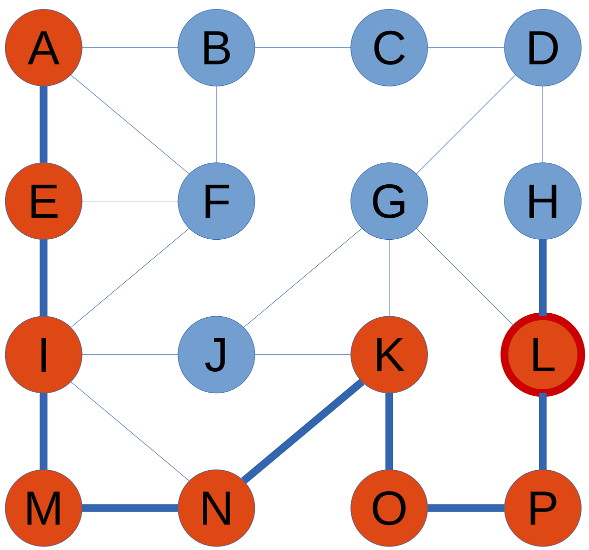
Pila

A, E, I, M, N, K, O, P, L

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



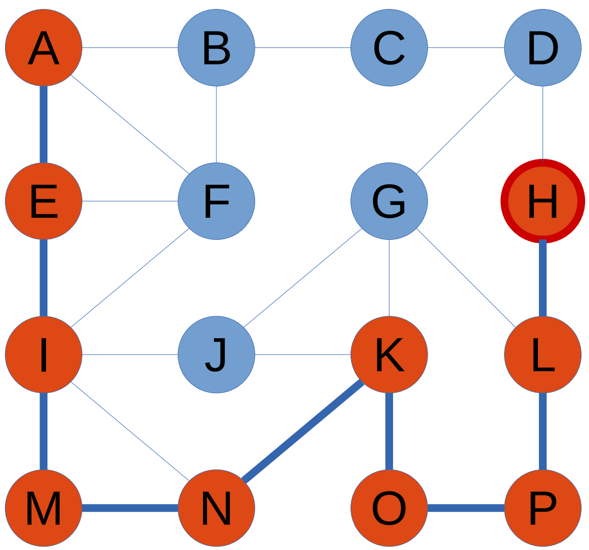
Pila

A, E, I, M, N, K, O, P, L

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



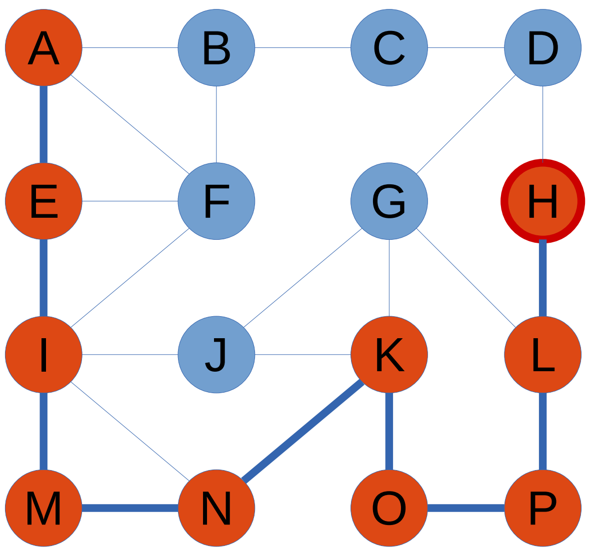
Pila

A, E, I, M, N, K, O, P, L, H

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	HL	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



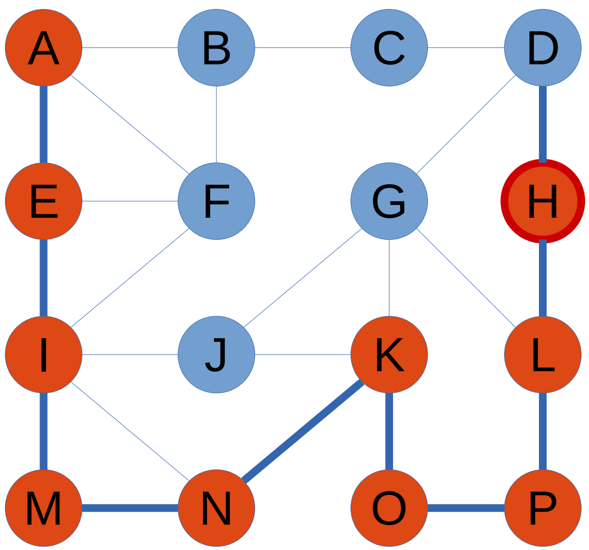
Pila

A, E, I, M, N, K, O, P, L, H

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



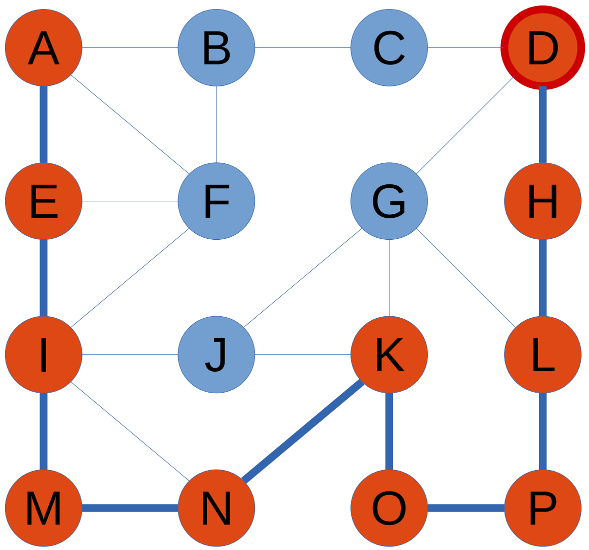
Pila

A, E, I, M, N, K, O, P, L, H

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



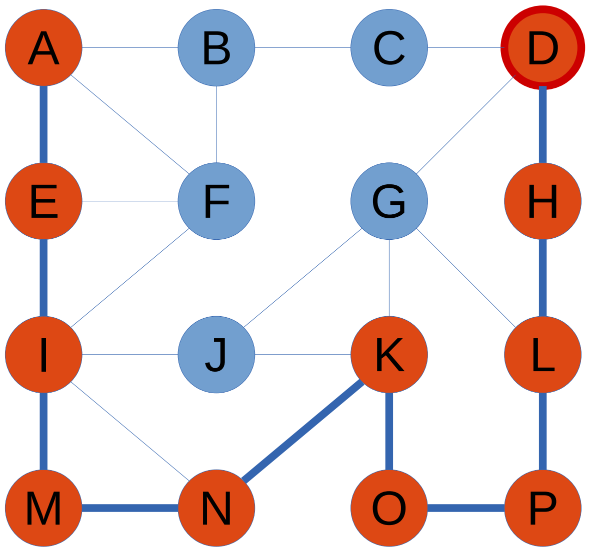
Pila

A, E, I, M, N, K, O, P, L, H, D

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



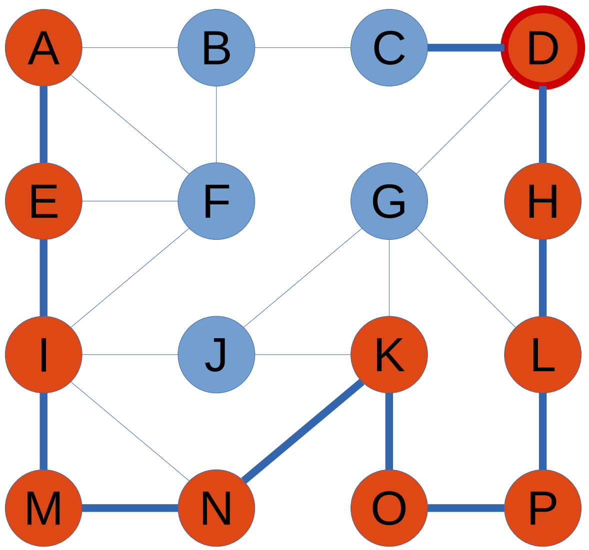
Pila

A, E, I, M, N, K, O, P, L, H, D

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	**DH**	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	CD	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



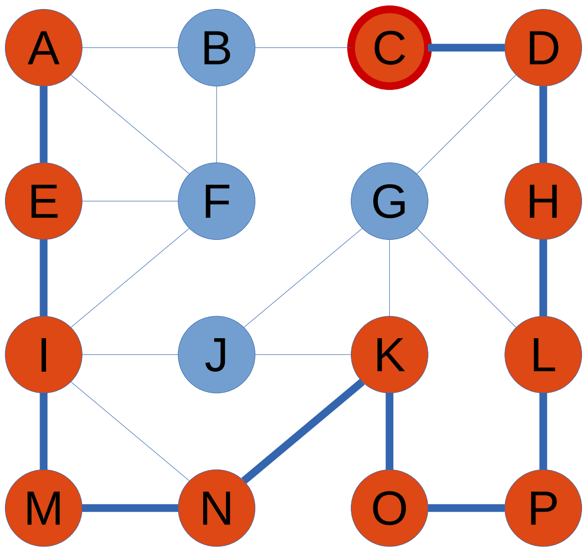
Pila

A, E, I, M, N, K, O, P, L, H, D

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	**DH**	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	**CD**	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



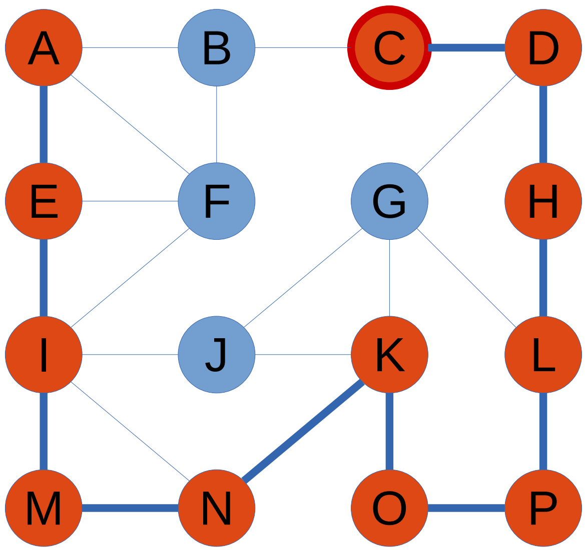
Pila

A, E, I, M, N, K, O, P, L, H, D, C

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	**DH**	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	**CD**	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



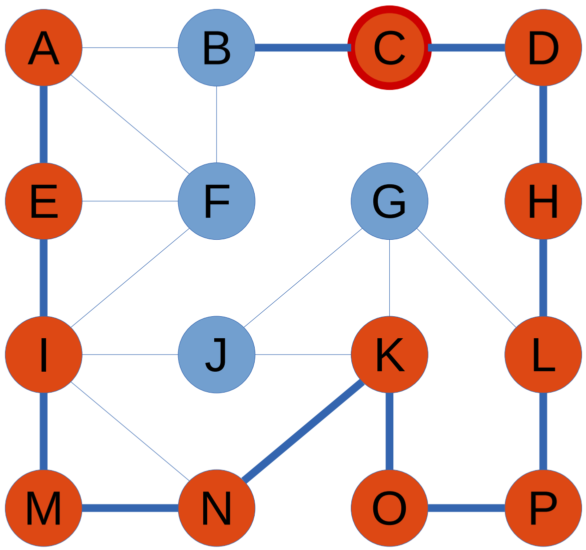
Pila

A, E, I, M, N, K, O, P, L, H, D, C

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	**CD**	**DH**	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	BC	**CD**	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



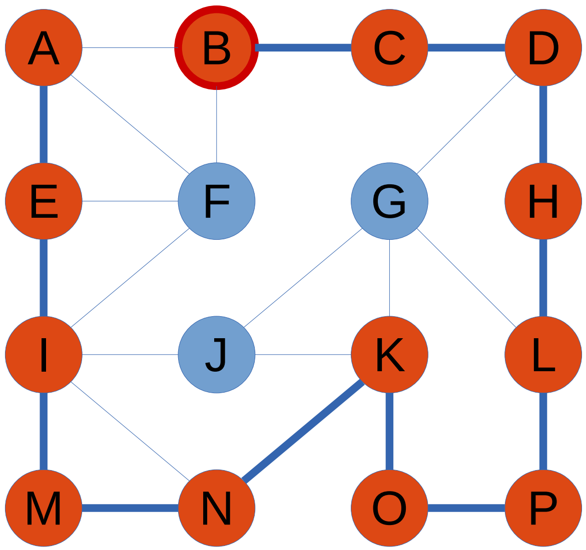
Pila

A, E, I, M, N, K, O, P, L, H, D, C

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	**CD**	**DH**	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



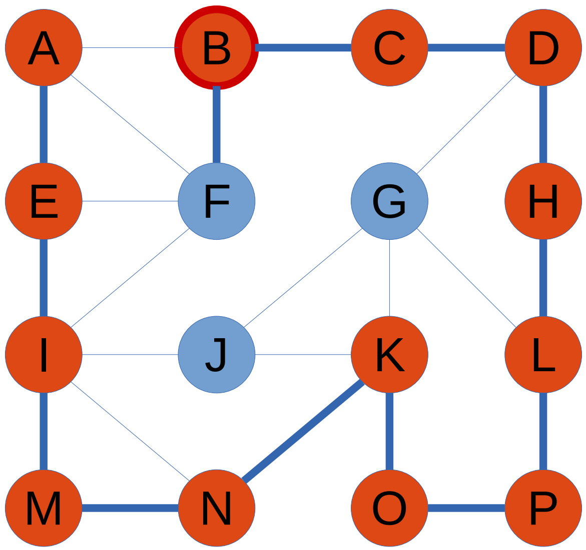
Pila

A, E, I, M, N, K, O, P, L, H, D, C, B

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	**CD**	**DH**	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



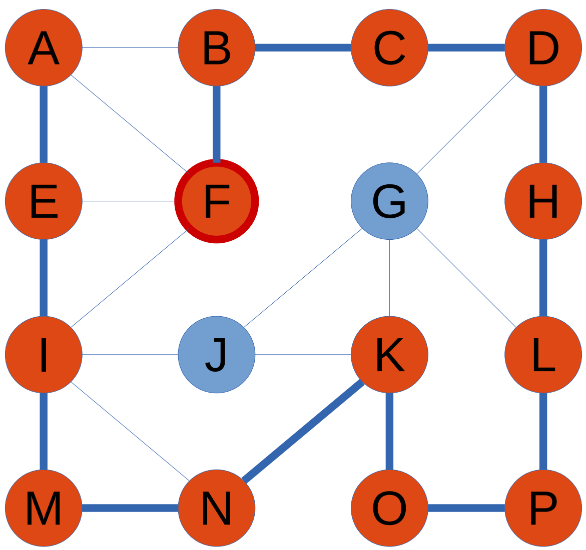
Pila

A, E, I, M, N, K, O, P, L, H, D, C, B

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



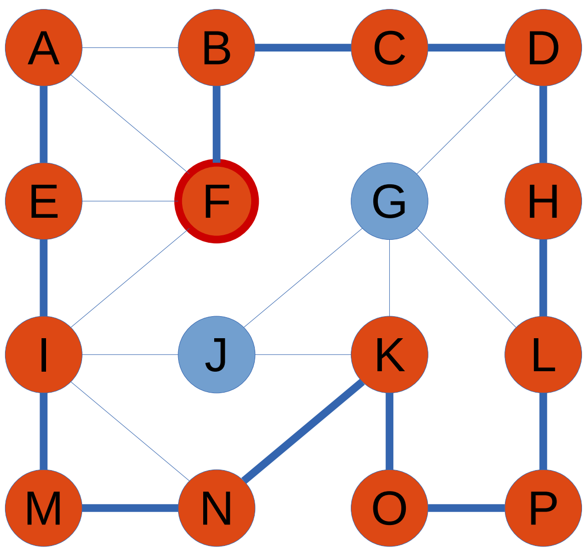
Pila

A, E, I, M, N, K, O, P, L, H, D, C, B, F

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	BF	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



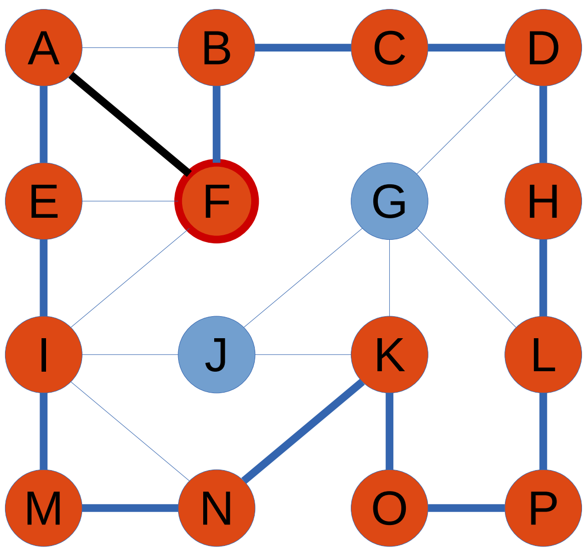
Pila

A, E, I, M, N, K, O, P, L, H, D, C, B, F

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	AF	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



Pila

A, E, I, M, N, K, O, P, L, H, D, C, B, F

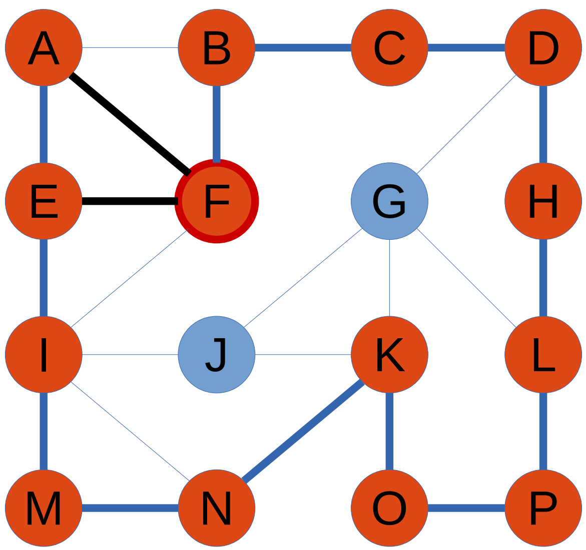
La aparición de una arista de vuelta
(en negro) indica la aparición de un ciclo

Observar que estas aristas llevan a
nodos ya analizados

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



Pila

A, E, I, M, N, K, O, P, L, H, D, C, B, F

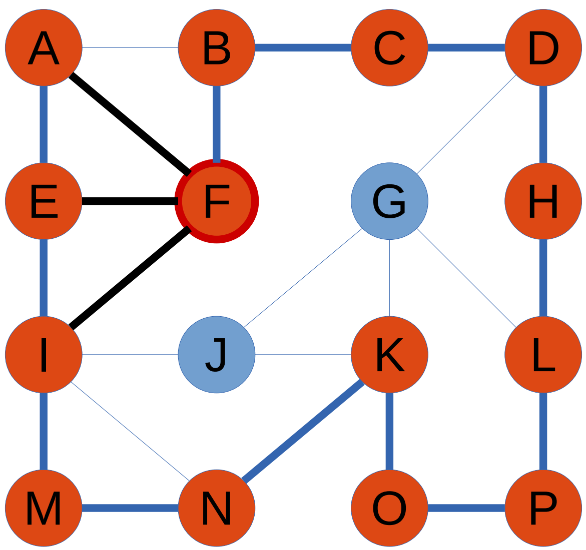
La aparición de una arista de vuelta (en negro) indica la aparición de un ciclo

Observar que estas aristas llevan a nodos ya analizados

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



Pila

A, E, I, M, N, K, O, P, L, H, D, C, B, F

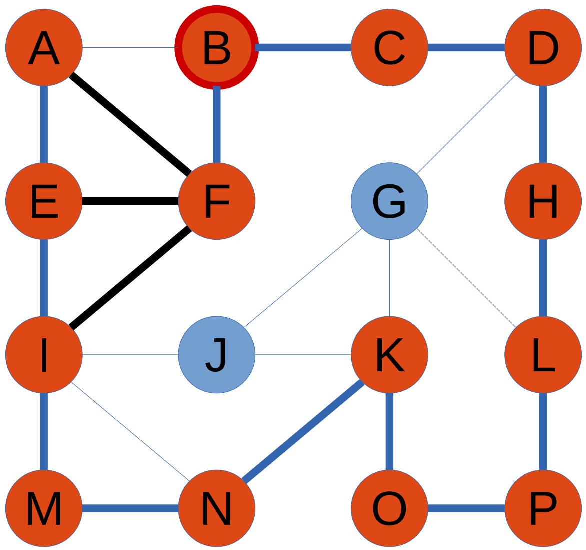
La aparición de una arista de vuelta (en negro) indica la aparición de un ciclo

Observar que estas aristas llevan a nodos ya analizados

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



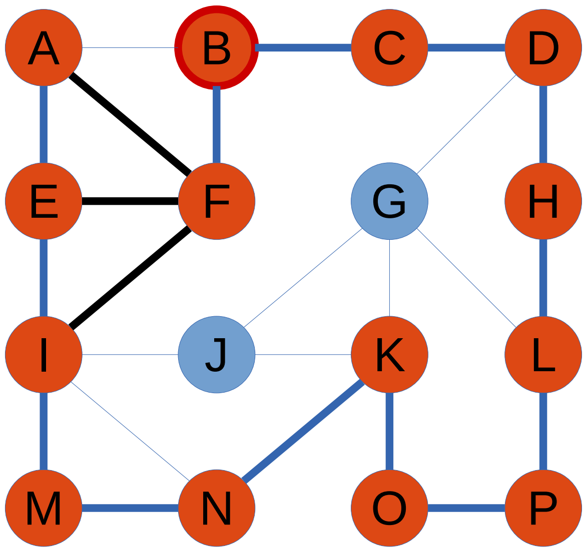
Pila

A, E, I, M, N, K, O, P, L, H, D, C, B

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	BC	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



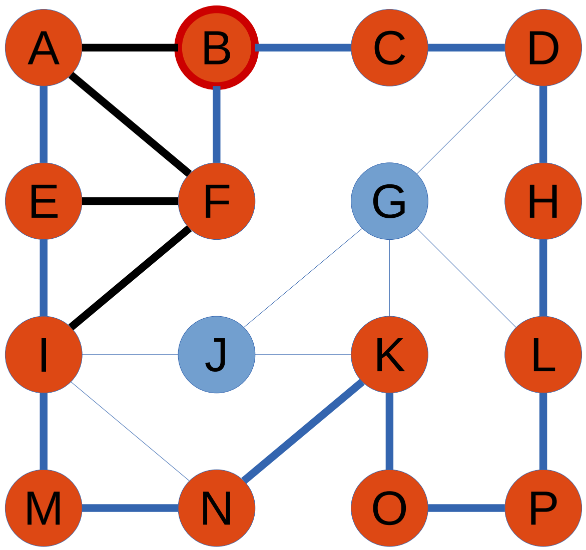
Pila

A, E, I, M, N, K, O, P, L, H, D, C, B

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	AB		DG	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



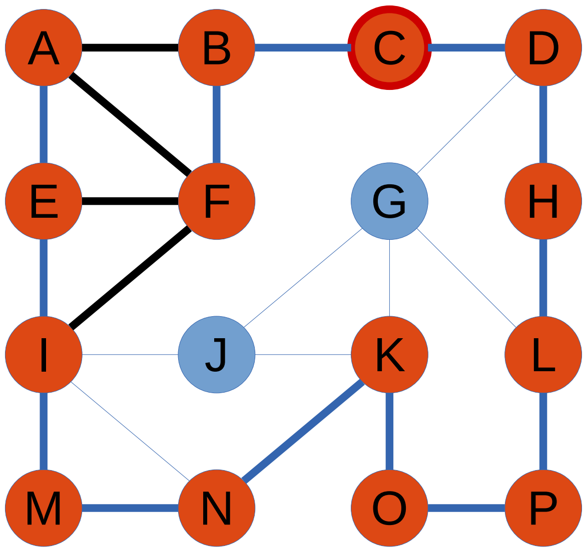
Pila

A, E, I, M, N, K, O, P, L, H, D, C, B

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		DG	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



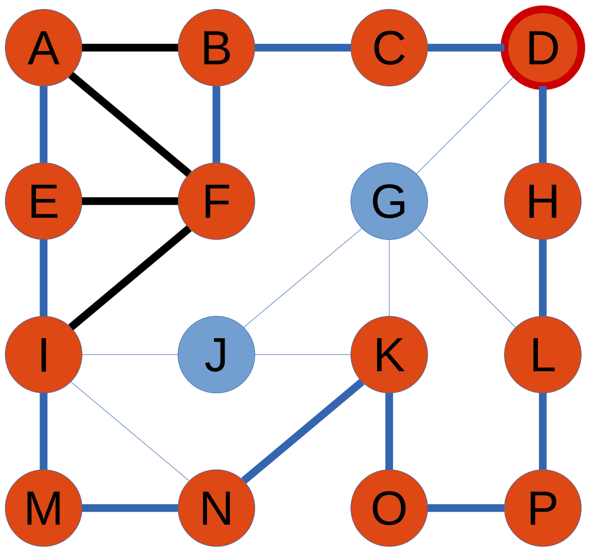
Pila

A, E, I, M, N, K, O, P, L, H, D, C

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		DG	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



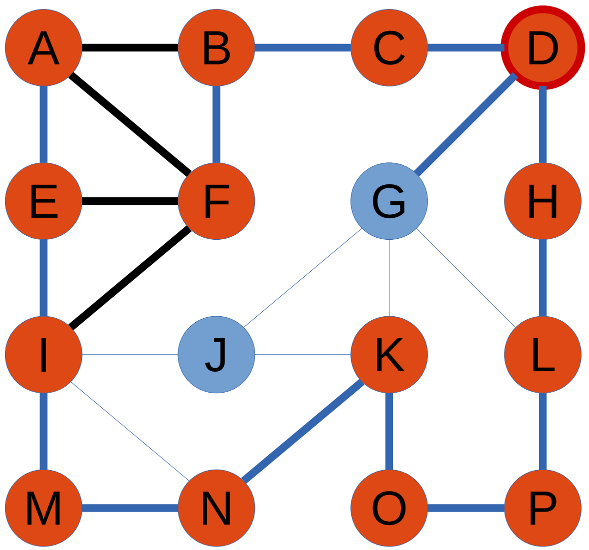
Pila

A, E, I, M, N, K, O, P, L, H, D

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		DG	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



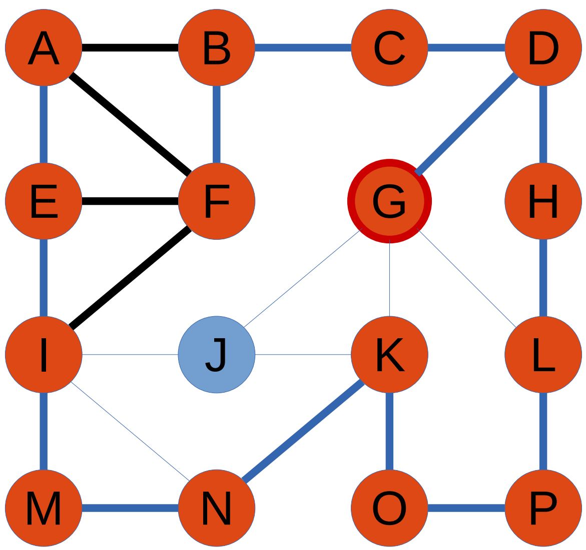
Pila

A, E, I, M, N, K, O, P, L, H, D

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



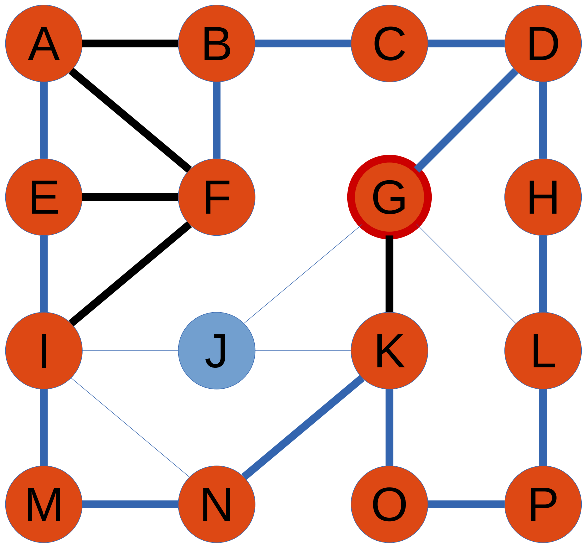
Pila

A, E, I, M, N, K, O, P, L, H, D, G

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	GK	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	GL	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



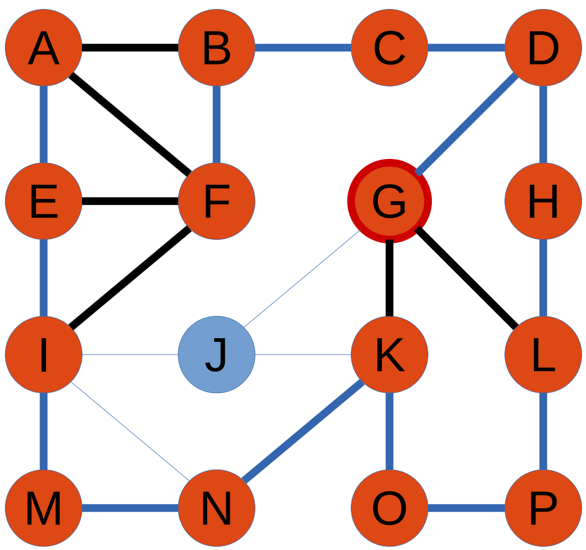
Pila

A, E, I, M, N, K, O, P, L, H, D, G

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



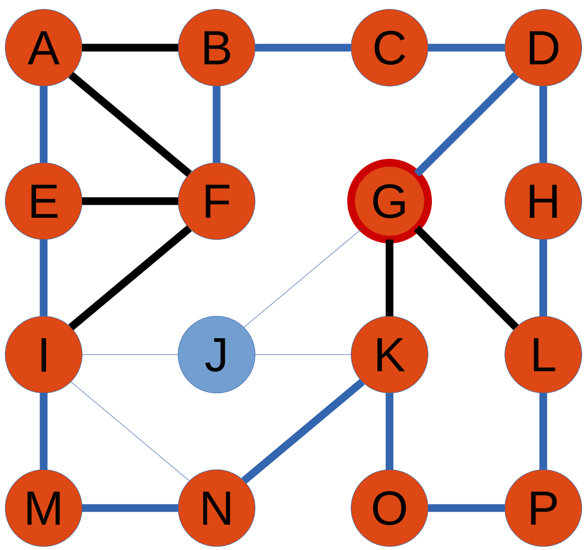
Pila

A, E, I, M, N, K, O, P, L, H, D, G

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



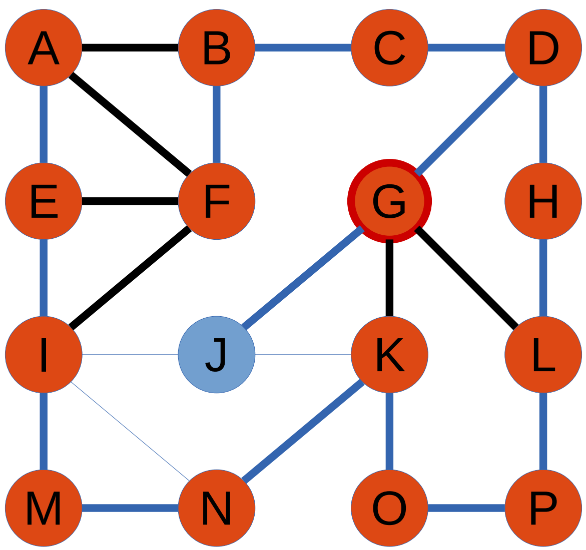
Pila

A, E, I, M, N, K, O, P, L, H, D, G

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



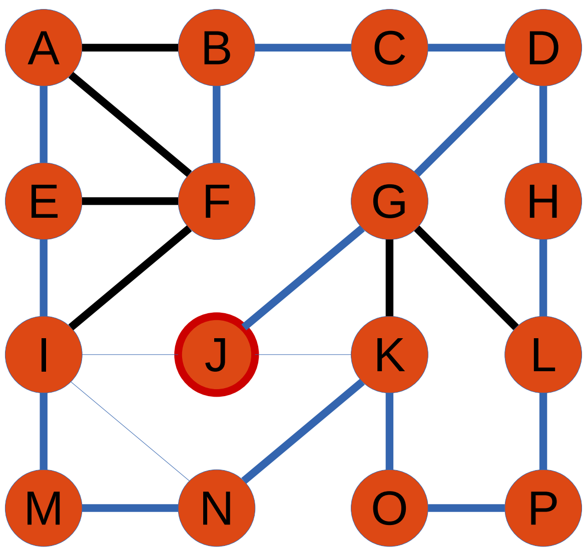
Pila

A, E, I, M, N, K, O, P, L, H, D, G

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	IJ	JK	GL		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



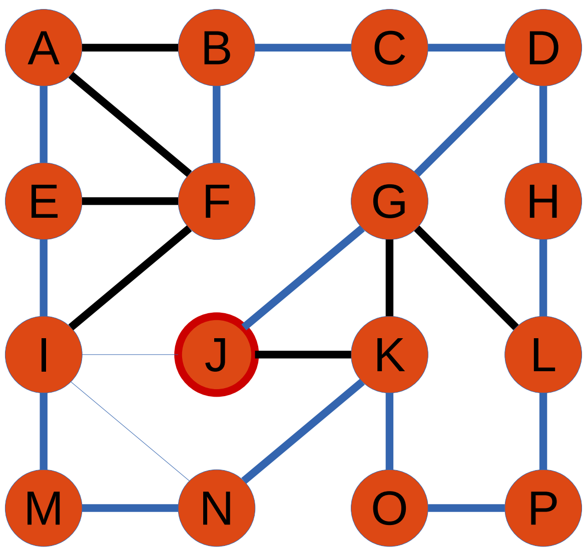
Pila

A, E, I, M, N, K, O, P, L, H, D, G, J

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	JK	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	IJ	JK	GL		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



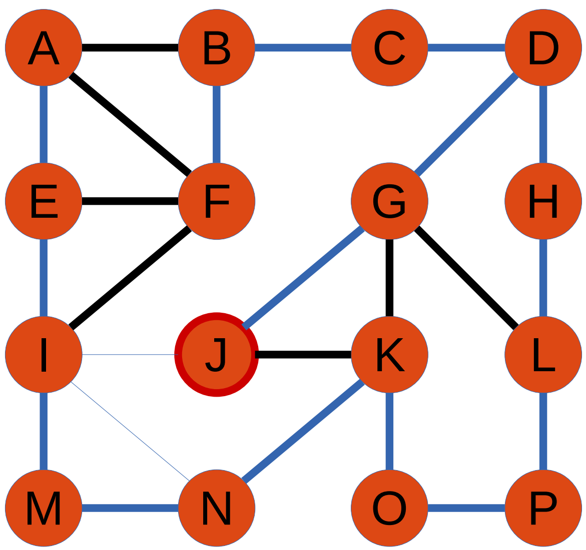
Pila

A, E, I, M, N, K, O, P, L, H, D, G, J

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	GJ	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	IJ	JK	GL		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



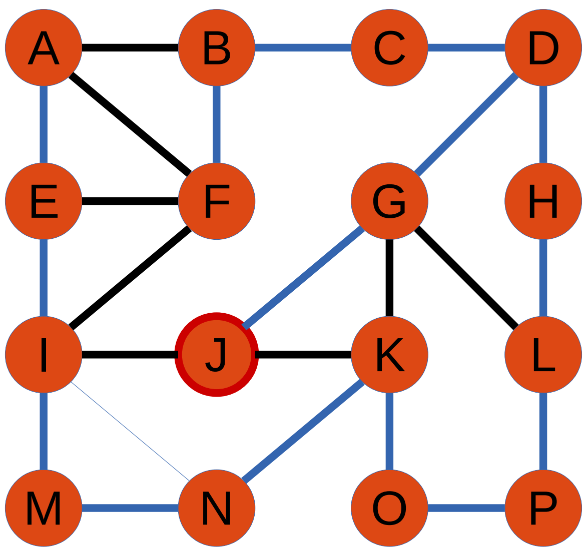
Pila

A, E, I, M, N, K, O, P, L, H, D, G, J

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	IJ	JK	GL		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



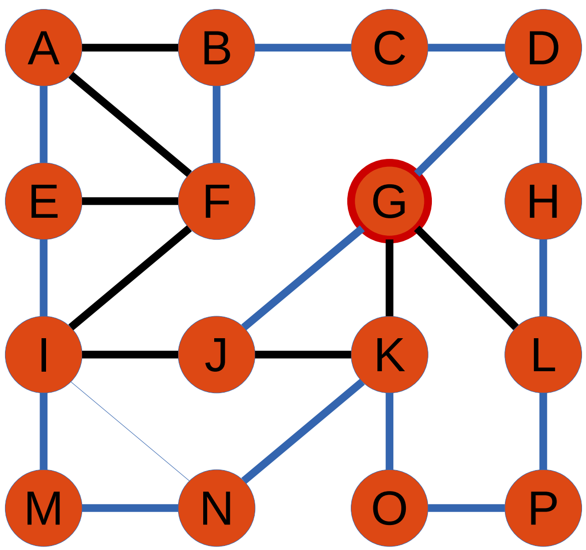
Pila

A, E, I, M, N, K, O, P, L, H, D, G, J

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	GL		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



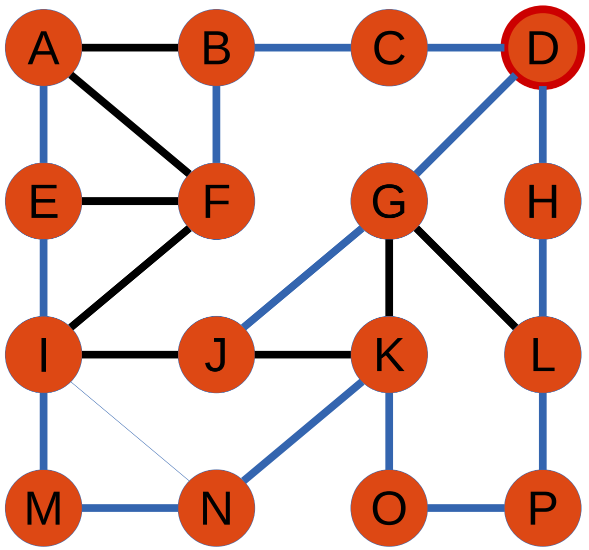
Pila

A, E, I, M, N, K, O, P, L, H, D, G

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	GL		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



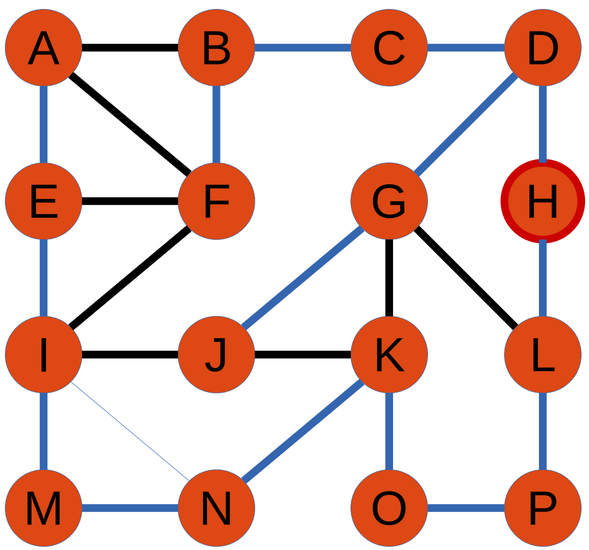
Pila

A, E, I, M, N, K, O, P, L, H, D

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



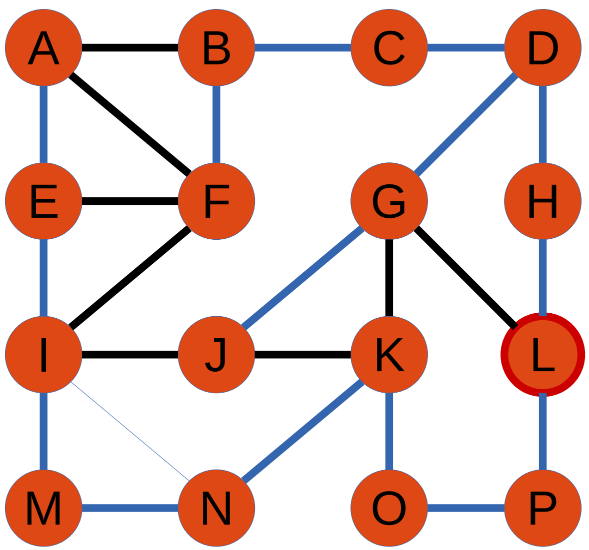
Pila

A, E, I, M, N, K, O, P, L, H

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	GL		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



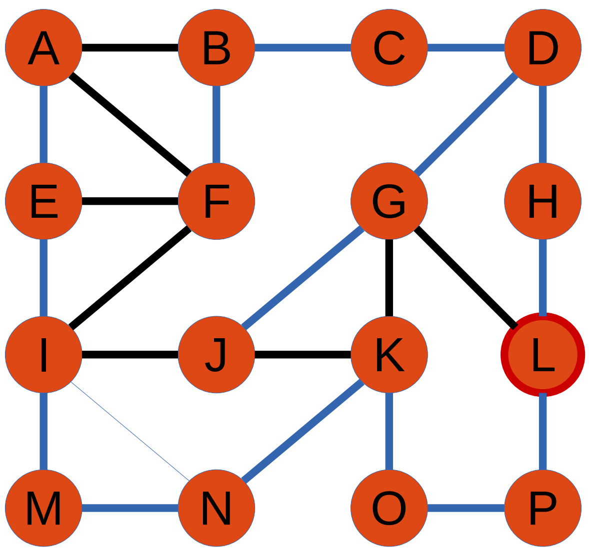
Pila

A, E, I, M, N, K, O, P, L

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	GL		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



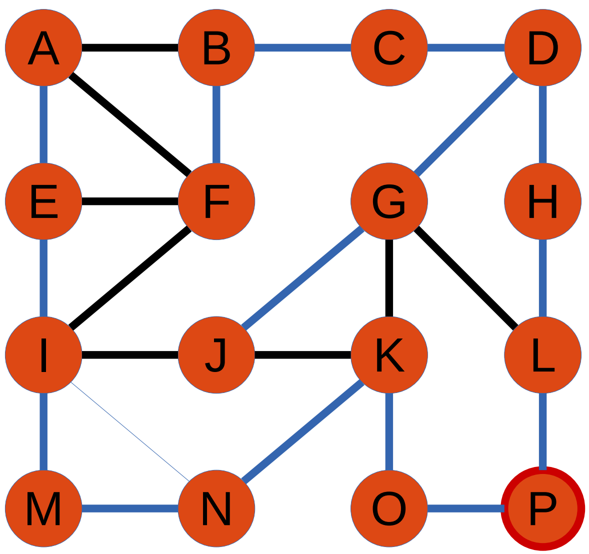
Pila

A, E, I, M, N, K, O, P, L

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	**GL**		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



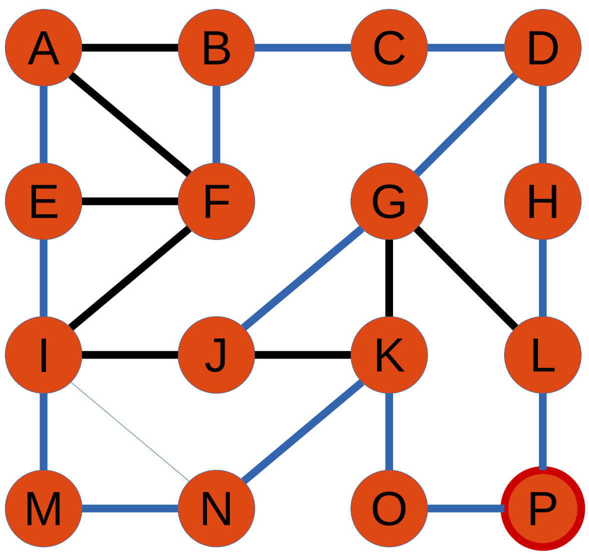
Pila

A, E, I, M, N, K, O, P

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	OP
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	**GL**		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



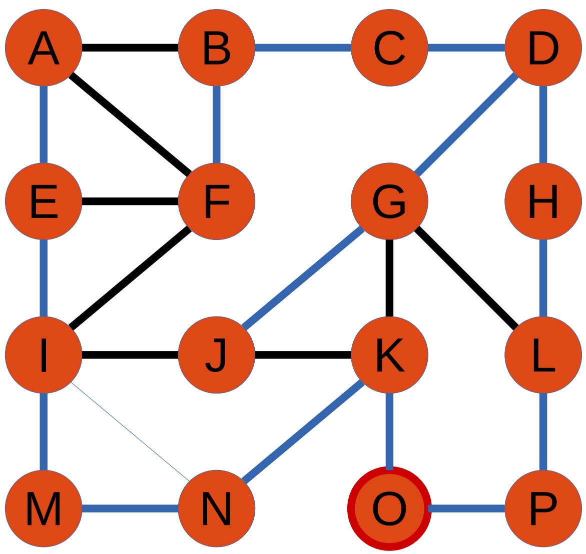
Pila

A, E, I, M, N, K, O, P

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	**GL**		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



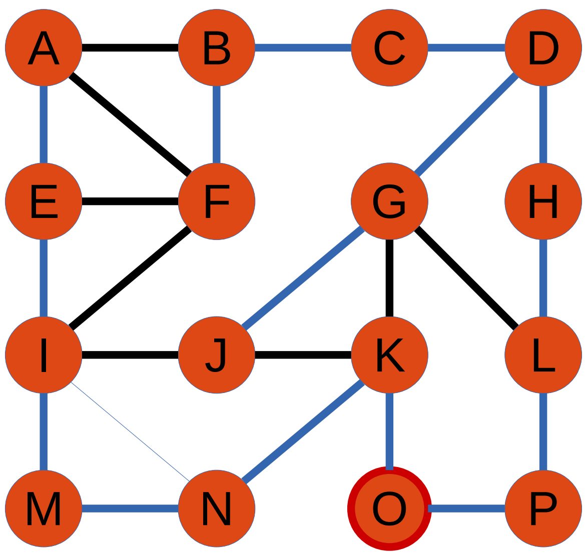
Pila

A, E, I, M, N, K, O

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	KO	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	**GL**		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



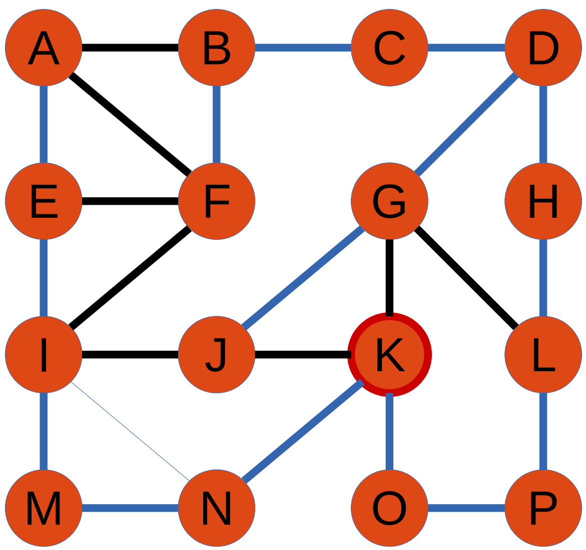
Pila

A, E, I, M, N, K, O

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	**KO**	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	**GL**		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



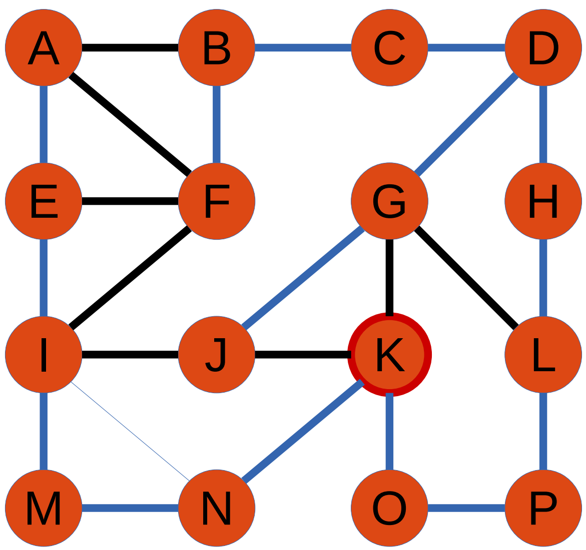
Pila

A, E, I, M, N, K

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	GK	**HL**	IM	IN	**KO**	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	JK	**GL**		MN		
					FI	**GJ**		EI		KN					

Recorrido en profundidad (DFS)



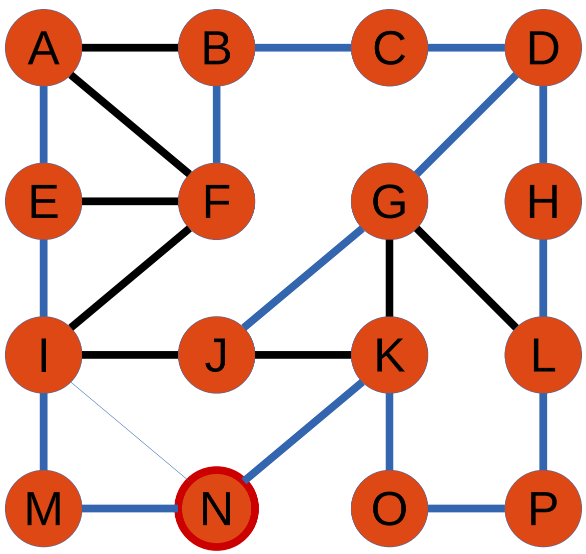
Pila

A, E, I, M, N, K

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	**GK**	**HL**	IM	IN	**KO**	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	**JK**	**GL**		MN		
					FI	**GJ**		EI		**KN**					

Recorrido en profundidad (DFS)



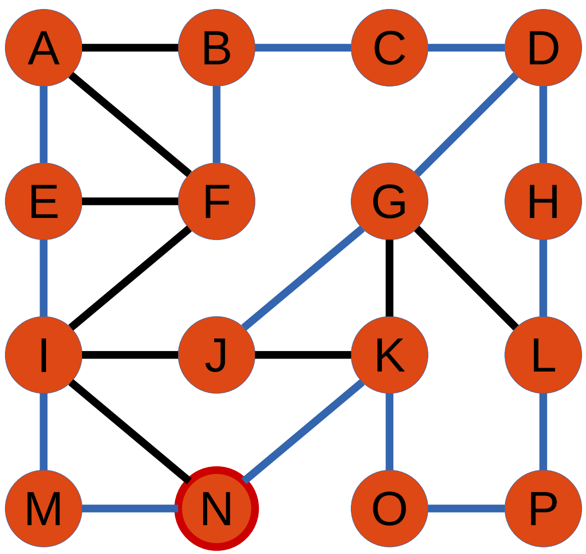
Pila

A, E, I, M, N

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	**GK**	**HL**	IM	IN	**KO**	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	**JK**	**GL**		MN		
					FI	**GJ**		EI		**KN**					

Recorrido en profundidad (DFS)



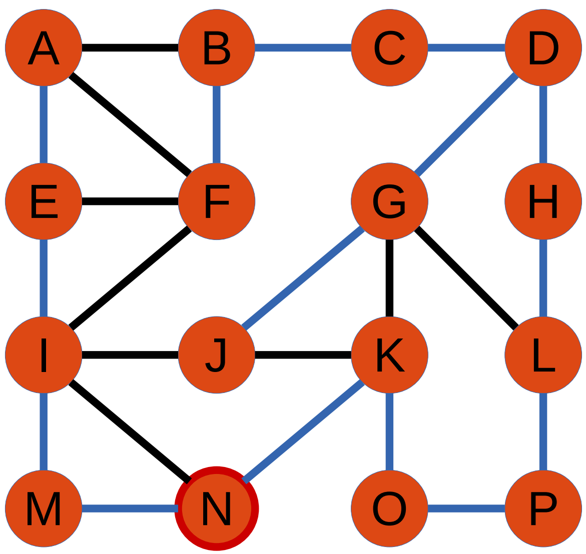
Pila

A, E, I, M, N

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	**GK**	**HL**	IM	**IN**	**KO**	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	**JK**	**GL**		MN		
					FI	**GJ**		EI		**KN**					

Recorrido en profundidad (DFS)



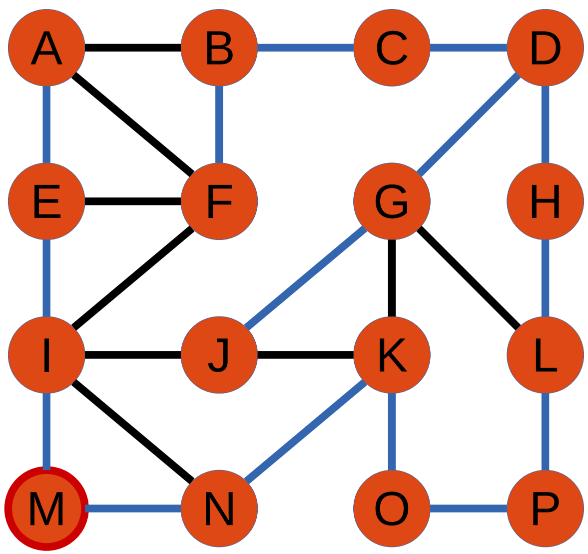
Pila

A, E, I, M, N

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



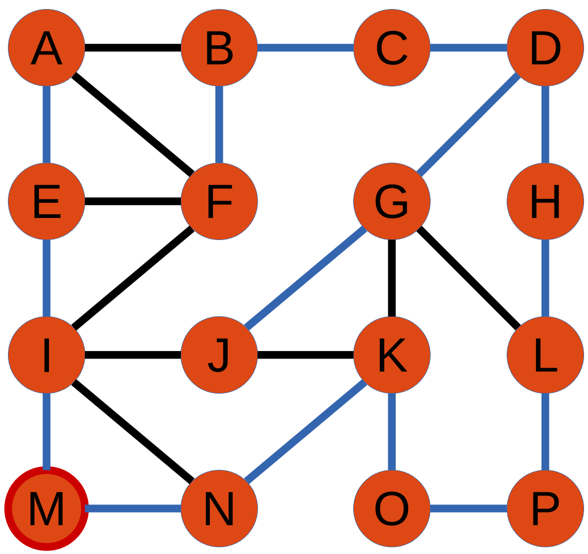
Pila

A, E, I, M

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	**GK**	**HL**	IM	**IN**	**KO**	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	**JK**	**GL**		**MN**		
					FI	**GJ**		EI		**KN**					

Recorrido en profundidad (DFS)



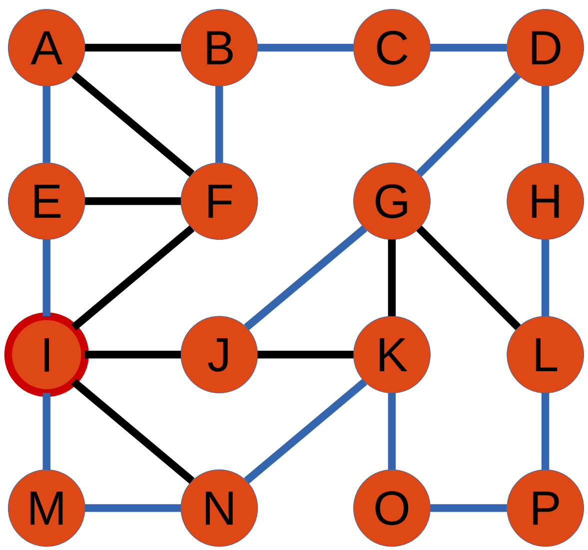
Pila

A, E, I, M

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	IN	**GJ**	**GK**	**HL**	**IM**	**IN**	**KO**	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		IJ	**IJ**	**JK**	**GL**		**MN**		
					FI	**GJ**		EI		**KN**					

Recorrido en profundidad (DFS)



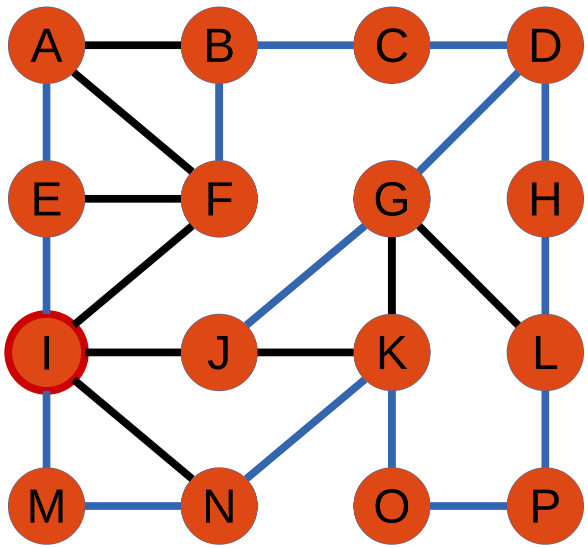
Pila

A, E, I

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



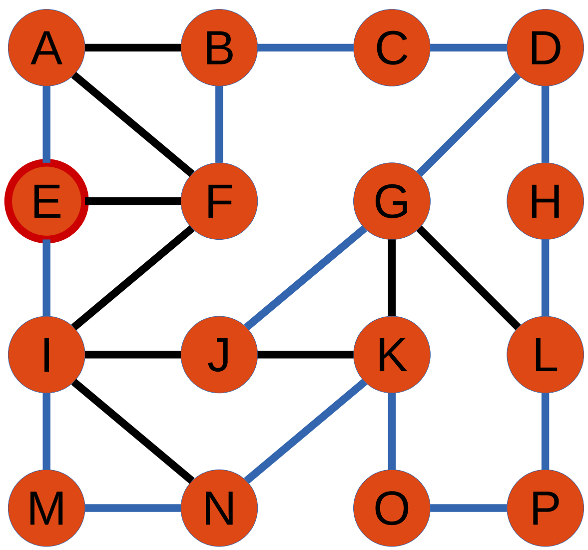
Pila

A, E, I

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	**BF**	**CD**	**DH**	**EI**	**BF**	**GK**	**HL**	**IM**	**JK**	**KO**	**LP**	**MN**	**KN**	**OP**	**LP**
AF	**BC**	**BC**	**CD**	EF	**AF**	**GL**	**DH**	**IN**	**GJ**	**GK**	**HL**	**IM**	**IN**	**KO**	**OP**
AB	**AB**		**DG**	AE	**EF**	**DG**		**IJ**	**IJ**	**JK**	**GL**		**MN**		
					FI	**GJ**		**EI**		**KN**					

Recorrido en profundidad (DFS)



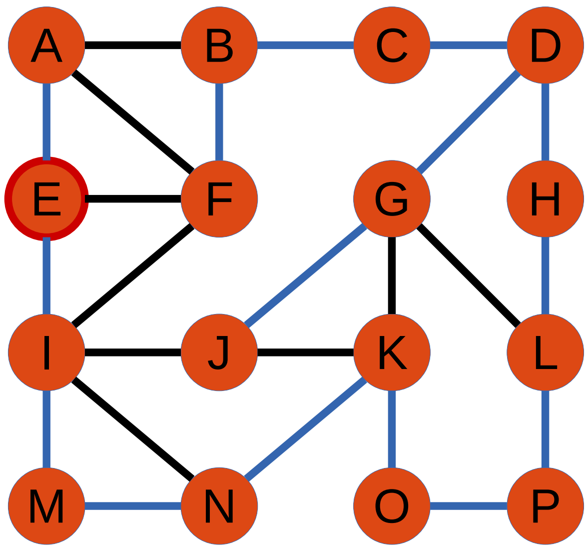
Pila

A, E

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



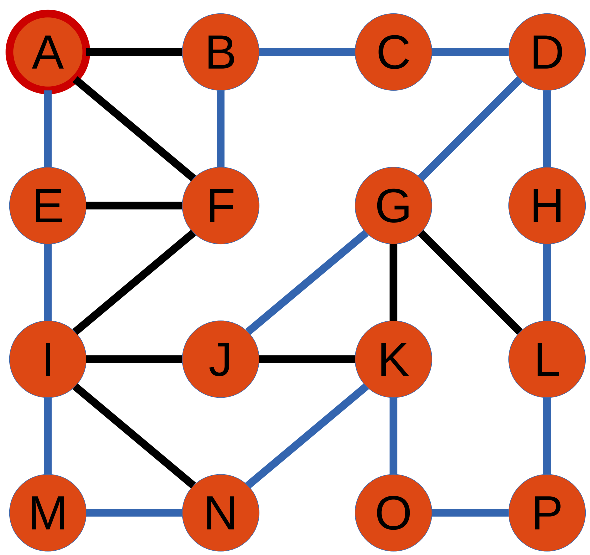
Pila

A, E

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



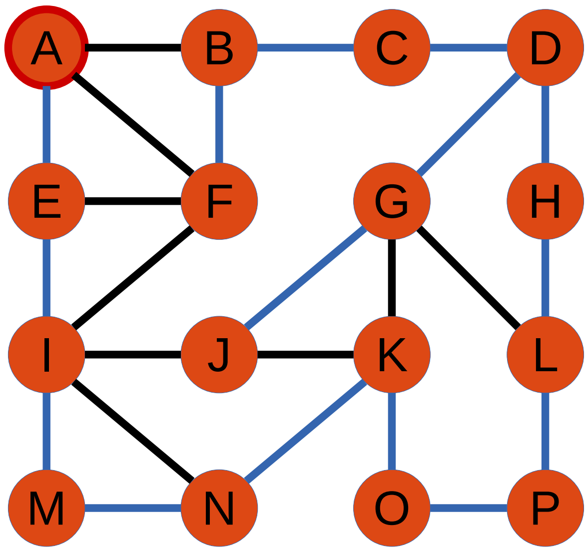
Pila

A

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)



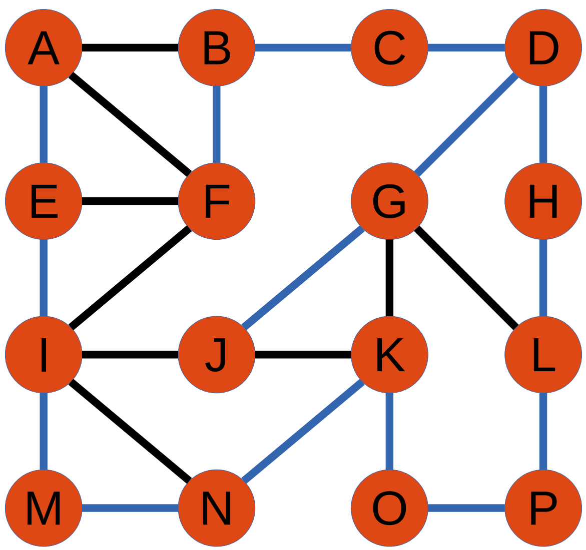
Pila

A

Lista de adyacencia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

Recorrido en profundidad (DFS)

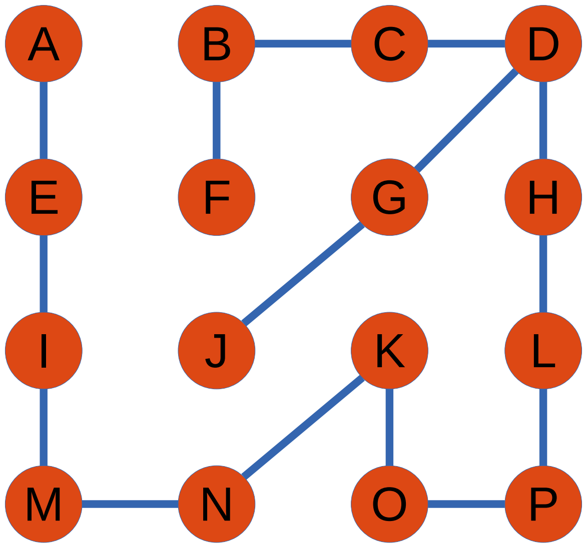


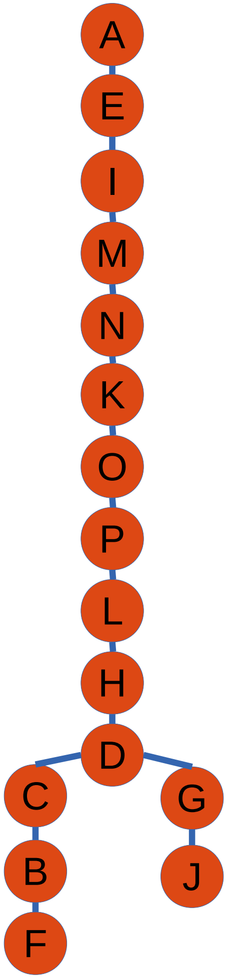
Pila

Lista de adyacencia

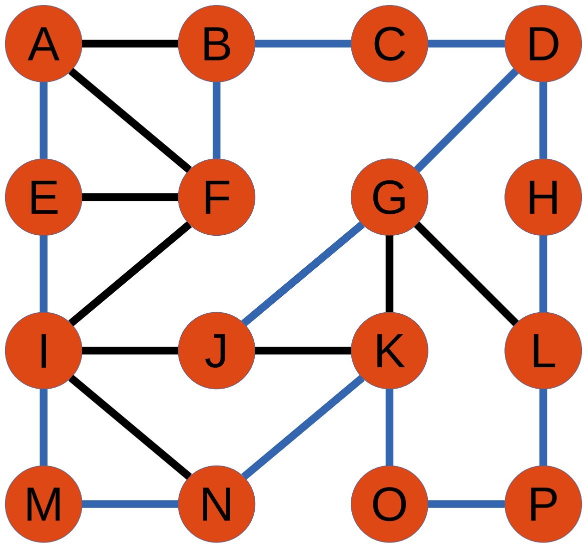
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
AE	BF	CD	DH	EI	BF	GK	HL	IM	JK	KO	LP	MN	KN	OP	LP
AF	BC	BC	CD	EF	AF	GL	DH	IN	GJ	GK	HL	IM	IN	KO	OP
AB	AB		DG	AE	EF	DG		IJ	IJ	JK	GL		MN		
					FI	GJ		EI		KN					

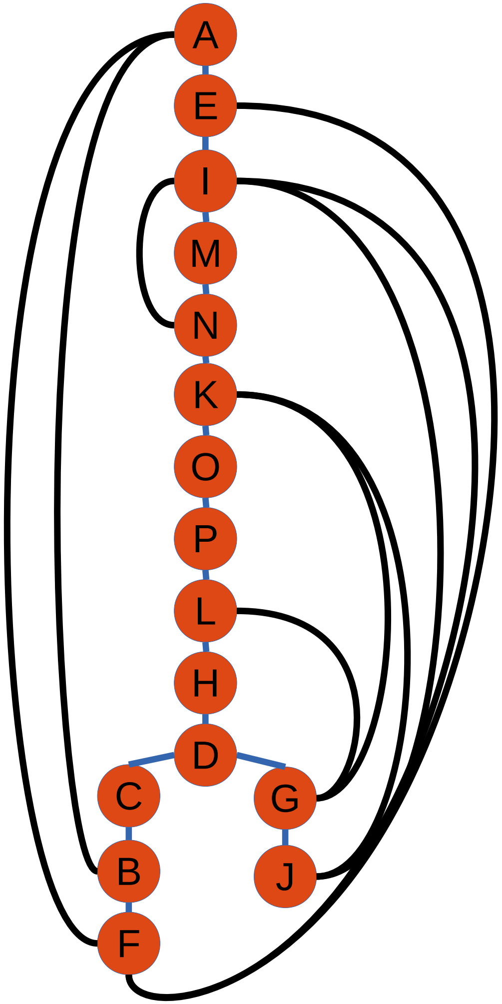
Recorrido en profundidad (DFS)





Recorrido en profundidad (DFS)





Recorrido en profundidad (DFS)

```
algorithm search(graph,node)
  node.label = explored
  stack.push(pair(node,node.incidentEdges().iterator()))
  while (!stack.isEmpty())
    currentPair = stack.peek()
    while currentPair.iterator.edge != null
      if currentPair.iterator.edge.label = null
        break
      currentPair.iterator.next()

    if (currentPair.iterator != null)
      nextNode = graph.opposite(currentPair.node,currentPair.iterator)
      if (nextNode.label = null)
        currentPair.edge.label = DiscoveryEdge
        nextNode.label = explored
        stack.push(pair(nextNode,nextNode.incidentEdges().iterator()))
      else
        currentPair.edge.label = BackEdge
    else
      stack.pop()
```

Recorrido en profundidad (DFS)

- De cada nodo se recorren todas sus aristas, por lo que la lista de aristas se recorre 2 veces. Además se recorren todos los nodos. Por ello, usando lista de adyacencia la complejidad es **O(v+a)**
- Usando matriz de adyacencia la complejidad es **$O(v^2)$** ya que la consulta "incidentEdges" tiene complejidad **O(v)** y se hace en un bucle que tiene complejidad **O(v)**
- Si la matriz es densa la complejidad es similar, pero si la matriz es dispersa el enfoque matricial es mucho peor

Recorrido en profundidad (DFS)

Usando una lista de adyacencia, DFS resuelve los siguientes problemas con complejida**d O(v+a)**:

- Encontrar un camino entre dos nodos
- Comprobar si un grafo es conexo
- Encontrar ciclos (presencia de arcos back)
- Obtener un árbol de expansión (arcos de exploración). Se puede observar que las aristas de vuelta siempre llevan a un antepasado del árbol

Recorrido en anchura

Recorrido en anchura (BFS)

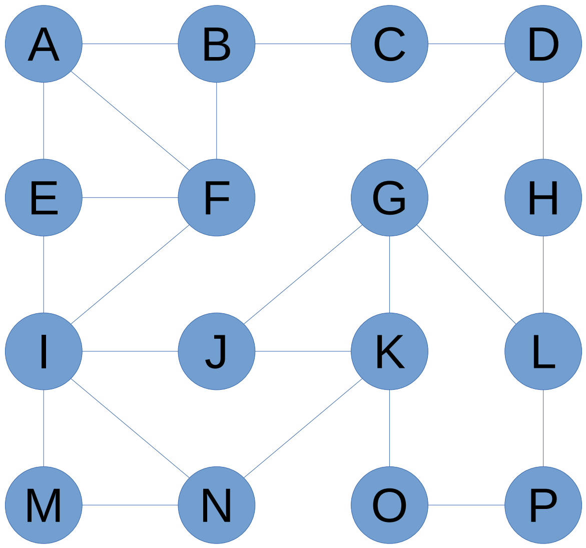
- Permite visitar todos los nodos del grafo conexo
- En anchura se recorren todos los adyacentes a un nodo, luego se recorren todos los adyacentes a éstos, y así sucesivamente
- Se obtiene un recorrido homogéneo, ya que caminos de la misma longitud se exploran en instantes de tiempo consecutivos

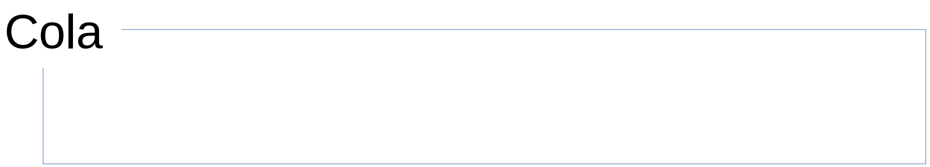
Recorrido en anchura (BFS)

1. Se toma el nodo desde el que se inicia la exploración, se etiqueta como explorado y se mete en una cola
2. Se etiquetan como explorados y se insertan en la cola los nodos adyacentes no visitados al de la cabeza de la cola.
3. Se extrae el nodo de la cabeza la cola. Si la cola queda vacía terminar y sino ir al paso 2

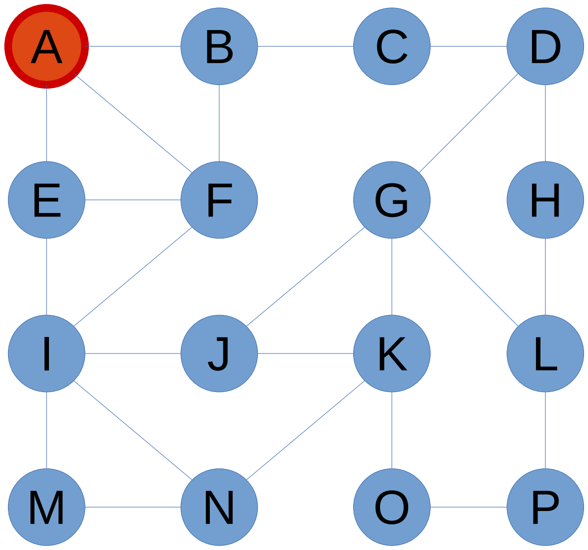
Durante el proceso, las aristas que llevan a nodos no visitados se etiquetan como aristas de exploración y las que llevan a nodos visitados se etiquetan como arista de cruce (cross)

Recorrido en anchura (BFS)





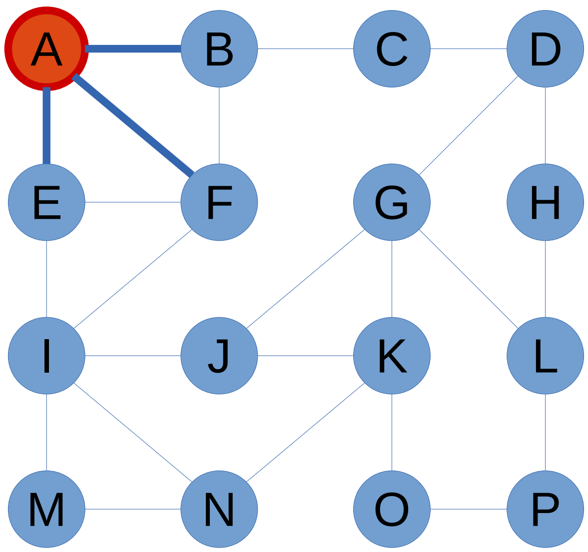
Recorrido en anchura (BFS)



Cola

A

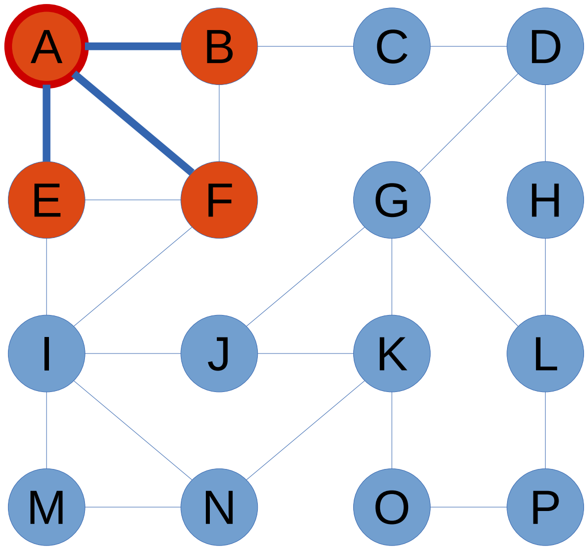
Recorrido en anchura (BFS)



Cola

A

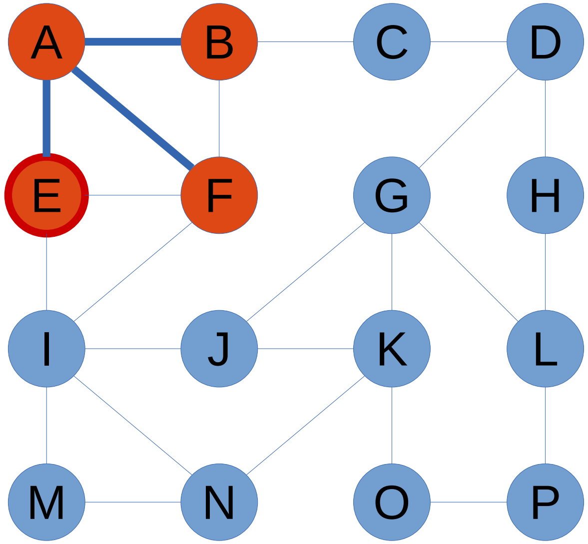
Recorrido en anchura (BFS)



Cola

A, E, F, B

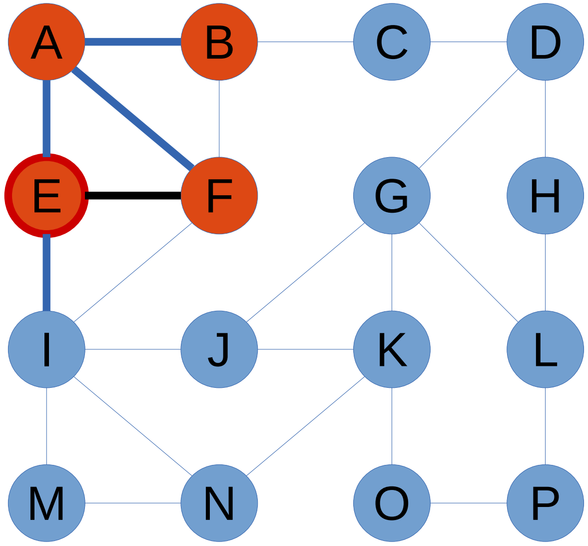
Recorrido en anchura (BFS)



Cola

E, F, B

Recorrido en anchura (BFS)



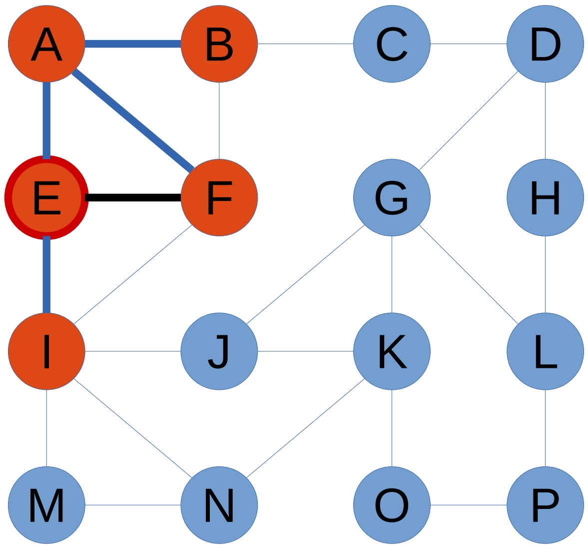
Cola

E, F, B

La aparición de una arista cross
(en negro) indica la aparición de un ciclo

Las aristas cross no suponen vuelta a
nodos ya analizados como ocurre en la
Recorrido en profundidad

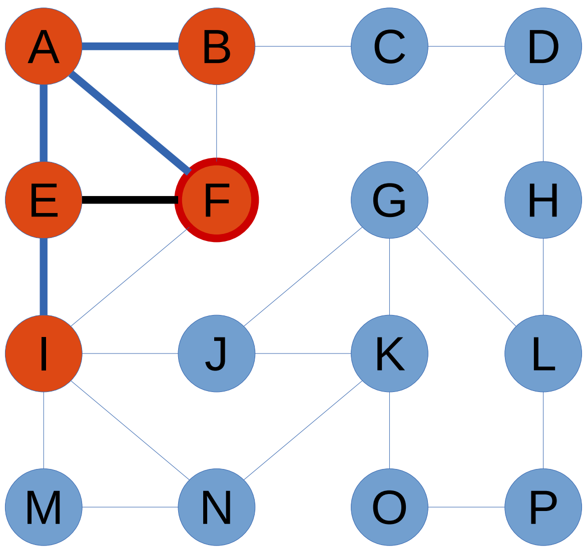
Recorrido en anchura (BFS)



Cola

E, F, B, I

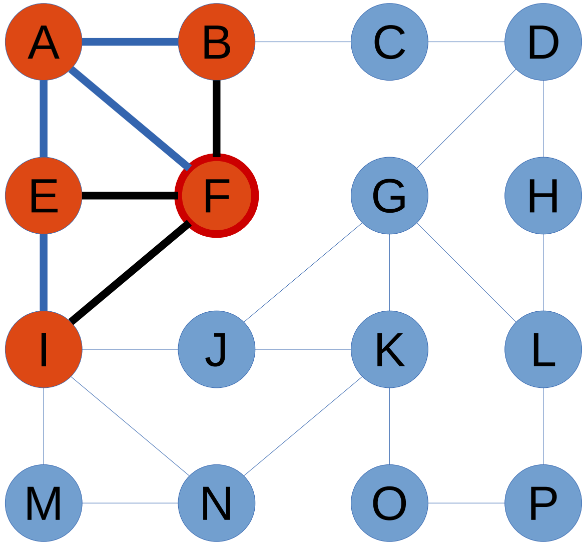
Recorrido en anchura (BFS)



Cola

F, B, I

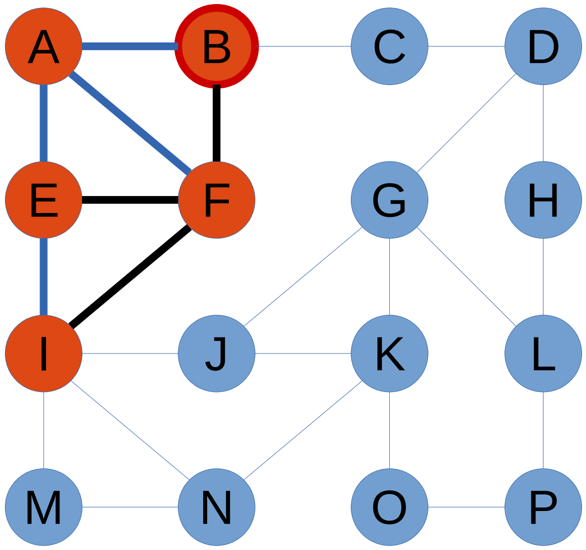
Recorrido en anchura (BFS)



Cola

F, B, I

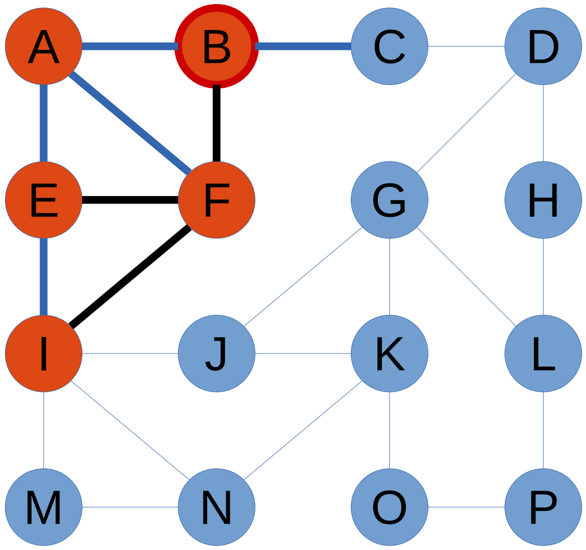
Recorrido en anchura (BFS)



Cola

B, I

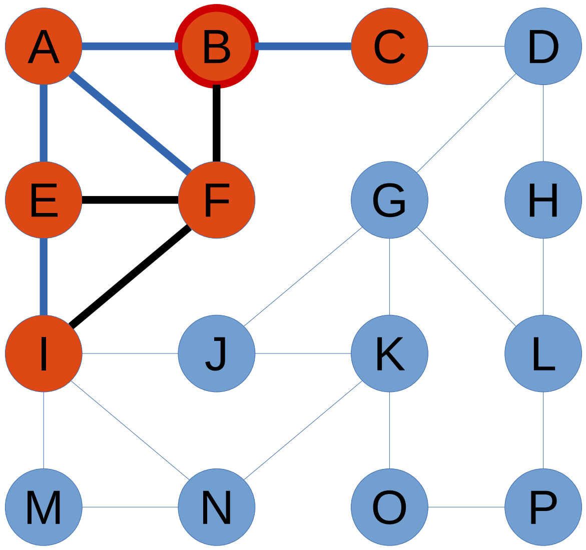
Recorrido en anchura (BFS)



Cola

B, I

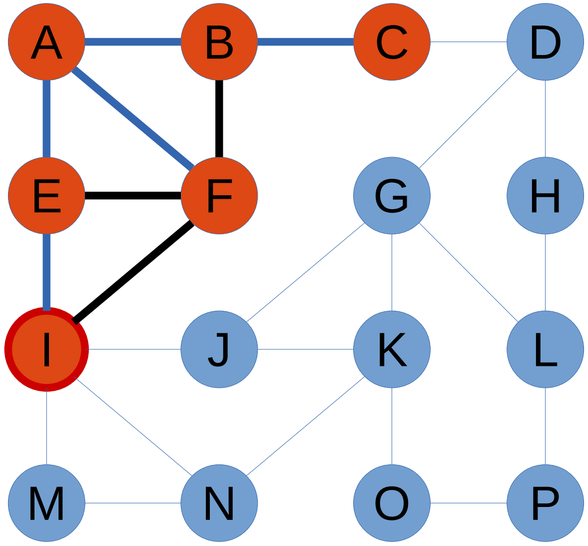
Recorrido en anchura (BFS)



Cola

B, I, C

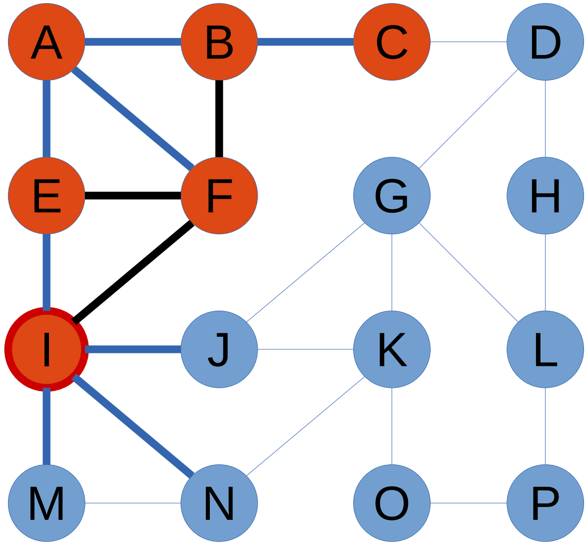
Recorrido en anchura (BFS)



Cola

I, C

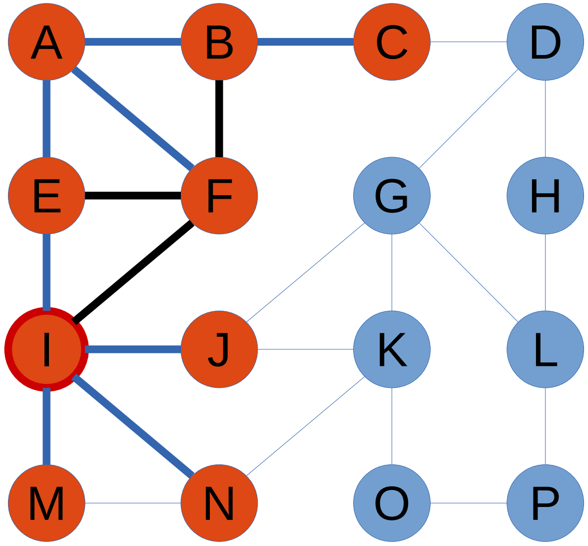
Recorrido en anchura (BFS)



Cola

I, C

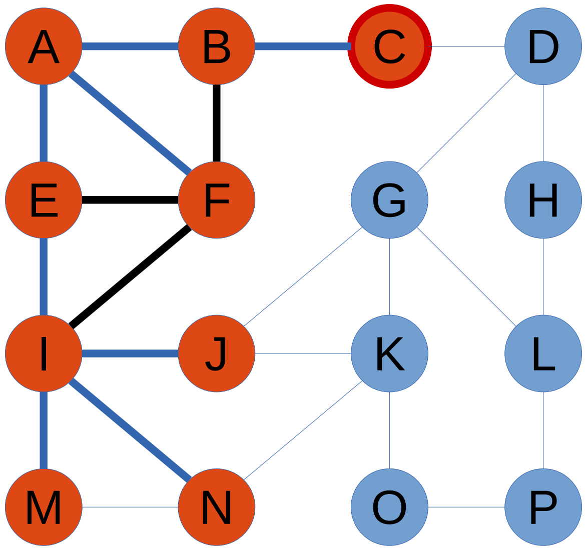
Recorrido en anchura (BFS)



Cola

I, C, M, N, J

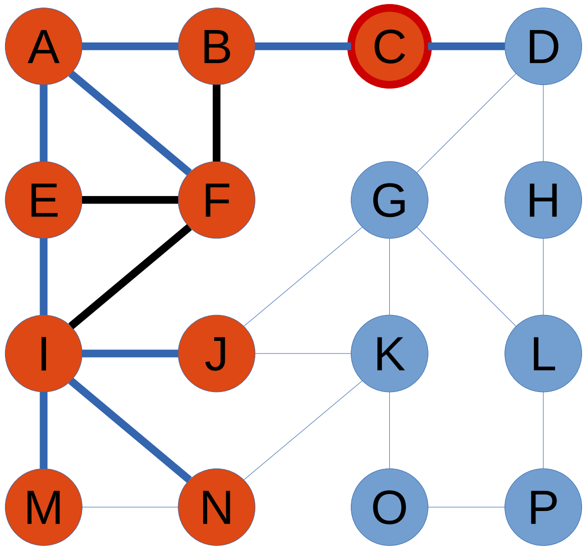
Recorrido en anchura (BFS)



Cola

C, M, N, J

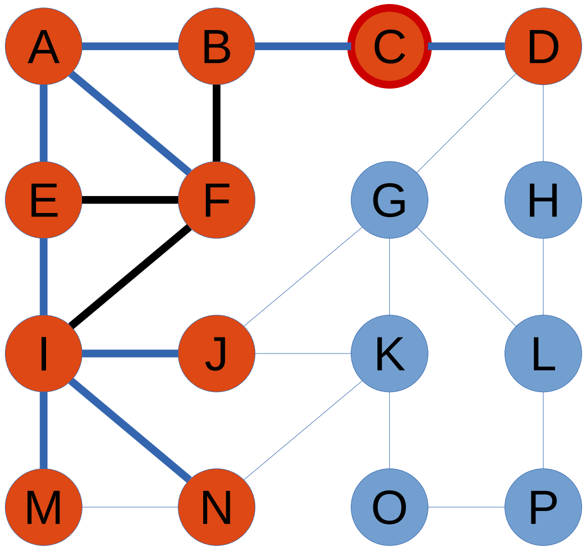
Recorrido en anchura (BFS)



Cola

C, M, N, J

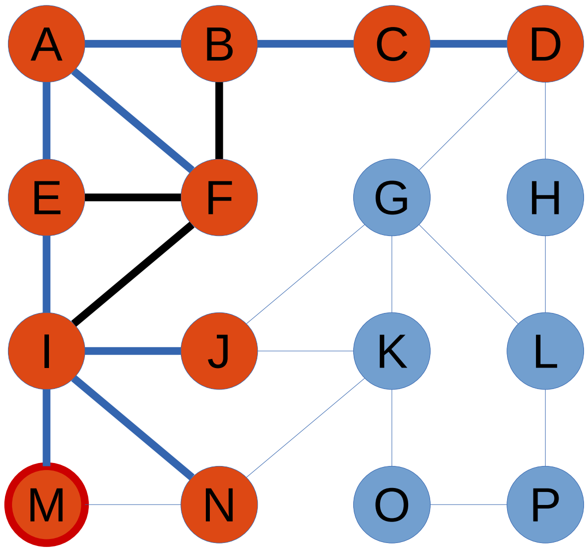
Recorrido en anchura (BFS)



Cola

C, M, N, J, D

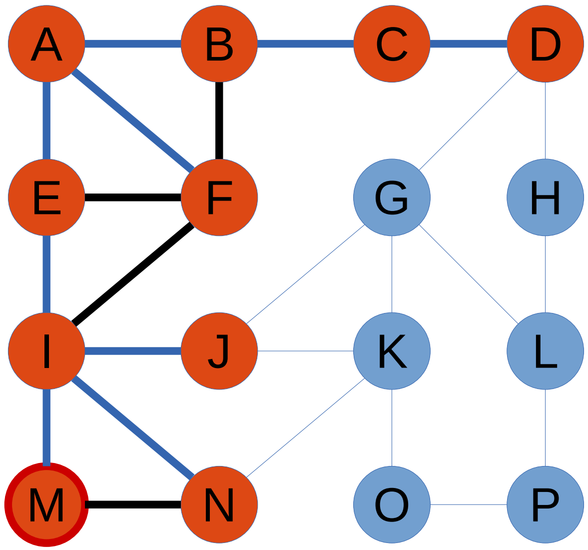
Recorrido en anchura (BFS)



Cola

M, N, J, D

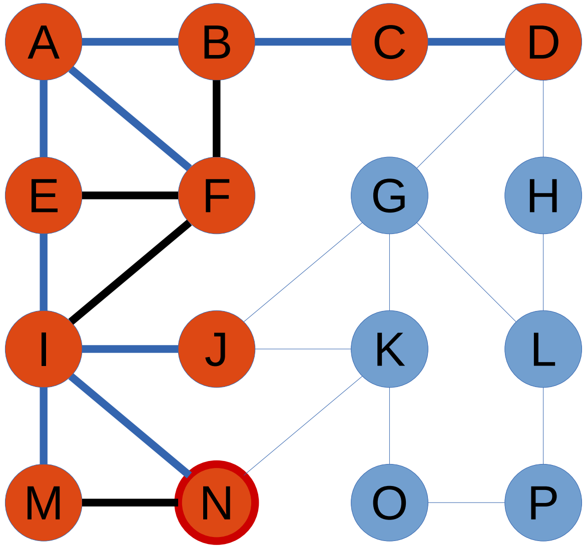
Recorrido en anchura (BFS)



Cola

M, N, J, D

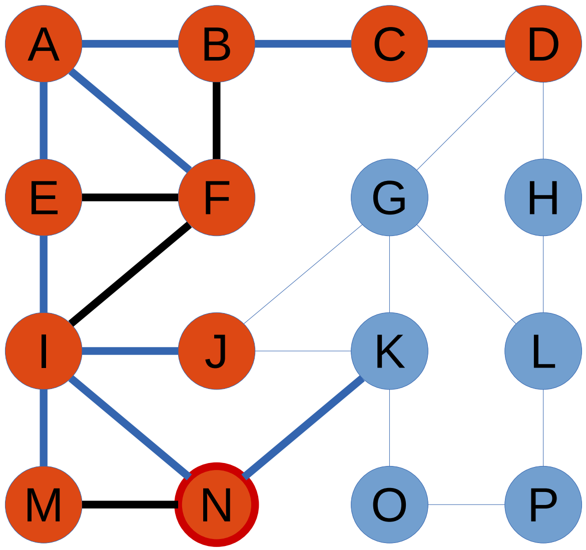
Recorrido en anchura (BFS)



Cola

N, J, D

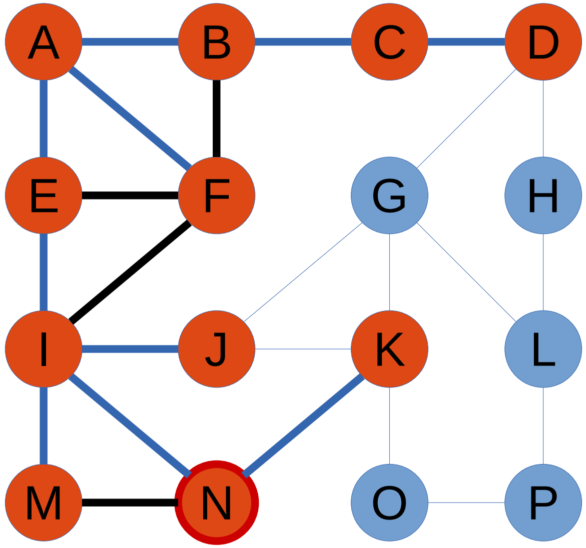
Recorrido en anchura (BFS)



Cola

N, J, D

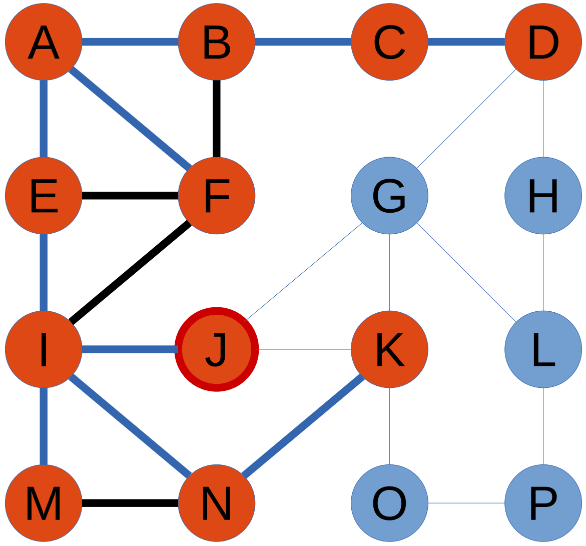
Recorrido en anchura (BFS)



Cola

N, J, D, K

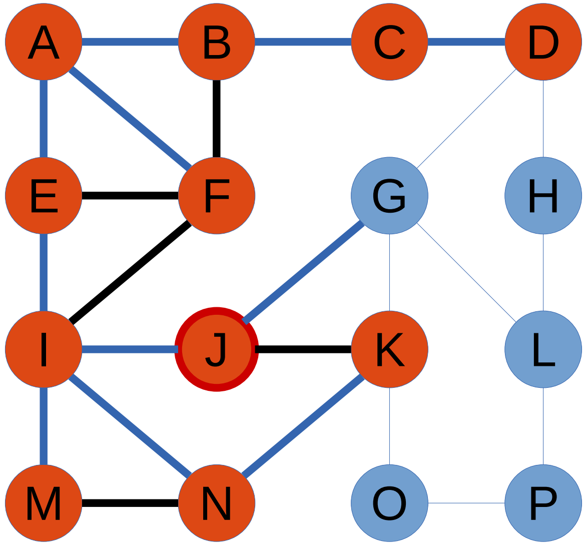
Recorrido en anchura (BFS)



Cola

J, D, K

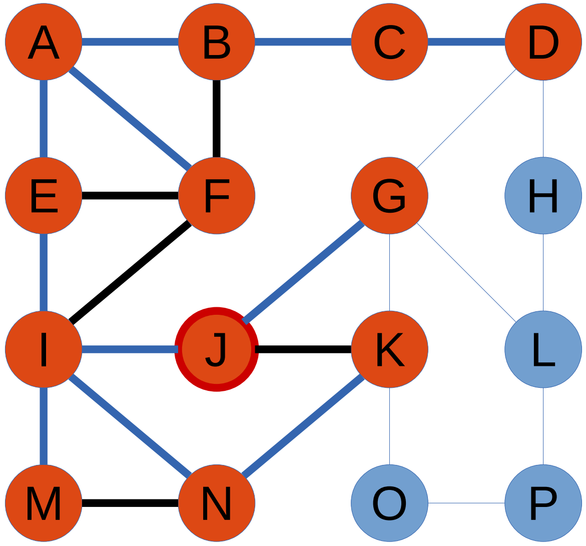
Recorrido en anchura (BFS)



Cola

J, D, K

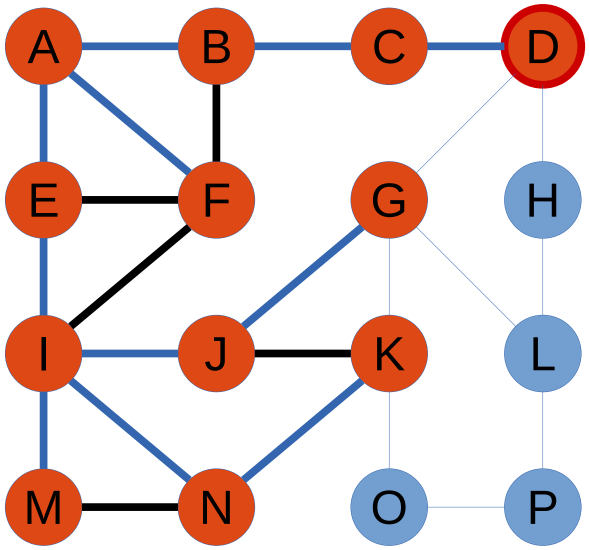
Recorrido en anchura (BFS)



Cola

J, D, K, G

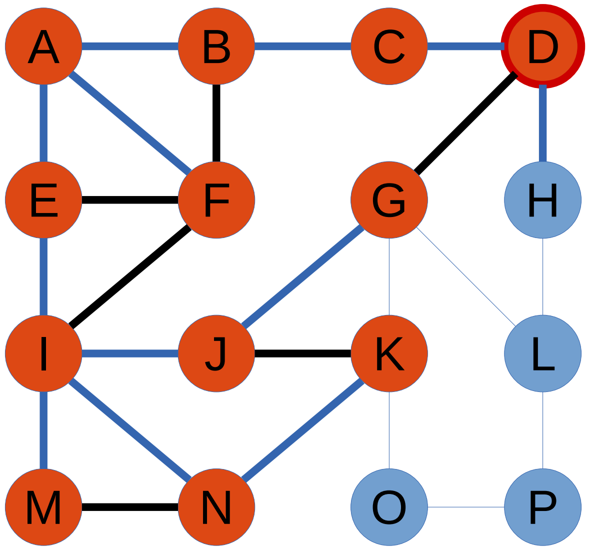
Recorrido en anchura (BFS)



Cola

D, K, G

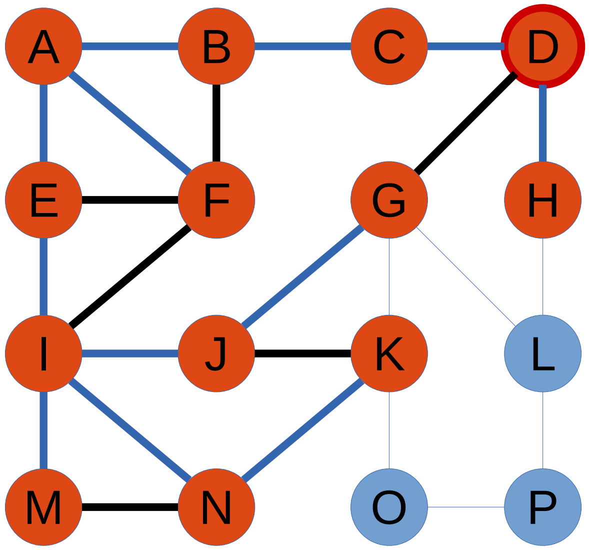
Recorrido en anchura (BFS)



Cola

D, K, G

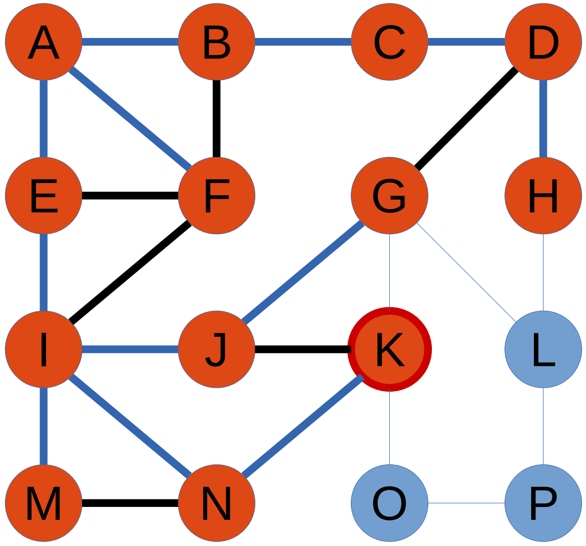
Recorrido en anchura (BFS)



Cola

D, K, G, H

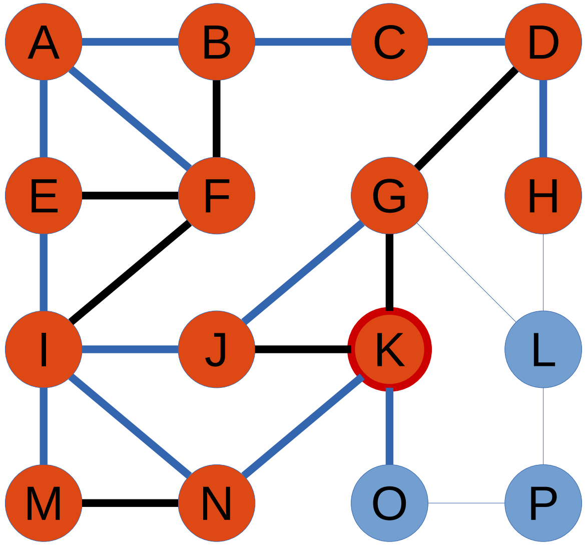
Recorrido en anchura (BFS)



Cola

K, G, H

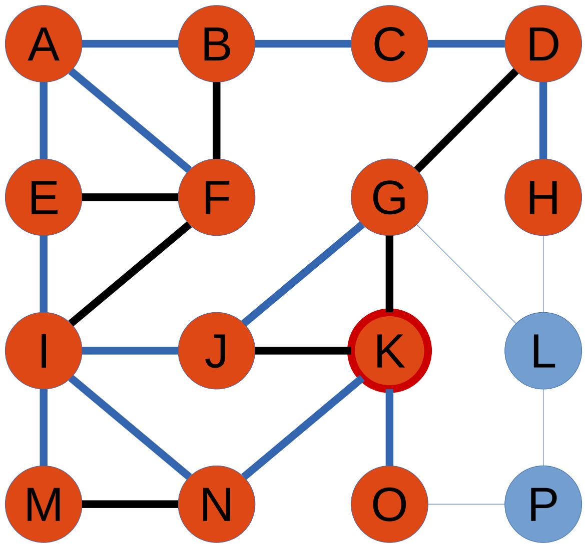
Recorrido en anchura (BFS)



Cola

K, G, H

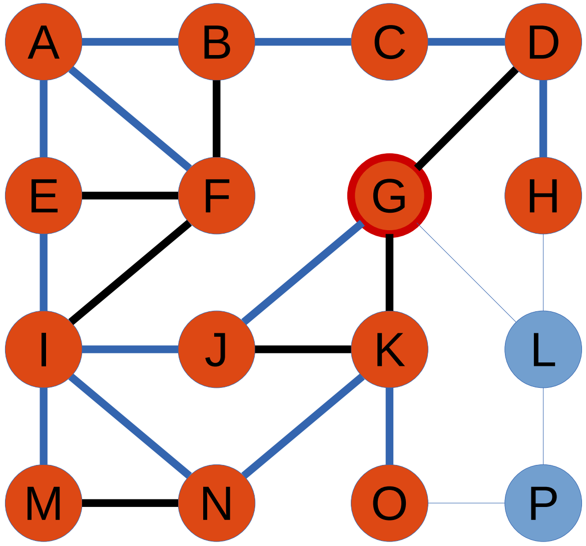
Recorrido en anchura (BFS)



Cola

K, G, H, O

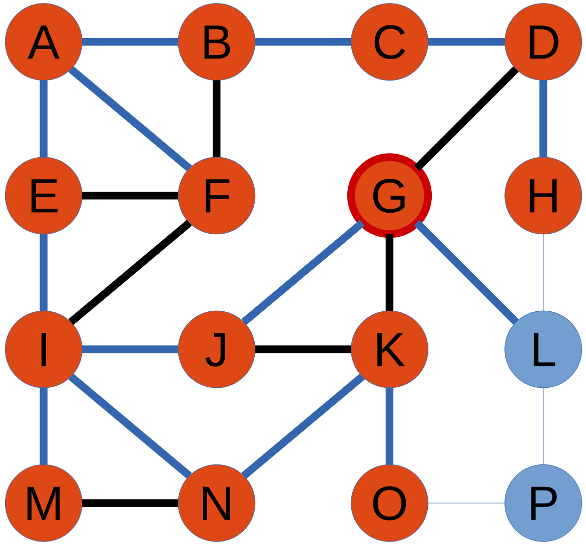
Recorrido en anchura (BFS)



Cola

G, H, O

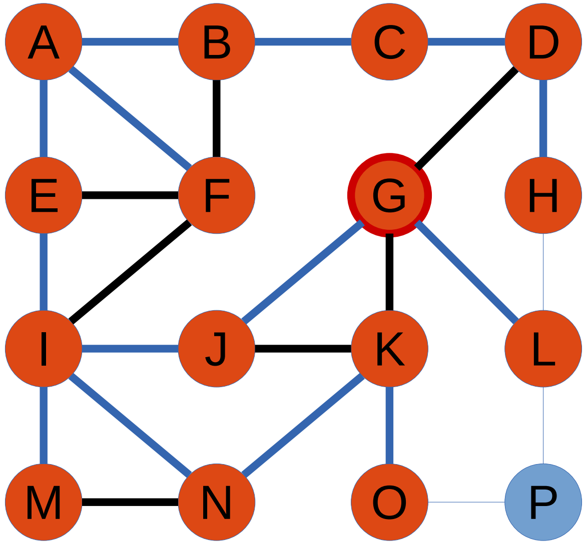
Recorrido en anchura (BFS)



Cola

G, H, O

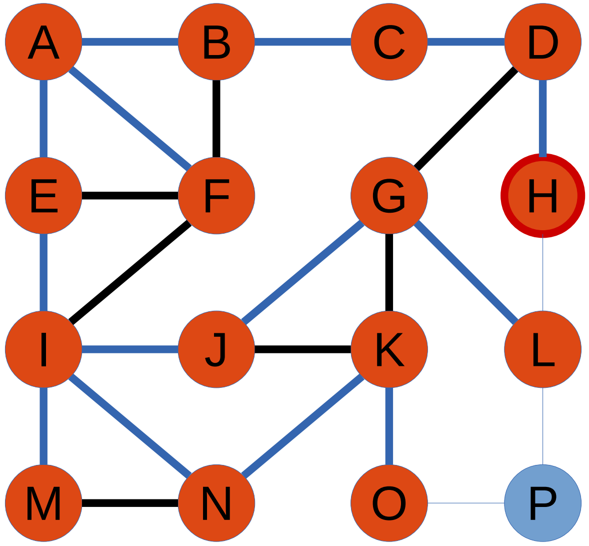
Recorrido en anchura (BFS)



Cola

G, H, O, L

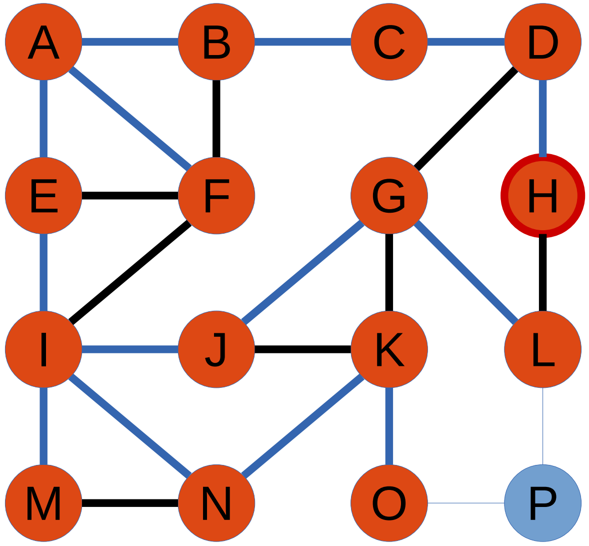
Recorrido en anchura (BFS)



Cola

H, O, L

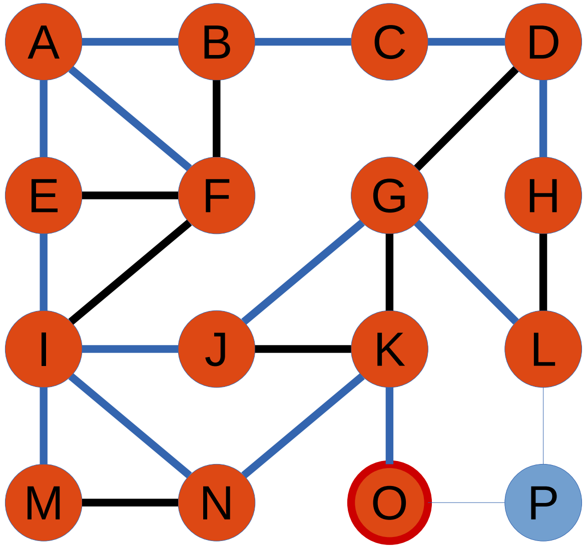
Recorrido en anchura (BFS)



Cola

H, O, L

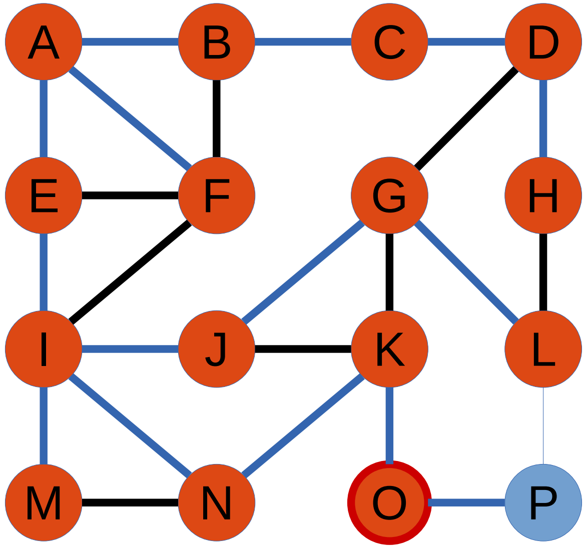
Recorrido en anchura (BFS)



Cola

O, L

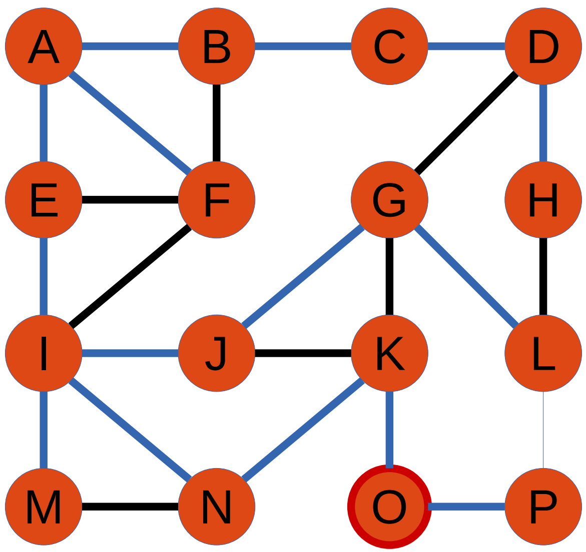
Recorrido en anchura (BFS)



Cola

O, L

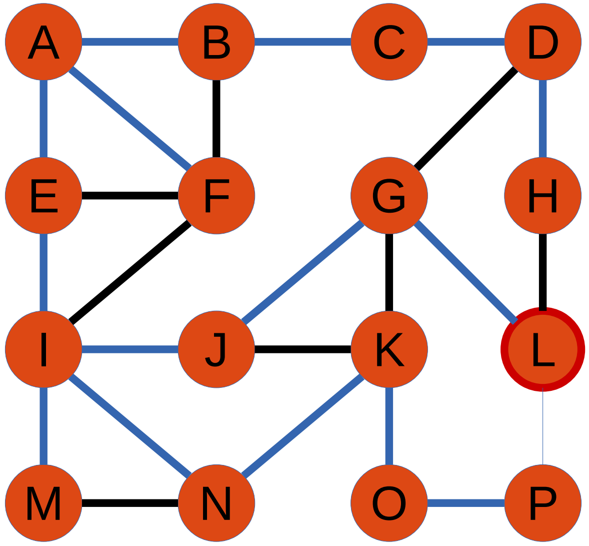
Recorrido en anchura (BFS)



Cola

O, L, P

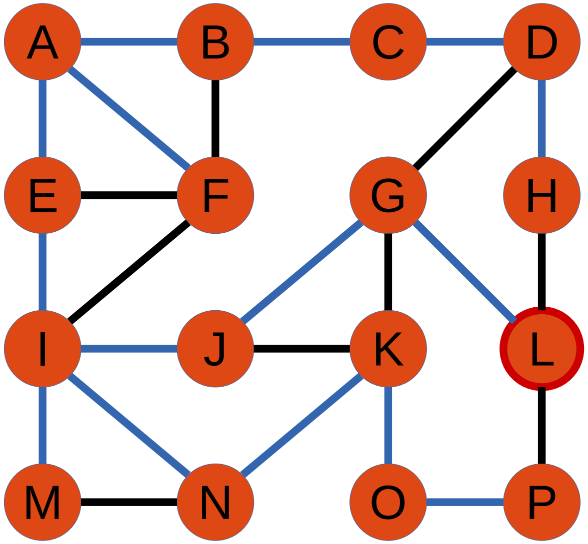
Recorrido en anchura (BFS)



Cola

L, P

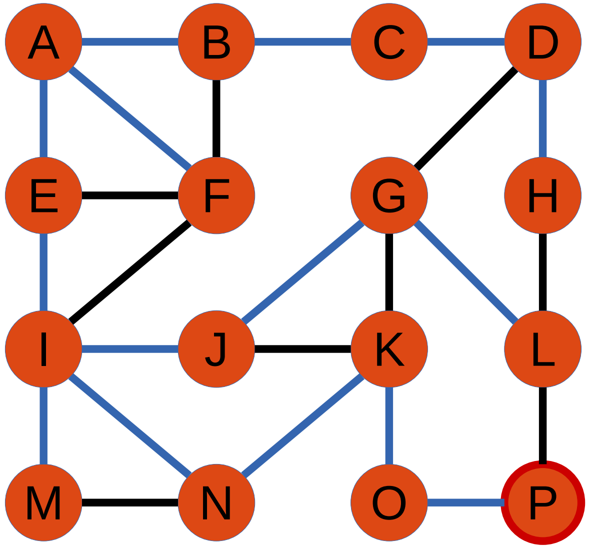
Recorrido en anchura (BFS)



Cola

L, P

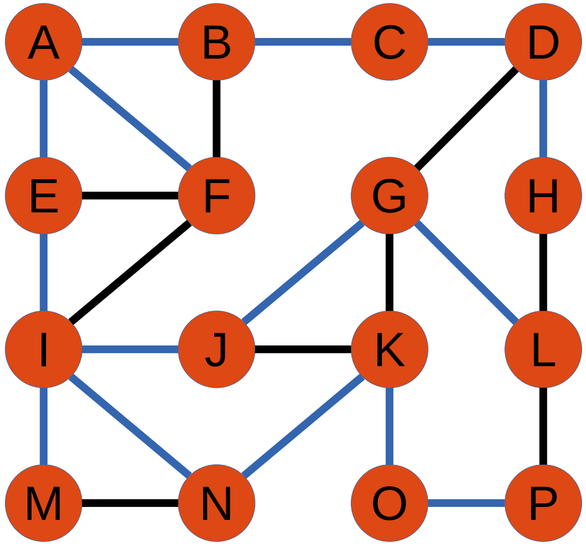
Recorrido en anchura (BFS)



Cola

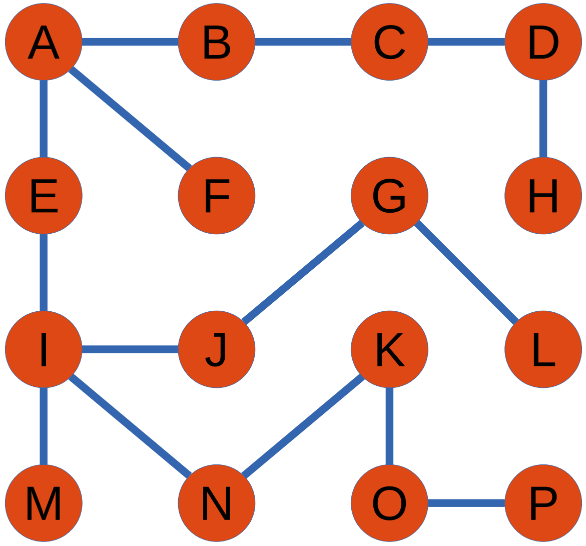
P

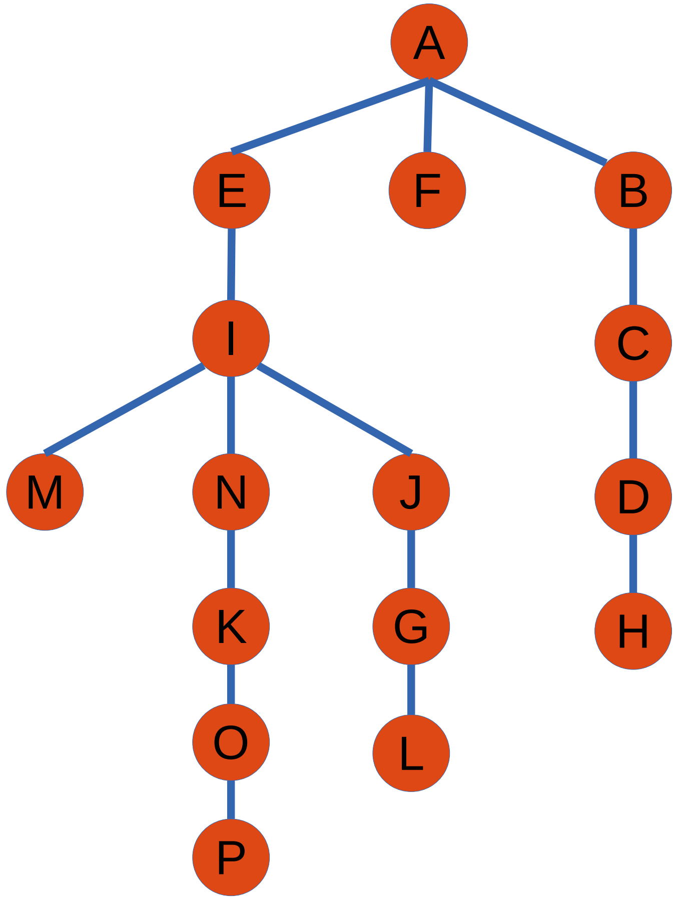
Recorrido en anchura (BFS)



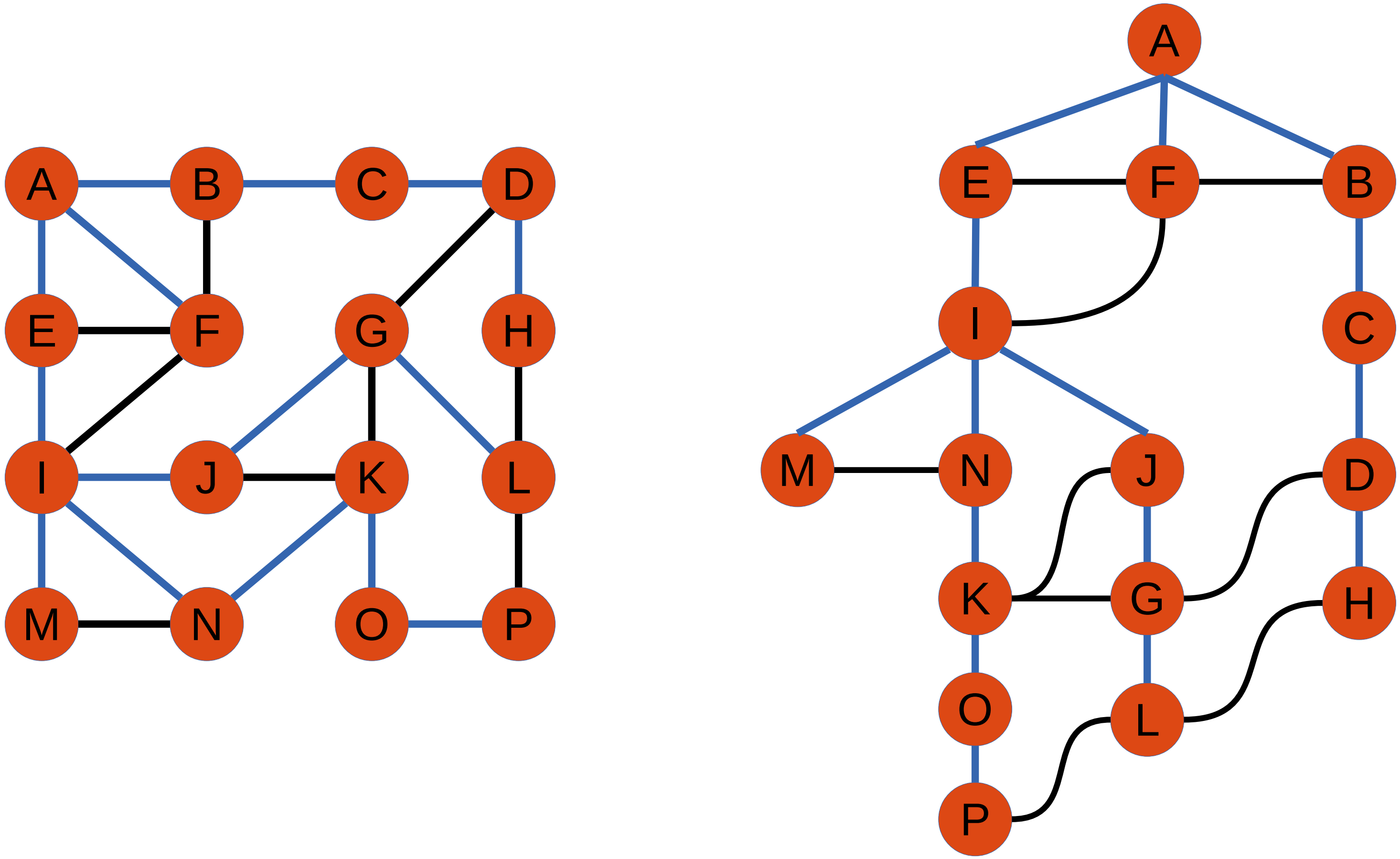
Cola

Recorrido en anchura (BFS)





Recorrido en anchura (BFS)



Recorrido en anchura (BFS)

El algoritmo en pseudocódigo:

```
algorithm search(graph,node)
  queue.pushBack(node)
  queue.front().label = explored

 while (!queue.isEmpty())

    for each edge in graph.incidentEdges(queue.front())
      if edge.label = null
        nextNode = graph.opposite(queue.front(),edge)
        if (nextNode.label == null)
          edge.label = DiscoveryEdge
          nextNode.label = explored
          queue.pushBack(nextNode)
        else
          edge.label = CrossEdge
    queue.removeFront()
```

Recorrido en anchura (BFS)

Utilizando listas de adyacencia se obtiene complejidad **O(*v*+*a*)** para la Recorrido en anchura y para los siguientes problemas:

- Encontrar, si existe, el camino más corto entre dos vértices
- Comprobar si es conexo
- Obtener un árbol de expansión. Observese que en este caso las aristas no llevan a antepasados ni a descendientes, y por eso las llamamos arcos cross
- Obtener las componentes conexas
- Encontrar ciclos

Camino más corto

Camino más corto

El Recorrido en anchura se puede utilizar para localizar el camino más corto desde un vértice al resto de vértices.

Este enfoque es útil cuando todos los arcos son indiferentes, pero no es útil cuando se usa sobre grafos ponderados donde unos arcos tienen mayor peso que otros.

En este caso el problema se redefine como el de encontrar el camino de menor coste.

Otro problema relacionado es el problema del viajante (que además busca visitar solo una vez determinados nodos).

Camino más corto

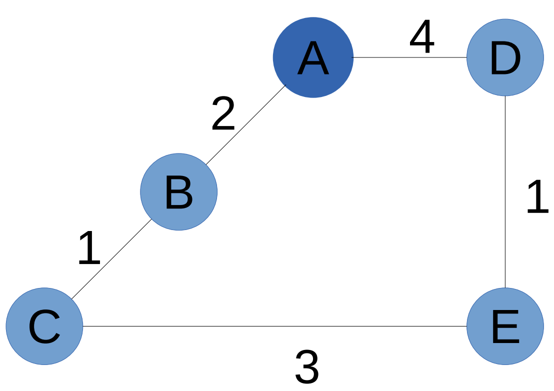
Dijkstra propuso un algoritmo de programación dinámica para encontrar el camino de menor coste entre un nodo origen ***v*** y el resto de nodos de un grafo con pesos.

Bellman-Ford propuso otro algoritmo para encontrar el camino de menor coste entre un nodo y el resto de nodos.

El algoritmo de Dijkstra crea una nube de vértices en torno a ***v***. La nube crece iterativamente. En cada iteración entran en la nube los vértices más cercanos a ella.

Camino más corto (Dijkstra)

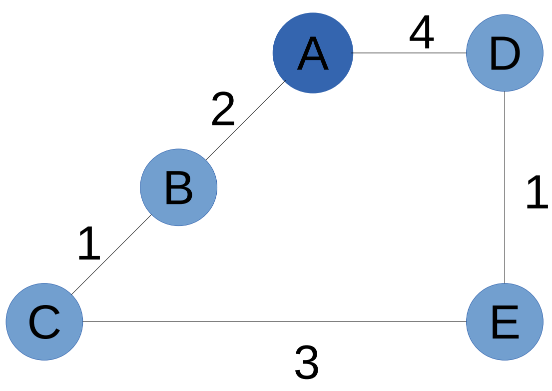
A continuación se presenta un ejemplo de aplicación del algoritmo de Dijksjtra para encontrar el camino más corto desde el nodo ***A*** al resto de nodos del grafo de la figura.



Camino más corto (Dijkstra)

Es preciso almacenar cierta información asociada a cada nodo:

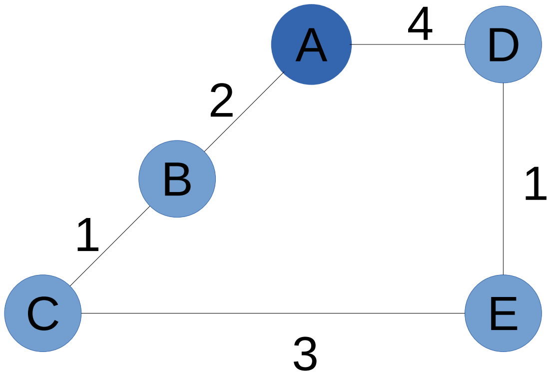
- Distancia al nodo ***A*** (inicialmente infinito)
- Si ha sido visitado (inicialmente a falso)
- Cuál es el nodo previo en el recorrido (inicialmente indefinido)

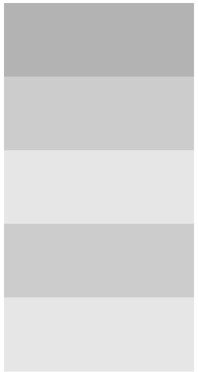


Nodo	Distancia	Visitado	Previo
A	∞	False	
B	∞	False	
C	∞	False	
D	∞	False	
E	∞	False	

Camino más corto (Dijkstra)

También es preciso una cola de prioridad para los vértices. Los vértices se ordenarán en esta cola utilizando su distancia al nodo ***A***.





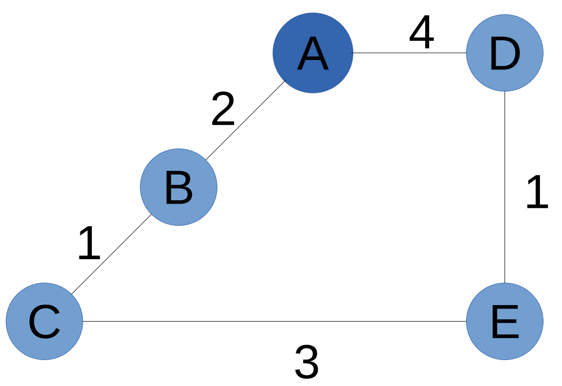
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	∞	False	
B	∞	False	
C	∞	False	
D	∞	False	
E	∞	False	

Camino más corto (Dijkstra)

El algoritmo se inicializa insertando el nodo inicial ***A*** en la cola de prioridad y apuntando que su distancia al nodo inicial es cero.

El algoritmo termina cuando la cola de prioridad queda vacía.



A	0

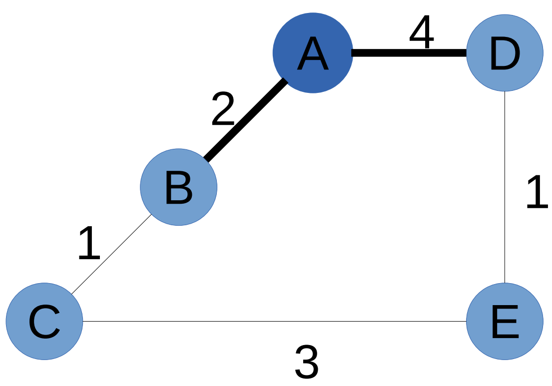
Cola de prioridad

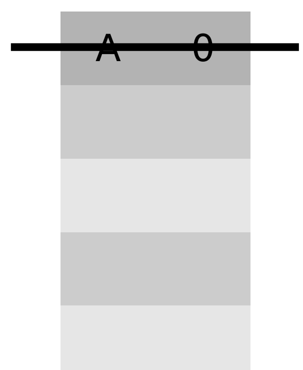
Nodo	Distancia	Visitado	Previo
A	0	False	
B	∞	False	
C	∞	False	
D	∞	False	
E	∞	False	

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice A la cabeza de la cola, se marca como visitado y se miran sus vecinos no visitados: B y D.





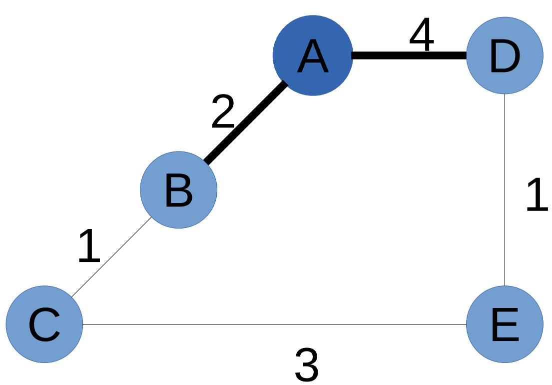
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	∞	False	
C	∞	False	
D	∞	False	
E	∞	False	

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice A la cabeza de la cola, se marca como visitado y se miran sus vecinos no visitados: B y D.
  - Como la distancia hasta A (0) más la de A a B (2) es 2 y 2 < ∞ se inserta B en la cola y se actualiza su distancia con el valor acumulado (2).



~~A~~	~~0~~
B	2

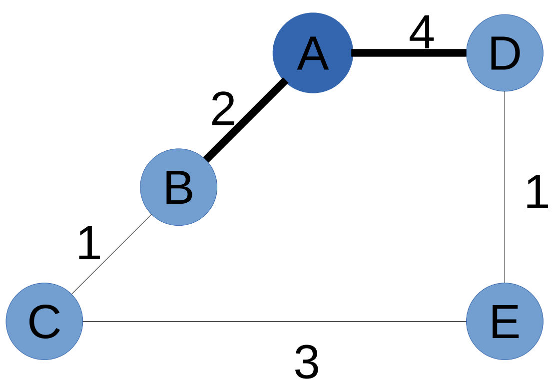
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	False	A
C	∞	False	
D	∞	False	
E	∞	False	

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice A la cabeza de la cola, se marca como visitado y se miran sus vecinos no visitados: B y D.
  - Como la distancia hasta A (0) más la de A a B (2) es 2 y 2 < ∞ se inserta B en la cola y se actualiza su distancia con el valor acumulado (2).
  - Como la distancia hasta A (0) más la de A a D (4) es 4 y 4 < ∞ se inserta D en la cola y se actualiza su distancia con el valor acumulado (4).



~~A~~	~~0~~
B	2
D	4

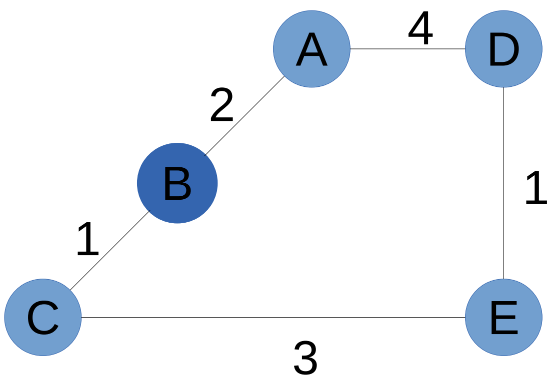
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	False	A
C	∞	False	
D	4	False	A
E	∞	False	

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice B la cabeza de la cola, se marca como visitado y se mira su único vecino no visitado C



~~A~~	~~0~~
~~B~~	~~2~~
D	4

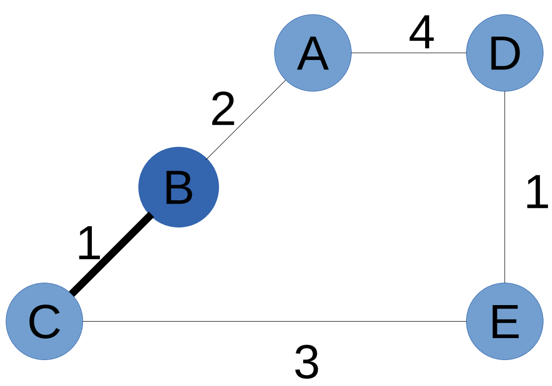
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	True	A
C	∞	False	
D	4	False	A
E	∞	False	

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice B la cabeza de la cola, se marca como visitado y se mira su único vecino no visitado C
- Como la distancia hasta B (2) más la de B a C (1) es 3 y 3 < ∞ se inserta C en la cola y se actualiza su distancia con el valor acumulado (3)



~~A~~	~~0~~
~~B~~	~~2~~
D	4
C	3

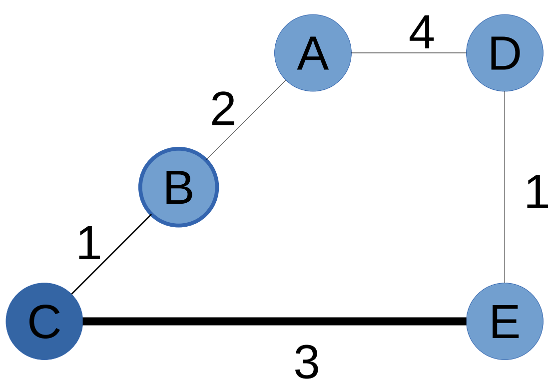
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	True	A
C	3	False	B
D	4	False	A
E	∞	False	

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice C la cabeza de la cola, se marca como visitado y se mira su único vecino no visitado E



~~A~~	~~0~~
~~B~~	~~2~~
D	4
~~C~~	~~3~~
E	6

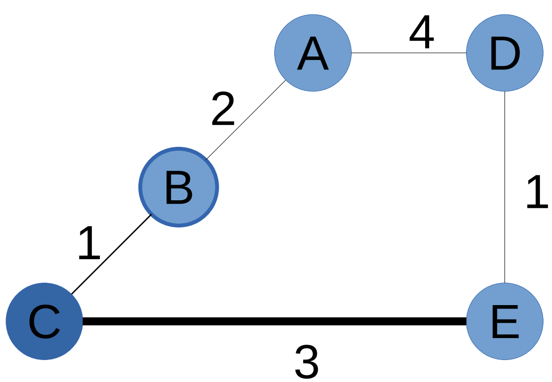
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	True	A
C	3	True	B
D	4	False	A
E	∞	False	

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice C la cabeza de la cola, se marca como visitado y se mira su único vecino no visitado E
- Como la distancia hasta C (3) más la de C a E (3) es 6 y 6 < ∞ se inserta E en la cola y se actualiza su distancia con el valor acumulado (6)



~~A~~	~~0~~
~~B~~	~~2~~
D	4
~~C~~	~~3~~
E	6

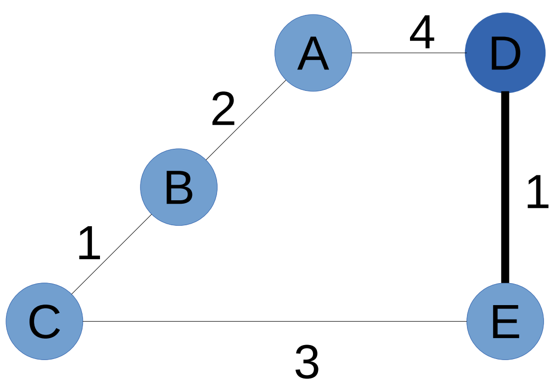
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	True	A
C	3	True	B
D	4	False	A
E	6	False	C

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice D la cabeza de la cola, se marca como visitado y se mira su único vecino no visitado E



~~A~~	~~0~~
~~B~~	~~2~~
~~D~~	~~4~~
~~C~~	~~3~~
E	6

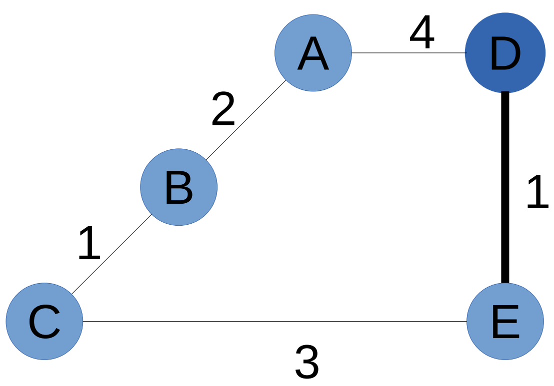
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	True	A
C	3	True	B
D	4	True	A
E	6	False	C

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Se saca el vértice D la cabeza de la cola, se marca como visitado y se mira su único vecino no visitado E
- Como la distancia hasta D (4) más la de D a E (1) es 5 y 5 < 6 se actualiza su distancia con el valor acumulado (5) y el previo es D.



~~A~~	~~0~~
~~B~~	~~2~~
~~D~~	~~4~~
~~C~~	~~3~~
E	5

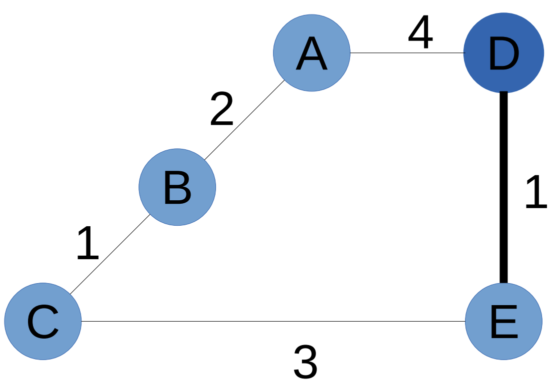
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	True	A
C	3	True	B
D	4	True	A
E	5	False	D

Camino más corto (Dijkstra)

Como la cola no está vacía:

- Como solo queda E y ya no quedan nodos por visitar, lo marcamos como visitado y hemos terminado.



~~A 0~~

~~B 2~~

~~D 4~~

~~C 3~~

~~E 5~~

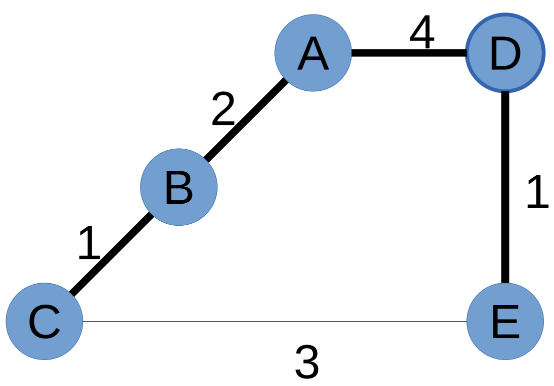
Cola de prioridad

Nodo	Distancia	Visitado	Previo
A	0	True	
B	2	True	A
C	3	True	B
D	4	True	A
E	5	True	D

Camino más corto (Dijkstra)

Ya tenemos los caminos mínimos desde A hasta todos los nodos del grafo. Para obtener la secuencia de nodos se recorren los caminos al revés hasta alcanzar el nodo A.

- Desde E, visitamos su nodo previo D
- Desde D, visitamos su nodo previo A



Camino más corto (Dijkstra)

```
function Dijkstra(graph, source)
  for each vertex v in Graph // Initializations
    dist[v]     = infinity   // Mark distances from source to v as not yet computed
    visited[v]  = false      // Mark all nodes as unvisited
    previous[v] = undefined  // Previous node in optimal path from source

  dist[source]  = 0;         // Distance from source to itself is zero
  insert source into Q;      // Start off with the source node in priority queue Q

  while Q is not empty       // The main loop
    u = vertex in Q with smallest distance in dist[] and !visited
    remove u from Q;
    visited[u] = true        // Mark this node as visited

    for each neighbor v of u
      alt = dist[u] + dist_between(u, v) // Accumulate shortest dist from source
      if alt < dist[v] && !visited[v]
        dist[v] = alt              // Keep the shortest dist from src to v
        previous[v] = u
        insert v into Q            // Add unvisited v into the Q to be processed
  return dist,previous
```

Árbol de expansión mínimo

Árbol mínimo de expansión

El árbol de expansión mínimo de un grafo con pesos **G** es un árbol que contiene todos los vértices de **G** y que minimiza la suma de los pesos de sus aristas.

Dos algoritmos voraces clásicos para resolver este problema son:

- El algoritmo de Kruskal
- El algoritmo de Prim-Jarník

Árbol mínimo de expansión (Kruskal)

Intuitivamente, el algoritmo de Kruskal va construyendo el árbol de manera iterativa.

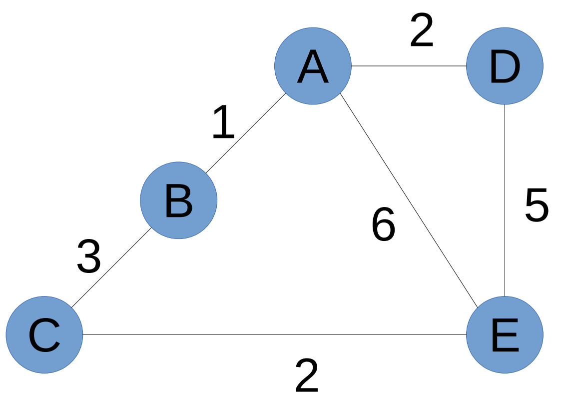
Inicialmente considera cada nodo como un árbol independiente.

En cada paso añade la arista de menor peso que conecta dos de estos árboles.

De esta forma los arboles se van uniendo hasta formar un árbol de expansión.

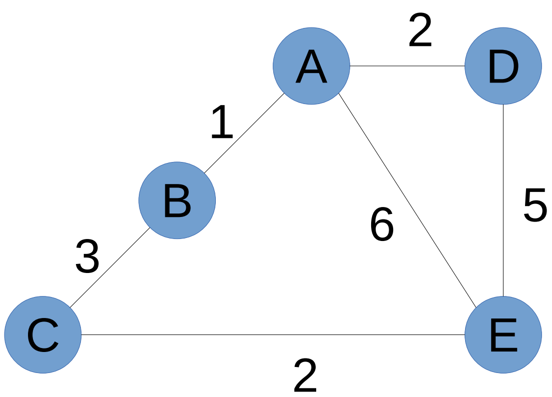
Árbol mínimo de expansión (Kruskal)

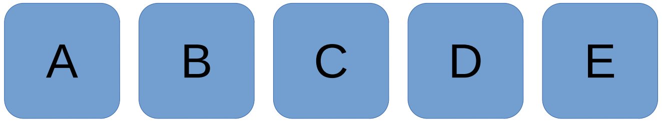
El siguiente ejemplo ilustra la ejecución del algoritmo de Kruskal sobre el grafo de la figura.



Árbol mínimo de expansión (Kruskal)

Inicialmente se crea un conjunto para cada nodo y una cola de prioridad que se inicia con todas las aristas priorizadas por su peso.





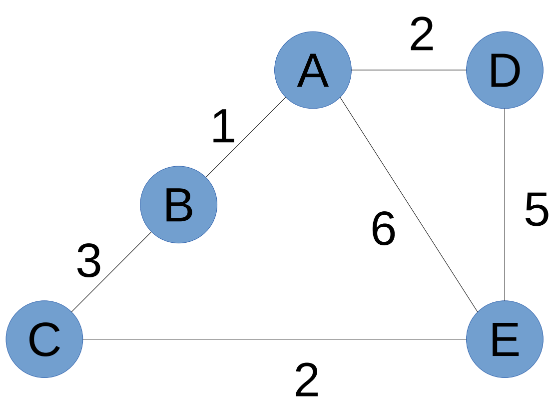
Conjuntos

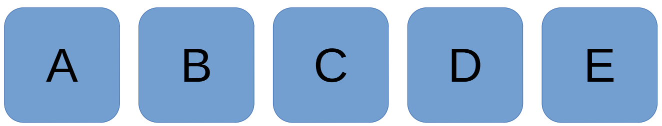
AB	1
AD	2
CE	2
BC	3
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Mientras haya más de un conjunto, se van extrayendo aristas de la cabeza de la cola. Si la arista extraída tiene los nodos en diferentes conjuntos se unen los conjuntos y se marca la arista como parte del árbol.





Conjuntos

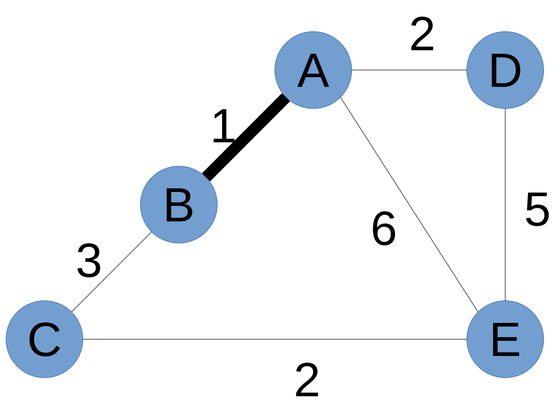
AB	1
AD	2
CE	2
BC	3
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Así, como tenemos 5 conjuntos:

- Se toma la arista **AB**, y como **A** y **B** están en conjuntos distintos se unen los conjuntos y se añade la arista al árbol



A, B	C	D	E

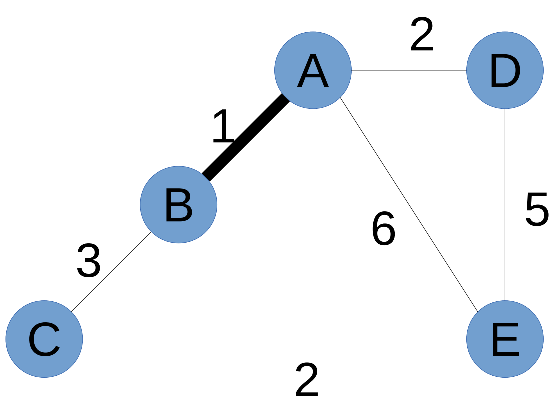
Conjuntos

~~AB~~	~~1~~
AD	2
CE	2
BC	3
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Ahora tenemos 4 conjuntos



A, B	C	D	E

Conjuntos

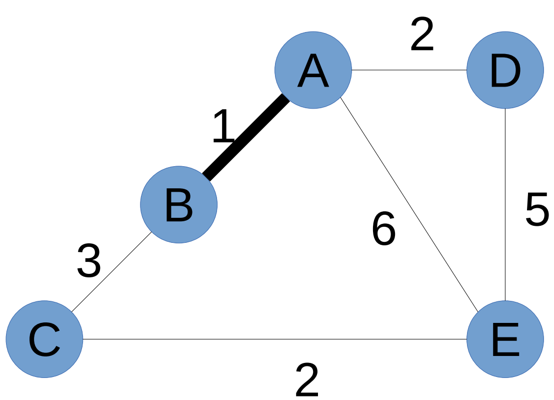
~~AB~~	~~1~~
AD	2
CE	2
BC	3
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Ahora tenemos 4 conjuntos

- Se toma la arista **AD**, y como **A** y **D** están en conjunto distintos se unen los conjuntos y se añade la arista al árbol



A, B | C | D | E

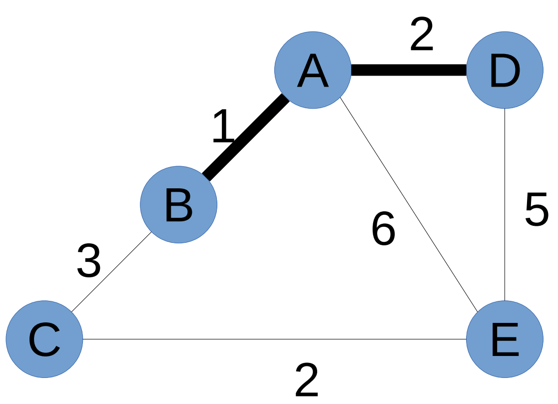
Conjuntos

~~AB~~	~~1~~
~~AD~~	~~2~~
CE	2
BC	3
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Quedan 3 conjuntos



A, B, D	C	E

Conjuntos

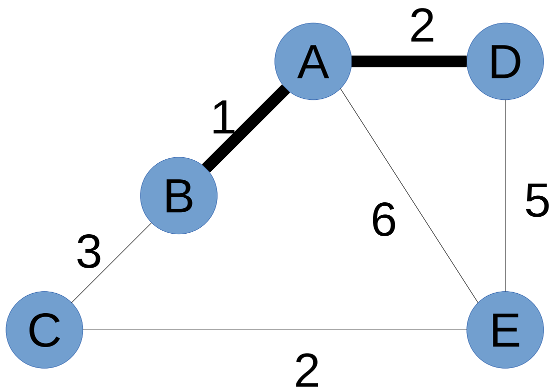
~~AB~~	~~1~~
~~AD~~	~~2~~
CE	2
BC	3
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Aún tenemos 3 conjuntos, por tanto:

- Se toma la arista **CE**, y como **C** y **E** están en conjunto distintos se unen los conjuntos y se añade la arista al árbol



A, B, D	C	E

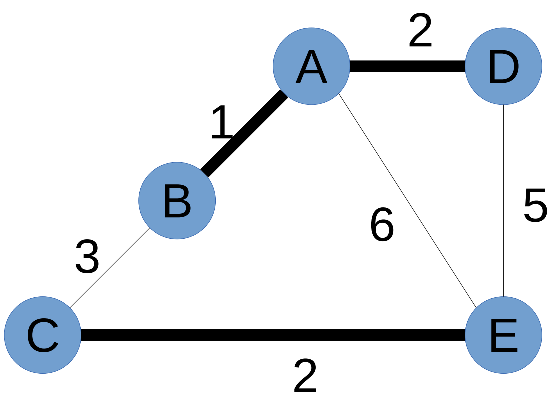
Conjuntos

~~AB~~	~~1~~
~~AD~~	~~2~~
~~CE~~	~~2~~
BC	3
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Ya solo quedan 2 conjuntos



A, B, D	C, E

Conjuntos

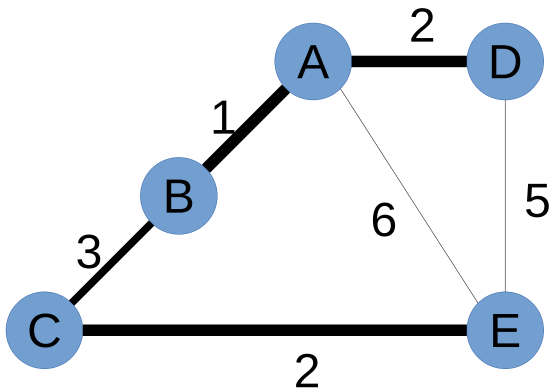
~~AB~~	~~1~~
~~AD~~	~~2~~
~~CE~~	~~2~~
BC	3
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Aún tenemos 2 conjuntos, así:

- Se toma la arista **BC**, y como **B** y **C** están en conjunto distintos se unen los conjuntos y se añade la arista al árbol



A, B, D

C, E

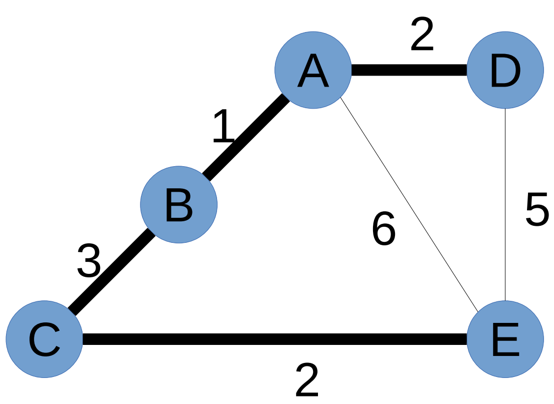
Conjuntos

~~AB~~	~~1~~
~~AD~~	~~2~~
~~CE~~	~~2~~
~~BC~~	~~3~~
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

Como ya solo queda un conjunto se considera que el algoritmo ha terminado y el árbol queda formado.



A, B, D, C, E

Conjuntos

~~AB~~	~~1~~
~~AD~~	~~2~~
~~CE~~	~~2~~
~~BC~~	~~3~~
DE	5
AE	6

Cola de prioridad

Árbol mínimo de expansión (Kruskal)

El algoritmo de Kruskal tiene complejidad **O(m log(m))**, donde **m** es el número de aristas.

```
function Kruskal(graph)
  Initialice a priority queue Q with all edges using weights as keys
  foreach v : graph.vertices()
    conjuntos.createconjunto(v)

  tree = ∅
  while tree.size() < (graph.vertices().size() - 1)
    (u,v) = Q.removeMin()
    if conjuntos.getconjunto(u) ≠ conjuntos.getconjunto(v)
      tree.add(u, v)
      conjuntos.joinconjuntos(u, v)
  return tree
```

Cierre transitivo de un digrafo

Algoritmos sobre grafos dirigidos

La mayor parte de los problemas habituales en los grafos dirigidos se refieren a la accesibilidad de los nodos. Así, suele ser de interés:

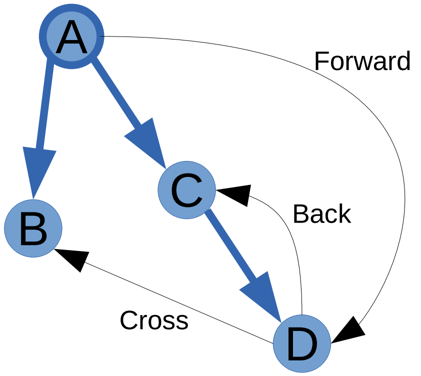
- Saber si desde un vértice se puede acceder a otro
- Encontrar todos los vértices accesibles desde un vértice dado
- Determinar si el grafo tiene ciclos
- Computar el cierre transitivo del gráfo
- Determinar si el gráfo esta fuertemente conectado

Algoritmos sobre grafos dirigidos

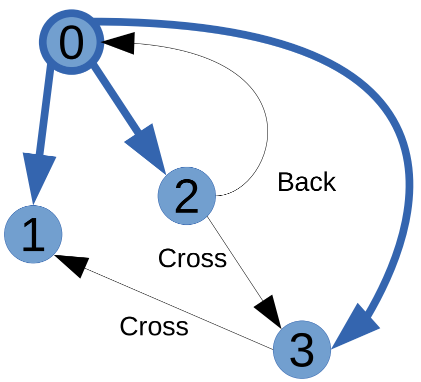
Los algoritmos de Recorrido en profundidad y Recorrido en anchura se pueden adaptar fácilmente para los digrafos.

En la Recorrido en profundidad aparecen 3 tipos de arcos que no pertenecen al árbol de expansión: back edges, forward edges y cross edges.

En la Recorrido en anchura solo aparecen 2 tipos de arcos: back edges y cross edges.



Explorando en profundidad un digrafo es posible llegar a un descendiente y encontrarlo ya visitado (como el arco AD, etiquetado como forward), pero explorando en anchura esto es imposible.



Cierre transitivo

Una posible solución para resolver el cálculo del cierre transitivo consiste en usar DFS desde cada uno de los vértices.

La complejidad de este enfoque es **$O(n^3)$** en el caso usar la implementación de matriz de adyacencia.

Si se usa la lista de adyacencia la complejidad es **$O(n(n+m))$**, aunque se aproxima a **$O(n^3)$** si la matriz es densa.

Cierre transitivo (Floyd-Warshall)

El algoritmo Floyd-Warshall presenta una alternativa muy simple para obtener el cierre transitivo de un grafo.

El algoritmo consiste en añadir para cada vértice K que posibilite la conexión entre 2 vértices I y J un arco que conecte directamente esos vértices I y J.

El proceso se repite iterativamente para toda combinación de vértices.

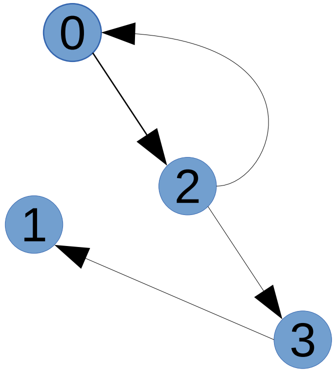
El resultado de este proceso es el cierre transitivo del grafo inicial.

Esta implementación surge de deshacer la recursividad del algoritmo que para cada posible combinación (i,j) mira recursivamente si existe un k intermedio.

La complejidad del algoritmo Floyd-Warshall también es **$O(n^3)$**.

Cierre transitivo (Floyd-Warshall)

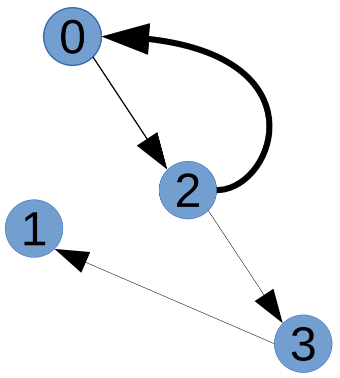
Cierre transitivo del grafo adjunto:



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

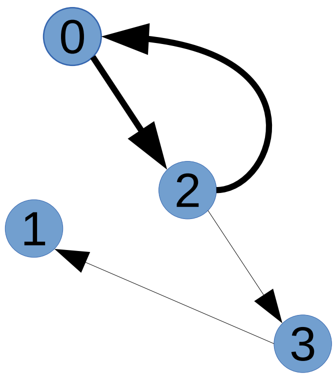
- Para cada arista que termina en 0: la (2,0)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

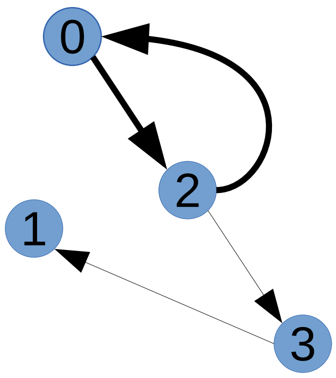
- Para cada arista que termina en 0: la (2,0)
  - Para cada arista que empieza en 0: la (0,2)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

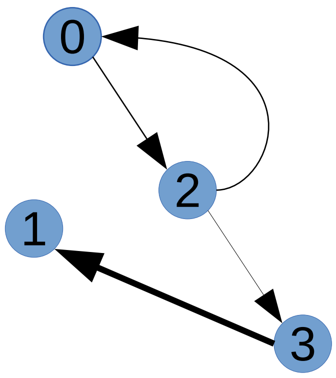
- Para cada arista que termina en 0: la (2,0)
  - Para cada arista que empieza en 0: la (0,2)
    - Añadir la (2,2) → no, porque no se permiten bucles



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

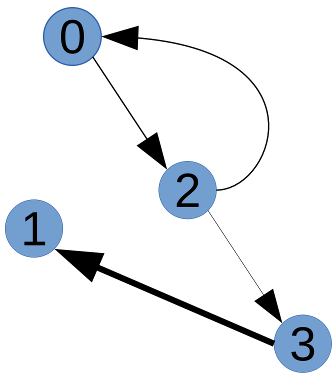
- Para cada arista que termina en 1: la (3,1)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

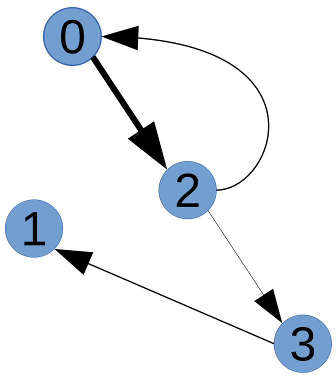
- Para cada arista que termina en 1: la (3,1)
  - Para cada arista que empieza en 1 (no hay)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

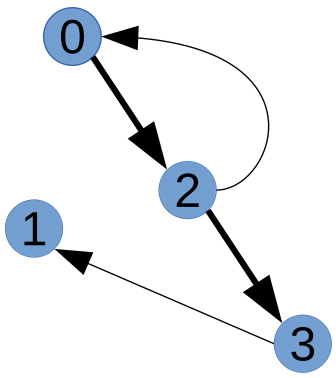
- Para cada arista que termina en 2: la (0,2)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

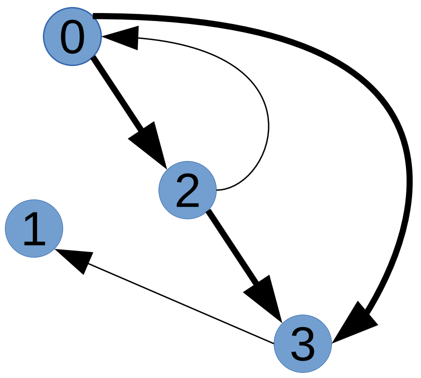
- Para cada arista que termina en 2: la (0,2)
  - Para cada arista que empieza en 2: la (2,3)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

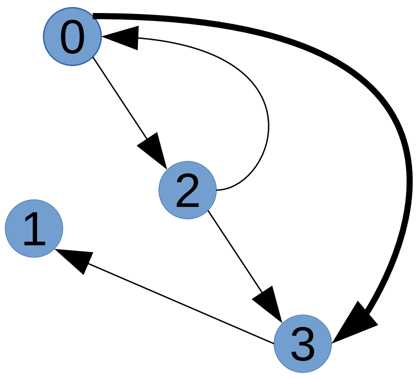
- Para cada arista que termina en 2: la (0,2)
  - Para cada arista que empieza en 2: la (2,3)
    - Añadir la arista (0,3) si no está



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

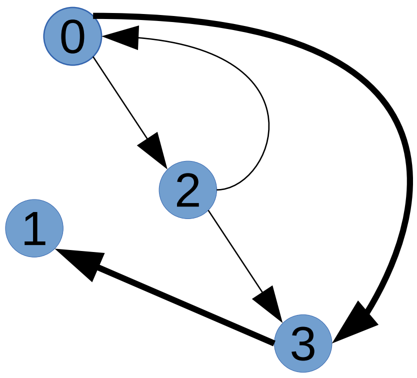
- Para cada arista que termina en 3: la (0,3)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

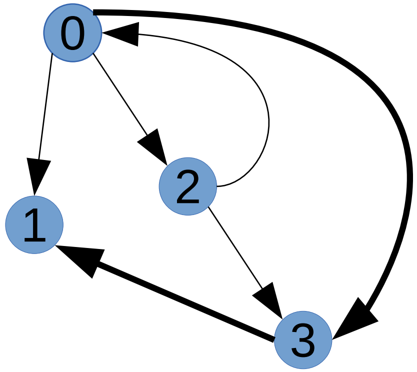
- Para cada arista que termina en 3: la (0,3)
  - Para cada arista que empieza en 3: la (3,1)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

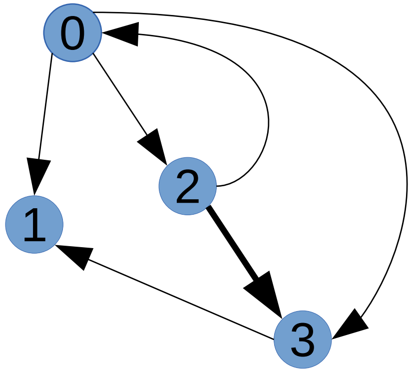
- Para cada arista que termina en 3: la (0,3)
  - Para cada arista que empieza en 3: la (3,1)
    - Añadir la arista (0,1) si no existe



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

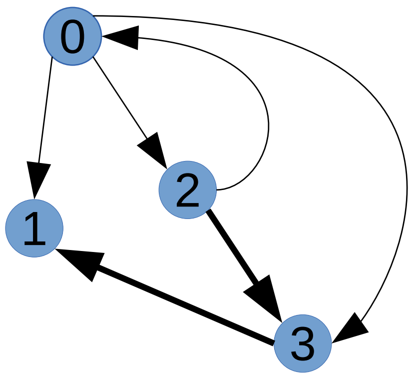
- Para cada arista que termina en 3: la (2,3)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

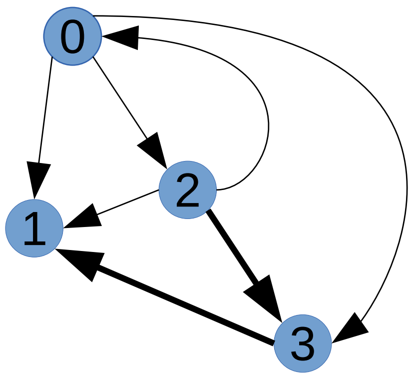
- Para cada arista que termina en 3: la (2,3)
  - Para cada arista que empieza en 3: la (3,1)



Cierre transitivo (Floyd-Warshall)

Cierre transitivo del grafo adjunto:

- Para cada arista que termina en 3: la (2,3)
  - Para cada arista que empieza en 3: la (3,1)
    - Añadir la arista (2,1) si no existe



Cierre transitivo (Floyd-Warshall)

```
algorithm FloydWarshall(graph)

  auxGraph[0] = graph

  for k=1 to graph.vertices().size()
    v = graph.vertices()
    auxGraph[k] = auxGraph[k-1]
    for i=1 to v.size()
      for j=1 to v.size()
        if (i != j) & (i != k) & (j != k)
          if auxGraph[k-1].areAdjacent(v[i],v[k]) &
             auxGraph[k-1].areAdjacent(v[k],v[j])
             auxGraph[k].insertEdge(v[i],v[j])

  return auxGraph[v.size()]
```

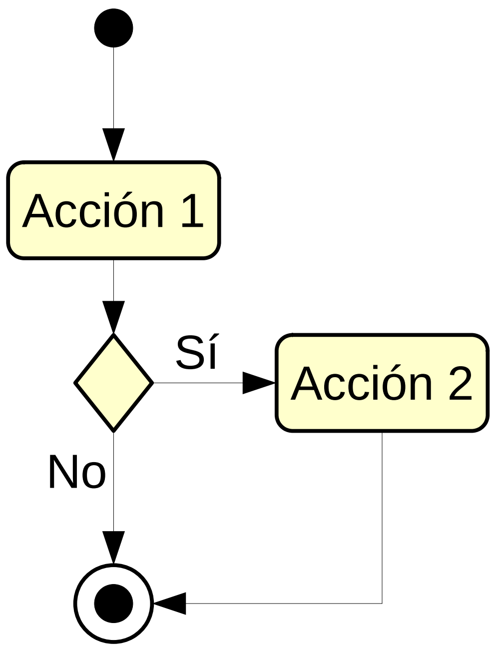
Aplicaciones

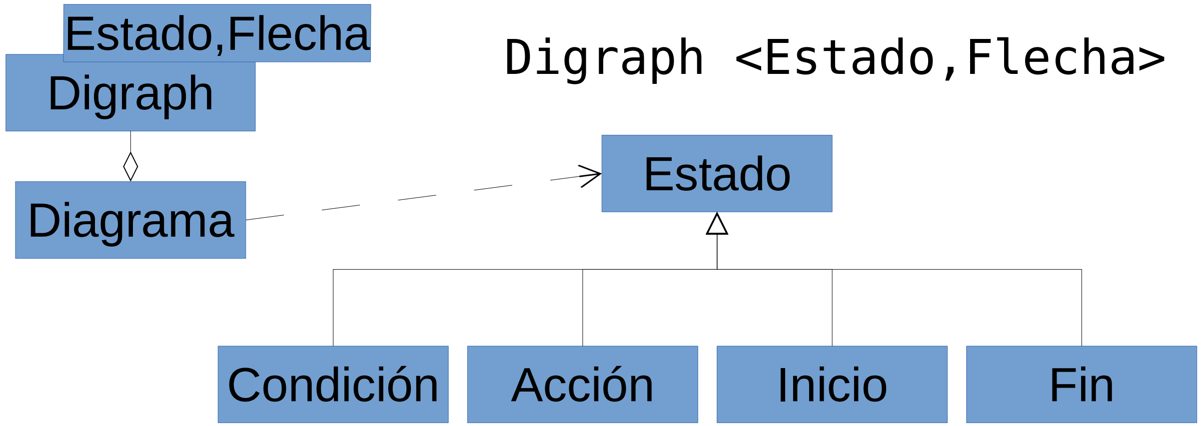
Identificar el tipo de grafo y los algoritmos útiles

- Un diagrama UML de actividad o de casos de uso
- Las relaciones en una red social
- Un laberinto
- Qué movimiento hacer en un tablero de parchís
- Una planificación de una serie de trabajos
- Un mapa de calles y carreteras
- Cómo trazar el cableado que conecte los ordenadores de una oficina

Identificar el tipo de grafo y los algoritmos útiles

Para un diagrama UML de actividad como el de la figura se podría utilizar un digrafo, en el que los vértices pertenecerían a una jerarquía de estados y en el que los arcos podrían contener cadenas.





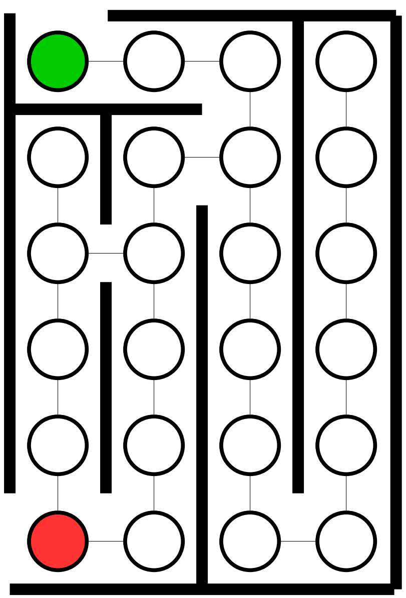
`Digraph <Estado,Flecha>`

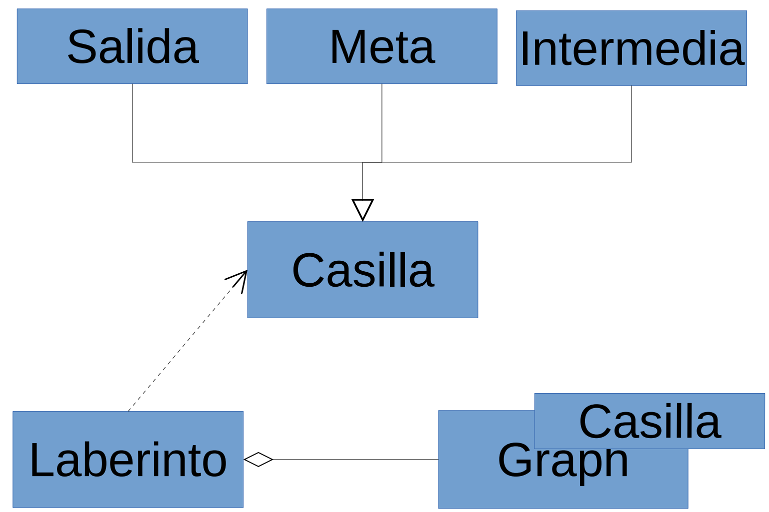
El cálculo del cierre transitivo permitiría detectar acciones a las que es imposible transitar desde el estado inicial, o comprobar si el estado final es accesible desde el inicial.

Aplicaciones

Identificar el tipo de grafo y los algoritmos útiles

Para un laberinto como el de la figura se podría utilizar un grafo, en el que los vértices pertenecerían a clase Casilla y los arcos podrían indicar simplemente la posibilidad de transito.





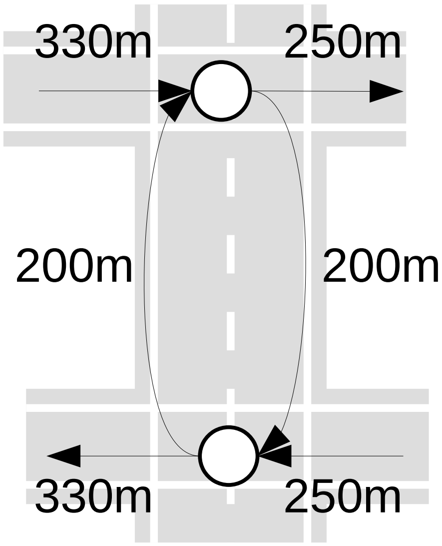
`Graph <Casilla>`

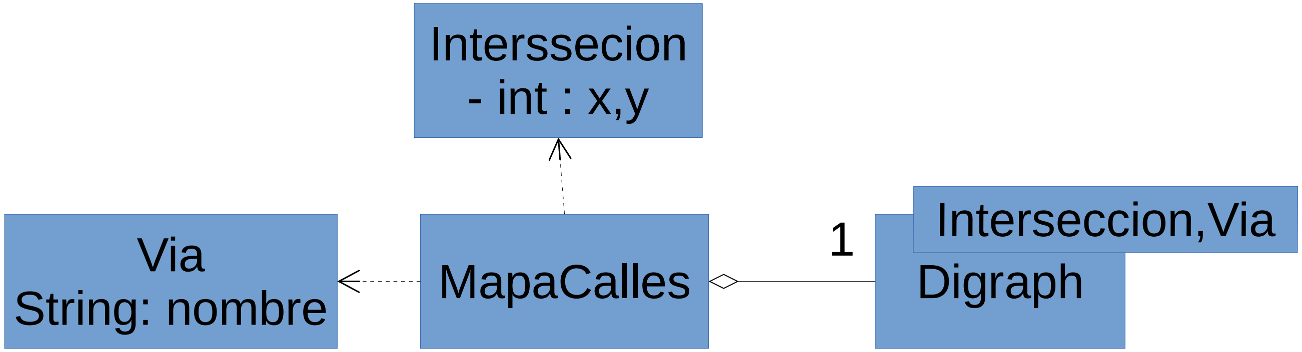
Un algoritmo de búsqueda en amplitud me permitiría localizar el camino más corto entre la salida y la meta.

Identificar el tipo de grafo y los algoritmos útiles

Para un plano de calles y carreteras como el de la figura se podría utilizar un digrafo, en el que los vértices pertenecerían a clase Intersección y los arcos podrían indicar el tipo de vía y su distancia.

`Digraph <Interseccion,Via>`





Una implementación que resolviese el problema del viajante permitiría trazar trayectorias óptimas entre cualquier par de intersecciones del mapa.

Referencias

- [GOO] Michael T. Goodrich and Roberto Tamassia, Data Structures and Algorithms in Java, Willey, 2010.
- [COR01] Cormen, T. H. Introduction to Algorithms. MIT press, 2001.
- JGrahT – http://jgrapht.sourceforge.net/.
- GUAVA - Google Collecctions - http://code.google.com/p/guava-libraries/
- Apache Collections - http://commons.apache.org/collections/