Carne: 2025072099

Grupo: 02

Ronny Espinoza Cordero

Aula invertida #11 **GRAFOS**

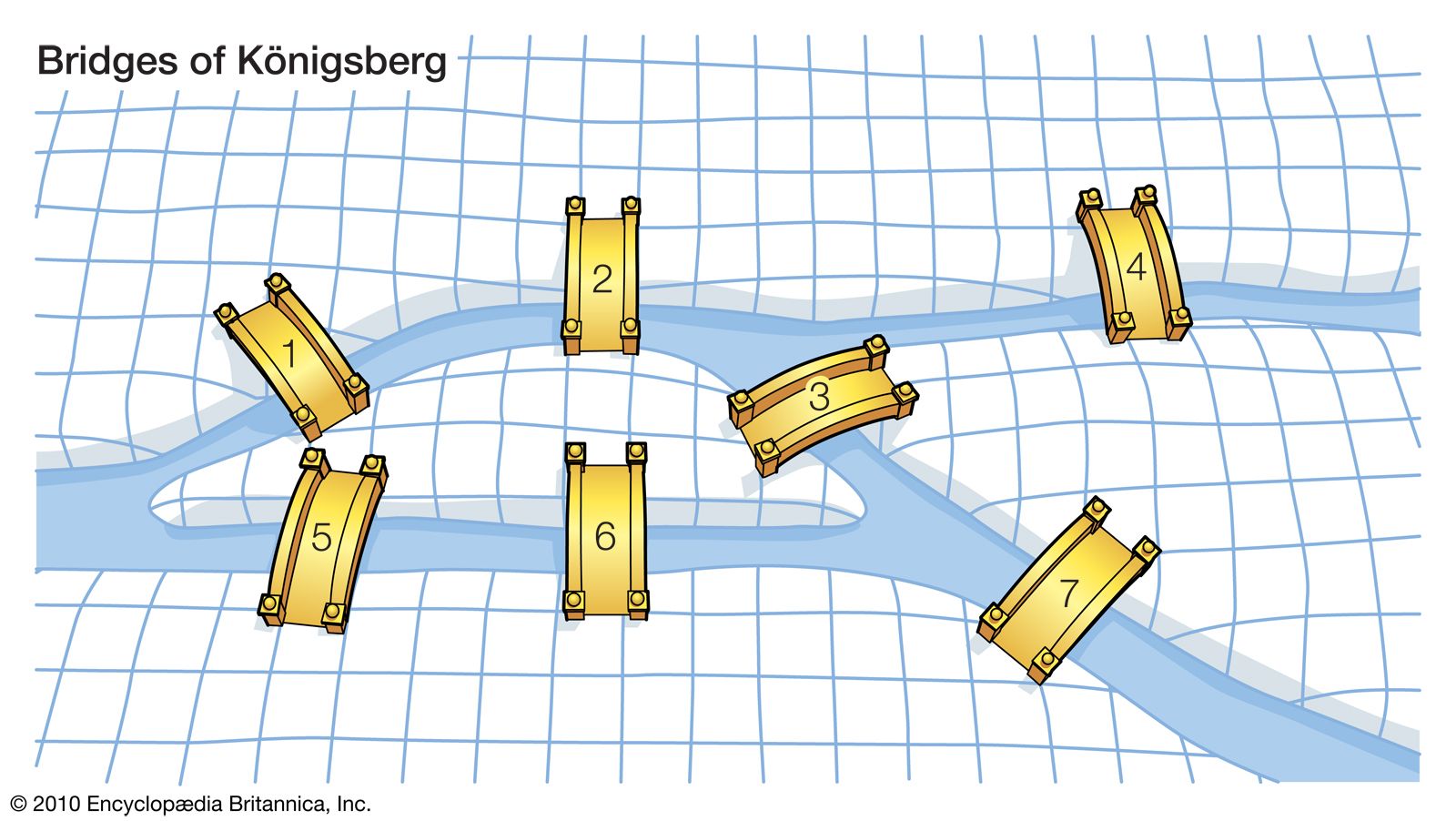
La teoría de grafos tiene muchos usos en aspectos como la ingenieria, computacion, biología, física, urbanismo, comunicaciones, economía y empresa incluso redes sociales.

Ademas se usan en elementos como el internet, los protocolos de comunicaciones, navegadores GPS, inteligencia militar y muchísimas cosas más. Casi que cualquier cosa electrónica se puede entender con un grafo.

**Historia de los Grafos**

El origen de los grafos se le atribuye el matemático Leonhard Euler que demostró que el esquema grafico de los 7 puentes de Königsberg no podía recorrerse partiendo de un punto cruzando cada puente una sola vez y volviendo al punto de partida.

Con este problema y el trabajo que se dio para resolverlo se dio la primera aplicación de la teoría de grafos.



El problema que se daba era que dado el mapa de Königsberg dividido en 4 regiones se planteaba la pregunta: ¿Es posible dar un paseo comenzando desde cualquiera de estas regiones, pasando por todos los puentes, recorriendo solo una vez cada uno, y regresando al mismo punto de partida?

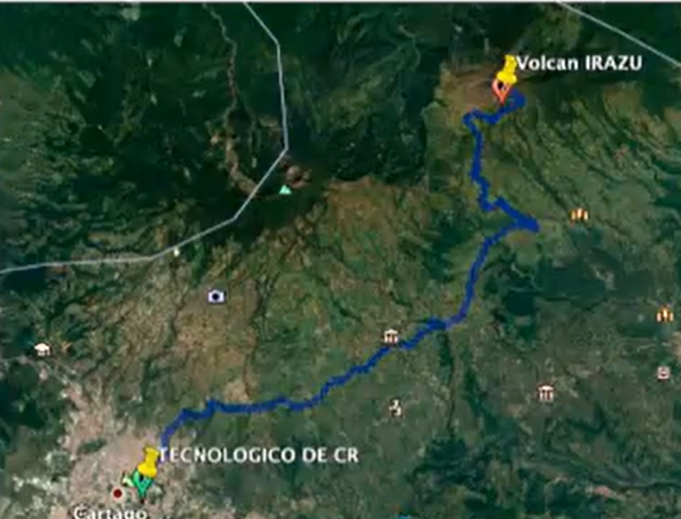
Cosa que Euler pudo resolver con un teorema matemático.

**Problema practico o “más cercano”**

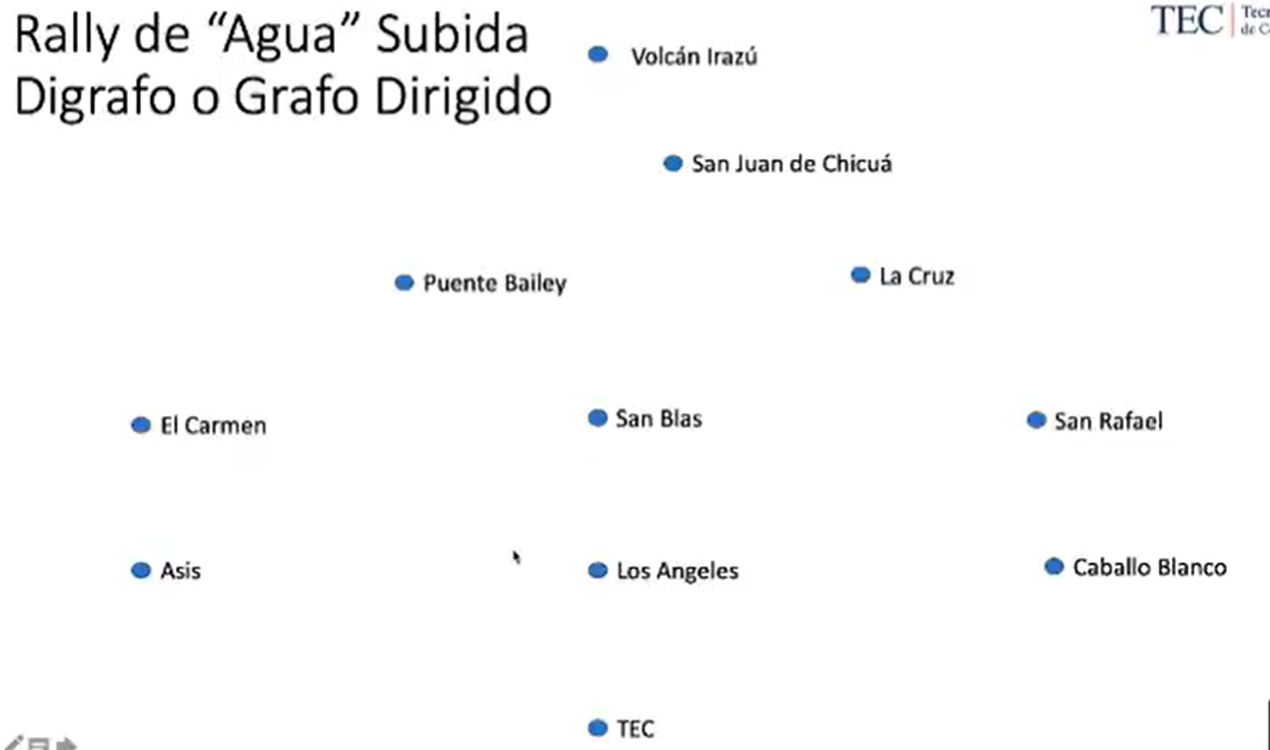
En el TEC se daba un Rally de “Agua” en el que la gente tenia que recorrer el campus del TEC y en ciertos puntos se debe de tomar una botella de agua.

Este ejemplo permitirá entender el tema del Grafo Dirigido.

Si a este rally se le hace una variación en el que no solo se debe recorrer el TEC sino que hay que pasar por la cúspide del volcán Irazú.

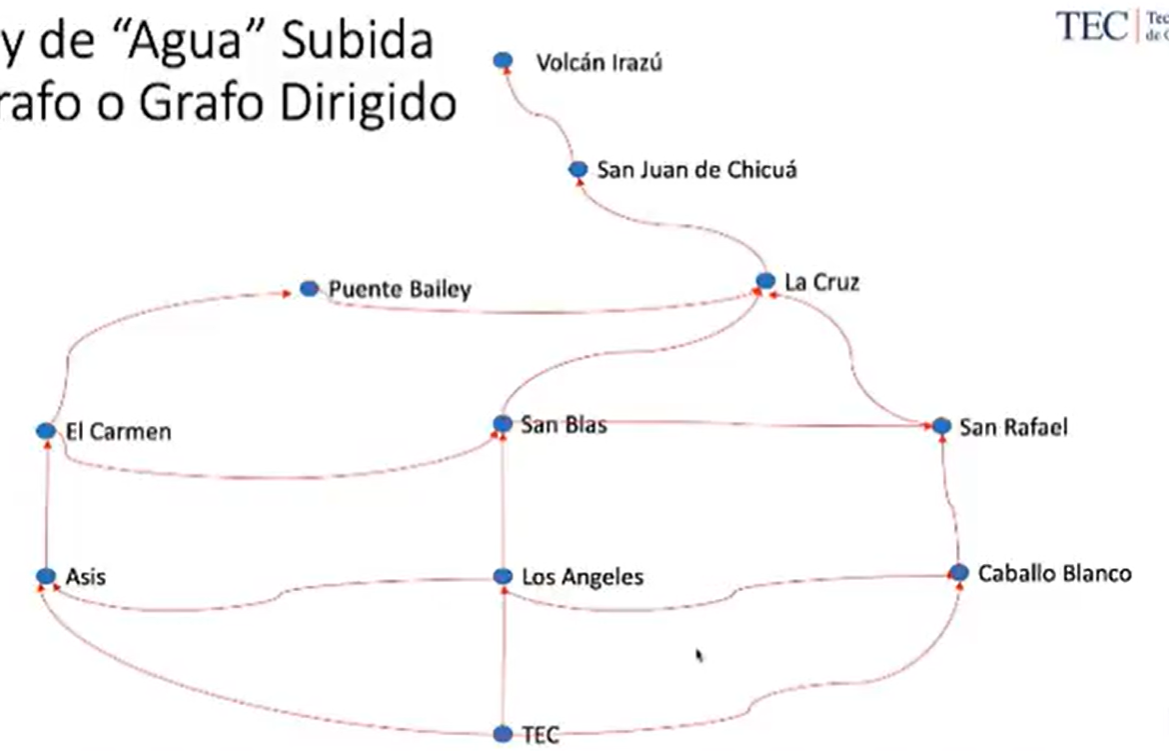


Entonces debemos analizar las diferentes rutas y para eso tenemos los siguientes puntos:

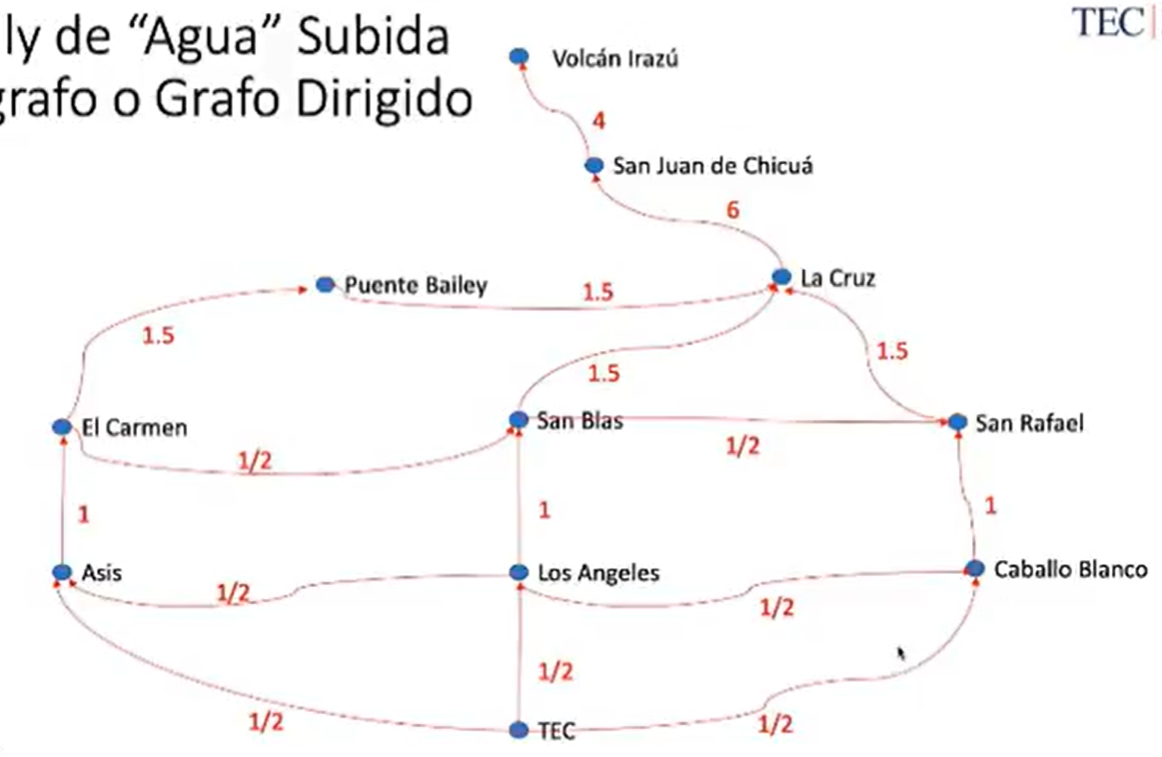


Aquí tenemos el punto de inicio (el TEC) y el punto de llegaba (el Volcán Irazú), en medio de estos hay ciertos barrios o lugares emblemáticos.

Con estos puntos se tienen las siguientes rutas o caminos entre puntos:



Ahora, como se explico antes en este rally se deben tomar botellas de agua, esto servirá para atribuir un costo a la acción de viajar desde un punto a otro, dándonos el siguiente resultado:

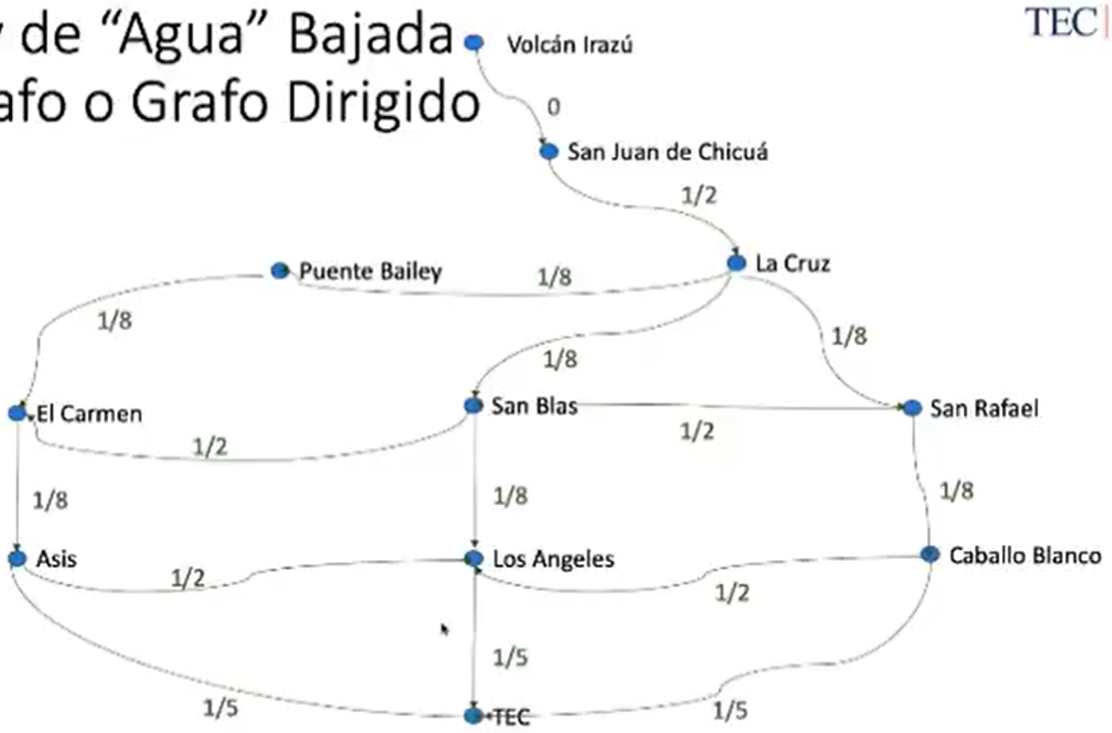


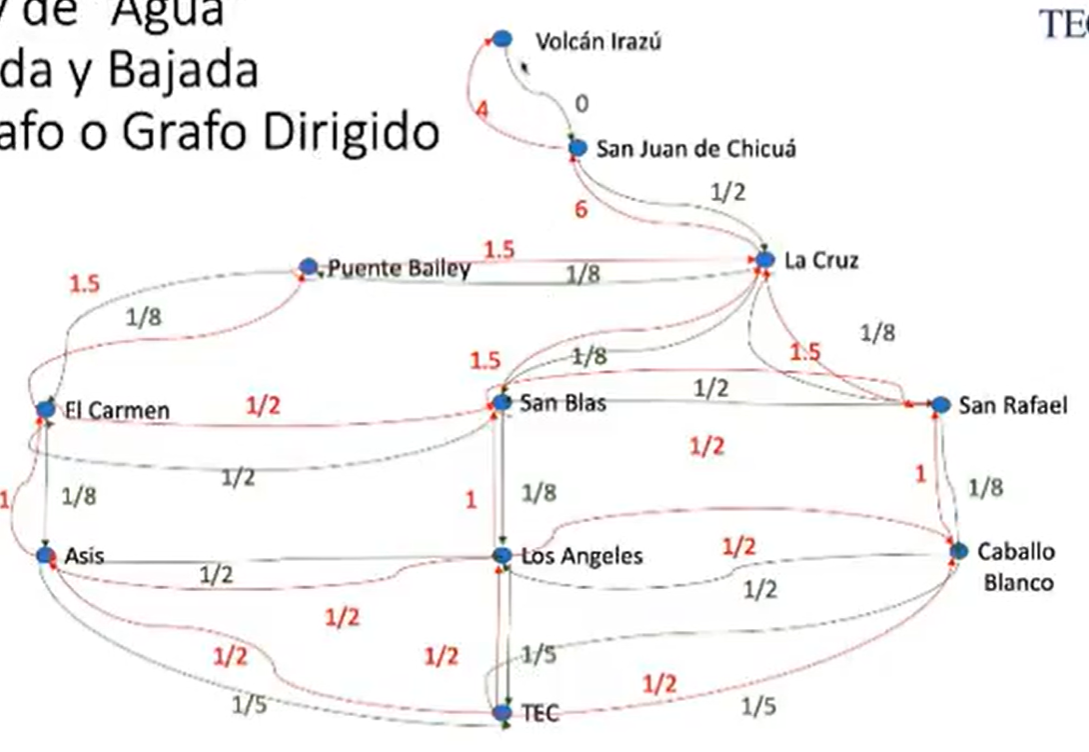
Entonces se podría decir que para ir desde el TEC hasta Los Angeles se consume media botella de agua igual que ir desde Los Angeles hasta otros puntos.

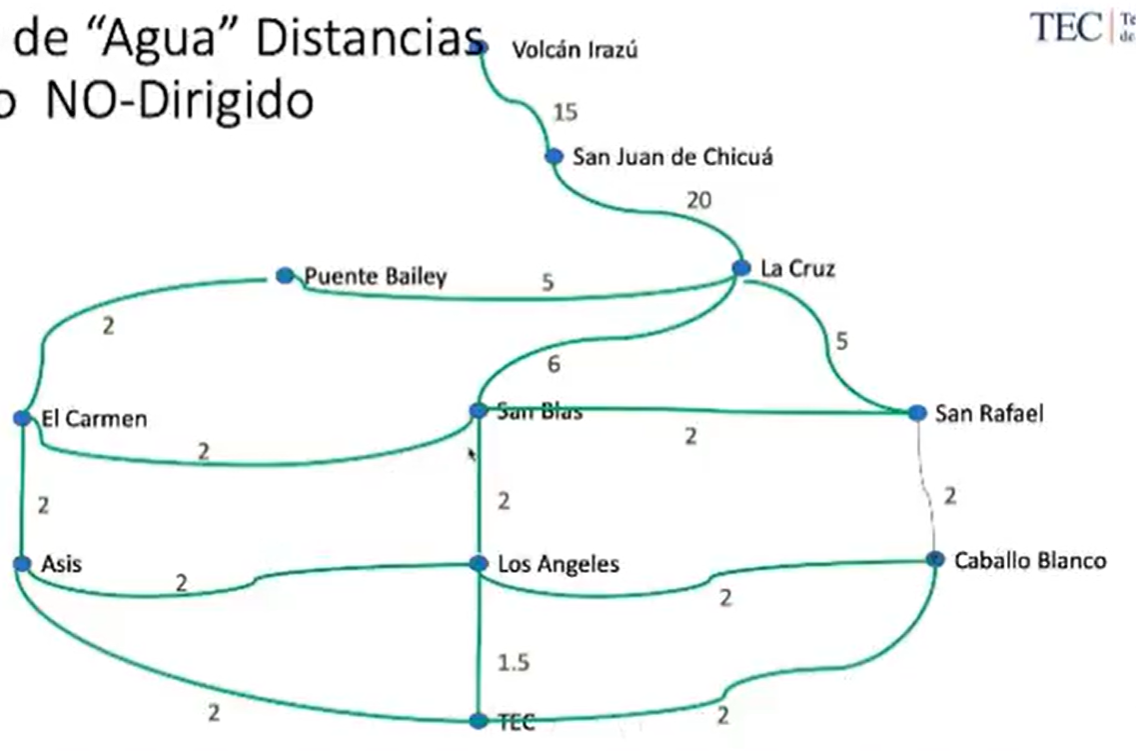
Ahora hablando de una representación clara del concepto de grafos se dice que los puntos azules son los nodos o vértices y las líneas rojas que conectan los diferentes nodos son arcos o aristas.

A este grafo se le llama Grafo Dirigido o Dígrafo porque los arcos o aristas tienen punta de flecha, es decir que las aristas tienen una dirección.

También se puede analizar el grafo contrario al que se acaba de mostrar, en el que se pasa del Irazu al TEC, es importante notar la diferencia en los valores de ir de un punto a otro:



La unión de estos dos dígrafos da como resultado el digrafo completo que muestran las dos direcciones:

Ahora al considerar un grafo no dirigido se puede entender este concepto como un grafo en el que los vértices van en ambos sentidos ósea en ambas direcciones:  


Analizando el ejemplo práctico se puede decir que no se están analizando botellas de agua sino distancias. La distancia no varia de un punto a otro por lo que no hace falta indicar flechas o algo parecido.

**Introducción y definiciones**

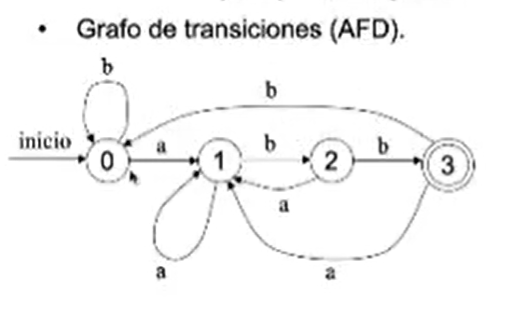
Un grafo es una tupla G=(V, A) donde V es un conjunto no vacío de vértices y A es el conjunto de aristas o arcos. Aquí cada arista es un par que tiene un punto de salida y un punto de llegada representado por un par (v, w), donde v es la cabeza de la arista y w es la cola de la arista, estos pertenecen a V. También tenemos que un vértice es adyacente sí y solo sí (v, w) pertenece a A.

Como otro concepto importante se tienen los caminos que son secuencias que se pueden dar entre los diferentes nodos. Basado en esto se puede hablar de la longitud de un camino que representa el número de aristas del camino y el concepto de camino simple que es un camino en el que todos los vértices pueden ser distintos.

En un grafo no dirigido las aristas no están ordenadas. Ósea (v,w) = (w,v).

En un grafo dirigido las aristas tienen un sentido, una dirección, ósea los pares si están ordenados. (v, w) no es lo mismo que (w, v).

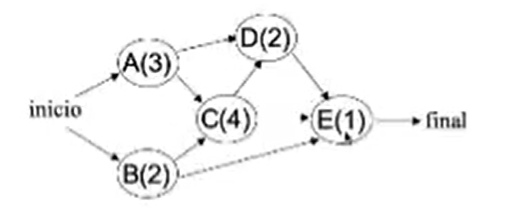
Se pueden analizar diferentes ejemplos de grafos como por ejemplo:

Grafo de transiciones: En este caso los nodos son estados en lo que puede estar un proceso y las aristas son la forma de pasar desde un estado a otro, permitiendo retroceder y avanzar en ciertos procesos.

Grafo para representar el tiempo de vuelos: Se representan los aeropuertos como nodos y las transiciones son los tiempos.



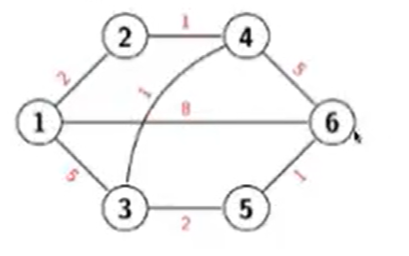
Grafo para la planificación de tareas: Permite controlar la planificación de tareas, aquí se dice que ciertas actividades no pueden arrancar hasta que se terminen otras actividades.



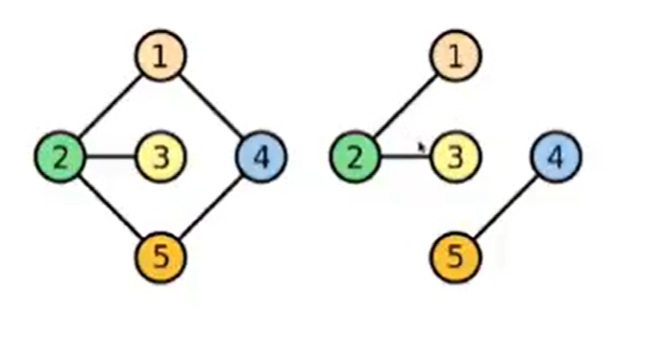
Grafos de ciudades: Se representan las ciudades y la distancia entre estas.

**Ciclos en los grafos**

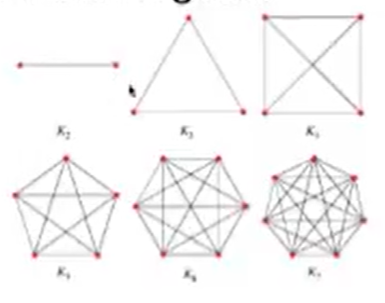
Un ciclo es un camino en el que el primer y el ultimo vértice son iguales. Ósea el origen y el final son el mismo. Dado dos vértices v, w se dicen que estos están conectados si existe un camino de v a w, ósea si hay nodos intermedios entre estos. Por ejemplo en este caso se tiene que 1 y 5 están conectados ya que tienen un nodo intermedio:



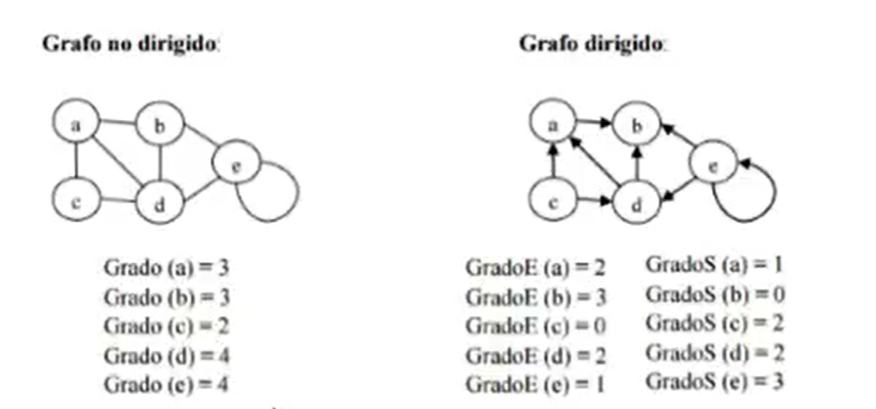
Se dice que un grafo es conexo o conectado si hay un camino entre cualquier par de vértices. Si es un grafo dirigido se dice que es fuertemente conexo. En el siguiente caso podemos observar un grafo conexo y uno no conexo respectivamente:



Un grafo es completo si existe una arista entre cualquier par de vértices. Ejemplo así:

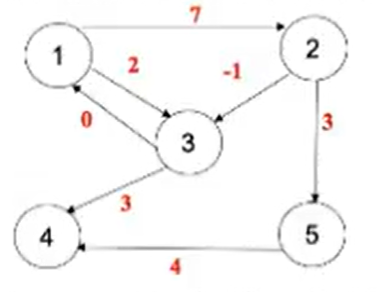
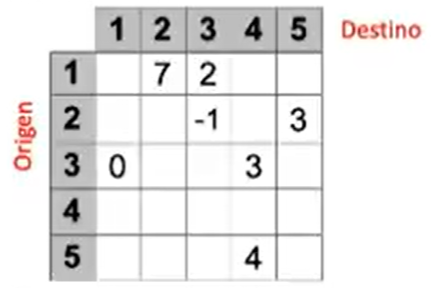


El grado de un vértice v: Corresponde al número de arcos que inciden en el. En el caso de un grafo dirigido se debe sacar el grado de entrada y el grado de salida. Ejemplo en estos grafos:



Además se tiene que un grafo esta etiquetado si se le asocia a cada arista un peso o valor, se dice que un grafo con pesos son grafos etiquetados con valores numéricos y un subgrafo de G (V, A) es un grafo de G’(V’A’) donde A’ pertenece a A y V’ pertenece a V. Se puede decir que un subgrafo es una porción de grafo.

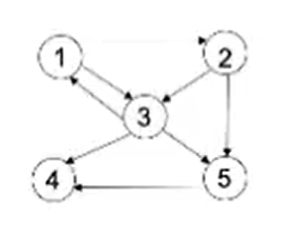
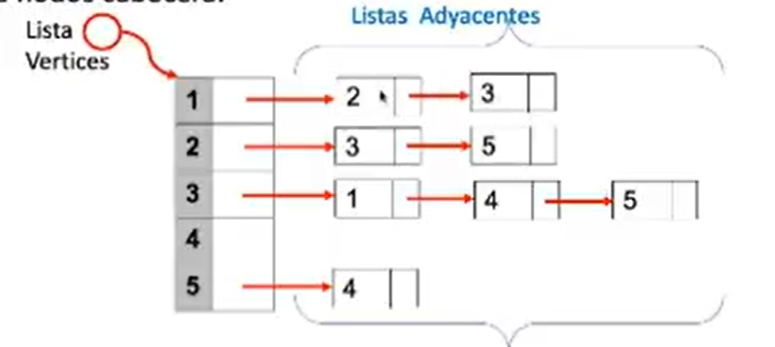
**Representación de grafos:**

Mediante matrices de adyacencia: Esta forma es la más simple pero que funciona con pocos nodos:

La matriz representa los pesos de pasar de un nodo a otro. Si el grafo es no dirigido la matriz es simétrica. Aun así se tienen ciertas desventajas como:

* Si el número de nodos es muy grande y hay poca conectividad se desperdicia memoria.
* Se deben conocer los tamaños aproximados que van a tener los grafos.

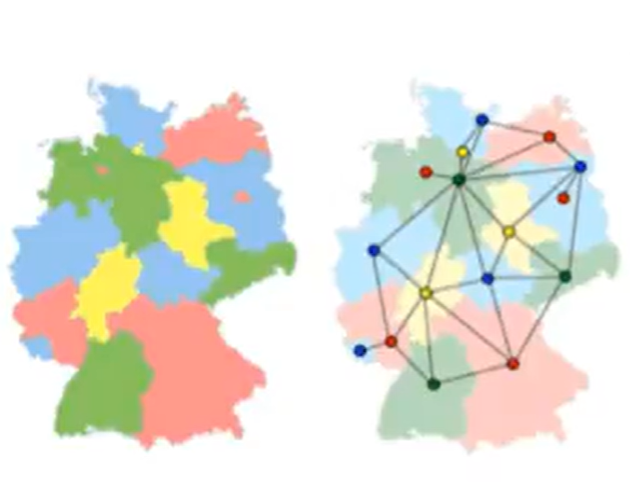
Representación mediante lista de vértices: Para cada nodo de V se tiene una lista de aristas que parten de ese nodo. Esas listas se guardan en un array.



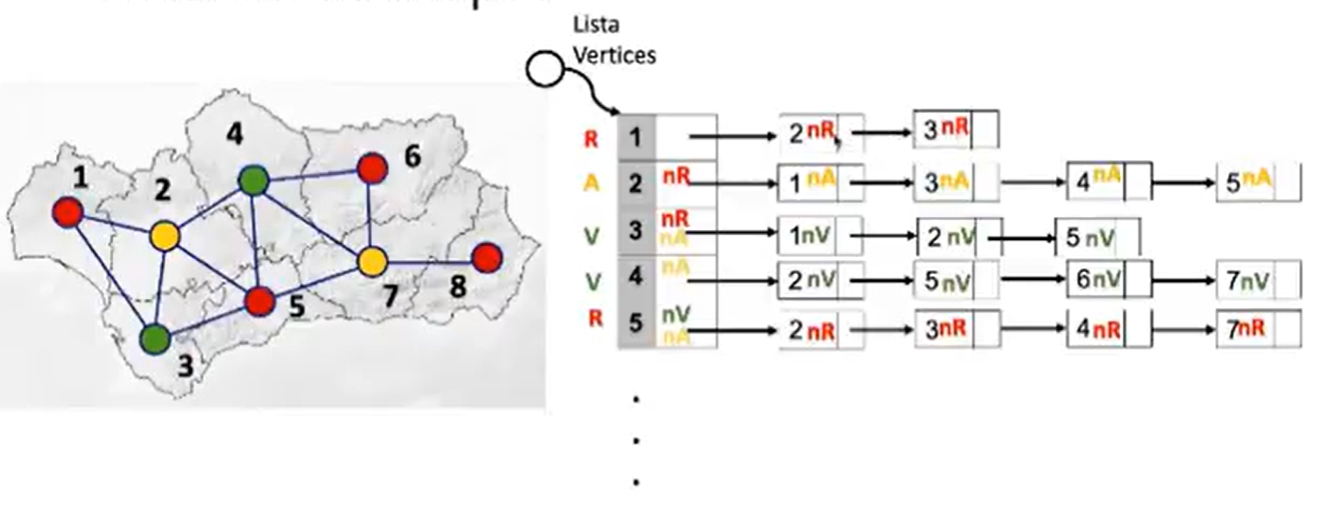
Si se estuviera trabajando con un grafo dirigido se debe calcular el grado de salida, recorriendo la lista correspondiente y el grafo de entrada recorriendo todas las listas.

**Coloreo de mapas**

En este algoritmo se asume una relación de vecindad o frontera entre regiones representándolas como aristas o relaciones y aquí cada región es un nodo.



La representación es muy parecida al método que se desarrolló más arriba:



**Algoritmos y problemas sobre grafos**

Recorridos sobre grafos: Al igual que con los arboles se empieza desde un nodo base y de ahí en adelante se visitan los arcos y los vértices de forma sistemática.

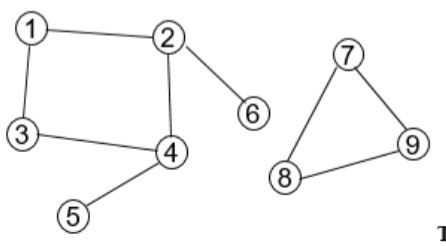
Existen 2 tipos de recorridos:

* Búsqueda en profundidad: Es como el equivalente a un recorrido en preorden. Se elige un nodo v de partida y se recorren los nodos adyacentes después de marcarlo como visitado.
* Búsqueda en anchura: Consiste en recorrer el árbol por niveles. Dado un nodo v, se primero los nodos adyacentes, luego los que estén a distancia 2 y luego a distancia 3.

**Búsqueda en profundidad**

Es importante recalcar que si un grafo es no dirigido entonces se generara un árbol por cada componente conexo. Se generará un bosque abarcador en profundidad

Para entender este ejemplo se va a usar un grafo no dirigido no convexo.



Como ya se dijo se está trabajando en un preorden por lo que siempre se pinta primero el nodo que se está visitando. En el caso que se está trabajando

Paso 1: Se empieza desde el nodo 1.

Paso 2: Del nodo 1 se pueden elegir 2 hijos pero se arranca por el hijo menor por etiqueta ósea el 2

Paso 3: Igual que en el paso anterior nos vamos por el hijo menor que es 4.

Paso 4: Se muestra 3 ya que es el hijo menor de 4.

Dado que no se han mostrado todos los elementos y que por el contrario, si se sigue en ese proceso se formará un ciclo se hace un arco de retroceso, entre el nodo inicial y el ultimo nodo visitado. Dado este arco se hace una secuencia en retroceso en la que se siguen pintando los hijos de los nodos, en este caso los hijos no mostrados ya.

Paso 5: Volvemos al nodo 4 y de ahí se visita el nodo 5.

Paso 6: Volvemos al nodo 2 y de ahí se visita el nodo 6.

En el subgrafo no visitado se hace el mismo proceso y se obtiene lo siguiente.

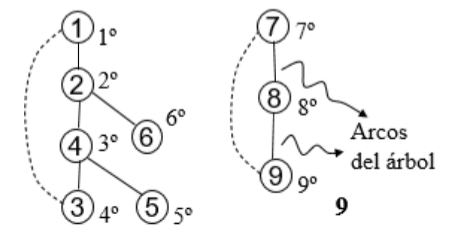
Paso 7: Se visita el nodo 7.

Paso 8: Se visita el hijo menor del nodo 7 osea el nodo 8.

Paso 9: A pesar de que el nodo 9 no es menor que 8, el nodo se visita desde ahí ya que es el único hijo que queda sin visitar.

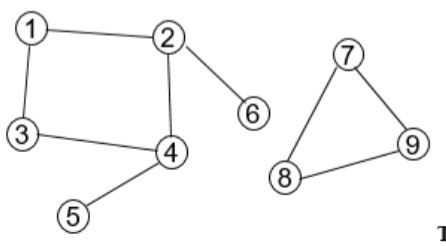
Paso 10: Desde el nodo 9 se genera un ciclo por lo que se hace un arco de retroceso.

Resultado:



**Búsqueda en anchura**

Trabajando con el mismo ejemplo y bajo el concepto del algoritmo de búsqueda en anchura.



Trabajando con capas y desde el mismo punto inicial tenemos el siguiente orden del recorrido:

Primera capa:1

Segunda capa:23

Tercera capa: 46

Cuarta capa: 5

Se obtiene: 1,2,3,4,6,5

Quinta capa: 7

Sexta capa: 8, 9

Se obtiene: 7, 8, 9

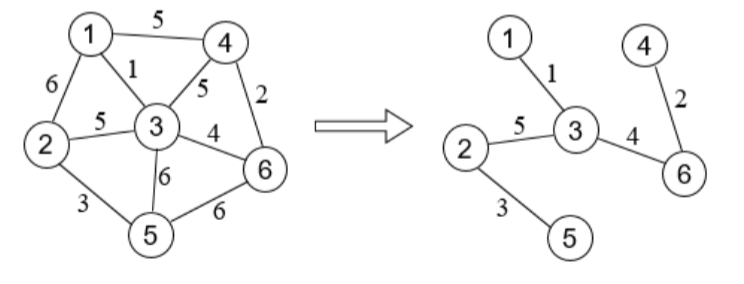
Resultado final: 1,2,3,4,6,5,7, 8, 9

**Algoritmos de expansión**

Un grafo puede tener muchas conexiones pero no todas estas conexiones son necesarias, algunos arcos son redundantes por decirlo de una forma. En este sentido se tienen los algoritmos de expansión que tratan de reducir el grafo de forma que sea mucho más eficiente.

Algoritmo Prim.

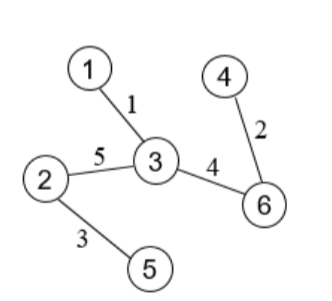
Para empezar se dice que en este algoritmo se usa una especie de “argolla”. Tenemos el siguiente ejemplo:



Igual que siempre empezamos en el 1 y de ahí nos vamos al nodo que tenga el menor costo de viaje, en este caso ese nodo vecino es 3, ya que para 4 se tiene un costo de 5 y para 2 se tiene un costo de 6.

Luego se dice que entre los nodos 1 y 3 el nodo más económico de visitar es el nodo 6, que tiene un costo de 4, luego se pasa al nodo 4 siguiendo este procedimiento.

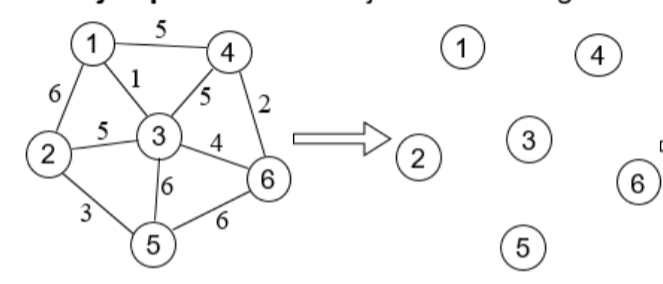
Al tener estos elementos decimos que si desde alguno de estos se toma el camino menor (5) se generaría un ciclo por lo que debemos salir de esos 4 nodos.

Esto nos llevaría al nodo 2 y finalmente al nodo 5

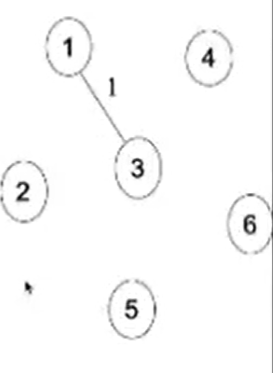
Aquí se puede notar que se redujo la cantidad de caminos, tomando en cuenta que nodos son más “económicos” que se pueden recorrer generando un grafo más eficiente.

Algoritmo de Kruskal

En este algoritmo se hace un “concurso” de aristas o de arcos, a partir del grafo original se hace un grafo vacío, ósea sin arcos

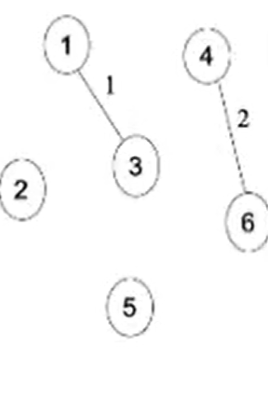


La idea del concurso es ver el orden de eficiencia de los caminos evitando caer en ciclos como se puede ver a continuación:

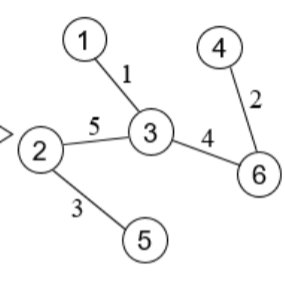


Se empieza por arco más económico de todos que en este caso es 1.

Pasamos al arco con coste 2.



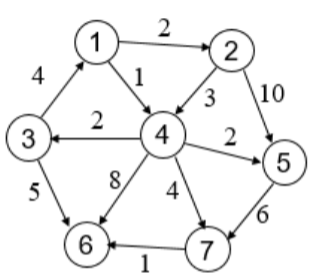
Y así se sigue hasta llegar al resultado:



Algoritmo de Dijkstra

En este algoritmo se trabajan con tablas de forma que los costos se van reemplazando.

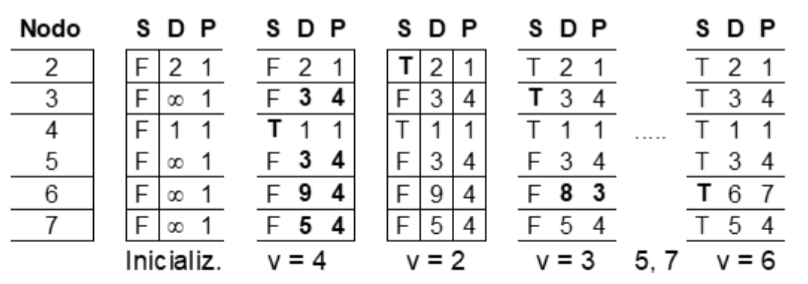
Veamos el siguiente ejemplo recalcando que es un grafo dirigido:



Al trabajar con el algoritmo de Dijkstra debemos hacer una tabla que contenga los siguientes valores.

|  |  |  |  |
| --- | --- | --- | --- |
| Nodos destino | Status | Peso | Proviene |

El Peso es lo que cuesta pasar del Proviene(nodo) al nodo destino. Evidentemente con un nodo no se puede llegar a todos los nodos por lo que la tabla debe de estar cambiando, usando los nodos disponibles, ósea los nodos vecinos. Estos nodos vecinos son los nodos cuyo destino no es infinito. Se analiza el vecino de menor Peso y se vuelve a hacer la tabla pero solo para encontrar los valores de la tabla cuyo Peso es infinito. Como se puede ver a continuación:



Aquí podemos analizar que el nodo 1 no se toma cuenta ya que empezamos a partir de ahí, además podemos ver que el Status(S) cambia si se toma en cuenta ese vecino especifico. Esto se hace hasta que toda la tabla tenga un Status S.