

Algoritmo de Kruskal

Objetivo: encontrar árbol de expansión mínima (traducido al español: encontrar un árbol dentro del grafo, el cual tenga la característica de que visite todos los nodos con la mínima suma de costes).

El algoritmo de Kruskal se realiza teniendo en cuenta lo siguiente: **encontrar la arista con menor valor que no forme ciclos dentro del grafo (ya que los árboles no contienen ciclos).**

La elección de las aristas mínimas dentro del grafo es libre, es decir, no deben estrictamente estar conectados con otro nodo ya visitado, debido a que posteriormente todo quedará conectado y sin ciclos.

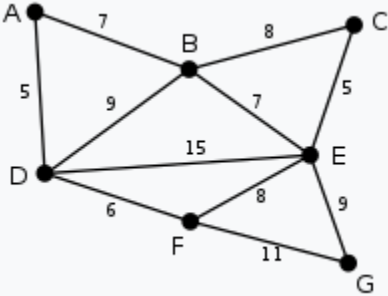
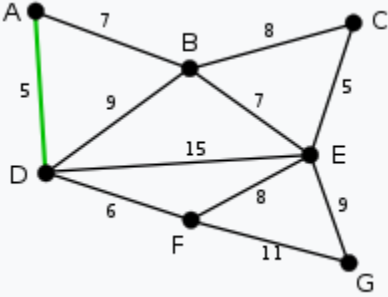
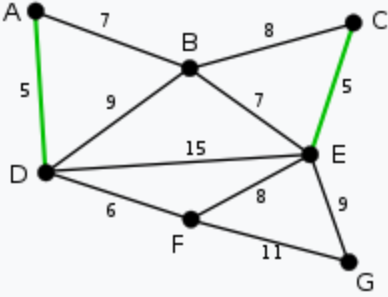
Por ahora todo bien, sin embargo, un tema importante es que la manera más conocida para detectar ciclos es recorriendo el grafo con DFS y verificar si llega a un mismo nodo por dos caminos. Este método va a tener problema para nosotros: y es que tendríamos que realizar este algoritmo **cada vez** que seleccionamos una nueva arista, lo cual no es para nada eficiente.

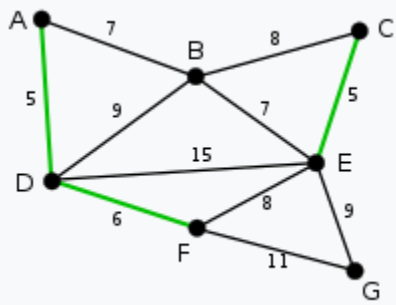
Respecto a esto, se utiliza una manera más óptima de detectar ciclos en una complejidad constante (**$O(1)$**), es decir, siempre tardará lo mismo, a diferencia de DFS que si bien es rápido, $O(1)$ es lo mejor. Ésta es la siguiente:

- Cada nodo pertenecerá a un árbol (al principio, todos los nodos pertenecen a su mismo árbol, por ejemplo, el nodo A al principio pertenece al árbol A).
- Si se desea elegir una arista, pero los nodos en los extremos pertenecen a un mismo árbol, quiere decir que, si elegimos esa arista, provocaremos un ciclo (no lo queremos).
- Si un nodo A perteneciente al árbol C cambia al árbol F, todos los nodos que pertenecían anteriormente al árbol C, pasan a formar parte también del mismo árbol F, registrando así que todos pasan a formar parte de un mismo árbol (F).

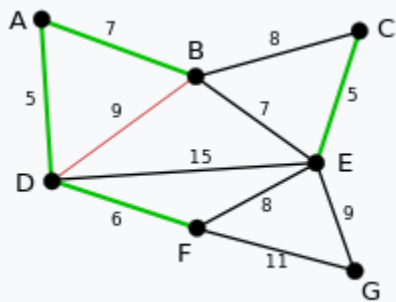
- Al final, todos pertenecerán a un mismo árbol.

Teniendo en cuenta esto, paso a explicar un ejemplo:

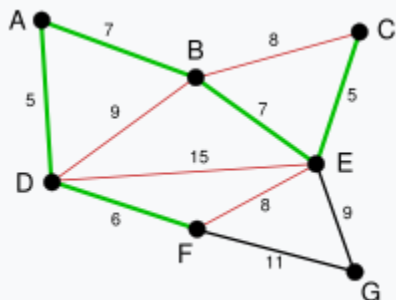
	<p>Este es el grafo original. Los números de las aristas indican su peso. Ninguna de las aristas está resaltada.</p>
	<p>AD y CE son las aristas más cortas, con peso 5, y AD se ha elegido <u>arbitrariamente</u>, por tanto se resalta.</p>
	<p>Sin embargo, ahora es CE la arista más pequeña que no forma ciclos, con peso 5, por lo que se resalta como segunda arista.</p>



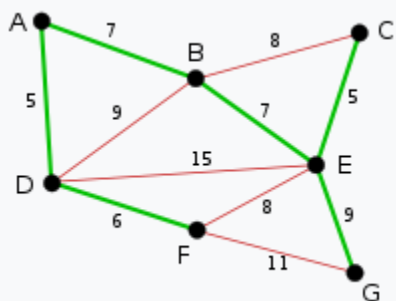
La siguiente arista, **DF** con peso 6, ha sido resaltada utilizando el mismo método.



Las siguientes aristas más pequeñas son **AB** y **BE**, ambas con peso 7. **AB** se elige arbitrariamente, y se resalta. La arista **BD** se resalta en rojo, porque formaría un ciclo **ABD** si se hubiera elegido.



El proceso continúa marcando las aristas, **BE** con peso 7. Muchas otras aristas se marcan en rojo en este paso: **BC** (formaría el ciclo **BCE**), **DE** (formaría el ciclo **DEBA**), y **FE** (formaría el ciclo **FEBAD**).



Finalmente, el proceso termina con la arista **EG** de peso 9, y se ha encontrado el **árbol expandido mínimo** con peso total de 39.

Se puede ver una implementación en el archivo `kruskal.cpp` de la misma carpeta (<https://github.com/mndzl/C-Codes/blob/master/Algorithms/kruskal.cpp>).