CIS 575. Introduction to Algorithm Analysis
Material for April 10, 2024

Minimum-Cost Spanning Trees

The topic of this note is presented in *Cormen*'s Section 21.1.

1 Minimum Spanning Tree

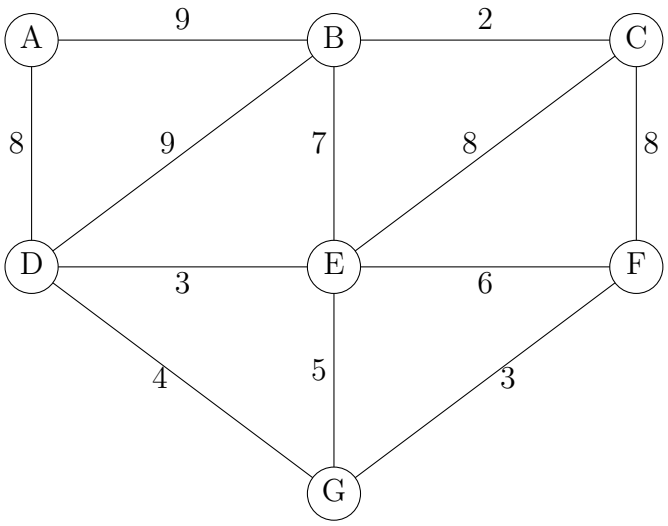
Given an **un**directed graph $G = (V, E)$ that is **connected** and which associates a **weight** > 0 to each edge in E, our *goal* is to find a subset E_0 of E that

- **spans**: each node in V is an end point of at least one edge in E_0
- forms a **tree**: the subgraph (V, E_0) is acyclic and connected
- has a **minimum** sum of the edge weights (no other spanning tree has lower weight).

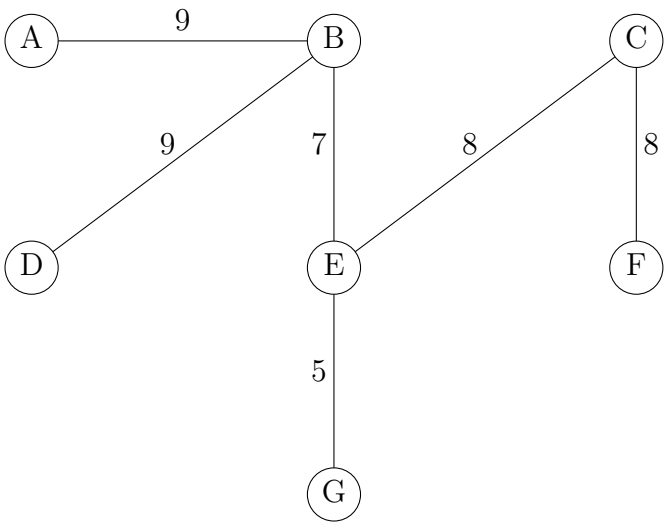
As usual, we shall let $n = |V|$ be the number of nodes, and let $a = |E|$ be the number of edges. We know that any tree must have $n - 1$ edges.

1.1 Example

As our running example, we shall use the graph



It is easy to find spanning trees; one such is:



but it has total weight 46 and is hardly the minimum one. And in fact, it may not appear obvious what the minimum spanning tree is. But this is why we need an algorithm to find it, and in subsequent lecture notes we shall develop not just one such algorithm but two: *Kruskal*'s, and *Prim*'s.

Both algorithms are *greedy*, that is they always make the right choice; once an edge has been selected for the minimum spanning tree, there is no regret. We shall now investigate *when* it is safe to add an edge to the list of edges already known to be in the tree.

1.2 Key Result

Assume we have already selected some edges T_0 for the minimum spanning tree. Assume that we can split the graph into (at least two) *components* such that all edges in T_0 are between two nodes from the *same* component. Let E_i be the edges in E that are *inter*-component, that is between nodes from two different components. Let e be an edge in E_i with minimum weight. Then it is *safe* to add e to T_0, in the following sense:

- assume that T is a spanning tree with $T_0 \subseteq T$
- then there exists a spanning tree T' with $T_0 \cup \{e\} \subseteq T'$ and with weight not higher than the weight of T.

That is, picking e will not prevent us from eventually having a minimum spanning tree.

Proof: If e belongs to T we can choose $T' = T$, so assume that $e \notin T$. Let e be between nodes u and v, and let C be the component that u belongs to. Since (V, T) is a spanning tree, there is an acyclic path π in T between u and v. Since u belongs to C but v does not (as e is an inter-component edge), there exists $w \in C$ and $x \notin C$ such that we can decompose π into

1. a path π_1 in T between u and w
2. an edge e' in T between w and x
3. a path π_2 in T between x and v.

Since e' is an inter-component edge, we infer that $e' \notin T_0$, and that e' cannot have lower weight than e. If we define T' by

$$T' = (T \setminus \{e'\}) \cup \{e\}$$

it will thus be the case that the weight of T' cannot be higher than the weight of T, and also that $T_0 \cup \{e\} \subseteq T'$. This will yield our claim, as we shall now show that T' is a spanning tree:

- Since (V, T) is connected, between any nodes in V there exists a path in T. But that path can be converted to a path in T', by replacing any occurrence of e' (between w and x) by a path formed by π_1 (between w and u), e (between u and v), and π_2 (between v and x). Hence also (V, T') is connected.
- Since (V, T) is a tree, it must contain $|V| - 1$ edges. But T' also contains $|V| - 1$ edges. Since we just saw that (V, T') is connected, this is possible only if T' is also acyclic.

□