

Recitation 5

Minimum Spanning Trees

Rebecca Lin | Friday, March 7th, 2025

How's everyone doing?

Agenda

MST Overview

Prim's Algorithm

Kruskal's Algorithm

MST Properties

MST Practice Problem

Overview

Definitions:

- A **spanning tree** T of a graph $G = (V, E)$ is an acyclic subset of G 's edges that connects all vertices of G
- T is a **minimum spanning tree** of a connected weighted graph $G = (V, E, w)$ if its weight $w(T) = \sum_{e \in T} w(e)$ is minimized

Overview

General Procedure:

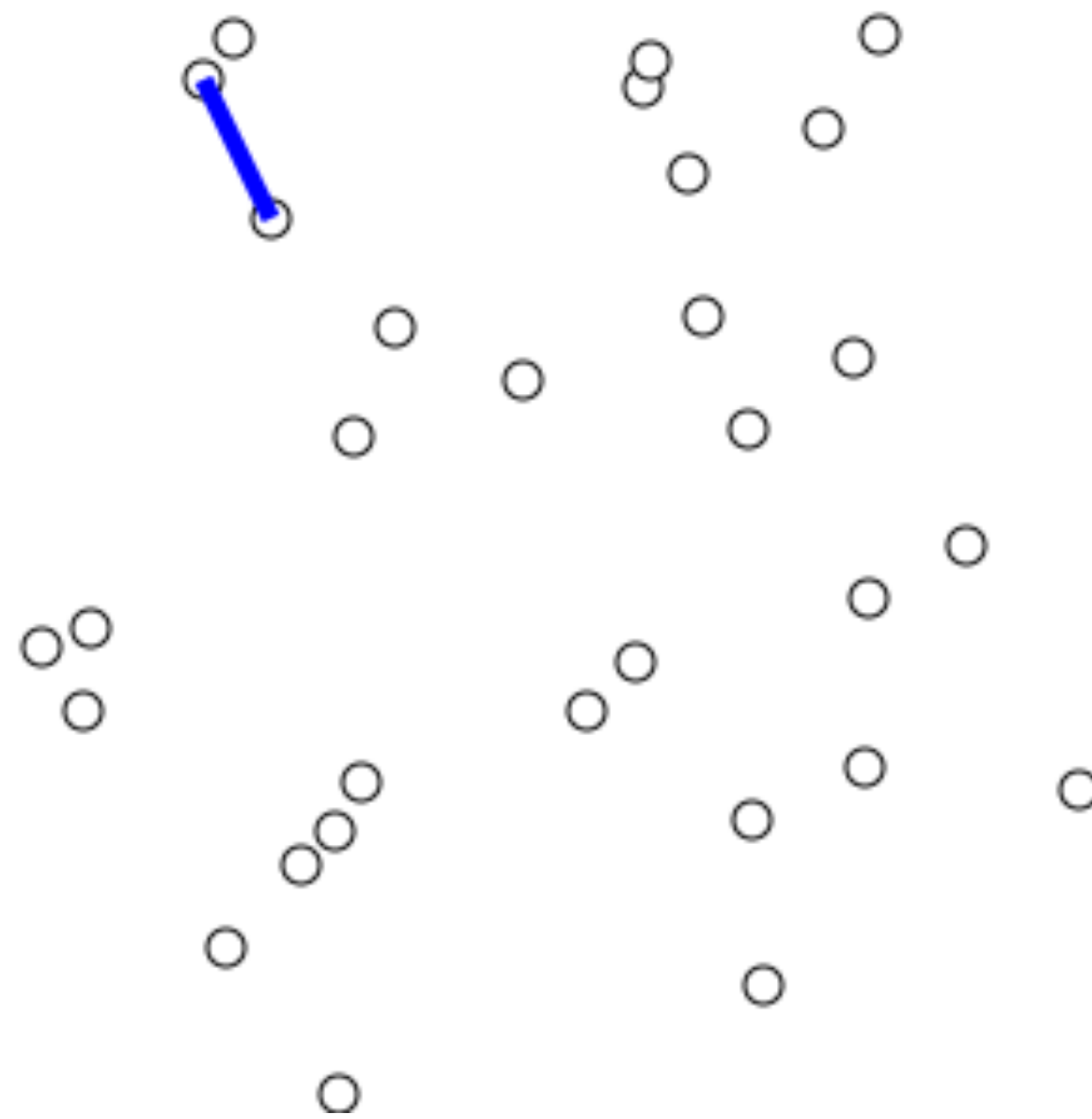
1. $T \leftarrow \emptyset$
2. **while** T is not spanning **do**
 Add a “safe” edge e to T such that T remains a subset of some MST
3. **return** T

How do we find safe edges?

Prim's Algorithm

Overview

Idea: Greedily select the cheapest edge that connects the tree to a new vertex.



Prim's algorithm applied to points in a
Euclidean plane

Prim's Algorithm

Algorithm

Procedure:

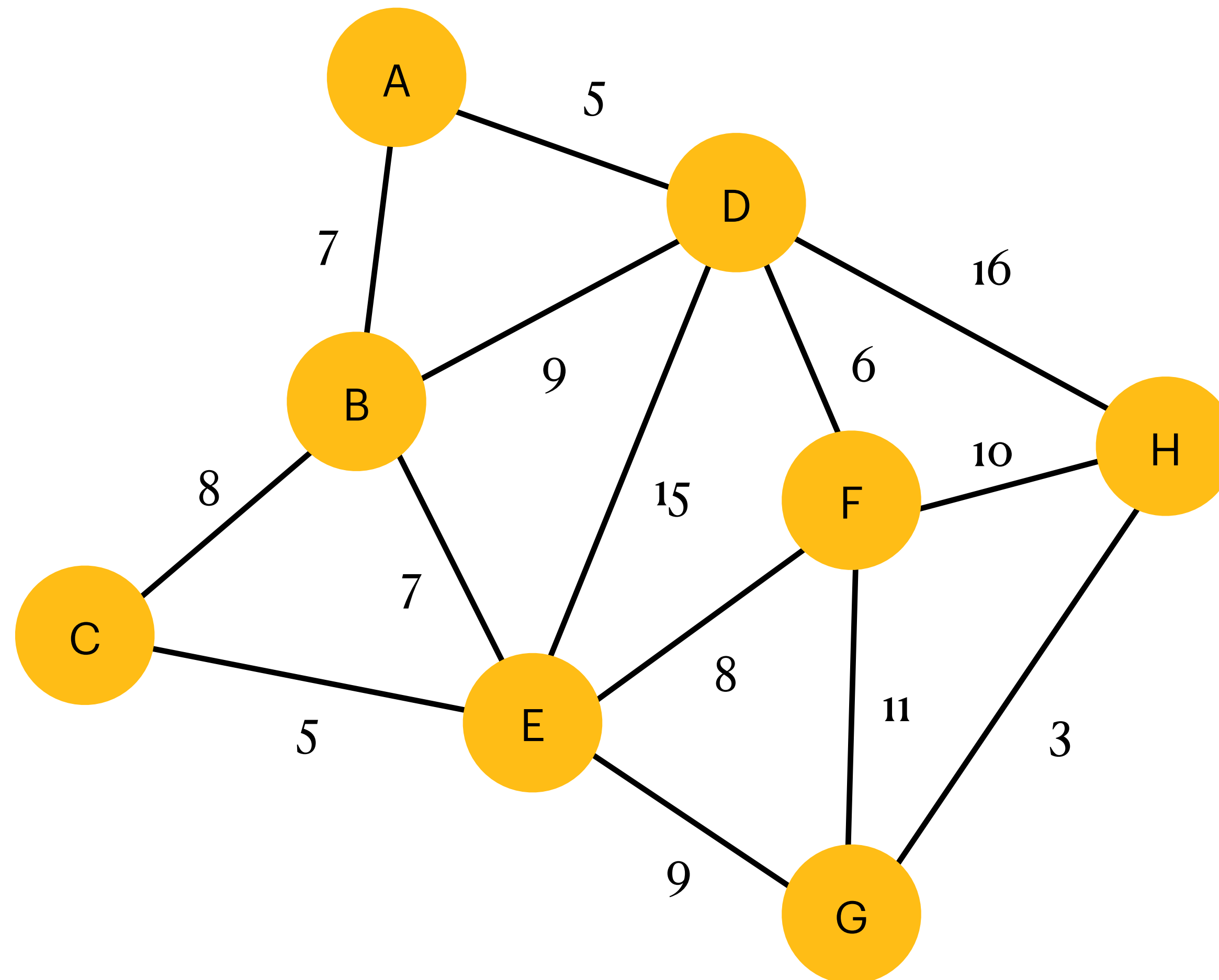
1. Choose an arbitrary starting point $s \in V$
2. $T_V \leftarrow \{s\}$
3. $T_E \leftarrow \emptyset$
4. **while** T_E is not spanning **do**
 - A. Find the lowest-weight edge $e = (u, v)$ such that $u \in T_V$ and $v \notin T_V$
 - B. $T_V \leftarrow T_V \cup \{v\}$
 - C. $T_E \leftarrow T_E \cup \{e\}$
5. **return** T_E

Prim's Algorithm

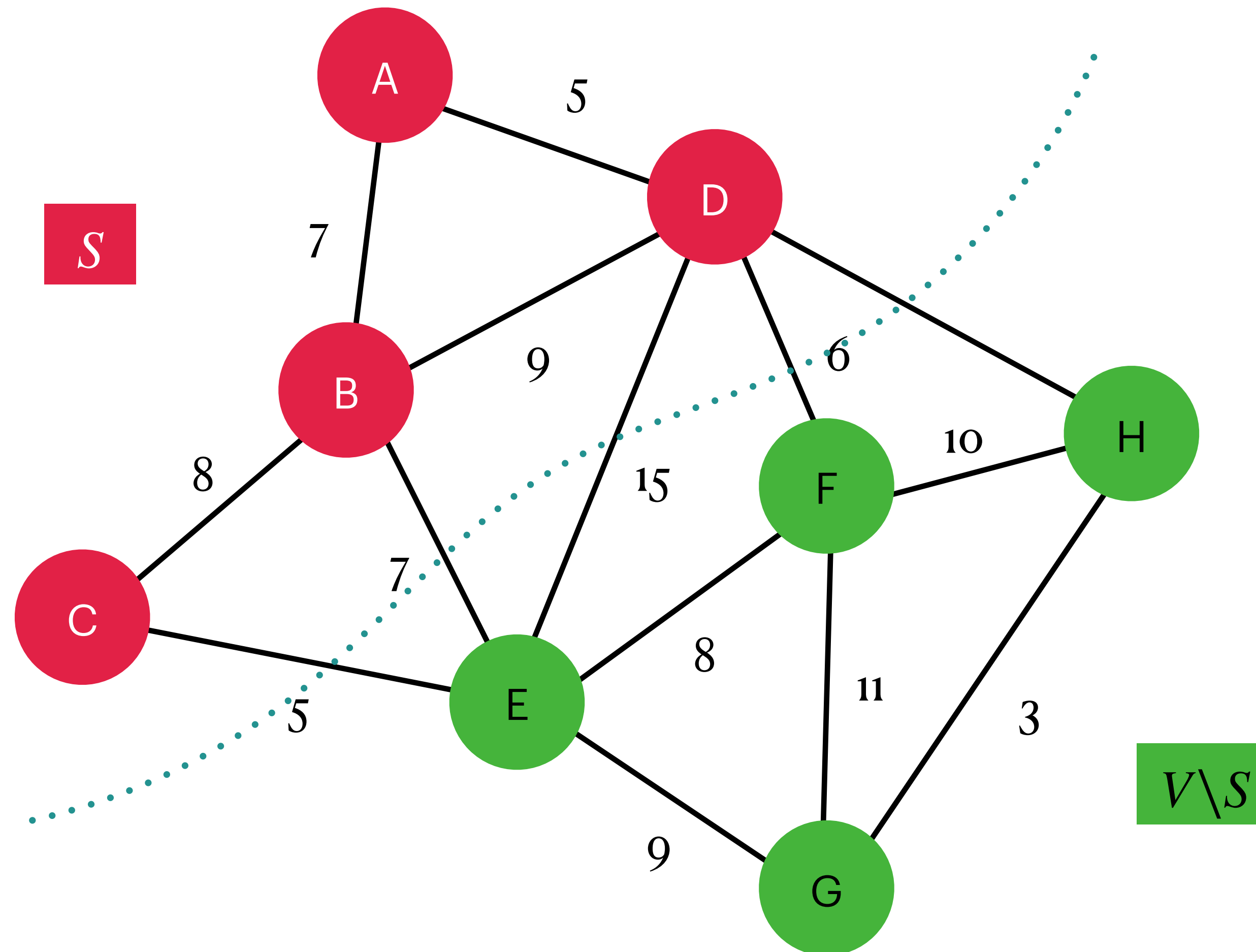
Correctness

Claim: Prim's algorithm always adds safe edges.

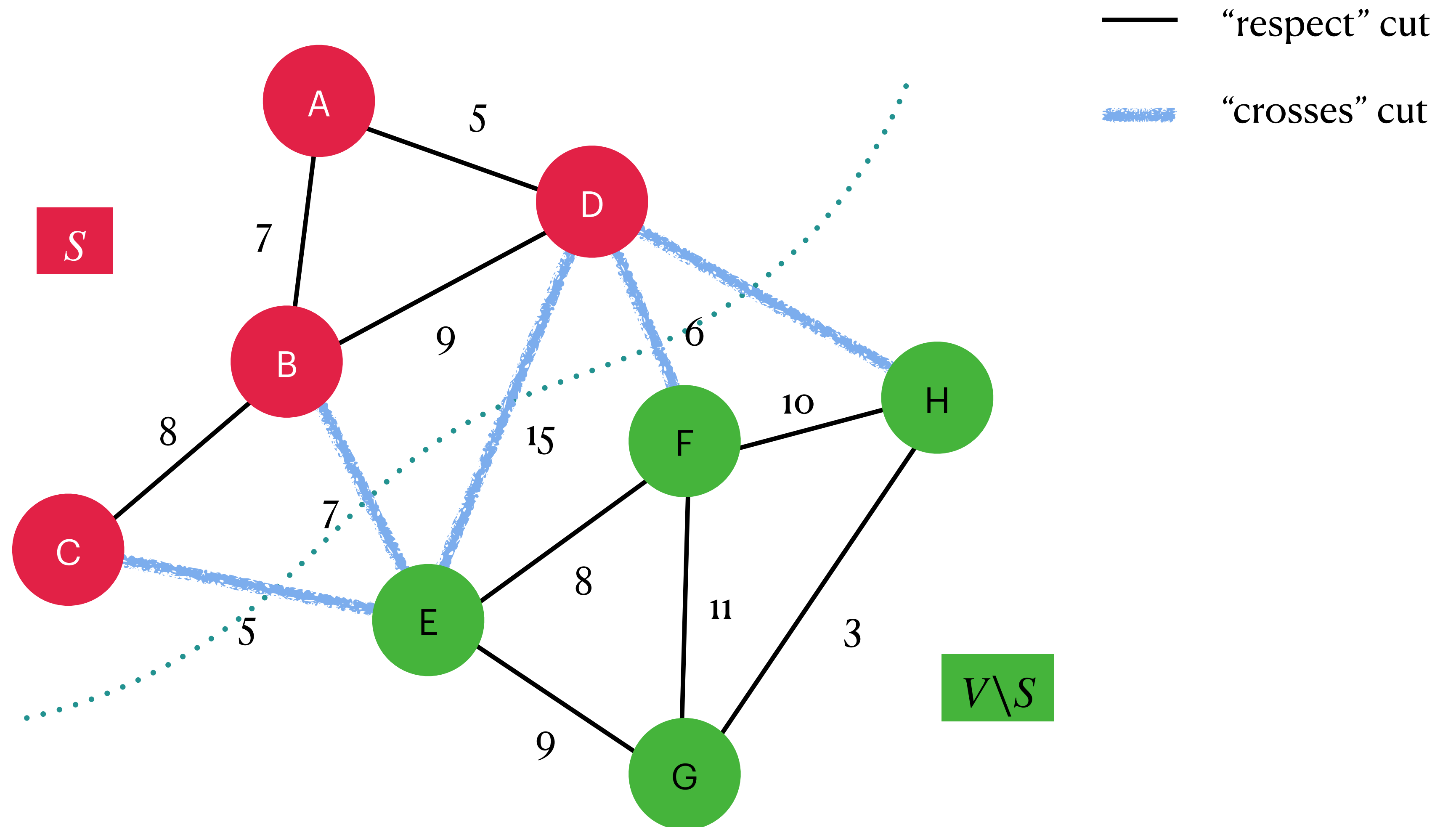
Aside: Review of Cuts



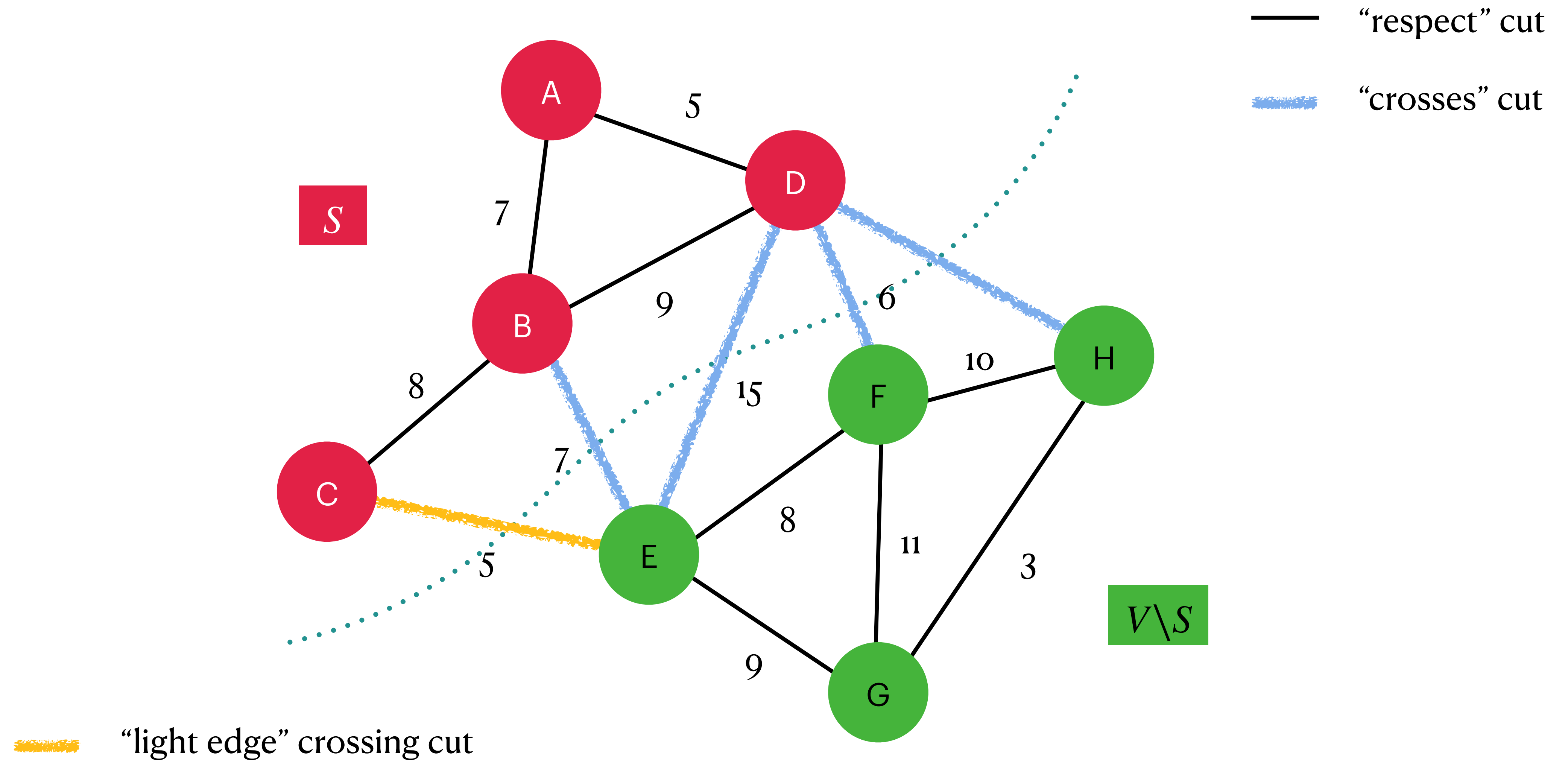
Aside: Review of Cuts



Aside: Review of Cuts



Aside: Review of Cuts



Prim's Algorithm

Correctness

Claim: Prim's algorithm always adds safe edges.

We use **CLRS₄ Theorem 31.1**: Given

1. a connected weighted graph $G = (V, E, w)$;
2. a set of edges $A \subseteq E$ belonging to some MST for G ;
3. a cut $(S, V \setminus S)$ of G that respects A ; and
4. a light edge e crossing the cut,

then e is a safe edge for A .

Prim's Algorithm

Correctness

Consider iteration k :

- Take $A = T_E$ and the cut to be $(T_V, V \setminus T_V)$
- Let $e_k = (u, v)$ be the cheapest edge such that $u \in T_V$ and $v \notin T_V$

We apply **CLRS4 Theorem 31.1**:

1. G is given as a connected weighted graph
2. Assuming only the addition of safe edges to T_E in past iterations, T_E included in some MST for G
3. Cut $(T_V, V \setminus T_V)$ respects T_E because all edges in T_E are between vertices in T_V
4. By definition, e_k is a light edge crossing the cut

Hence e_k is a safe edge.

Prim's Algorithm

Runtime Analysis

Procedure:

1. Choose an arbitrary starting point $s \in V$

2. $T_V \leftarrow \{s\}$

3. $T_E \leftarrow \emptyset$

4. **while** T_E is not spanning **do** $O(|V|)$

A. Find the lowest-weight edge $e = (u, v)$ such that $u \in T_V$ and $v \notin T_V$ $O(|E|)$

B. $T_V \leftarrow T_V \cup \{v\}$

C. $T_E \leftarrow T_E \cup \{e\}$

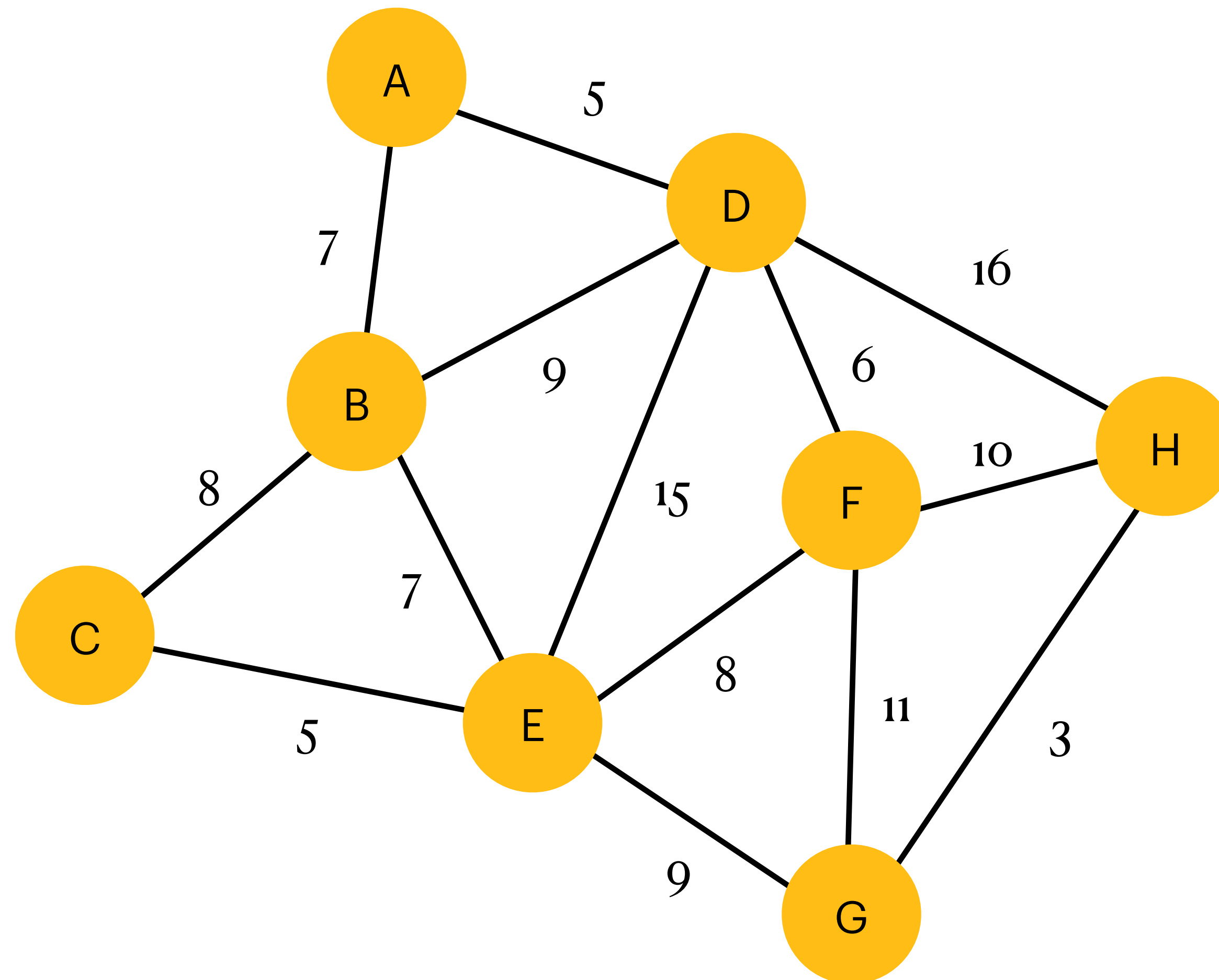
5. **return** T_E

What's the runtime?

Naively, it's $O(|V||E|)$ —how do we **improve** it?

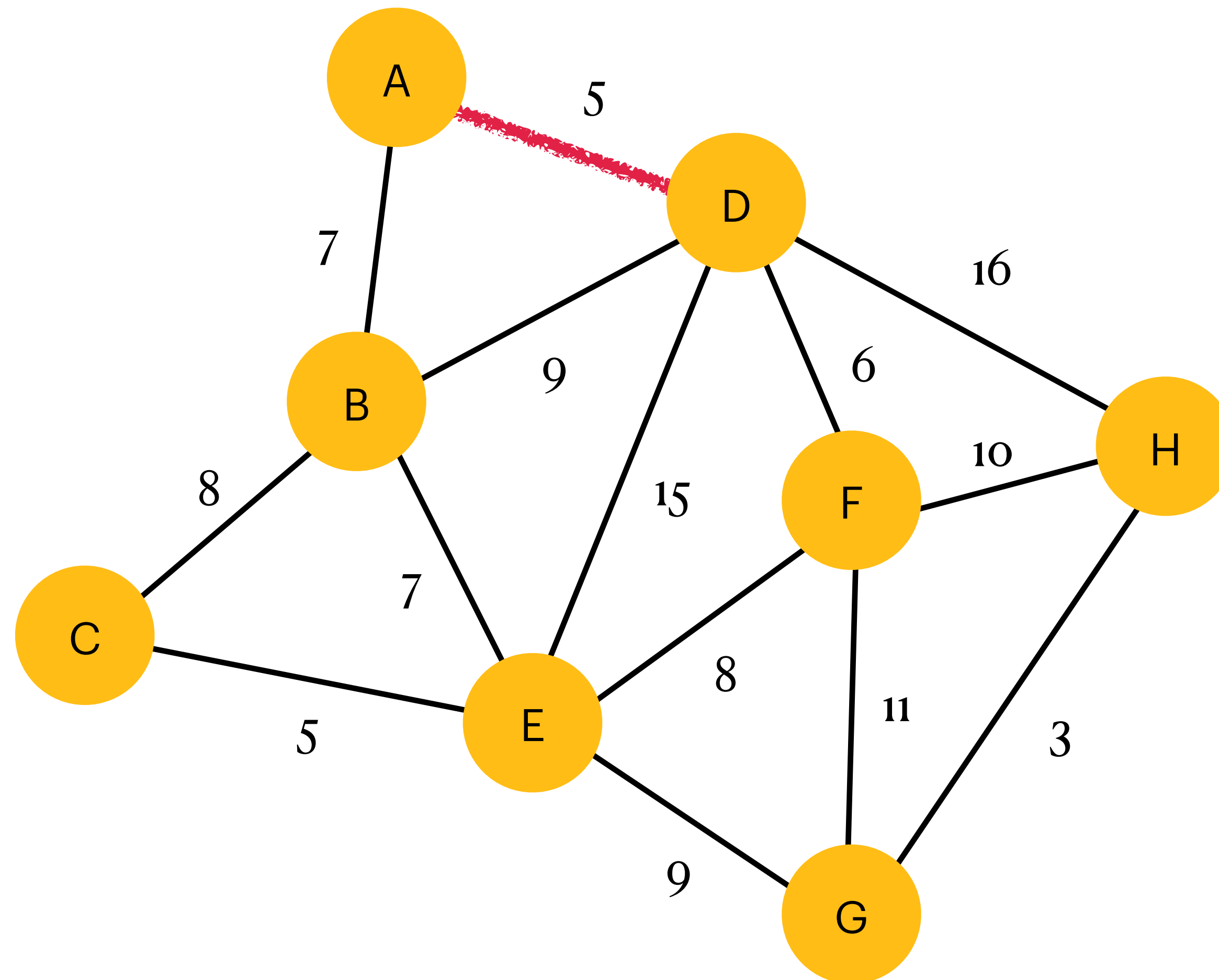
Prim's Algorithm

Example



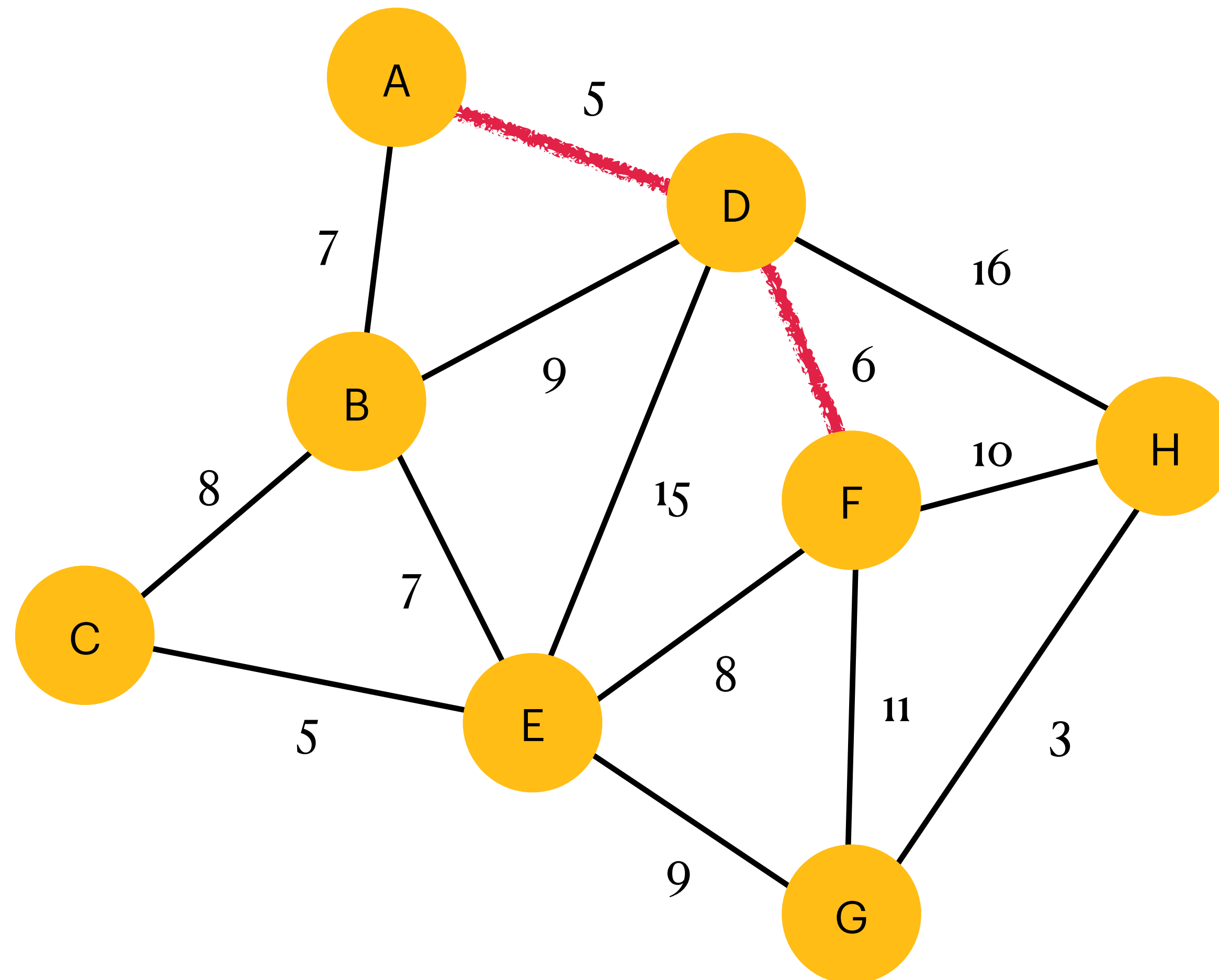
Prim's Algorithm

Example



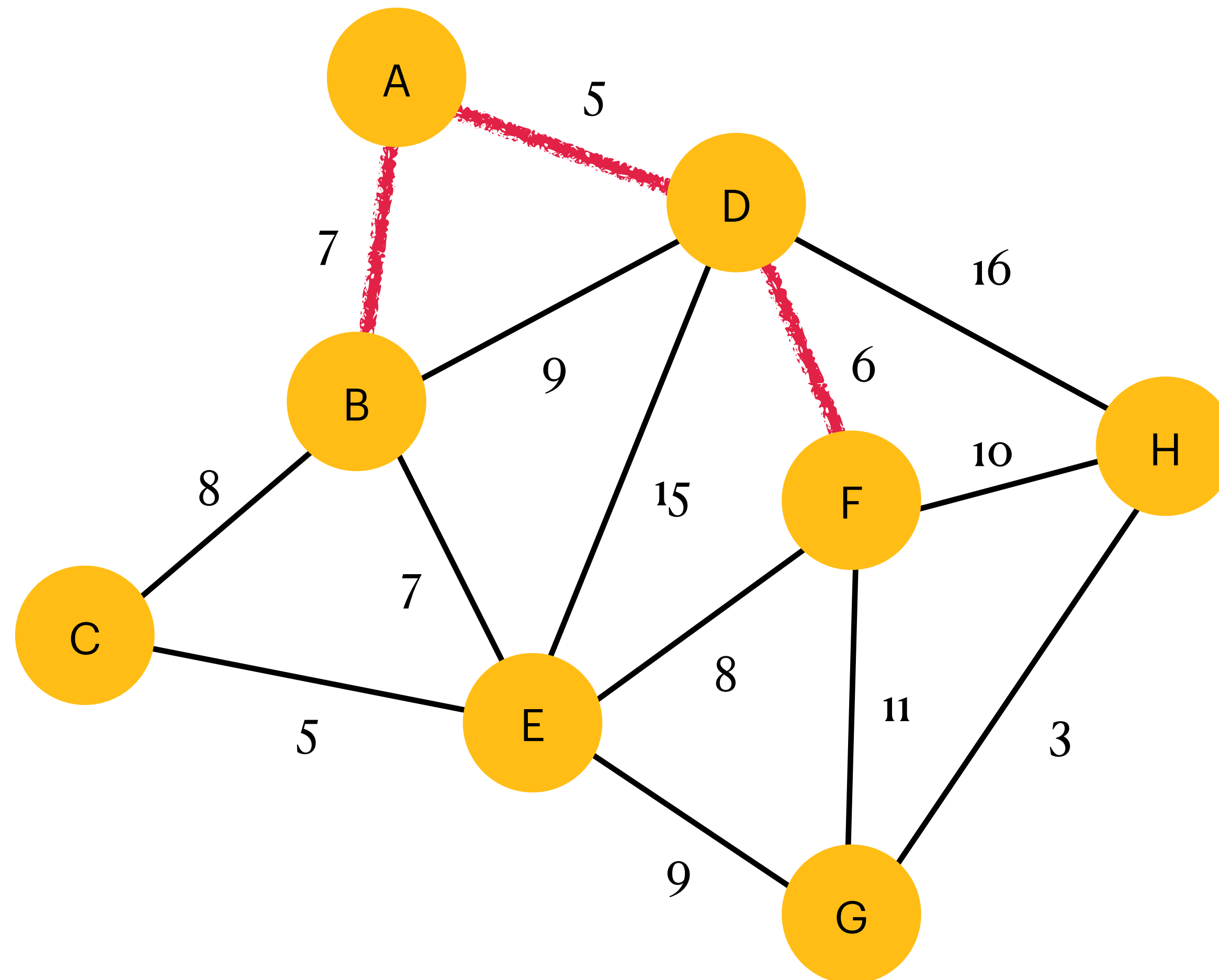
Prim's Algorithm

Example



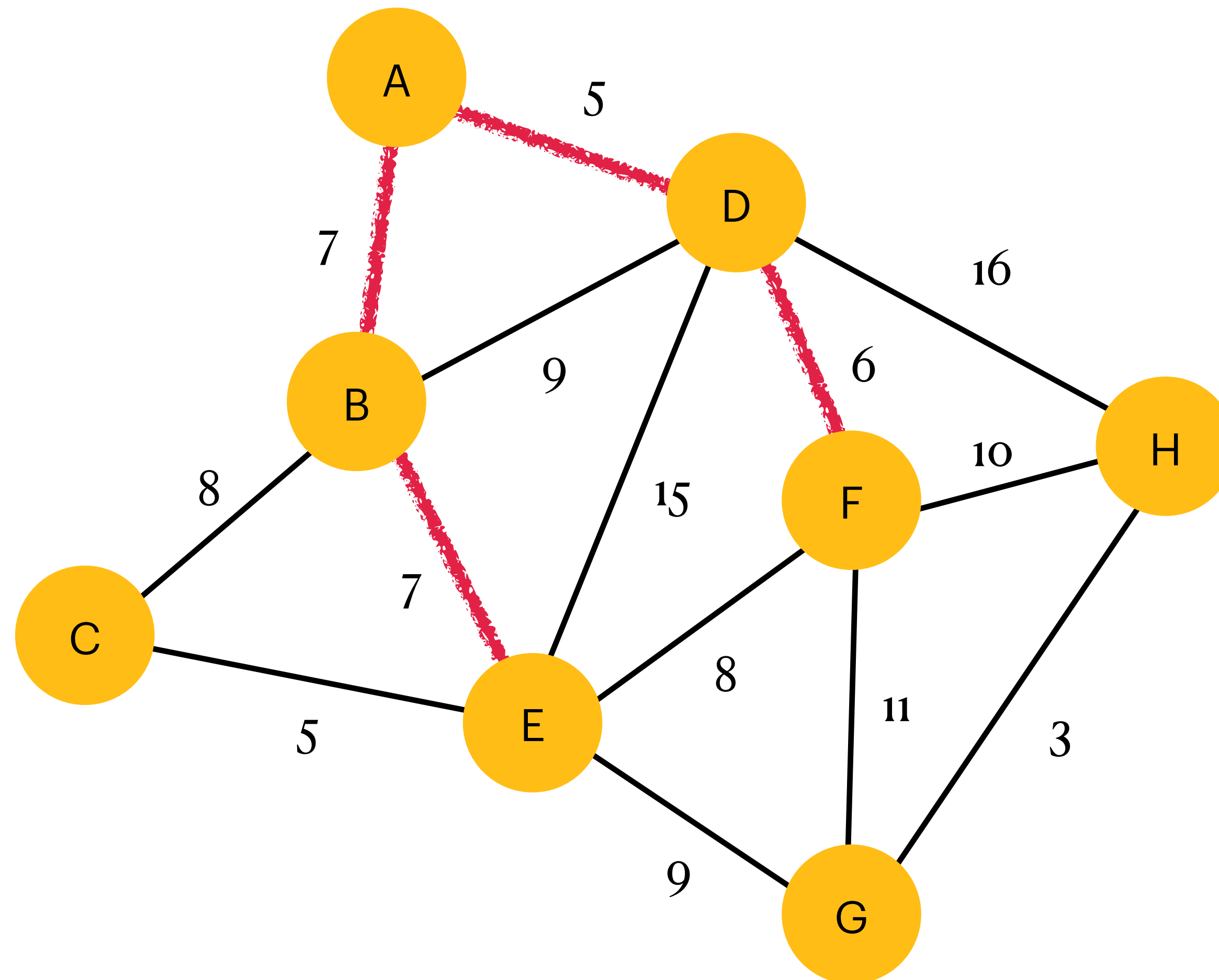
Prim's Algorithm

Example



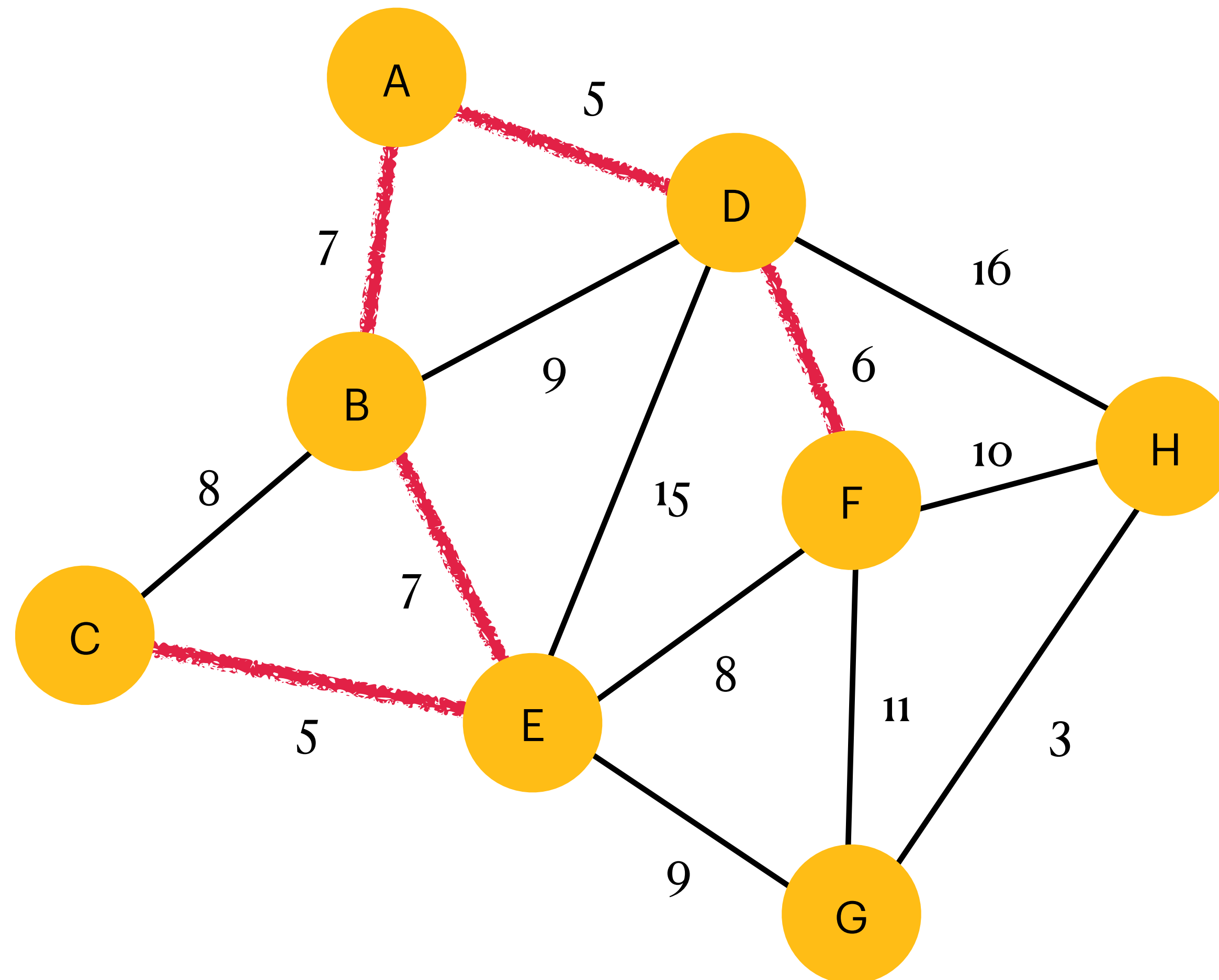
Prim's Algorithm

Example



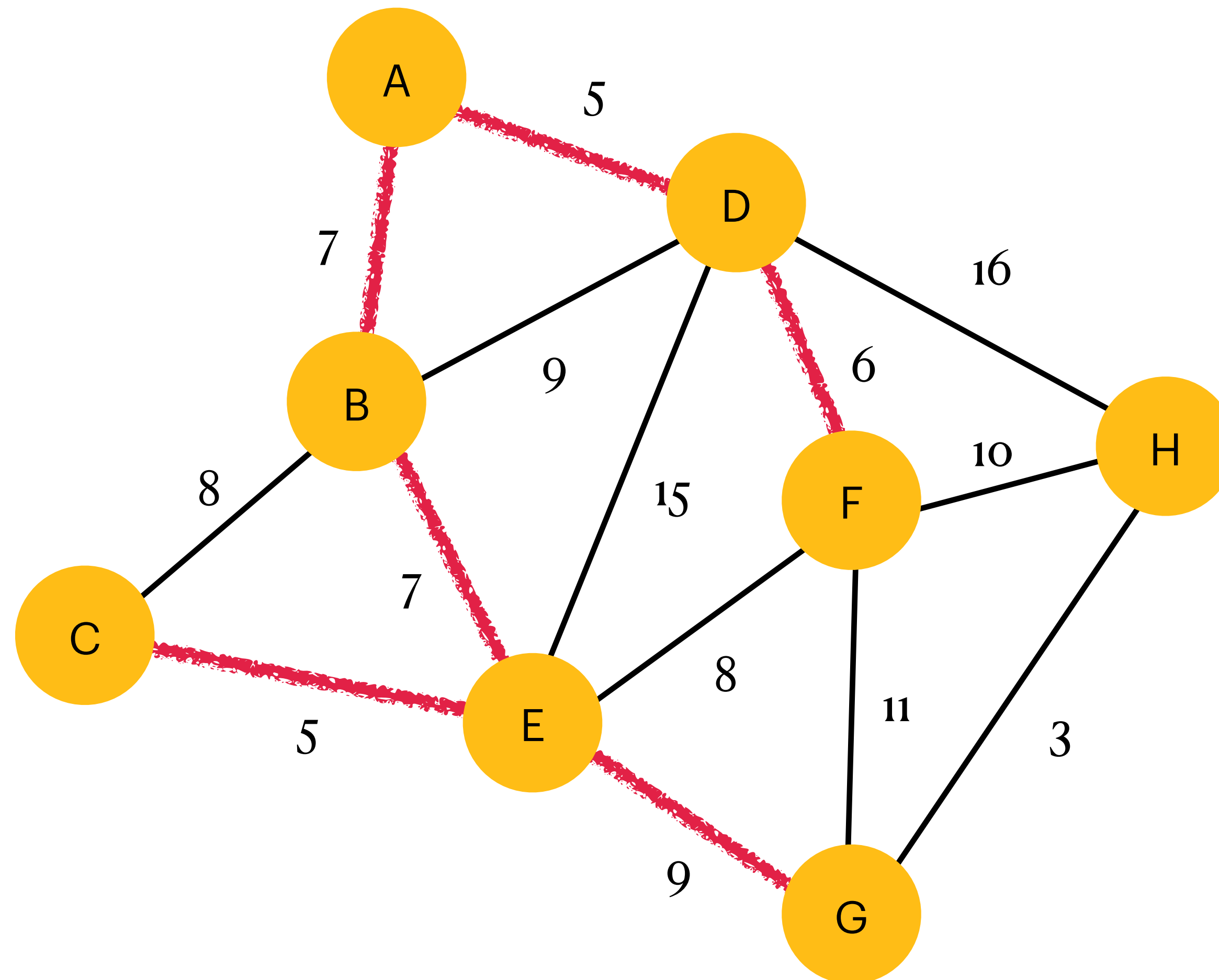
Prim's Algorithm

Example



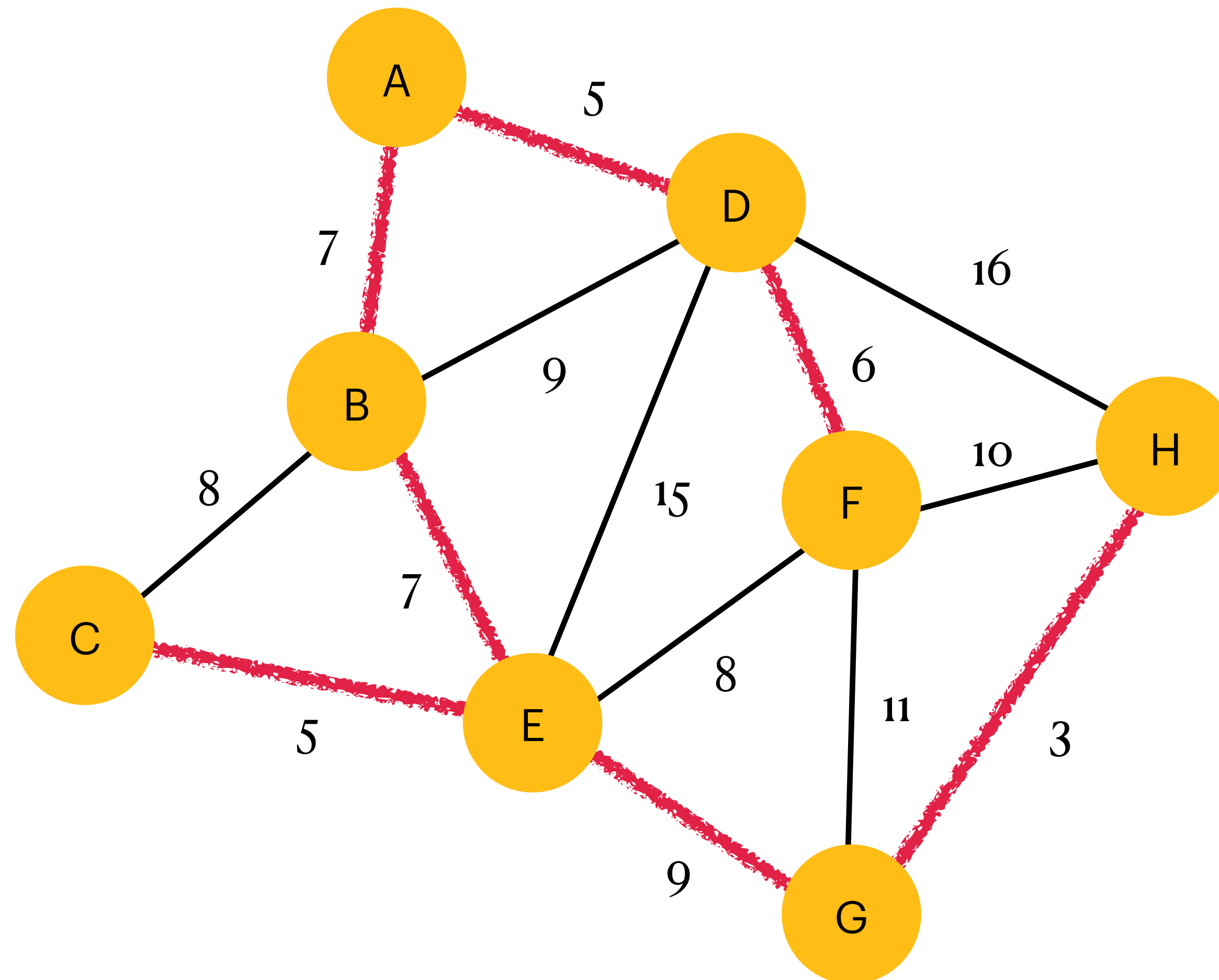
Prim's Algorithm

Example



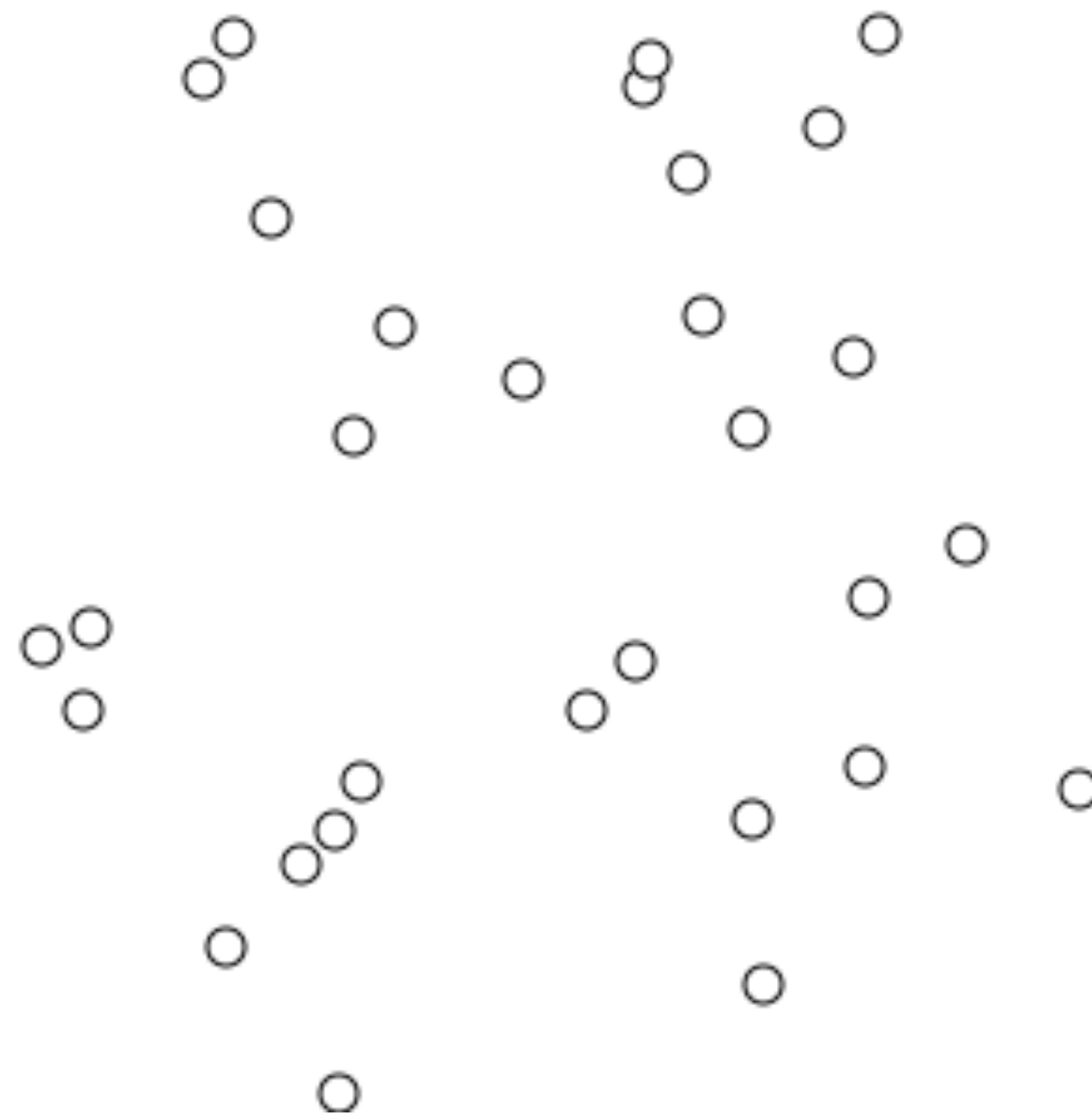
Prim's Algorithm

Example



Kruskal's Algorithm

Idea: Greedily add to the forest a lowest-weight edge that will not form a cycle.



Kruskal's algorithm applied to points in a
Euclidean plane.

Kruskal's Algorithm

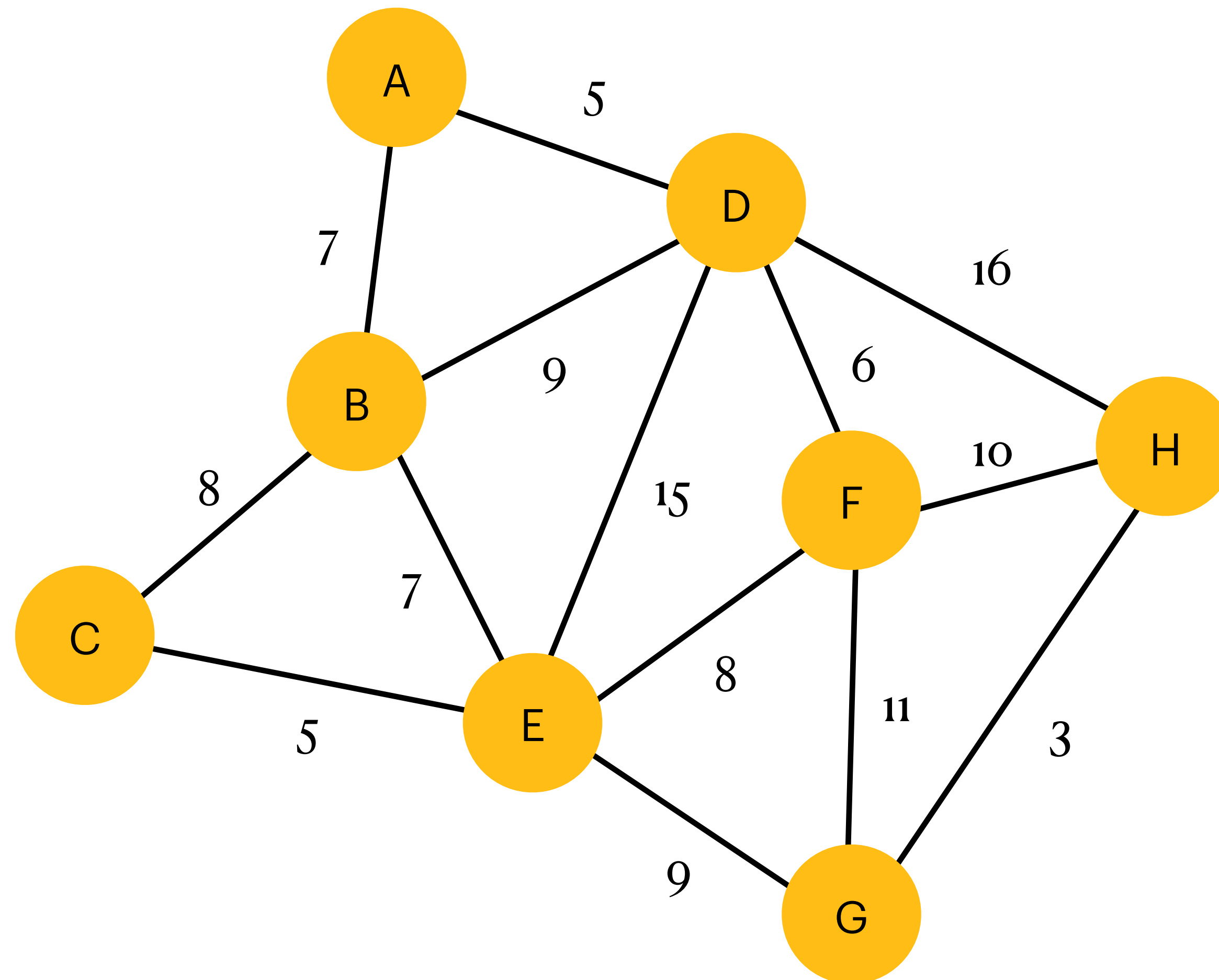
Algorithm

Procedure:

1. $T \leftarrow \emptyset$
2. **for** $v \in V$ **do**
 MAKE-SET(v)
3. **for** $e = (u, v)$ in E , sorted by weight in non-decreasing order **do**
 if FIND-SET(u) $\neq$ FIND-SET(v) **then**
 $T \leftarrow T \cup \{e\}$
 UNION(u, v)
4. **return** T

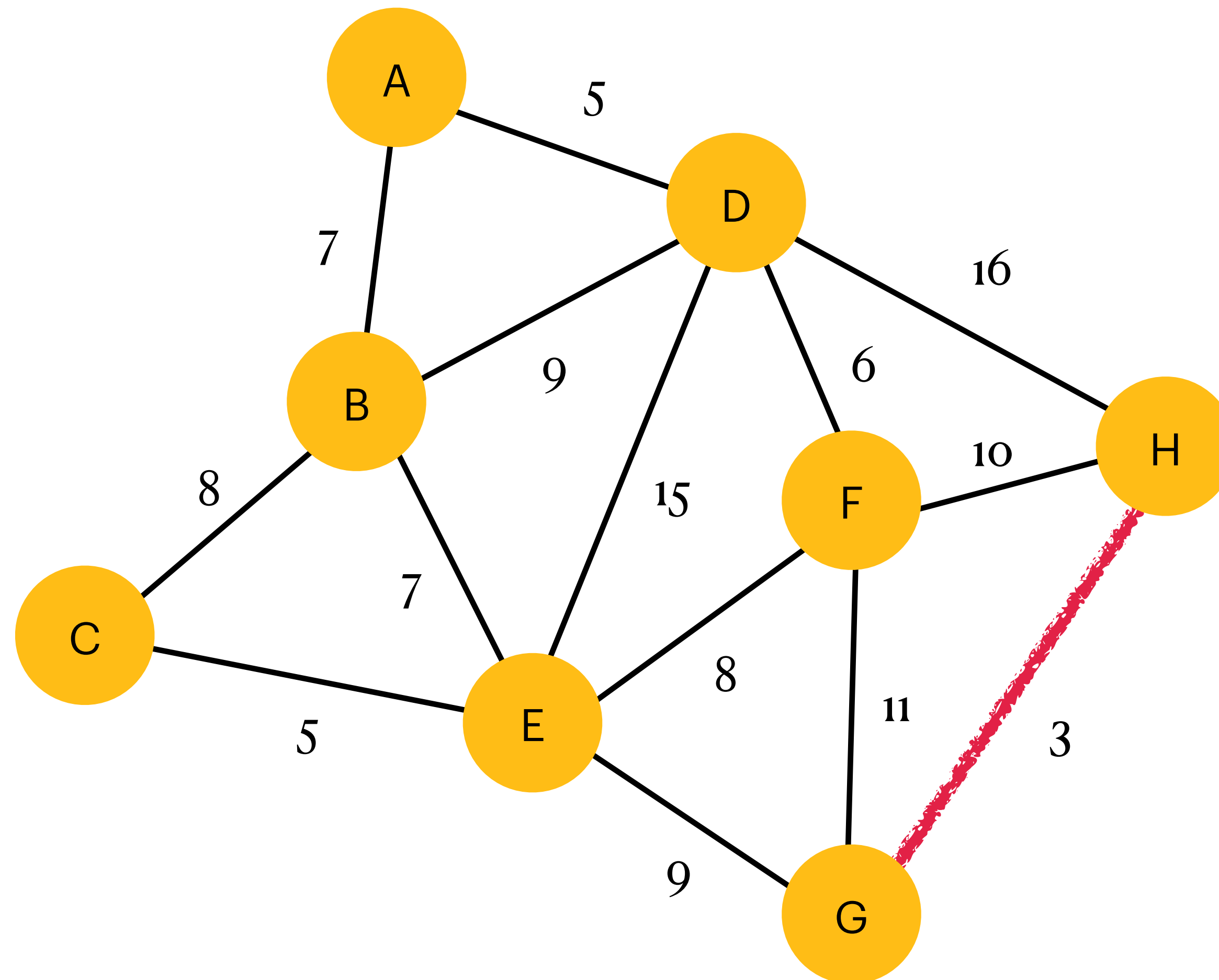
Kruskal's Algorithm

Example



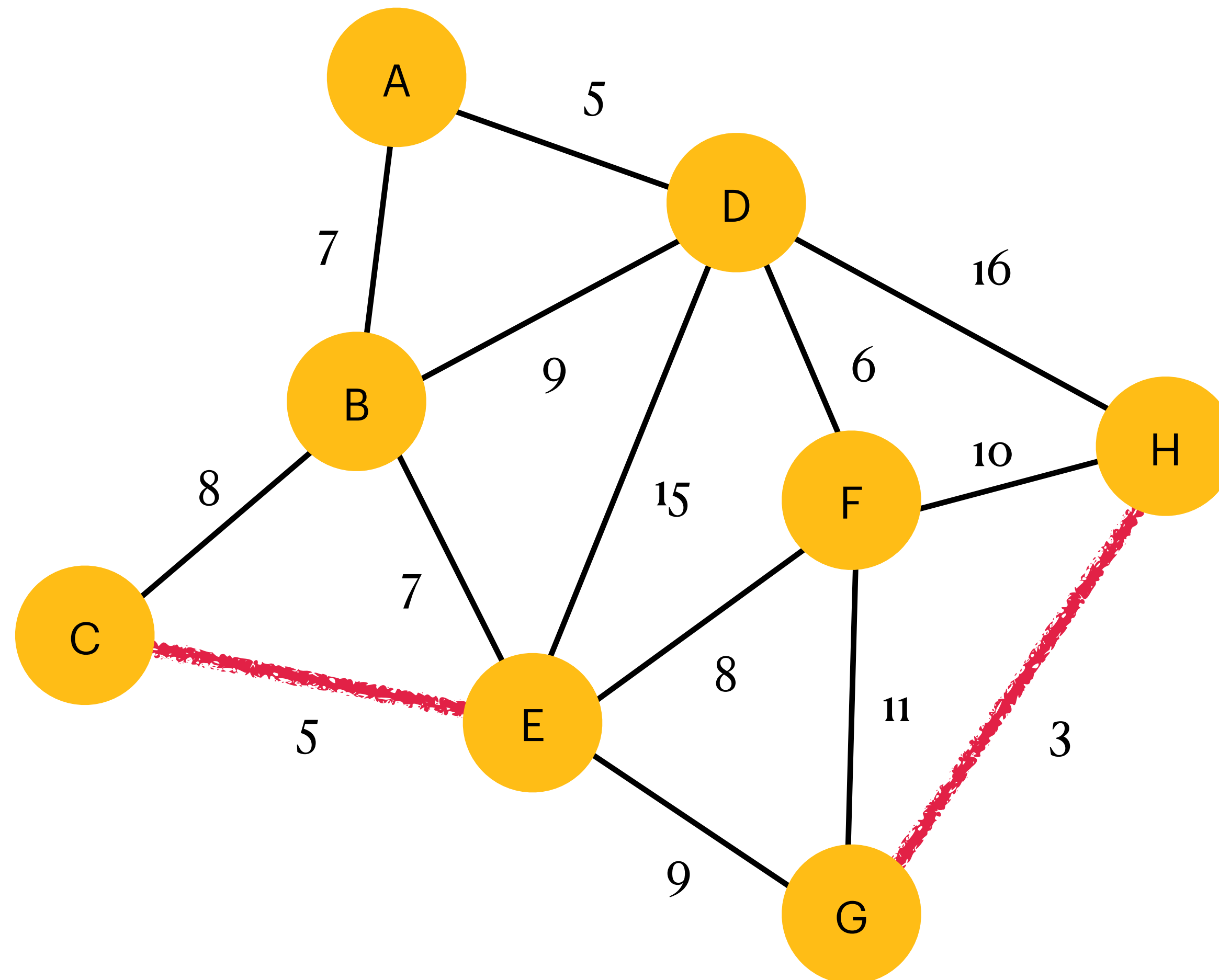
Kruskal's Algorithm

Example



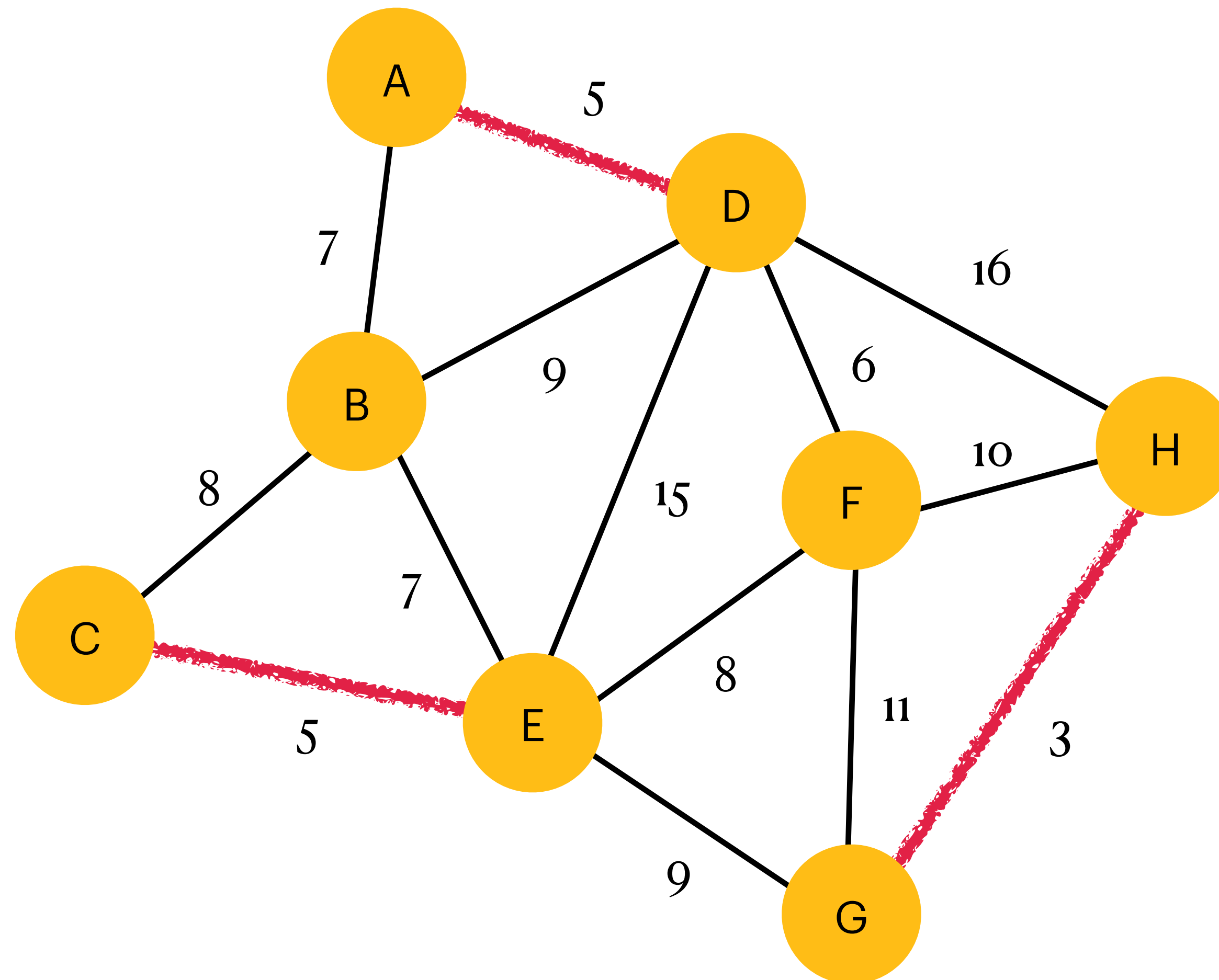
Kruskal's Algorithm

Example



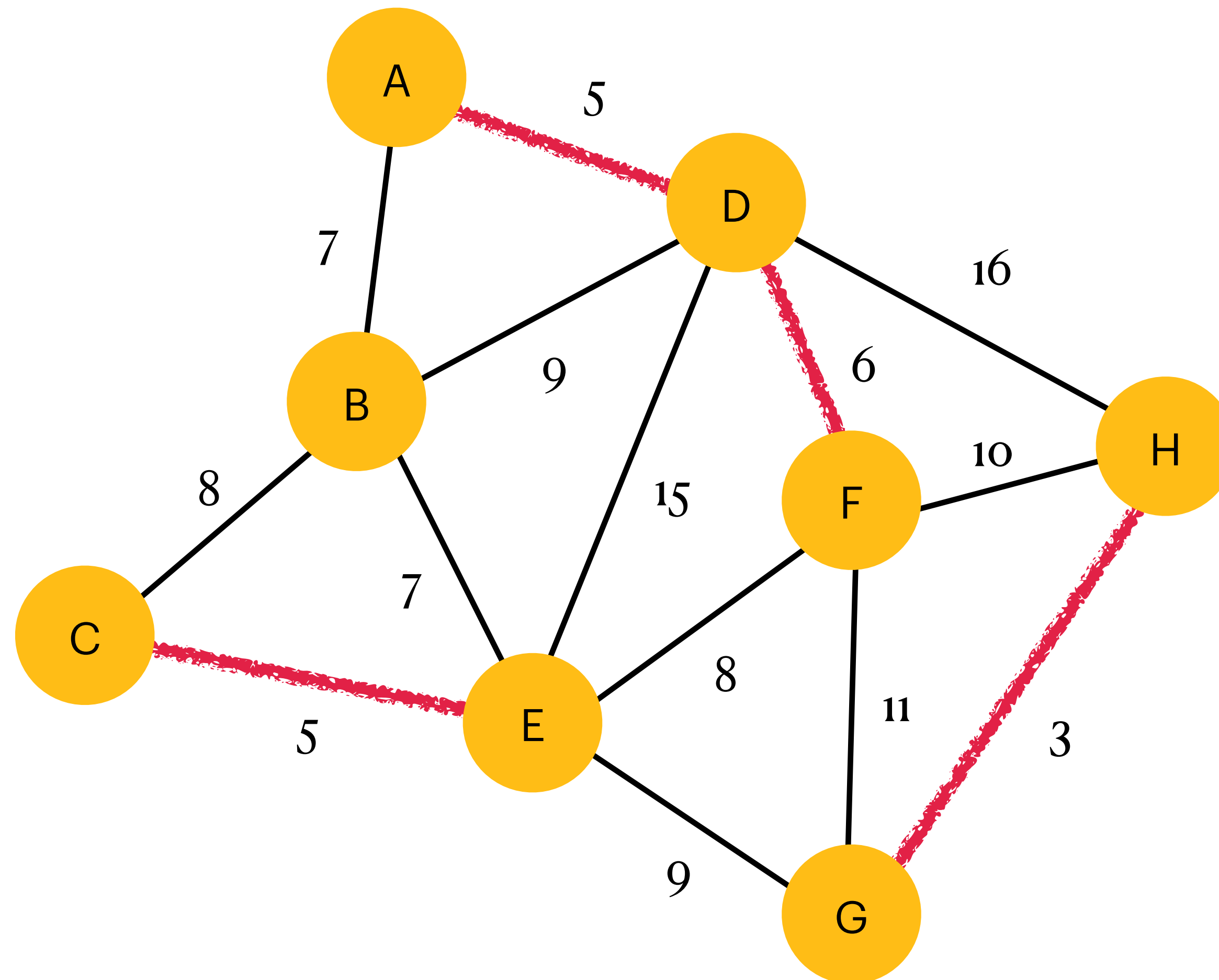
Kruskal's Algorithm

Example



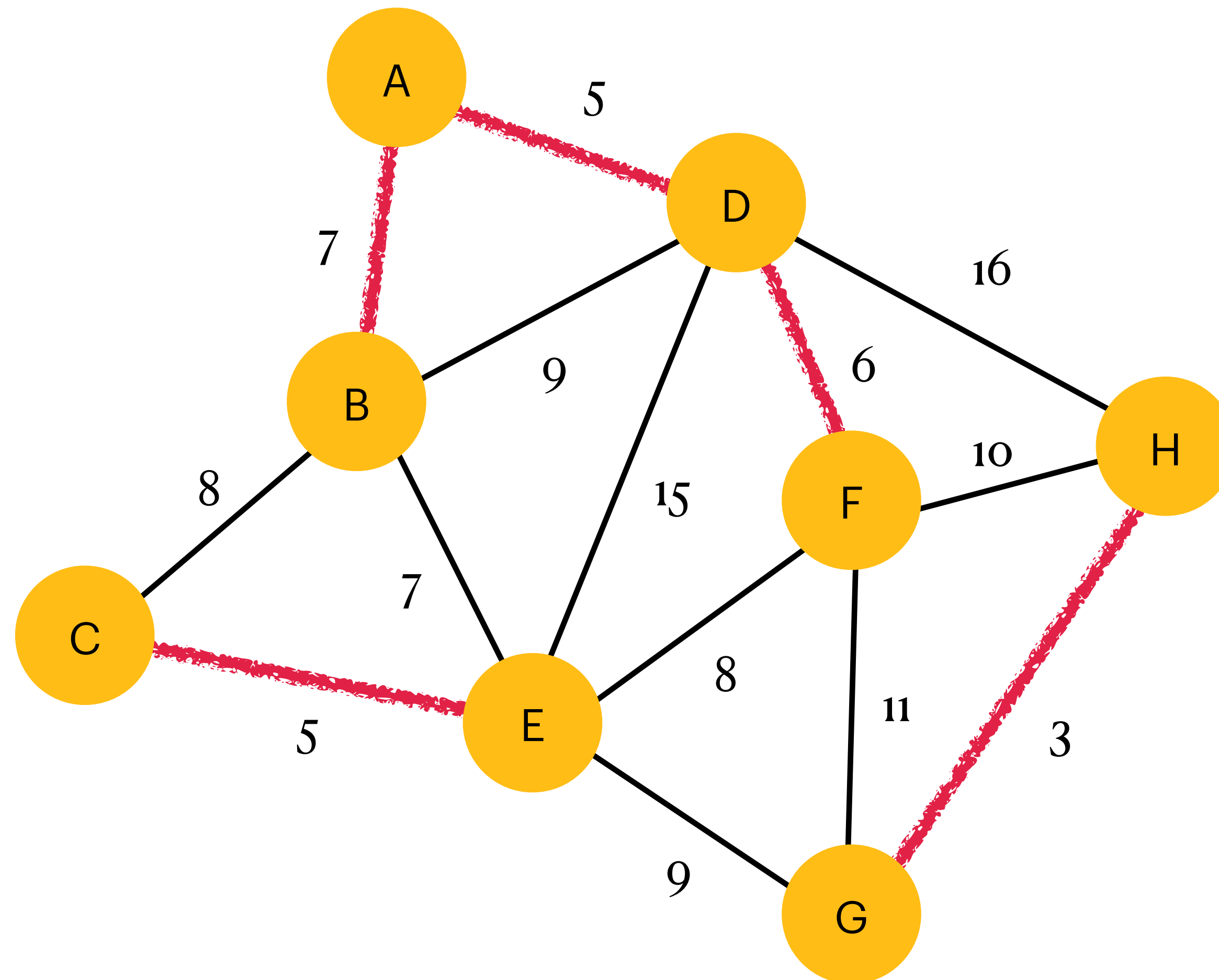
Kruskal's Algorithm

Example



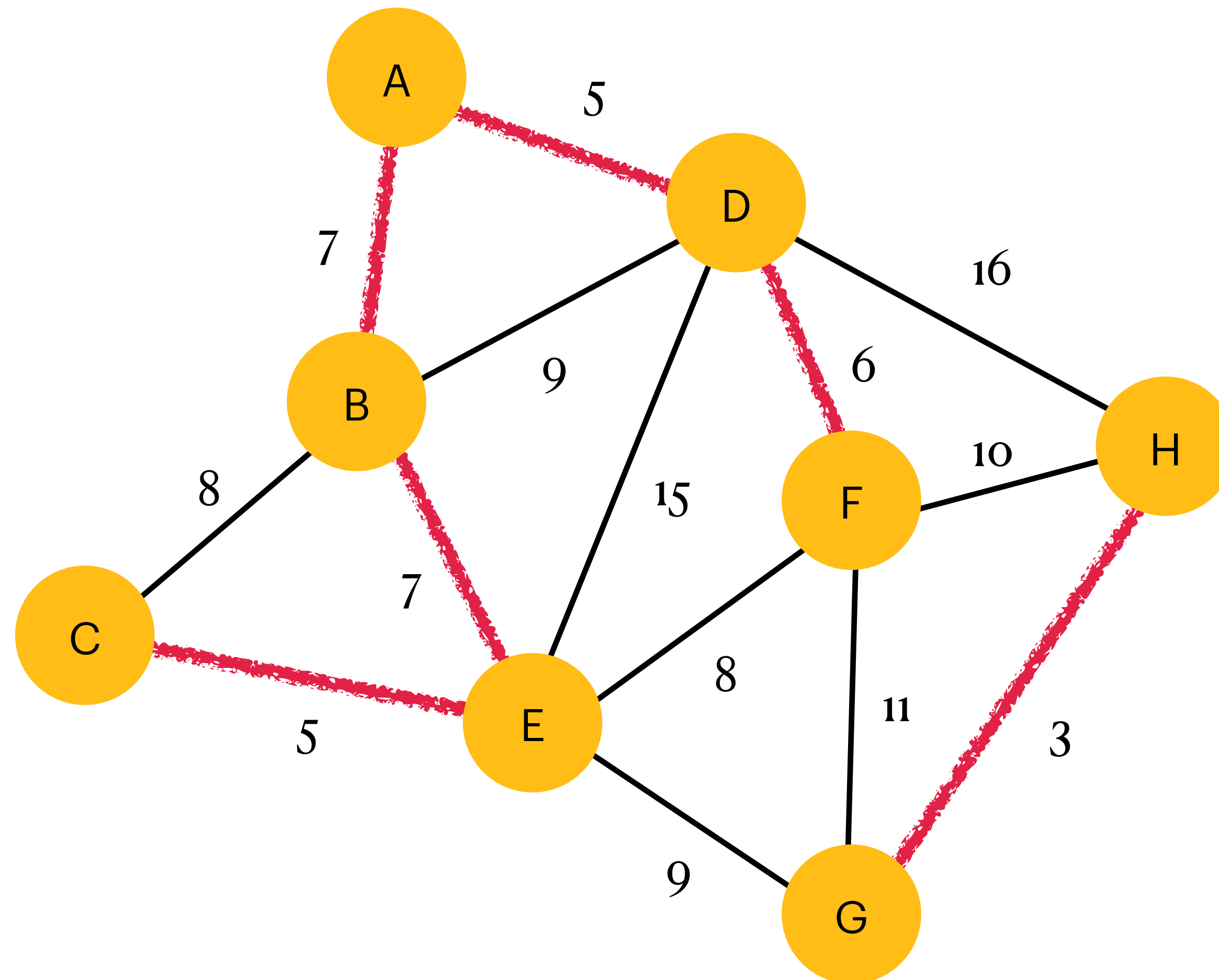
Kruskal's Algorithm

Example



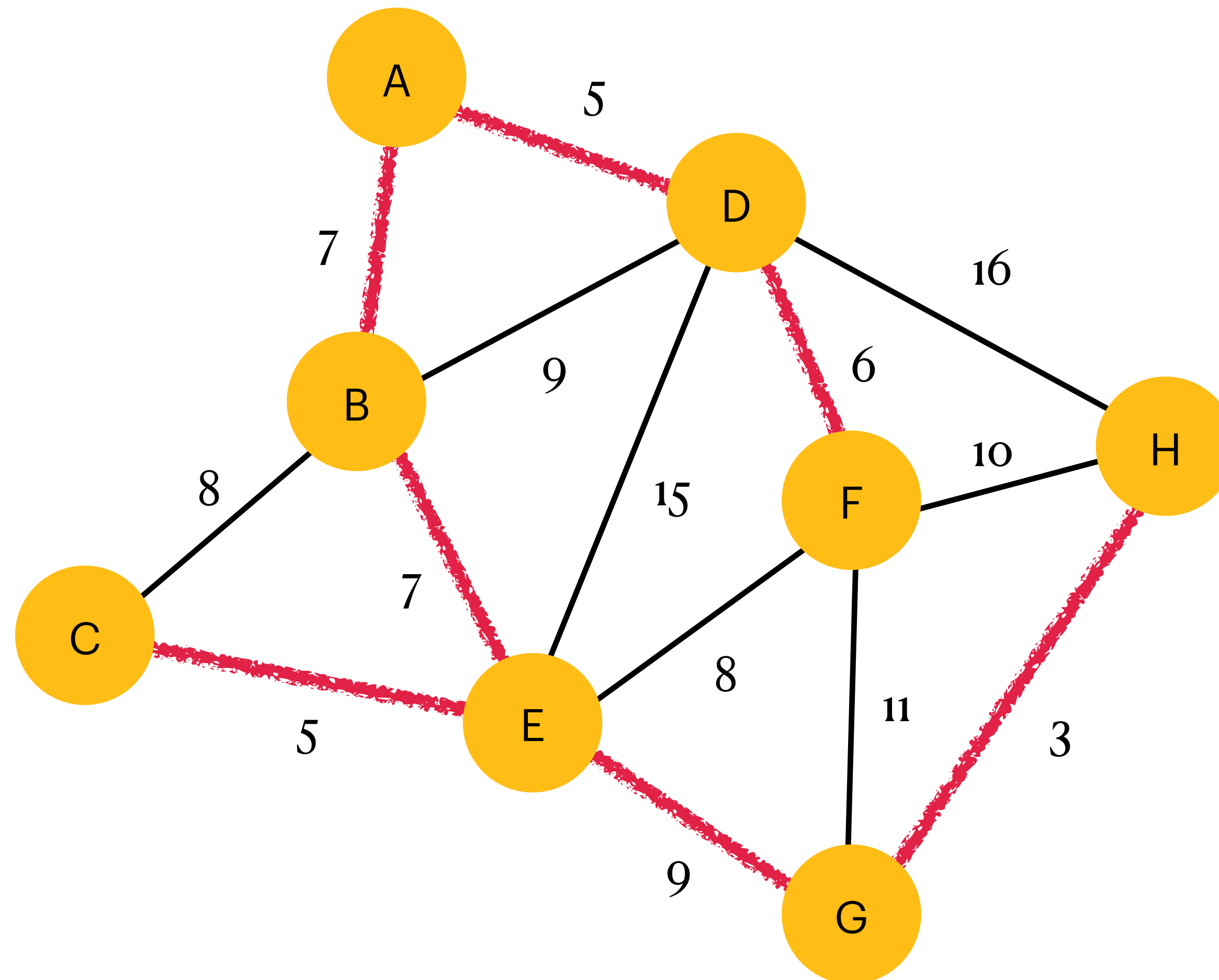
Kruskal's Algorithm

Example



Kruskal's Algorithm

Example



Properties of MSTs

Theorem [Cut Property]: Let S be some non-empty proper subset of the vertices. For the cut $(S, V \setminus S)$, any least-weight edge $e = (u, v)$ with $u \in S, v \notin S$ (i.e., a “crossing” edge) must belong to some MST.

We can greedily construct globally optimal solutions by making locally optimal choices.

Properties of MSTs

Theorem [Optimal Substructure Property]: If an edge e belongs to some MST T^* of G and T' is some MST of $G' = G \setminus e$, then $T = T' \cup \{e\}$ is an MST of G .

The optimal solution for the entire problem contains optimal solutions to subproblems.

Properties of MSTs

Theorem [Cycle Property]: Consider any cycle C in a graph G . If C contains an edge e whose weight is strictly larger than the weight of every other edge in C , then e cannot belong to any MST.

Properties of MSTs

Theorem [Uniqueness Property]: If a graph has distinct edge weights, then its MST is unique.

Practice Problem

What tunnels should I build?

Our subway is described as a graph $G = (V, E, c)$, where the stations are nodes and each possible tunnel between stations is an edge $e \in E$ with unique cost $c(e) > 0$. We find the cheapest way to connect all stations by computing the (unique) MST T of G .

Part A: What if a new edge $e \notin E$ with cost $c(e)$ is now added? Show that we can obtain the MST T' of the modified graph G' by adding e to T and then deleting the heaviest edge in the cycle created.

Proof. Two steps:

1. Only the edges in the set $T'' = T \cup \{e\}$ can be in T' . Why?
2. Removing the heaviest edge in the single cycle in T'' yields the MST. Why?

Part B: Say that the new network is now described by $G' = (V \cup \{v'\}, E \cup E', c)$, where v' is a new station, and E' is the set of possible tunnels between the new station and existing ones. We want to find the MST T' of G' given MST T of G in $O(|V| \log |V|)$ time.