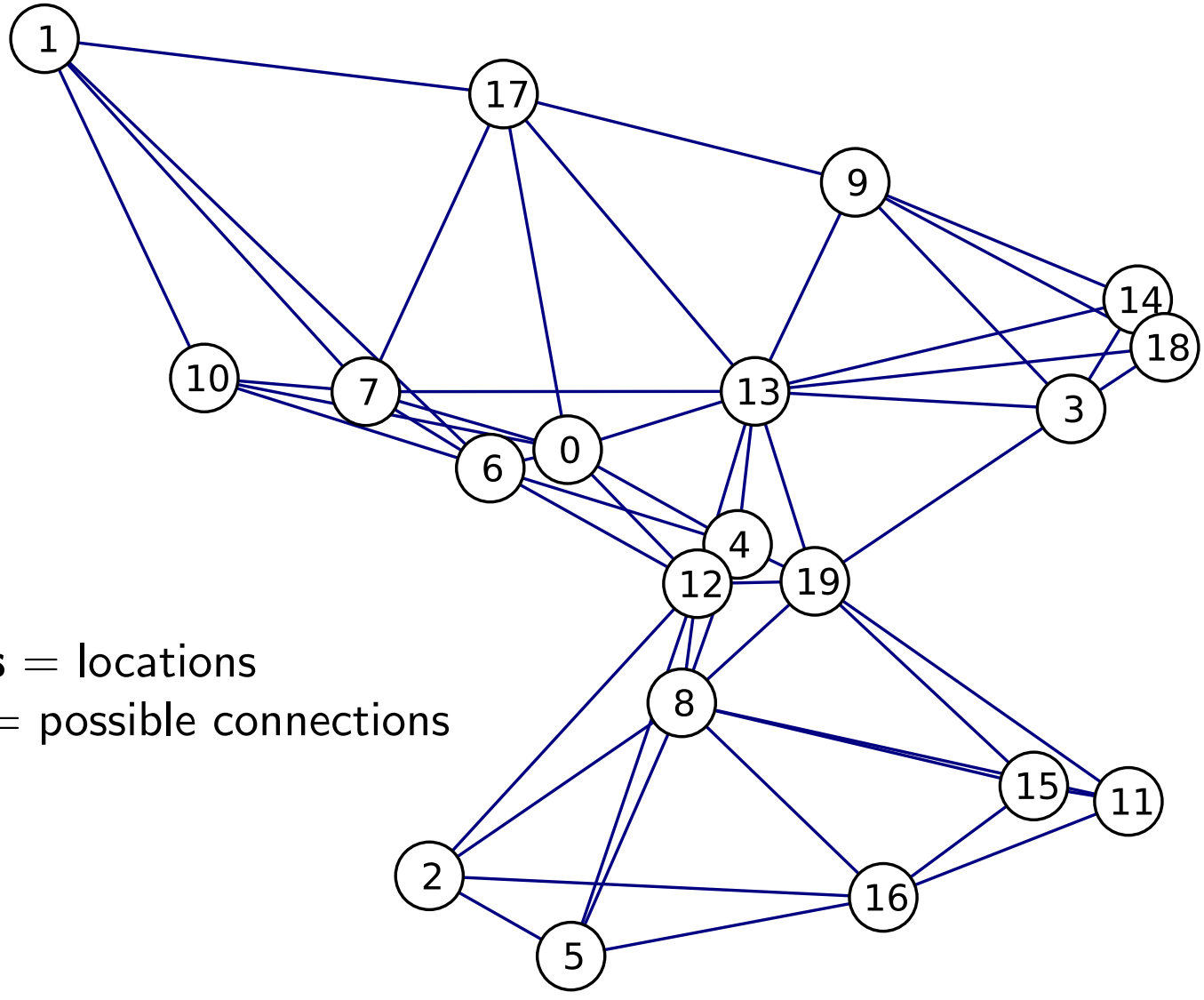
Minimum Spanning Trees

Slides by Carl Kingsford

Jan. 15, 2014

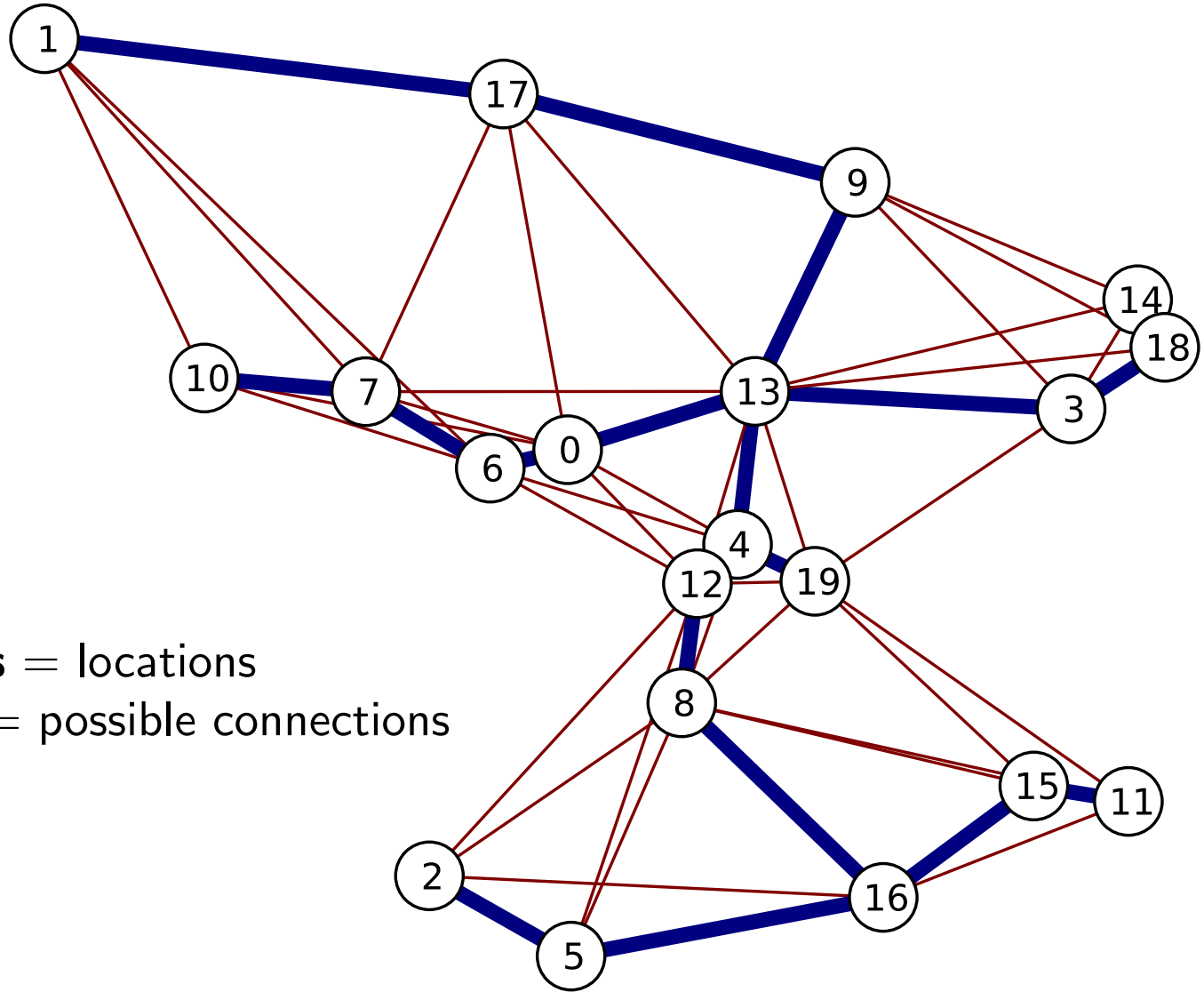
KT 3.1,4.5,4.7

Problem: Cost effective wiring of a network



circles = locations
lines = possible connections

Problem: Cost effective wiring of a network



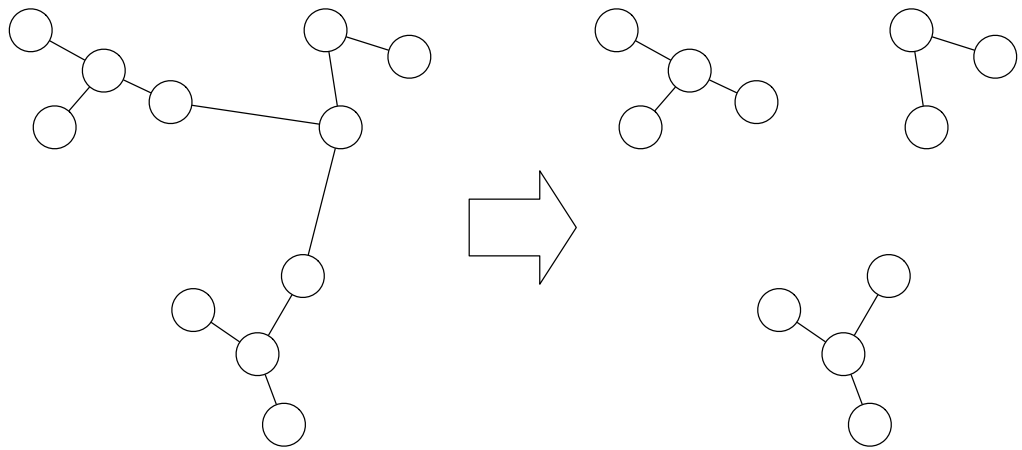
circles = locations
lines = possible connections

Other applications for the problem

1. **Circles** = DNA sequences
   Edge $u - v$ weighted by the similarity of sequences u and v.

   Find the most parsimonious way to explain how the sequences are evolutionarily related.

2. Remove long edges $\Rightarrow$ clustering:



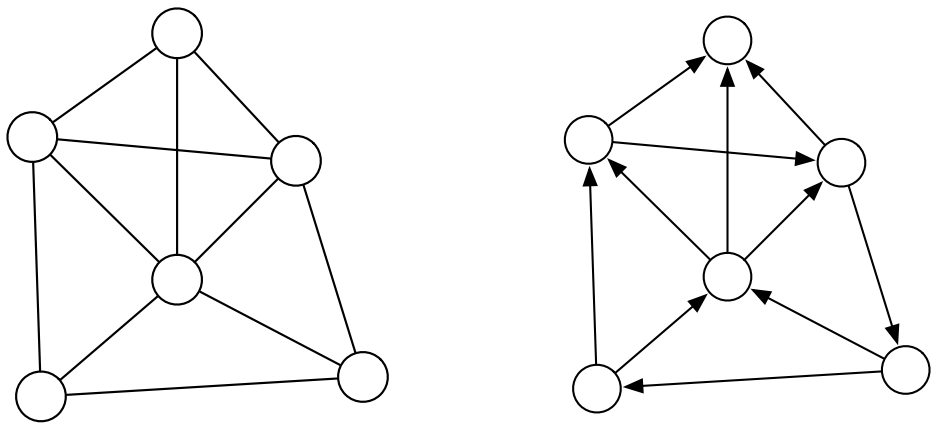
Graphs

Graphs specify pairwise relationships between objects.

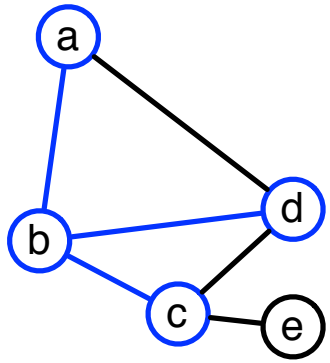
An undirected graph $G = (V, E)$ is a pair of sets:

- V is a set of nodes, aka vertices.
- E is a set of two-element subsets of V.
  An element of E is of the form: $e = \{u, v\}$ with $u, v \in V$.

A graph is directed if E is a set of ordered pairs (u, v), $u, v \in V$:



Subgraphs and components



$V_G = \{a,b,c,d,e\}$

$E_G = \{ \{a,b\}, \{a,d\}, \{b,d\}, \{b,c\}, \{c,d\}, \{c,e\} \}$

$V_H = \{a,b,c,e\}$

$E_H = \{ \{a,b\}, \{b,d\}, \{b,c\} \}$

A graph $H = (V_H, E_H)$ is a subgraph of $G = (V_G, E_G)$ if

- $V_H \subseteq V_G$, and
- $E_H \subseteq E_G$.

Note: $\{u, v\} \in E_H$ implies $u, v \in V_H$.

Connected component: a maximal connected subgraph.

Graph modeling examples

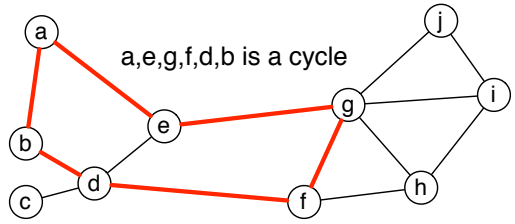
Graphs naturally model many concepts:

1. Social networks
2. Geographic adjacency
3. Polyhedra
4. Chemical molecules
5. Assigning jobs to applicants
6. Food webs
7. Finite-state machines
8. Markov processes
9. Project dependencies
10. World Wide Web
11. Telephone network
12. Roads
13. Graphical models (e.g. Bayesian networks)
14. Phylogenetic relationships
15. Mesh approximations to surfaces

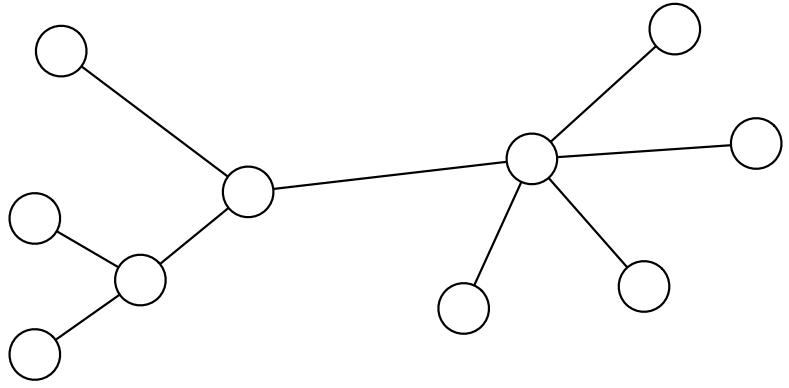
Trees

Trees are a special type of graph that occur often in algorithms.

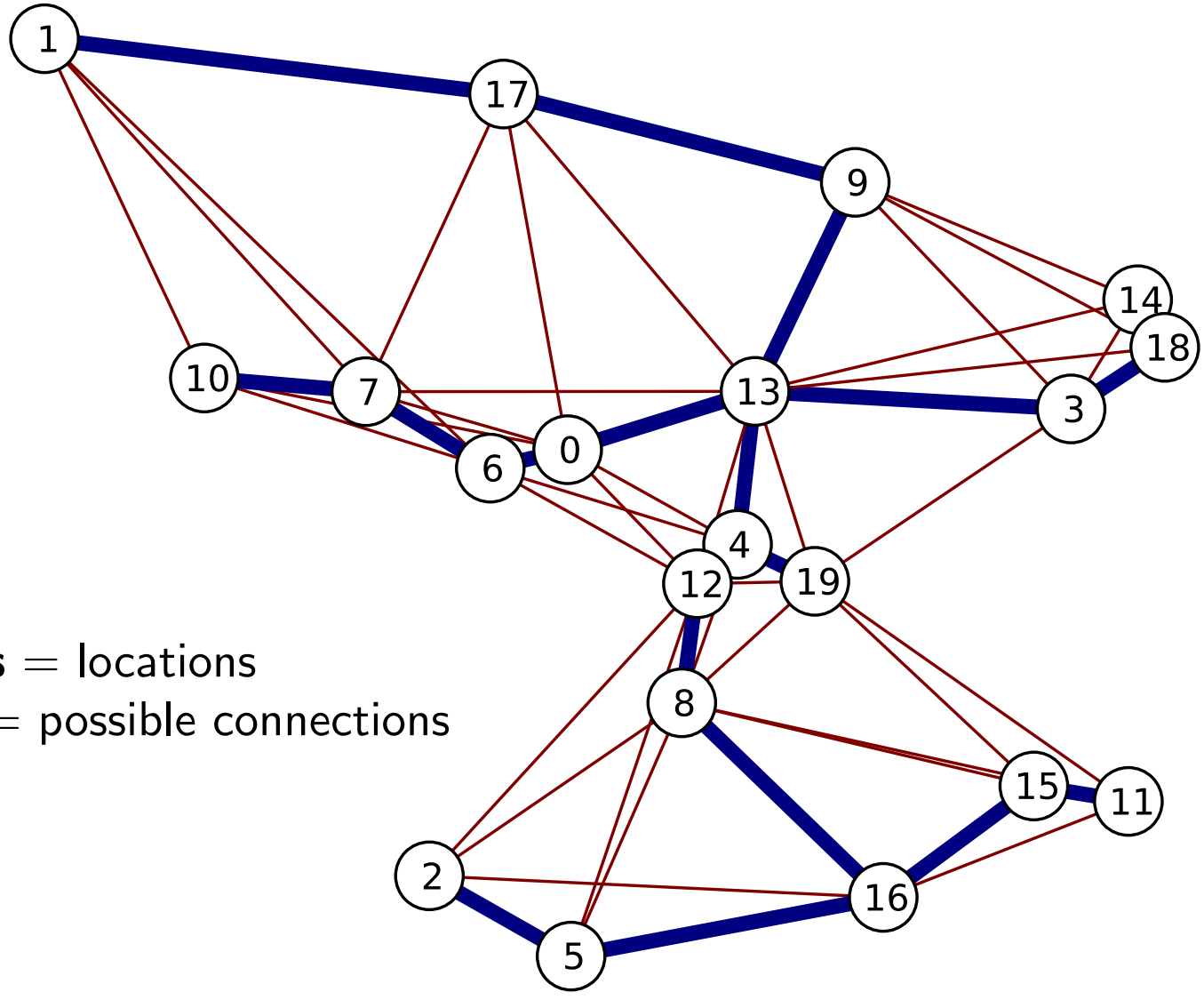
Definition (Cycle). A cycle of a graph $G = (V, E)$ is a sequence of distinct vertices $v_1, \ldots, v_k \in V$ such that $\{v_i, v_{i+1}\} \in E$ for all $i = 1, \ldots, k$ and $\{v_k, v_1\} \in E$.



Definition (Tree). A graph G is a tree if it is connected and contains no cycles.



Problem: Cost effective wiring of a network



circles = locations
lines = possible connections

The Minimum Spanning Tree problem

Given:

- undirected graph G with vertices for each of n objects
- weights $d(u, v) =$ cost of using edge {u,v}.

Find the subgraph T that connects all vertices and minimizes

$$\text{cost}(T) := \sum_{\{u,v\} \in T} d(u, v).$$

T will be a tree. Why?

The Minimum Spanning Tree problem

Given:

- undirected graph G with vertices for each of n objects
- weights $d(u, v) =$ cost of using edge {u,v}.

Find the subgraph T that connects all vertices and minimizes

$$\text{cost}(T) := \sum_{\{u,v\} \in T} d(u, v).$$

T will be a tree. Why?

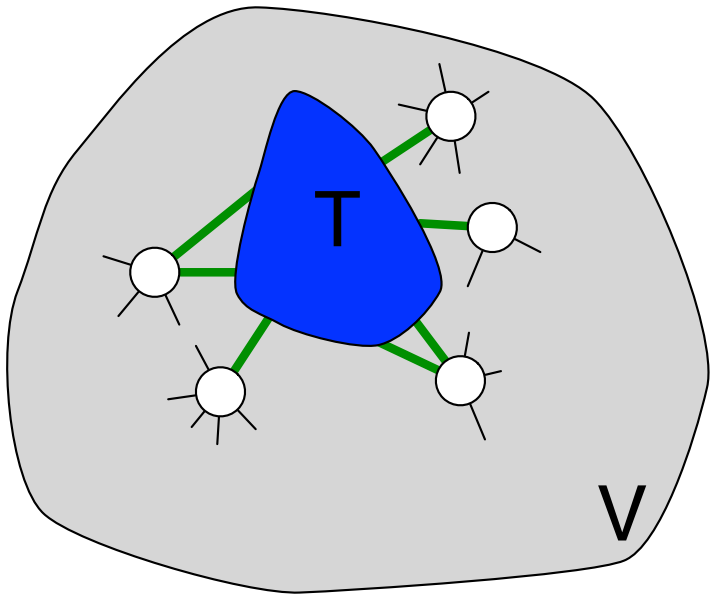
If there was a cycle, we could remove any edge on the cycle to get a new subgraph T' with smaller $\text{cost}(T')$.

Prim's MST algorithm

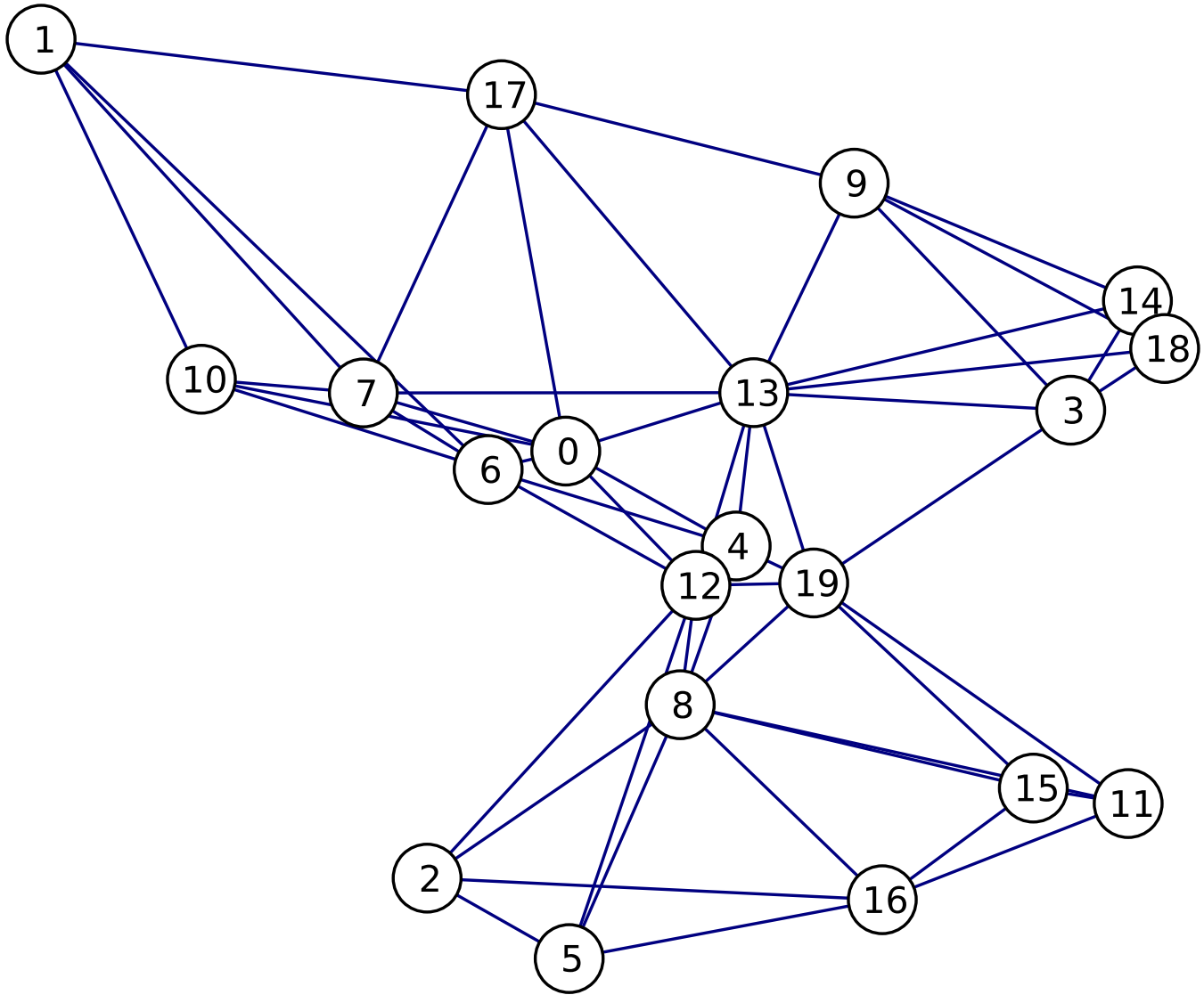
1. Given graph $G = (V, E)$, select an arbitrary vertex $s \in V$ and let T be a "tree" that contains only s.

2. Repeat the following $|V| - 1$ times:

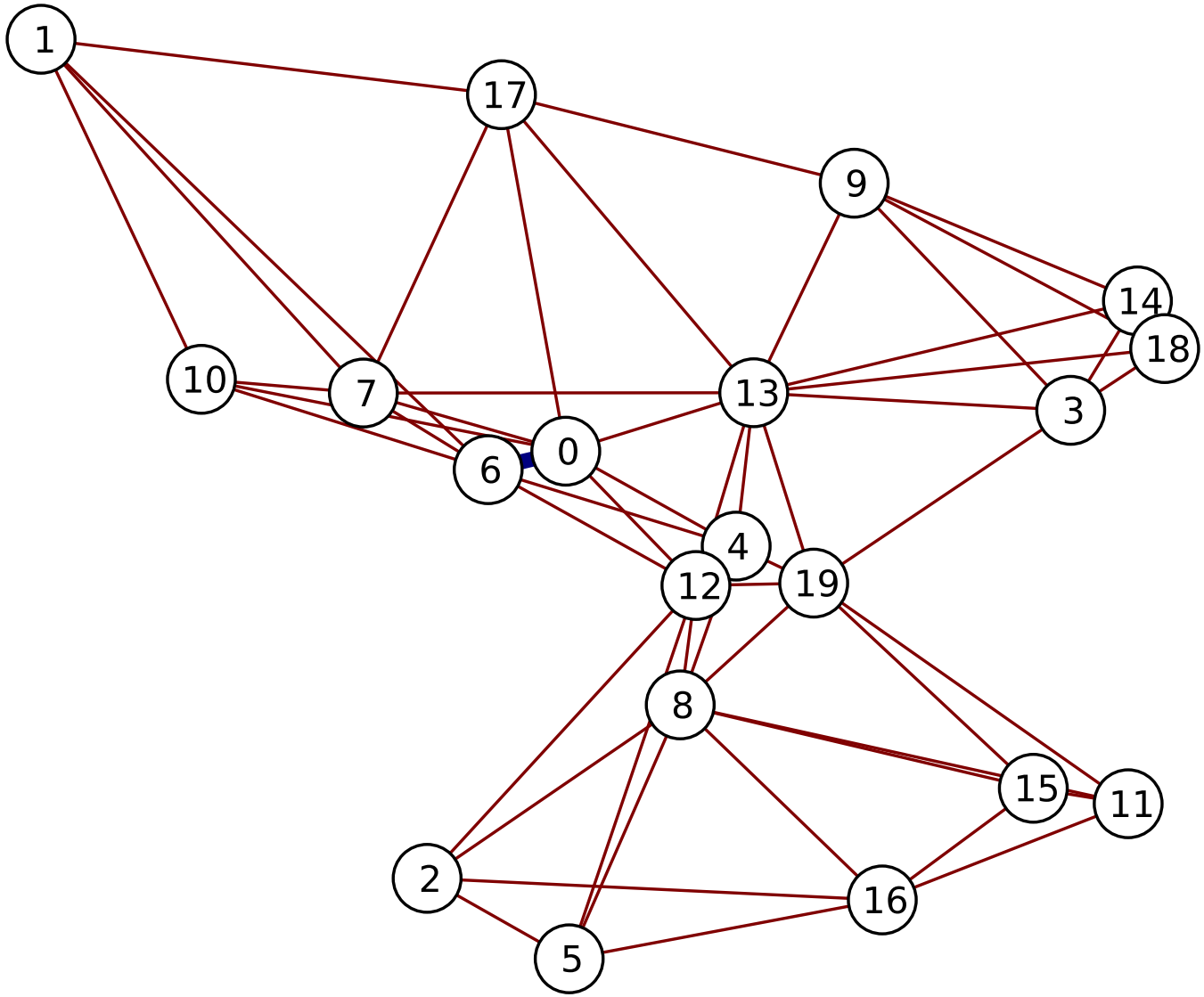
   Add to T the lowest-cost edge $\{u, v\}$ where $u \in T$ and $v \notin T$.



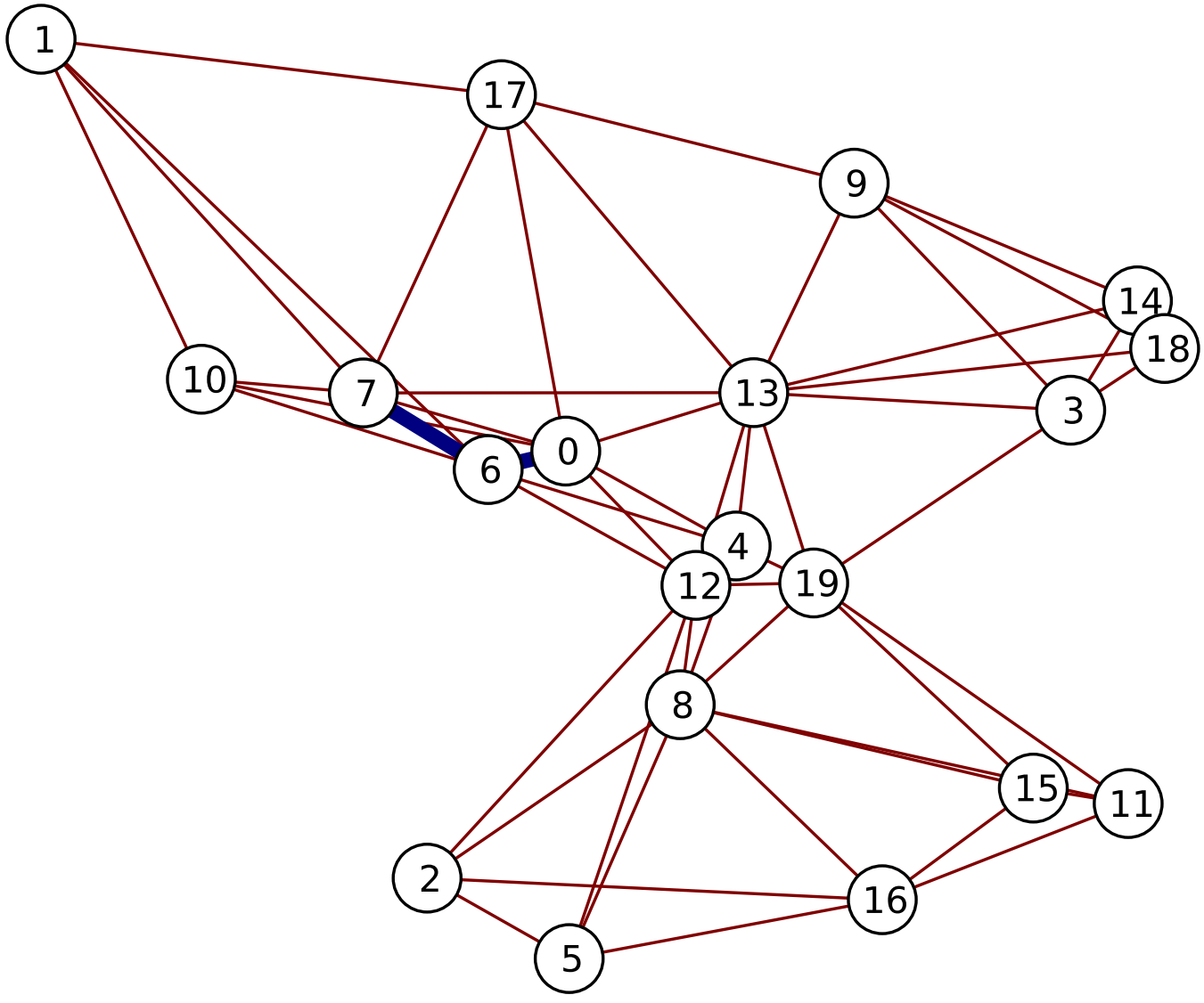
Example run of Prim's



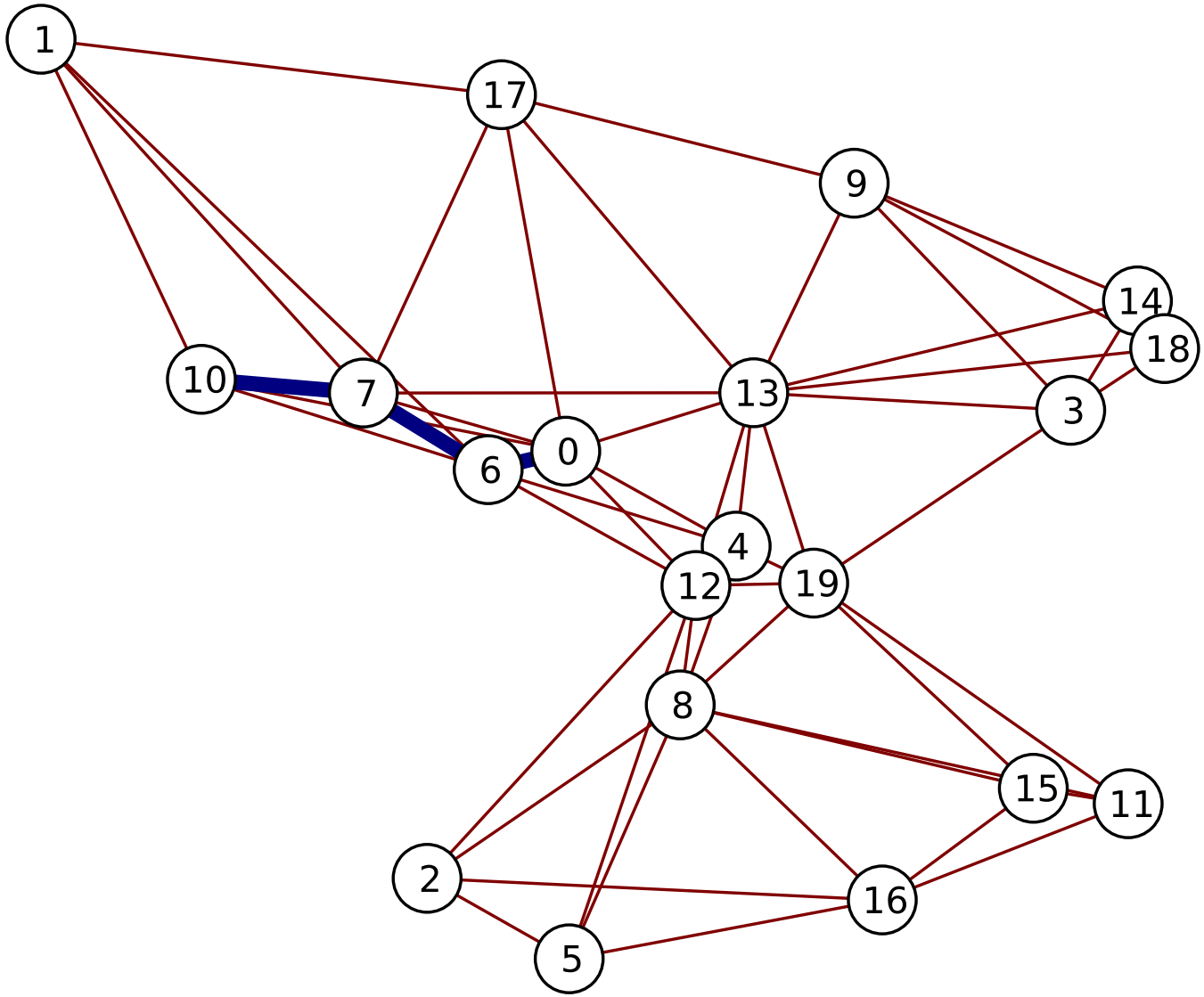
Example run of Prim's



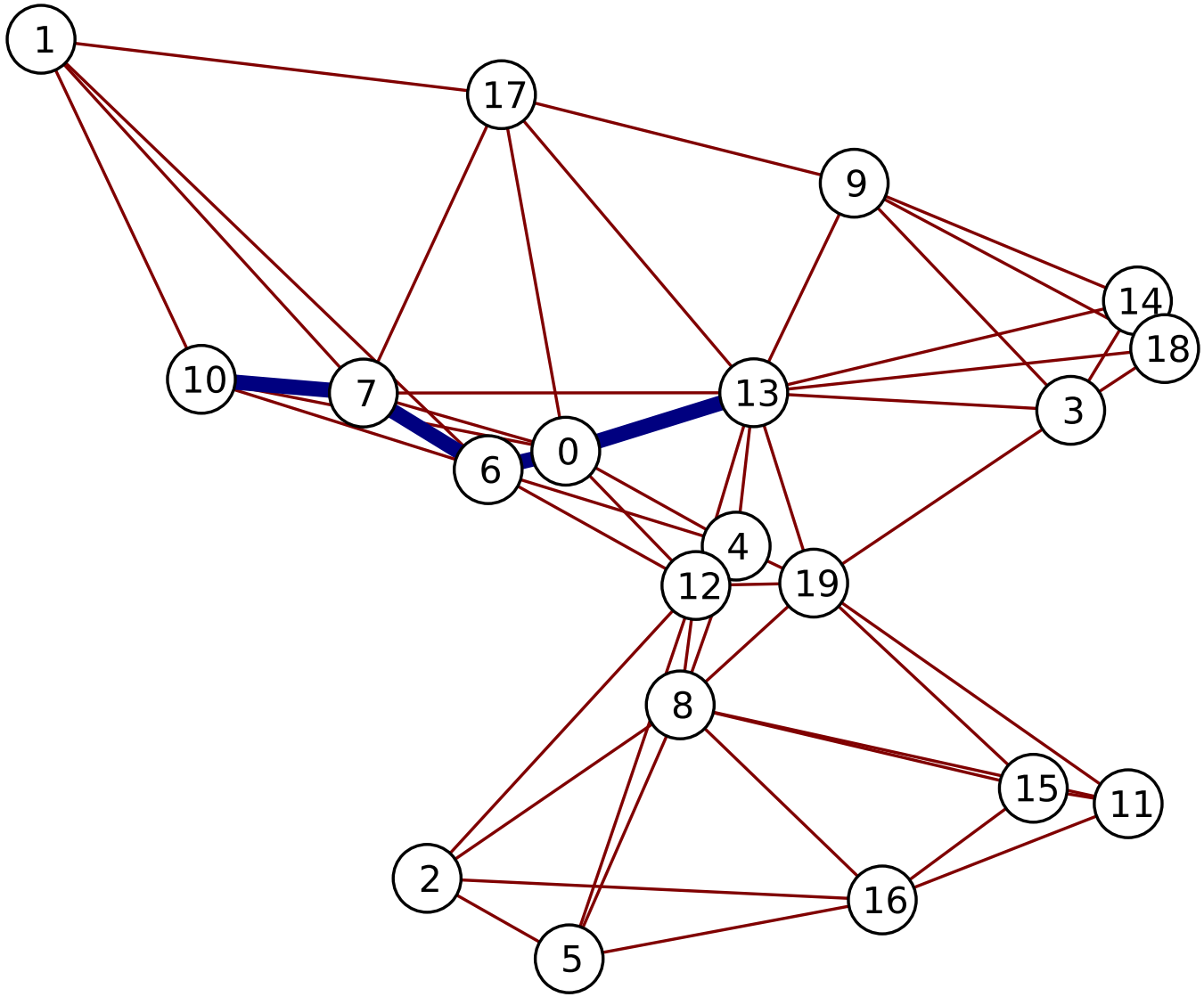
Example run of Prim's



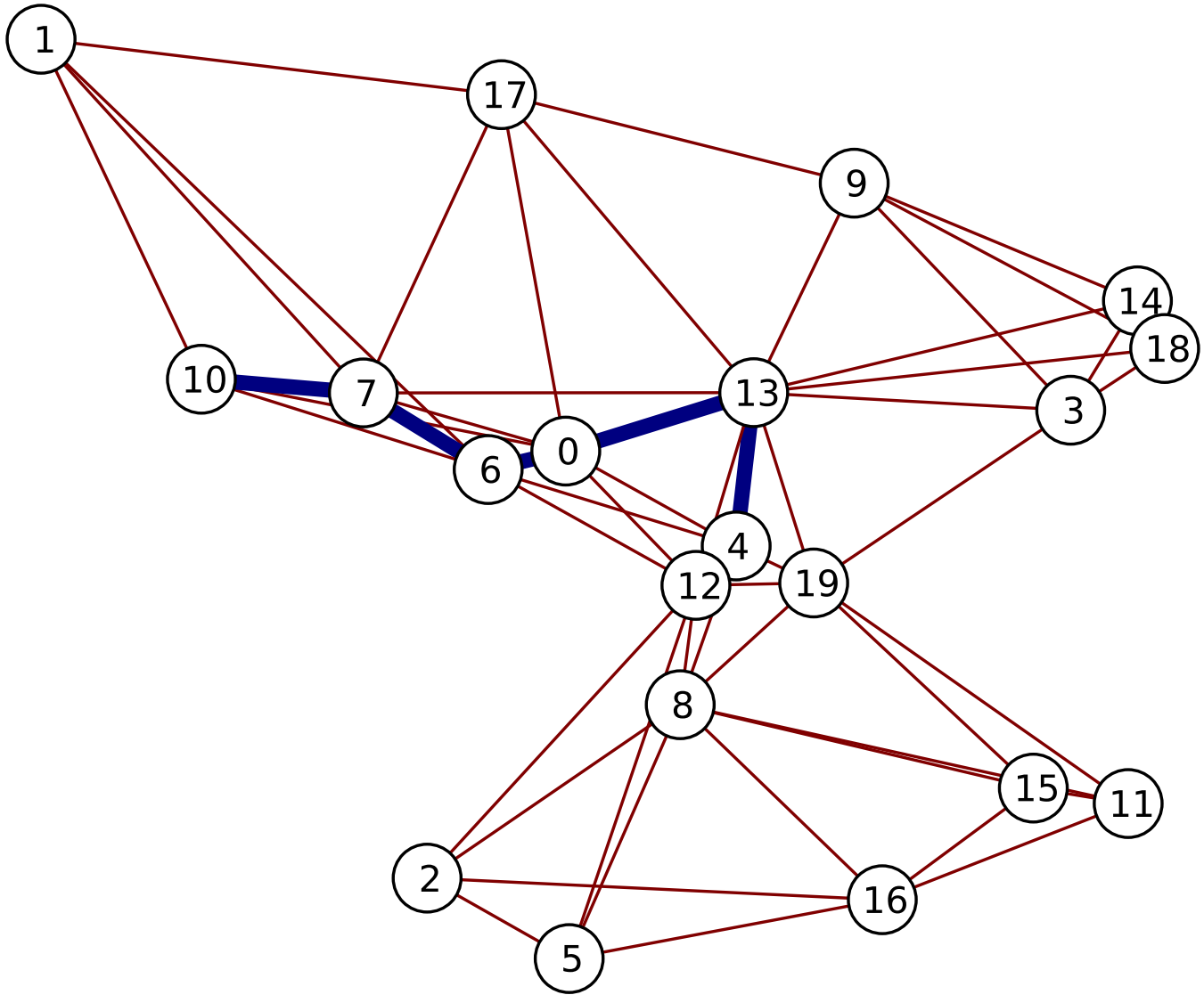
Example run of Prim's



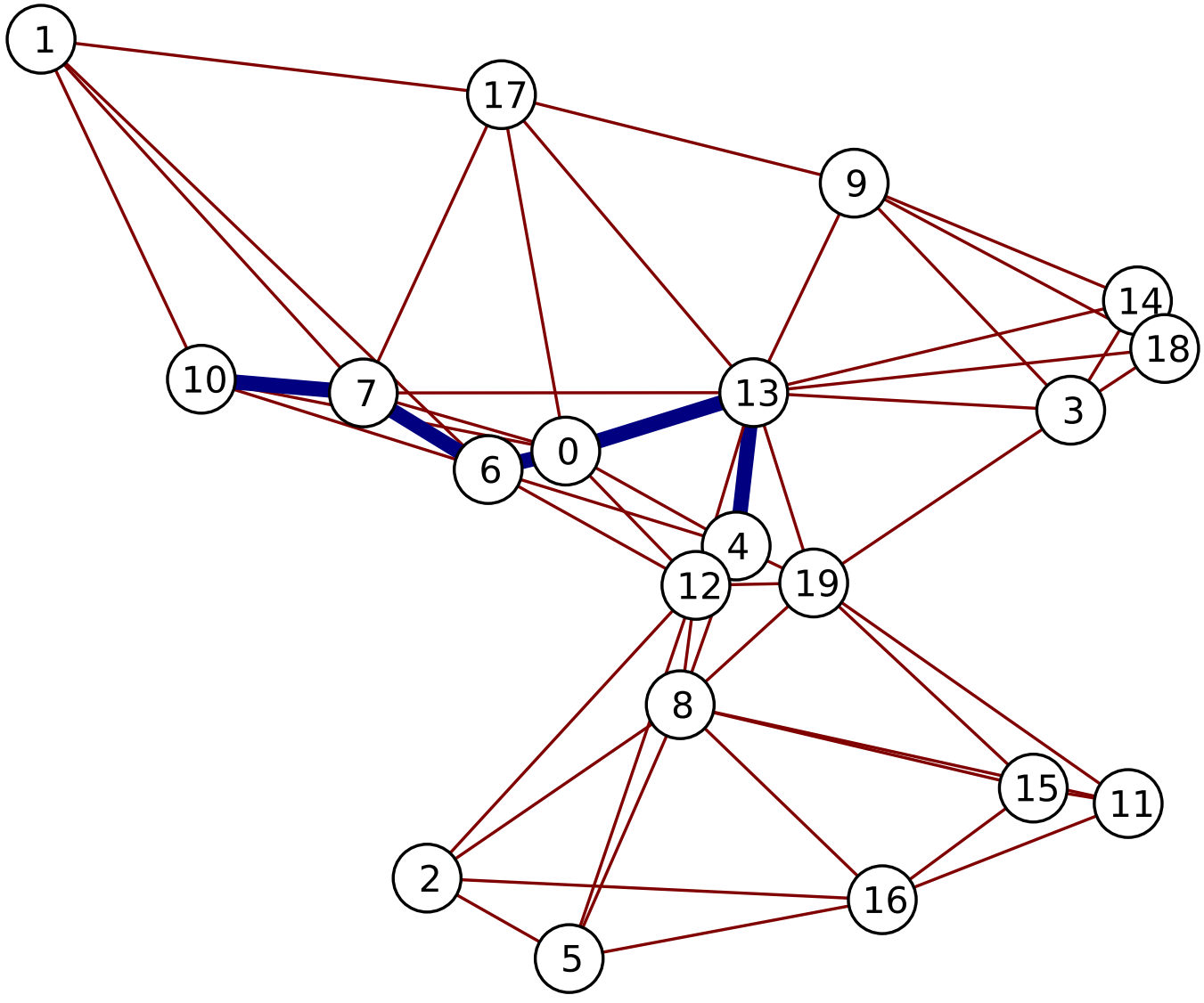
Example run of Prim's



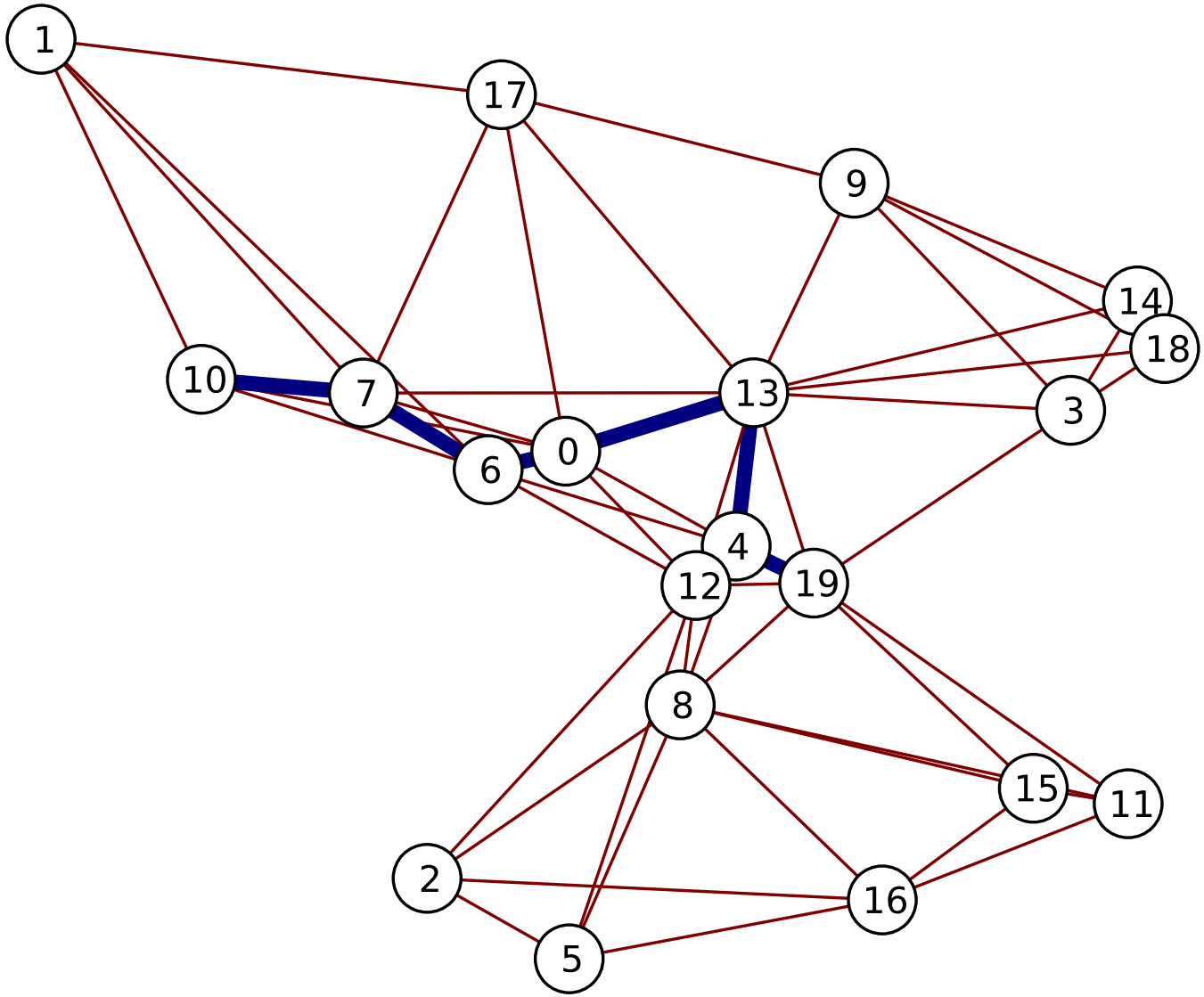
Example run of Prim's



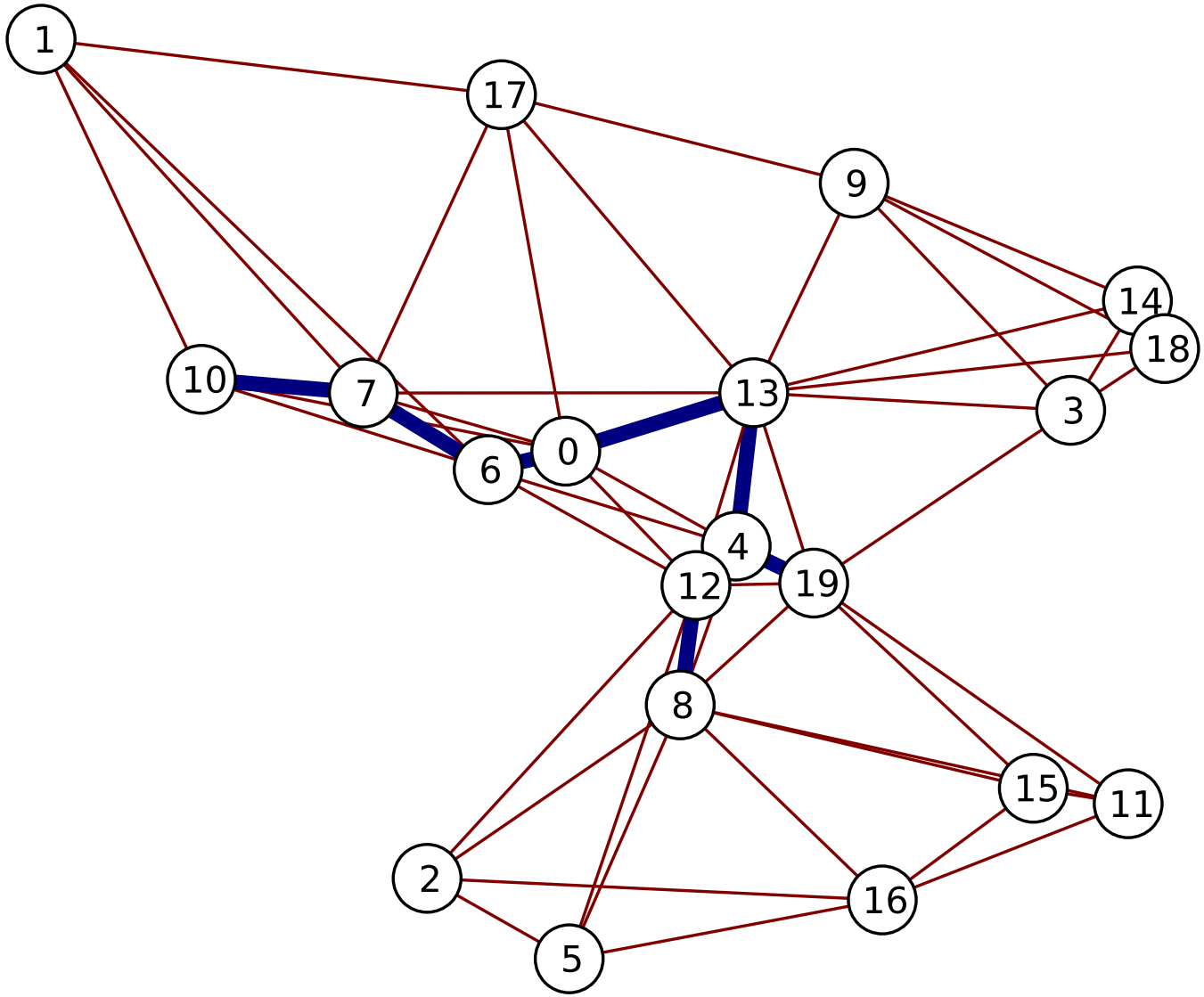
Example run of Prim's



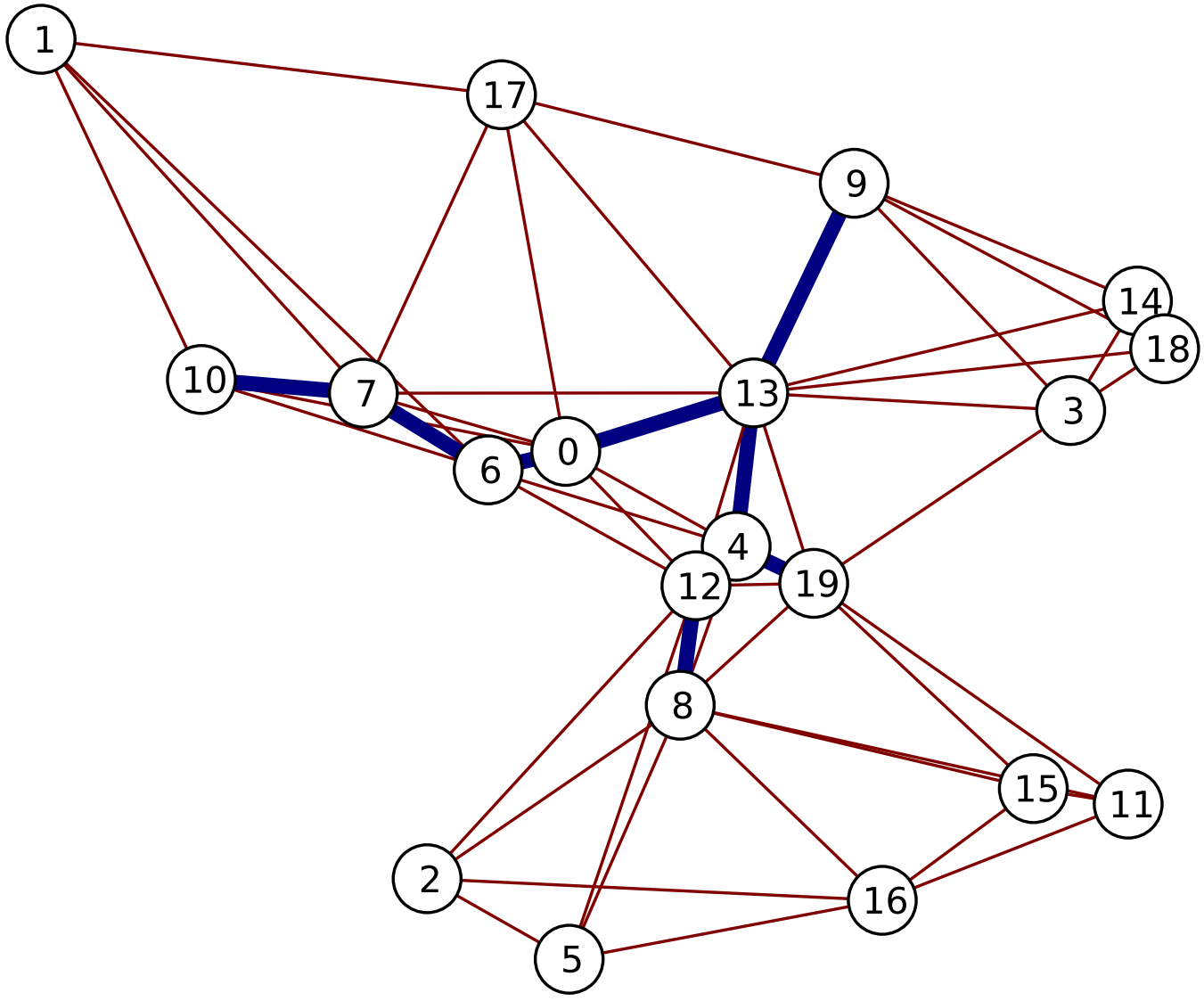
Example run of Prim's



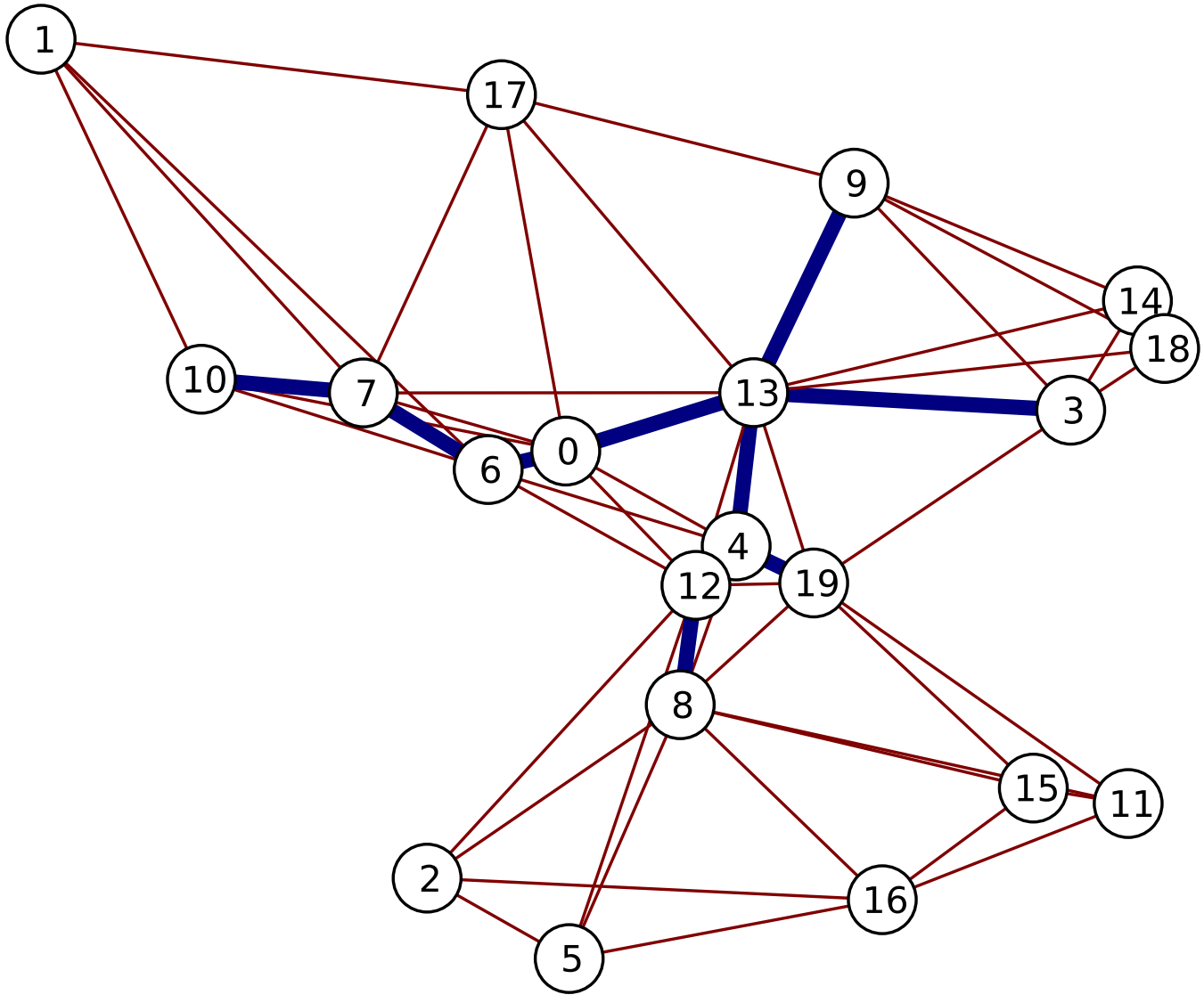
Example run of Prim's



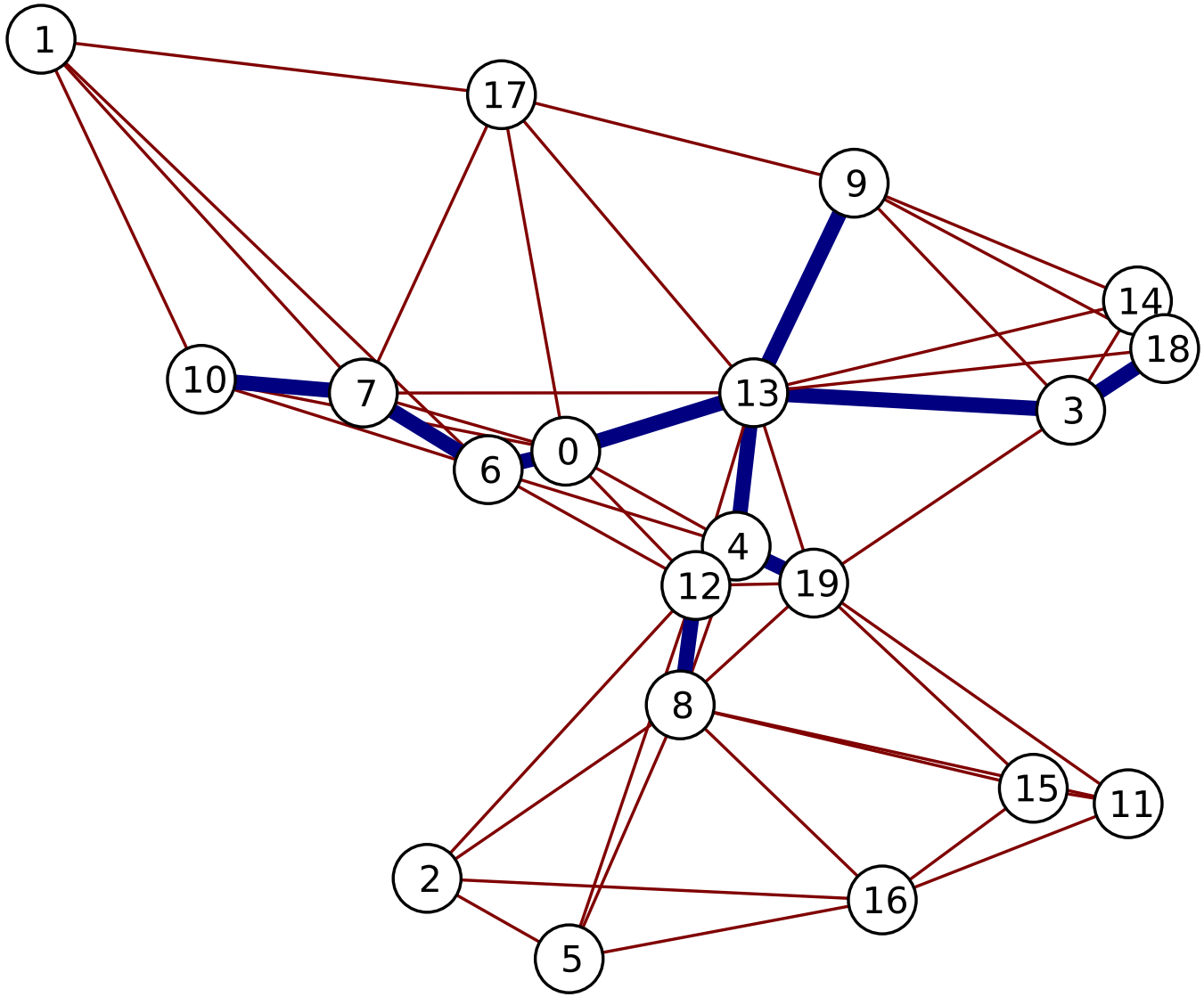
Example run of Prim's



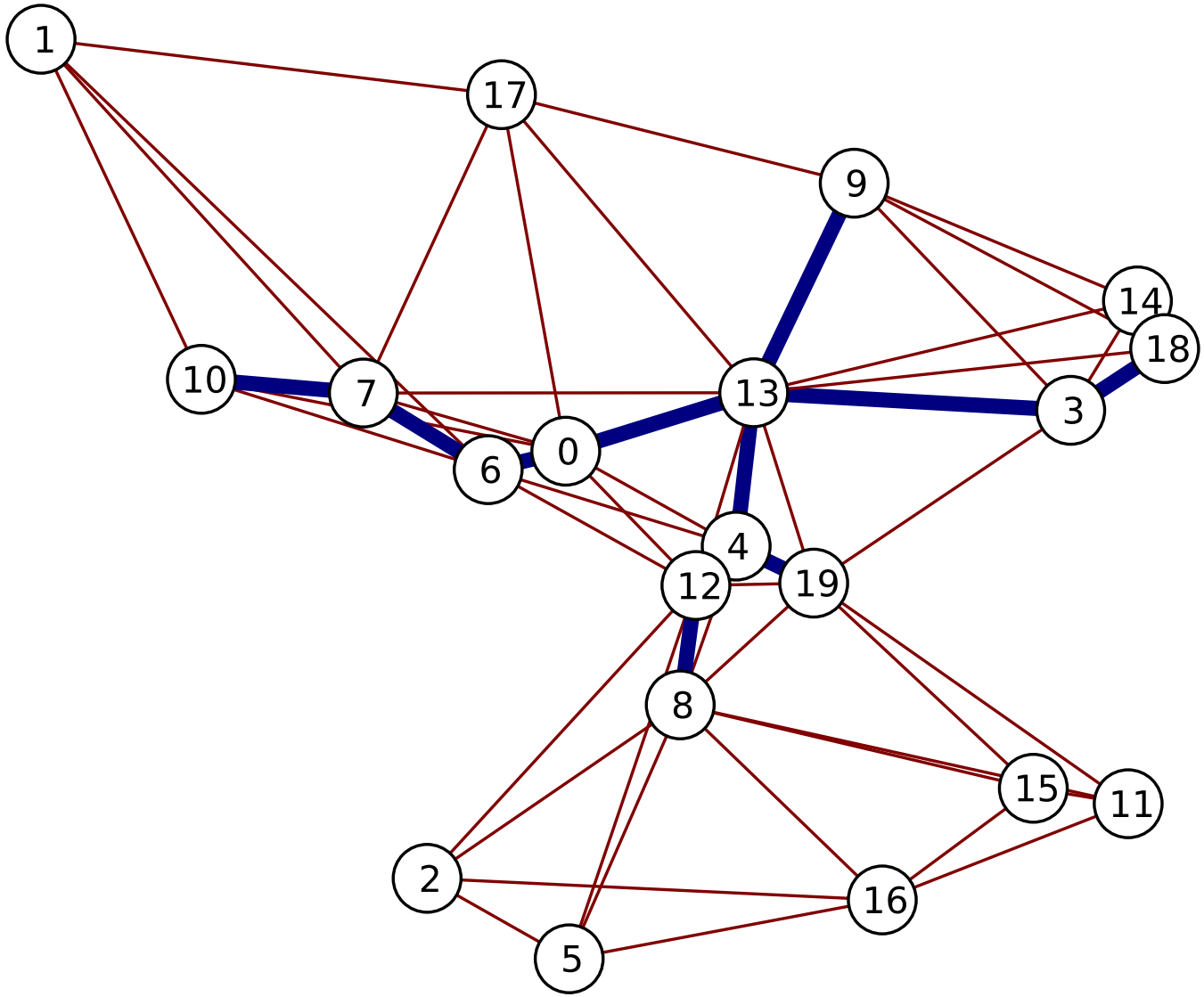
Example run of Prim's



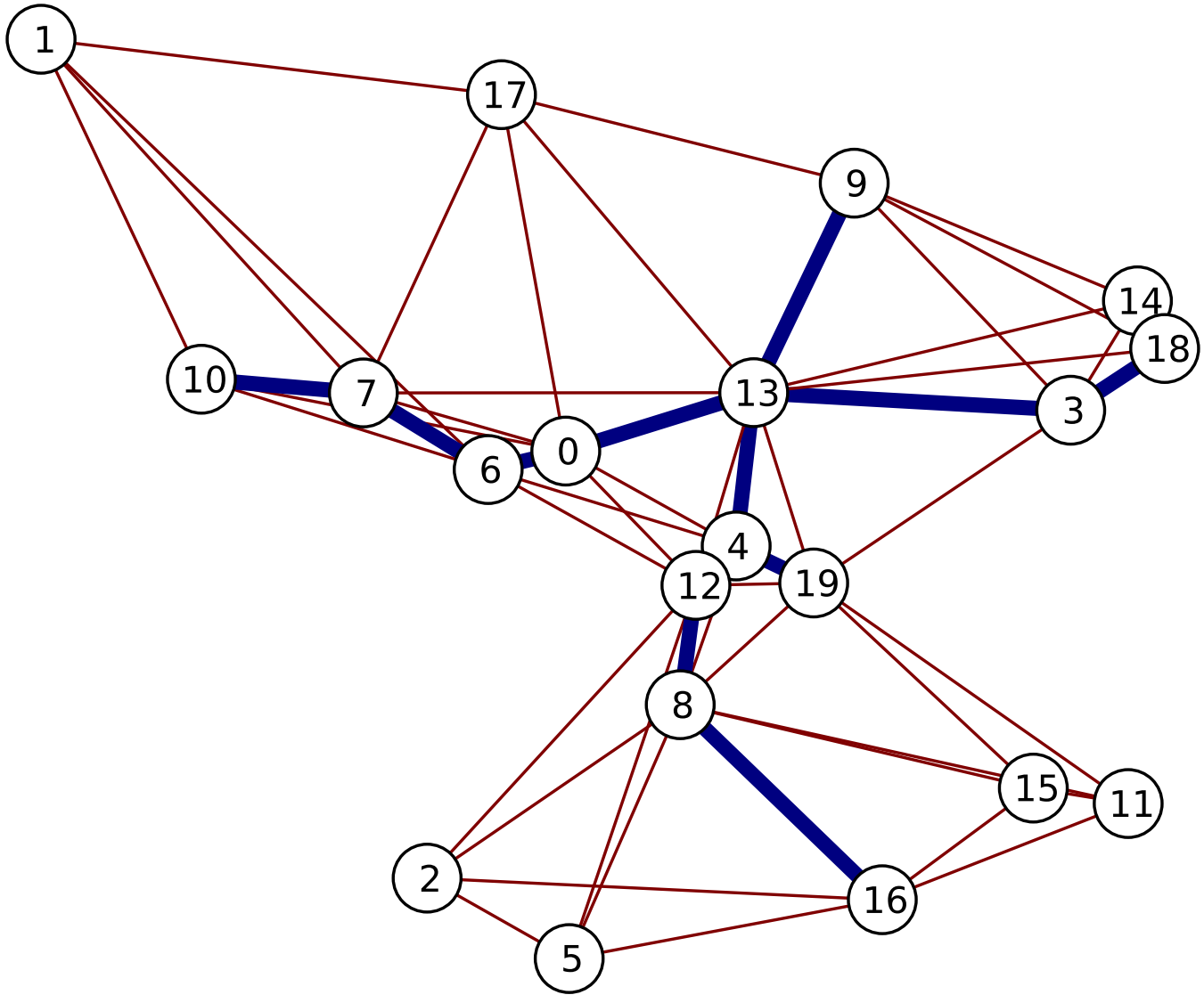
Example run of Prim's



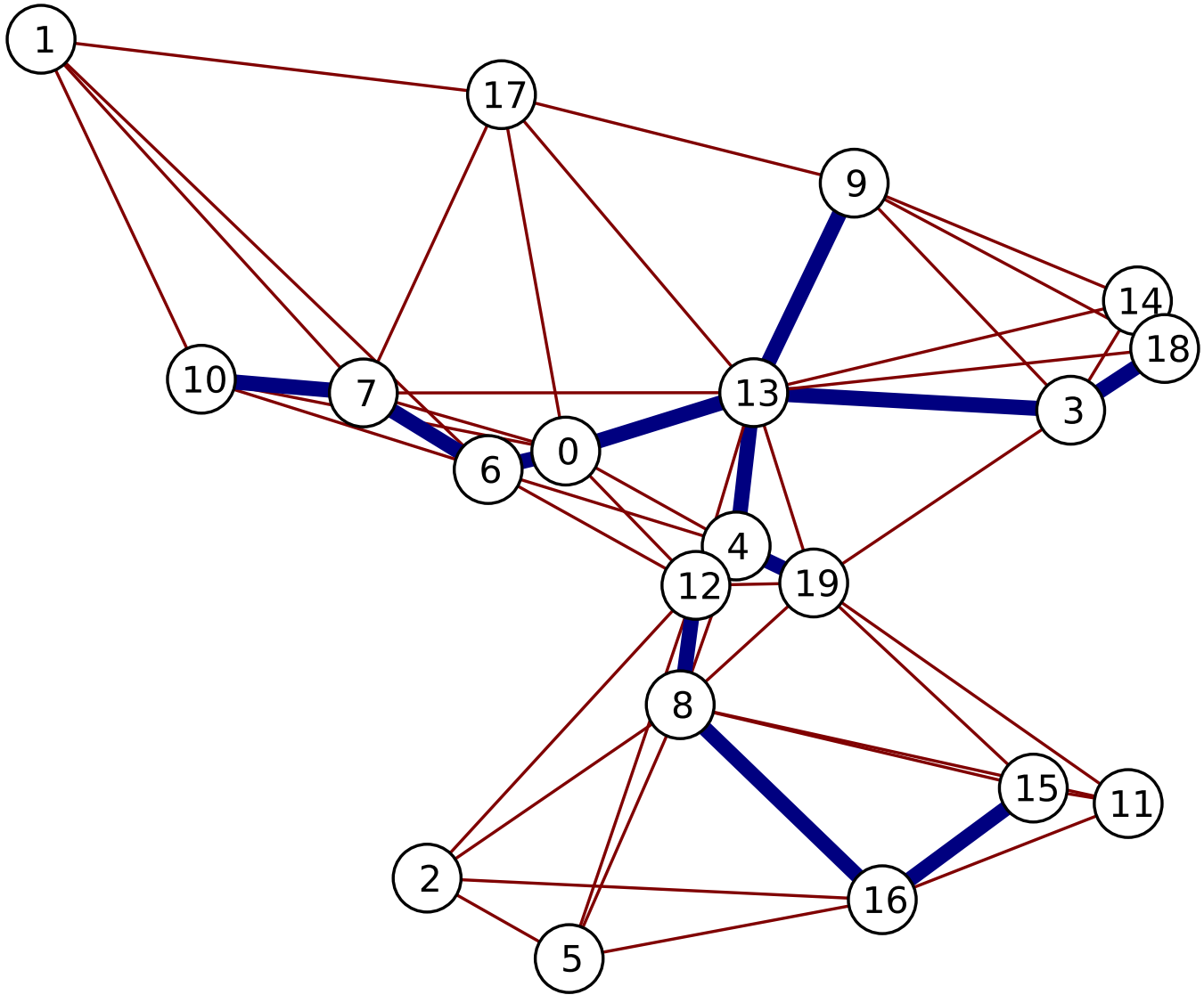
Example run of Prim's



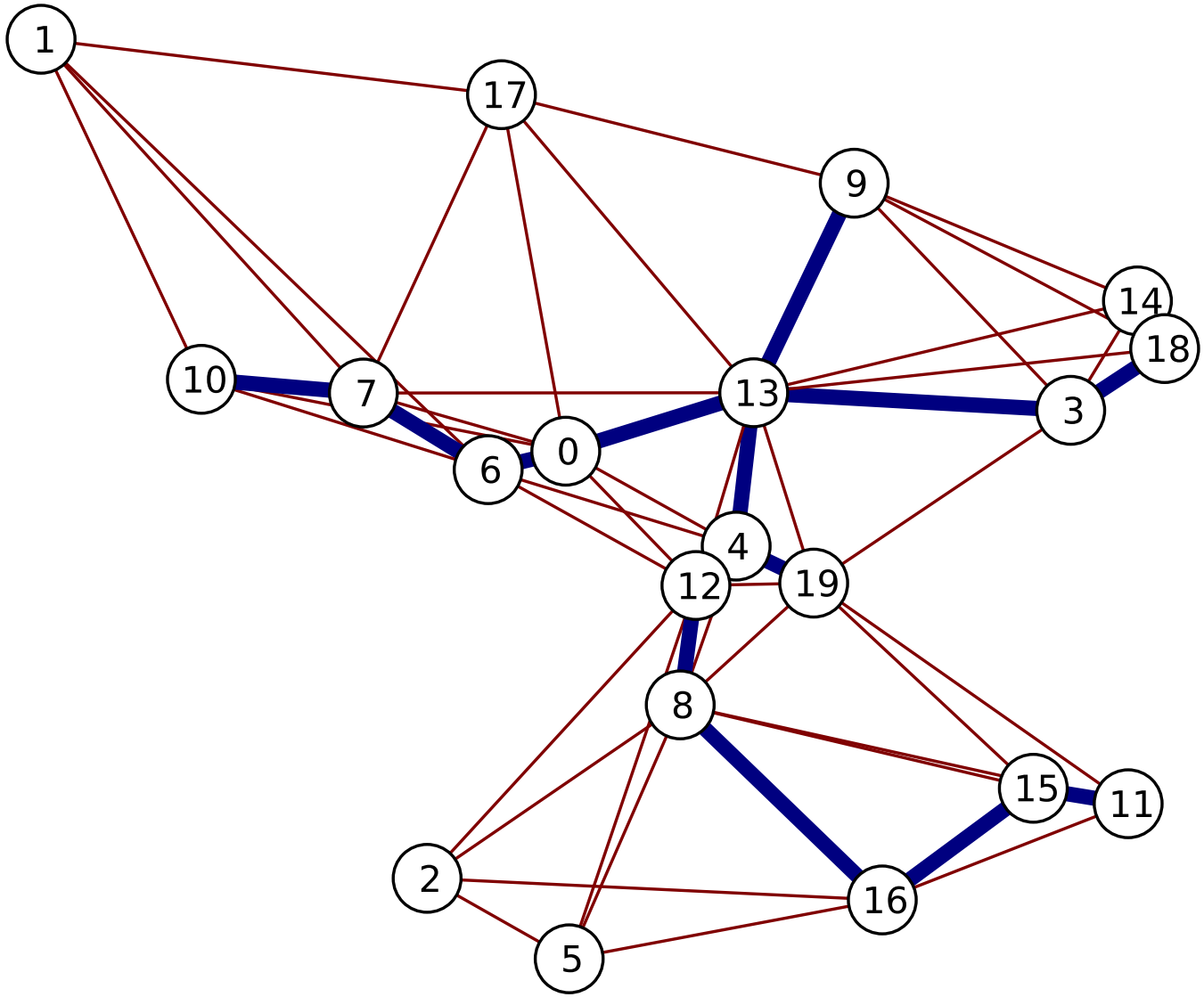
Example run of Prim's



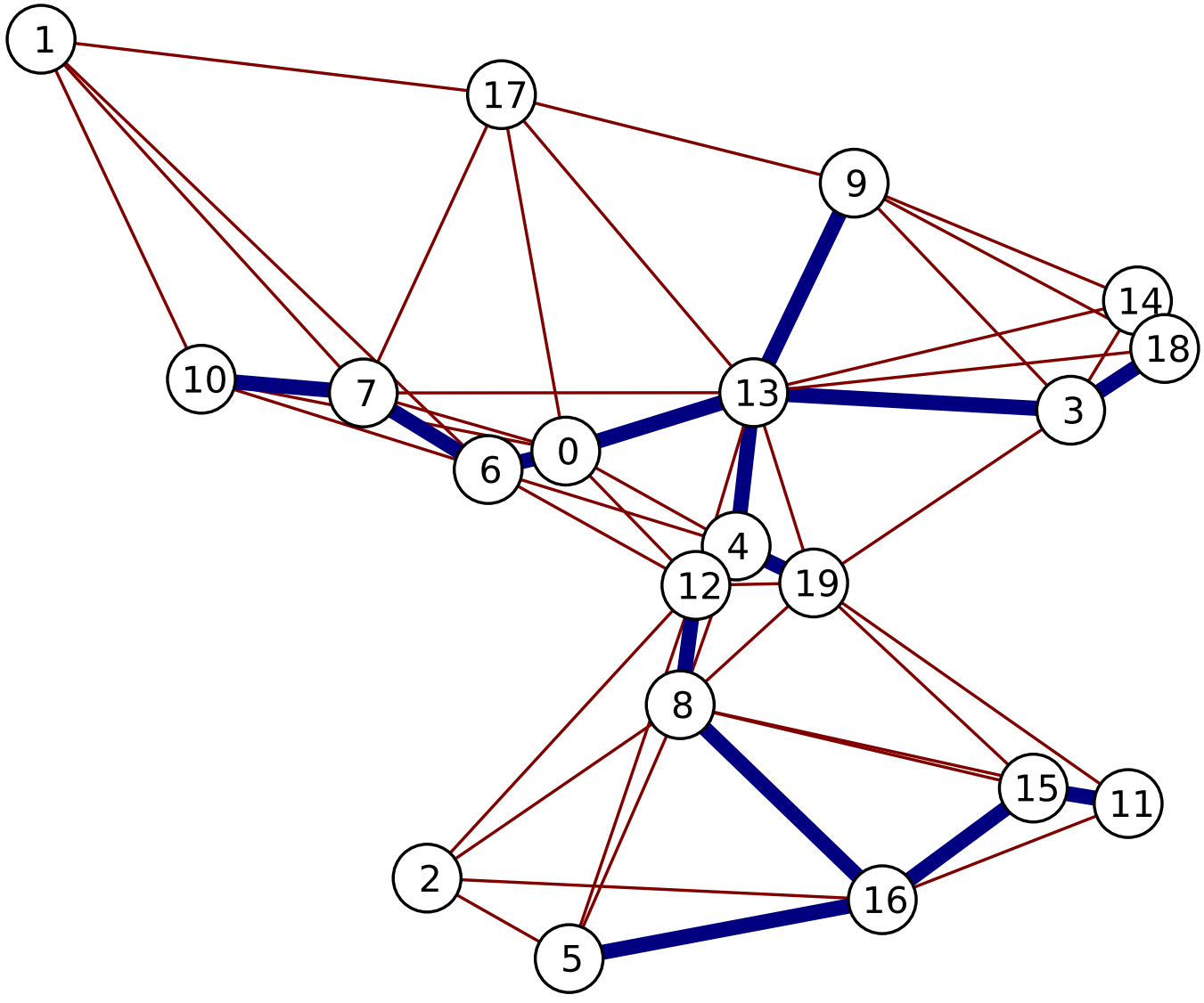
Example run of Prim's



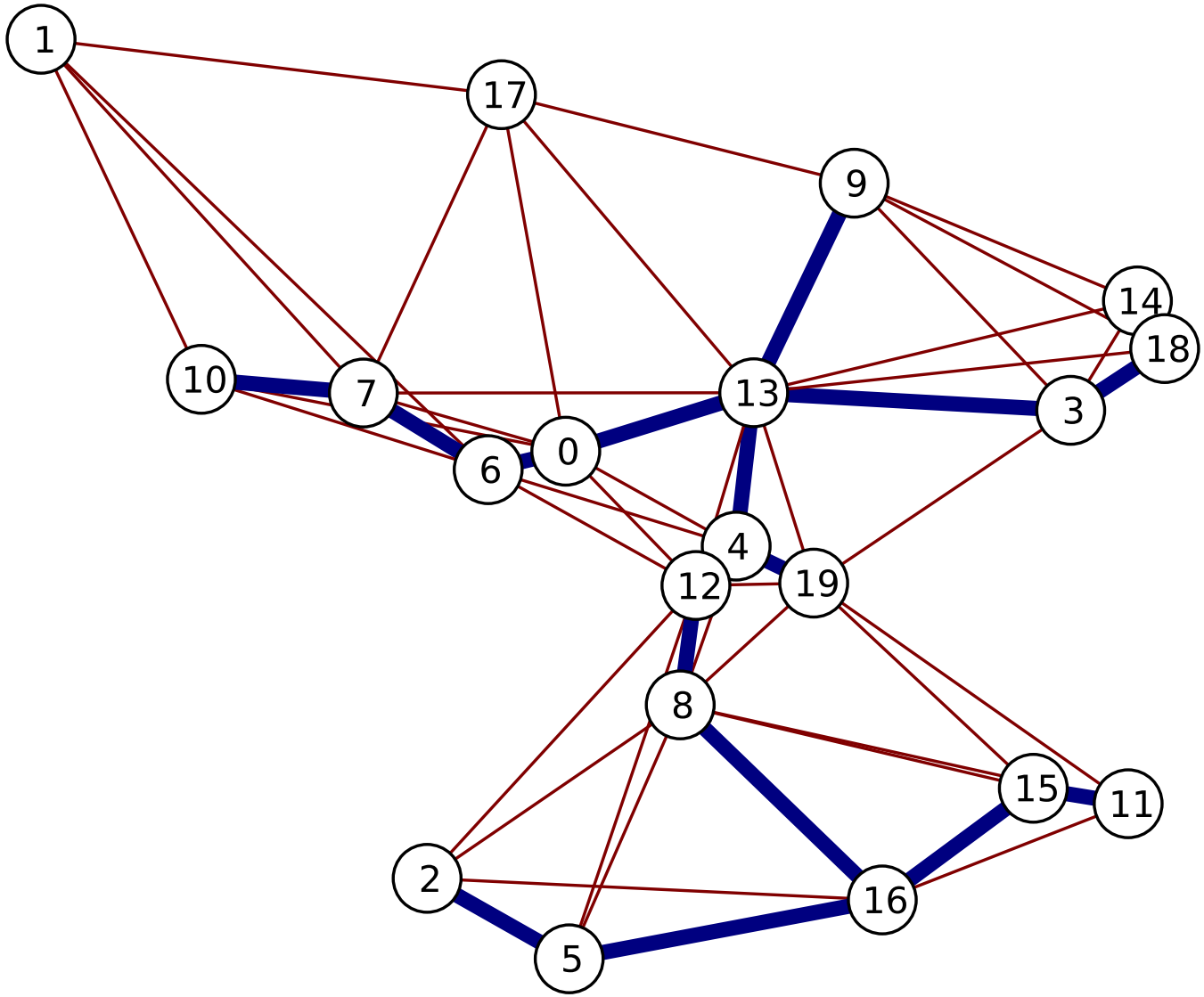
Example run of Prim's



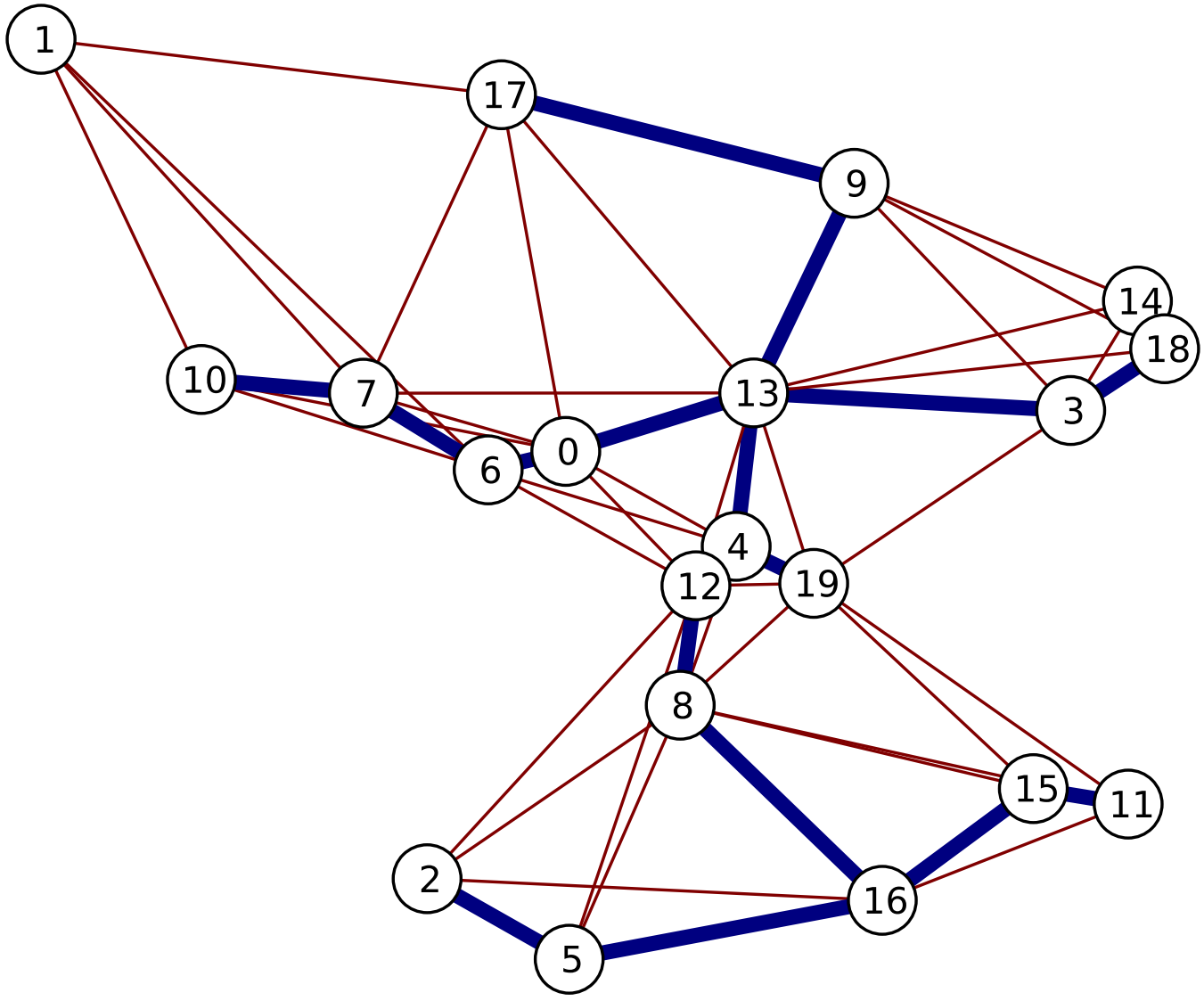
Example run of Prim's



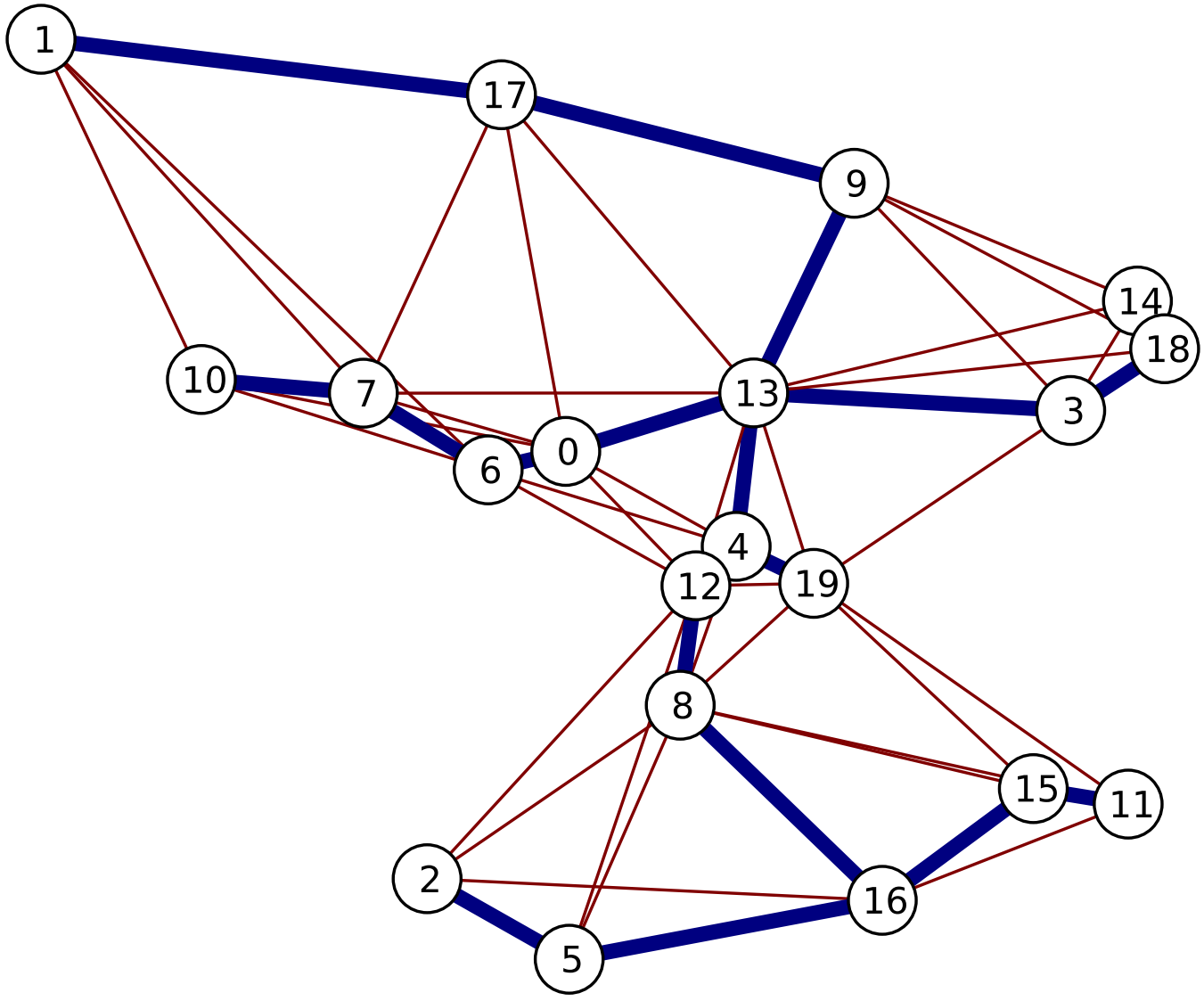
Example run of Prim's



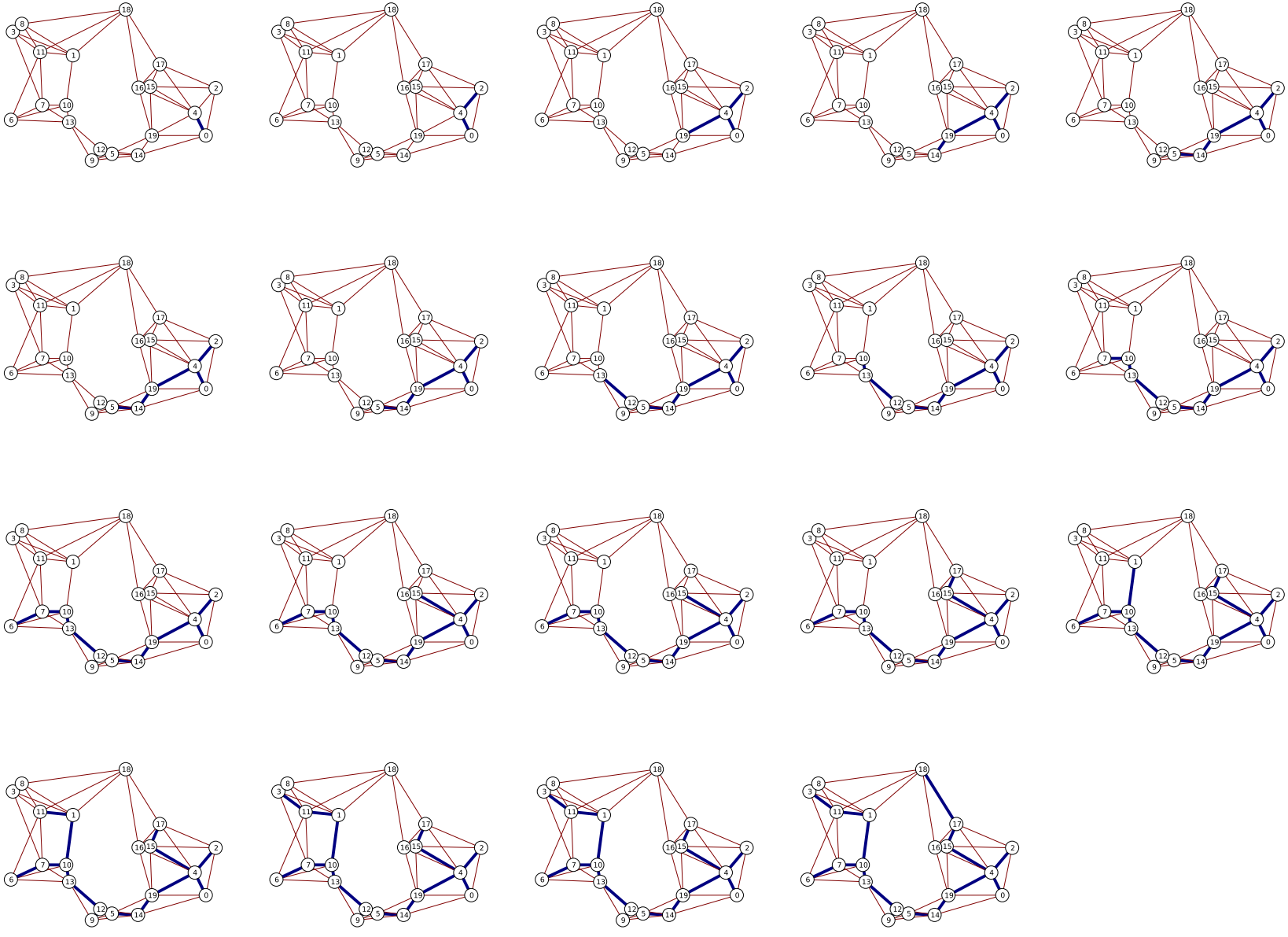
Example run of Prim's



Example run of Prim's



Another example



Prim's MST algorithm pseudocode

```
1:  # distToT[u] is distance from current tree to u
2:  for u ∈ V do distToT[u] ← ∞
3:  u ← s # (s is an arbitrary start vertex)
4:  while u ≠ null do
5:      # We put u in the tree, so distance is −∞
6:      distToT[u] ← −∞
7:      # Each of u's neighbors v are now incident to the current tree
8:      for v ∈ Neighbors(u) do
9:          # If the distance is smaller than before, we have to update
10:         if d(u,v) < distToT[v] then
11:             distToT[v] ← d(u,v)
12:             parent[v] ← u
13:     u ← ClosestVertex(distToT)
14: return parent
```

Questions

- How do we know it will always find the minimum?
  (Correctness)
- How can we find quickly the ClosestVertex and iteratate ?
  (Choice of data structure)
- What's the longest amount of time the algorithm will take?
  (Worst-case running time)
- Is this the fastest possible algorithm?
  (Complexity theory)

Correctness of Prim's MST Algorithm

Trees

Theorem (Characterization of Trees). *The following statements are equivalent:*

1. *T is a tree.*
2. *T contains no cycles and $n - 1$ edges.*
3. *T is connected and has $n - 1$ edges.*
4. *T is connected and removing any edge disconnects it.*
5. *Any two nodes in T are connected by exactly 1 path.*
6. *T is acyclic, and adding any new edge creates exactly one cycle.*

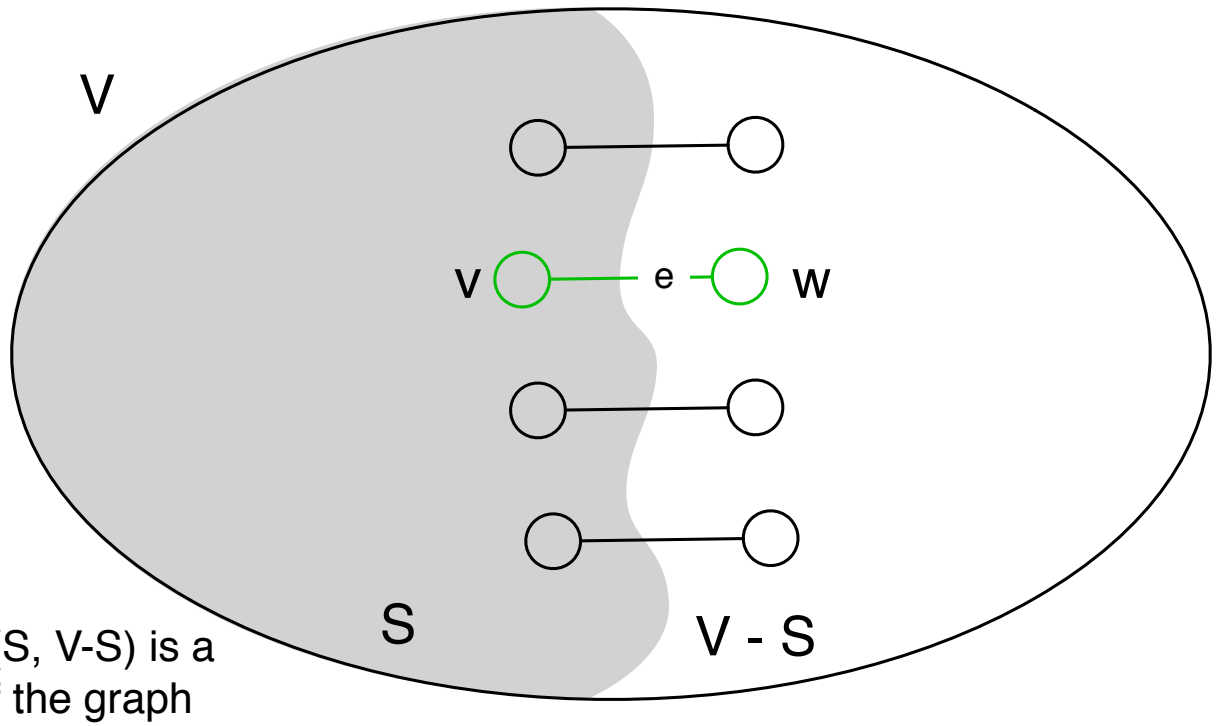
Assumption

We assume no two edges in G have the same edge cost.

If this doesn't hold true, we can add a very small value ϵ_e to the weight of every edge e.

Cut Property

Theorem (MST Cut Property). *Let S be a subset of nodes, with $|S| \geq 1$ and $|S| < |V|$. Every MST contains the edge $e = \{v, w\}$ with $v \in S$ and $w \in V - S$ that has minimum weight.*

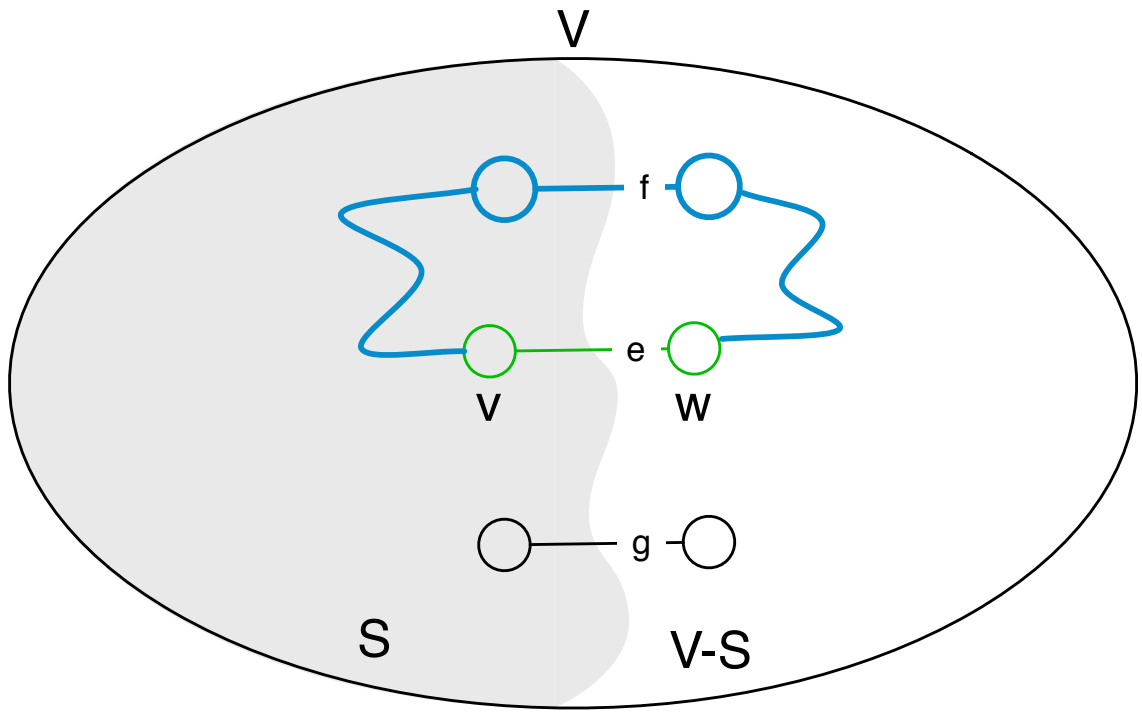


A pair (S, V-S) is a ***cut*** of the graph

Cut Property, Proof

Suppose T doesn't contain e. Because T is connected, it must contain a path P between v and w. P must contain some edge f that "crosses the cut."

The subgraph $T' = T - f \cup e$ has lower weight than T. T' is acyclic because the only cycle in $T' \cup f$ is eliminated by removing f.



Correctness

Theorem (Prim's correctness). *At termination, Prim's algorithm returns a subgraph T that is a minimum spanning tree.*

Proof. At any point, $T = (V_T, E_T)$ is a subgraph that is a tree.

T grows by 1 vertex and 1 edge after each iteration, so it will stop after $|V_G| - 1$ iterations and at that point T will be a spanning tree.

The pair $(V_T, V_G - V_T)$ is a cut of G.

By the cut property, the MST contains the lowest cost edge crossing this cut.

This is exactly the edge Prim's adds to T. So, Prim's only adds edges that are in the MST. □