# Minimum Spanning Trees

connected, undirected graph **G (V, E),** where V is the set of pins, E is the set of possible interconnections between pairs of pins, and for each edge (u,v) belongs to E, we have a weight w(u,v) specifying the cost (amount of wire needed) to connect u and v.

We then wish to find an acyclic subset T is a subset of E that connects all of the vertices and whose total weight is minimized .

Since T is acyclic and connects all of the vertices, it must form a tree, which we call a ***spanning tree*** since it “spans” the graph G. We call the problem of determining the tree T the ***minimum-spanning-tree problem***.

solving the minimum- spanning-tree problem: **Kruskal’s algorithm and Prim’s algorithm**. We can easily make each of them run in time **O(E lg V) using ordinary binary heaps**. By using **Fibonacci heaps, Prim’s algorithm runs in time O(E + V lg V )**, which improves over the binary-heap implementation **if |V| is much smaller than |E|.**

The two algorithms are **greedy algorithms**

Assume that we have a connected, undirected graph G (V,E) with a weight function w : E -> R, and we wish to find a minimum spanning tree for G

The generic method manages a set of edges A, maintaining the following loop invariant:

Prior to each iteration, A is a subset of some minimum spanning tree.