BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Data Structures and Algorithms
CS F211

Vishal Gupta

Department of Computer Science and Information Systems

Birla Institute of Technology and Science

Pilani Campus, Pilani

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Agenda: Minimum Spanning Trees

Four classes of graph problem

.. that can be solved efficiently (in polynomial time)

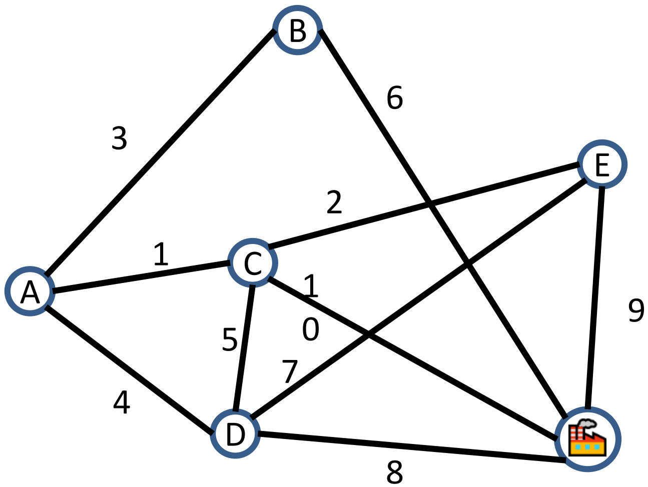
1. Shortest path – find a shortest path between two vertices in a graph
2. Minimum spanning tree – find subset of edges with minimum total weights
3. Matching – find set of edges without common vertices
4. Maximum flow – find the maximum flow from a source vertex to a sink vertex

A wide array of graph problems that can be solved in polynomial time are variants of these above problems.

In this class, we'll cover the first two problems – shortest path and minimum spanning tree

Minimum Spanning Trees

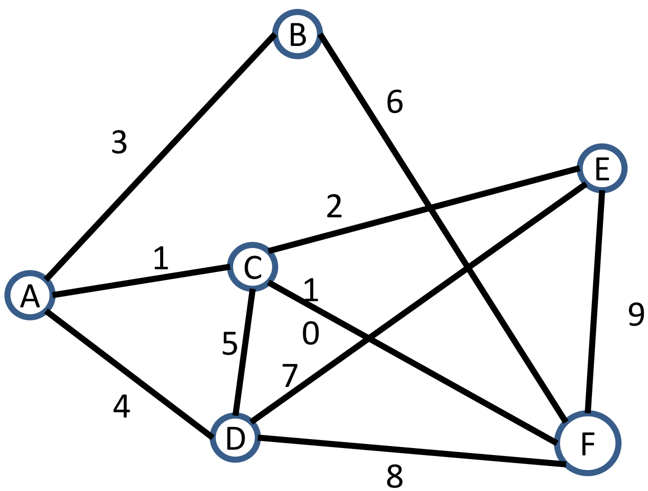
- It's the 1920's. Your friend at the electric company needs to choose where to build wires to connect all these cities to the plant.



She knows how much it would cost to lay electric wires between any pair of locations, and wants the cheapest way to make sure electricity from the plant to every city.

Minimum Spanning Trees

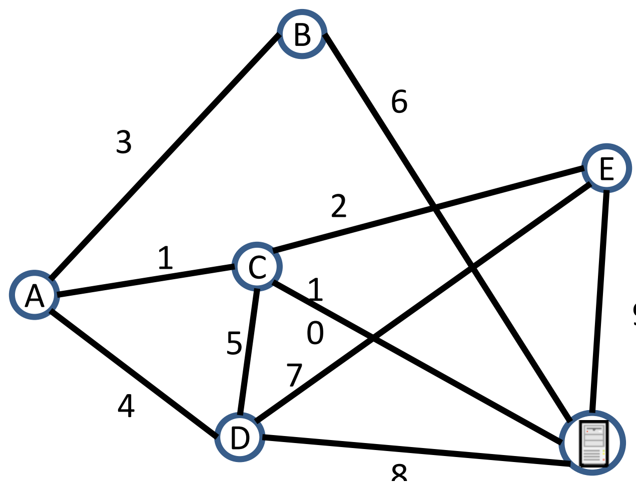
- It's 1950's Your boss at the phone company needs to choose where to build wires to connect all these cities each other.



She knows how much it would cost to lay phone wires between any pair of locations, and wants the cheapest way to make sure elect Everyone can call everyone else.

Minimum Spanning Trees

It's today Your friend at the ISP , needs to choose where to build wires to connect all these cities to the Internet with fiber optic cable



She knows how much it would cost to lay Cable between any pair of locations, and wants the cheapest way to make sure electric Everyone can reach the server

Minimum Spanning Trees

- What do we need? A set of edges such that:
  - Every vertex touches at least one of the edges. (the edges **span** the graph)
  - The graph on just those edges is **connected**.
  - The minimum weight set of edges that meet those conditions.
- Assume all edge weights are positive.
- Claim: The set of edges we pick never has a cycle. Why?

Aside: Trees

- Our BSTs had:
  - A root
  - Left and/or right children
  - Connected and no cycles
- Our heaps had:
  - A root
  - Varying numbers of children (but same at each level of the tree)
  - Connected and no cycles
- On graphs our tees:
  - Don't need a root (the vertices aren't ordered, and we can start BFS from anywhere)
  - Varying numbers of child
  - Connected and no cycles

Tree (when talking about graphs)

An undirected, connected acyclic graph.

MST Problem

Given an undirected graph G = (V, E), and for each edge (u, v) ∈ E, we have a weight w(u, u) specifying the cost between u and v. We then wish to find an acyclic subset T ∈ E that connects all of the vertices and whose total weight

$$w(T) = \sum_{(u,v)\in T} w(u, v)$$

is minimized.

Generic MST Algorithm

GENERIC_MST (G, w)

A = ϕ

while A does not form a spanning tree

find an edge (u, v) that is safe for A

A = A U {(u, v)}

return A

Definitions

- A cut (S, V - S) of an undirected graph G = (V, E) is a partition of V.
- We say that an edge (u, v) $\in$ E crosses the cut (S, V - S) if one of its endpoints is in S and the other is in V - S.
- We say that a cut respects a set A of edges if no edge in A crosses the cut
- An edge is a light edge crossing a cut if its weight is the minimum of any edge crossing the cut. Note that there can be more than one light edge crossing a cut in the case of ties.

Theorem

Let G = (V, E) be a connected, undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, let (S, V - S) be any cut of G that respects A, and let (u, v) be a light edge crossing (S, V - S). Then, edge (u, u) is safe for A.

Kruskal's Algorithm

MST_Kruskal (G, w)

```
A = ϕ
for each vertex v ∈ G.V
    MAKE-SET (v)
Sort the edges of G.E into non-decreasing order by weight w
for each edge (u, v) ∈ G.E taken in non-decreasing order by weight
    if FIND-SET(u) ≠ FIND-SET (v)
        A = A U {(u, v)}
return A
```

Prim's Algorithm

MST_PRIM (G, w, r)

for each u $\in$ G.V

u.key = ∞

u.π = NIL

r.key = 0

Q = G.V

while Q $\neq \phi$

u = EXTRACT_MIN (Q)

for each v $\in$ G.adj[u]

if v $\in$ Q and w(u, v) < v.key

v.π = u

v.key = w(u, v)