Chapter 4, Graphs:

- Concepts and implementation (Friday, Feb 14)
- Traversal (Wednesday, Feb 19 (plus lab))
- Minimum spanning trees (**Friday, Feb 21, and Monday, Feb 24**)
- Single-source shortest paths (Wednesday, Feb 26, and Friday, Feb 28)
- (Test 1 Wednesday, Mar 5)

"Today" (Friday and Monday):

- Finish graph traversal (analysis)
- MST problem definition
- Brute-force solution
- General structure of good solutions
- Kruskal's algorithm, plus proof and analysis
- Prim's algorithm, plus proof and analysis
- Performance comparison

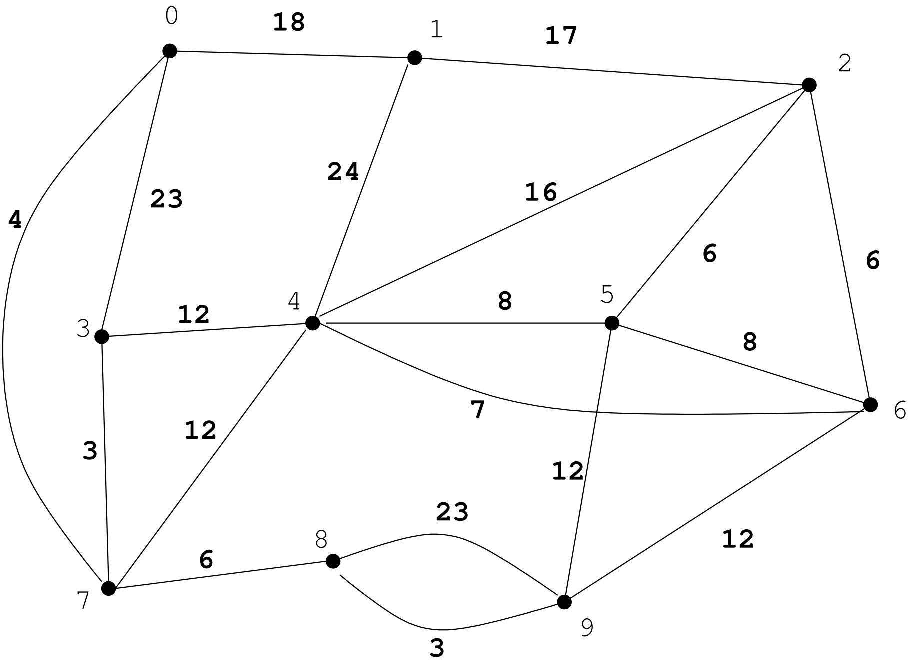
Minimum spanning tree problem

Given a weighted, undirected, connected graph, find minimum spanning tree:

Tree: A (sub)graph with no cycles. [We represent the tree a a set of edges.]

Spanning: All vertices in the original graph are included in the tree.

Minimum: For all spanning trees, this has least total weight.


0
18
1
17
2
24
23
16
4
6
6
4
8
5
12
3
8
7
3
12
6
12
23
12
8
6
7
9
3

General strategy for MST (both algorithms):

- Maintain a set of edges A that is a subset of a MST
- At each step, add one edge to A until it's a MST

Invariant (General MST main loop)

There exists $T \subseteq E$ such that T is a minimum spanning tree of G and $A \subseteq T$.

General algorithm outline:

$A = \emptyset$
While A isn't a MST
    add an edge to A that maintains the invariant

Insight 1: A implicitly partitions vertices into connected components. The lightest edge that connects two components is safe.

Lemma (Safe edges in Kruskal's algorithm.)

If $G = (V, E)$ is a graph, A is a subset of a minimum spanning tree for G, and (u, v) is the lightest edge connecting any distinct connected components of A, then (u, v) is a safe edge for A, that is, $A \cup \{(u, v)\}$ is a subset of a minimum spanning tree.

Proof. Suppose everything in the hypothesis, in particular that A is a subset of some minimum spanning tree T and that u and v are in distinct connected components of A, call them A_u and A_v. Let w_T be the total weight of T, that is, the sum of the weights of all the edges of T. We want to prove that adding (u, v) to A makes something that is still a subset of some minimum spanning tree.

If $(u, v) \in T$, then we're done. Suppose, then, that T does not contain (u, v). Since T is a spanning tree, it means that u and v are connected in T. Pick the lightest edge on the path from u to v that is not in A, call it (x, y). Essentially (x, y) is an edge that was picked instead of (u, v) that contributed to connecting A_u and A_v.

Snip out (x, y). This would disconnect T, that is, the graph $T - \{(x, y)\}$ is not a tree, but rather contains two connected components, one with u in it and the other with v in it. Now splice in (u, v). That will reconnect u and v and make it into a tree again. Formally we've made a new spanning tree $(T - \{(x, y)\}) \cup \{(u, v)\}$.

The hypothesis says that (u, v) was the lightest edge connecting distinct components of A. That means $w(u, v) \leq w(x, y)$. That in turn means that the total weight of the new spanning tree is also just as good, if not better, than the old one: $w_{T-\{(x,y)\})\cup\{(u,v)\}} \leq w_T$. Since it ties or beats a (supposed) minimum spanning tree, $(T - \{(x, y)\}) \cup \{(u, v)\}$ must be a minimum spanning tree. Therefore (u, v) is safe. $\square$

Invariant (General MST main loop)

There exists $T \subseteq E$ such that T is a minimum spanning tree of G and $A \subseteq T$.

Insight 1: A implicitly partitions vertices int connected components. The lightest edge that connects two components is safe.

Invariant (Prim's algorithm main loop)

A is a (single) tree.

Insight 2: The lightest edge that connects a new vertex to A is safe.

		Kruskal		Prim	
		Unoptimized	Optimized	Unoptimized	Optimized
Sparse	Adjacency list	31579	28841	72364	58089
	Adjacency matrix	49128	35493	67887	49537
Medium	Adjacency list	147527	54877	180407	113555
	Adjacency matrix	127485	59821	146358	75906
Dense	Adjacency list	136762	69867	191617	123762
	Adjacency matrix	162468	78154	130984	72245

Minimum Spanning Tree Problem

Given a weighted, undirected graph, find the tree with least-total weight that connects all the vertices, if one exists.

Single-Source Shortest Paths Problem

Given a weighted directed graph and a source vertex, find the tree comprising the shortest paths from that source to all other reachable vertices.

- Both are defined for weighted graphs
- Both produce trees as a result
- Both minmize by weight
- For each we have two algorithms

Input is only a graph
Problem usually is described on an undirected graph
Goal is to minimize total weight
There is no clear winner between the algorithms

Input is a graph and a starting point
Problem usually is described on a directed graph
Goal is to minimize weight on each path
One algorithm is clearly more efficient

Coming up:

Catch up on other projects...
Do **MST** *project (due Wed, Feb 26)*

Due **Mon, Feb 24**
Read Section 4.4
Do Exercises 4.(40, 42, 43)
(See comments on Canvas)
Take MST quiz

Due **Fri, Feb 28** *(end of day)*
Read Section 4.5
Do Exercises 4.(55–57)
Take SSSP quiz