CS 340 - lecture 28 - Nov 18

- no office hour on Nov 20
- instead: Nov 23, 12.30 pm - 2.00 pm

Kruskal's algorithm can be implemented with a union-find structure...

UNION-FIND / DISJOINT SET STRUCTURE

(for details, see Chapter 8 of textbook)

maintains a collection $S_1, S_2, \ldots, S_n$ of sets of objects,

where
- $S_1 \cup \ldots \cup S_n$ is the set of all objects
- $S_i \cap S_j = \emptyset$ for all i, j with $i \neq j$
  ($\rightsquigarrow$ "disjoint set structure")
- each set S_i has a distinguished object $r_i \in S_i$, called the "representative" of S_i

supports three main operations

(1) makeset(x) : creates a set S containing only x, with x as a representative

(2) union(r_i, r_j) : replace S_i, and S_j, by merged set $S_i \cup S_j$ and assigns a representative to $S_i \cup S_j$

(3) find(x) : finds the representative r of the set containing object x

There are clever implementations of UNION-FIND in time $O(N \log N)$, where N is the total number of objects.

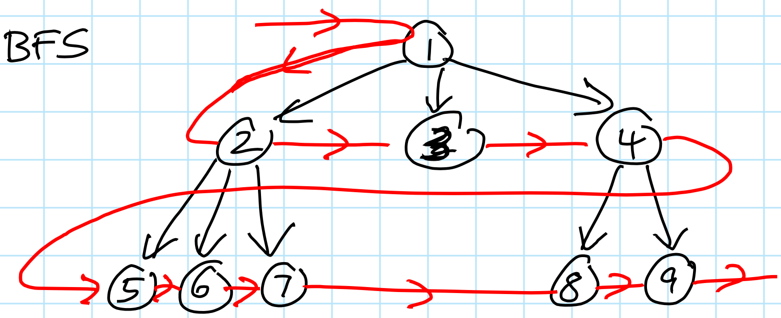
in Kruskal's algorithm:

- objects = edges, i.e., $S_1 \cup \ldots \cup S_n = E$
- a set S_i is a maximal set of edges added so far, represented as a tree
- "union" merges two sets when an edge is added that connects the two corresponding graph components

running time $O(|E| \log |E|)$

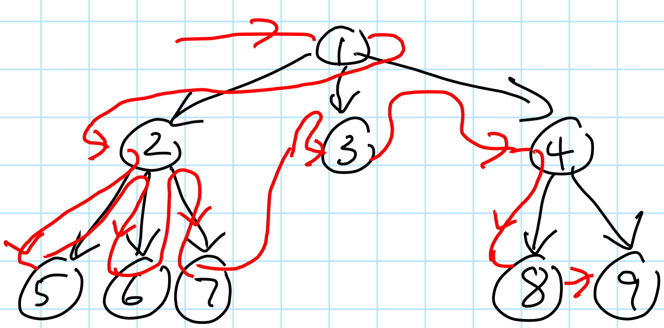
4.6 Depth-First Search

BFS



1, 2, 3, 4, 5, 6, 7, 8, 9

DFS



pre-order:

1, 2, 5, 6, 7, 3, 4, 8, 9

DFS on a general graph: need to keep track of visited vertices (to avoid cycles)

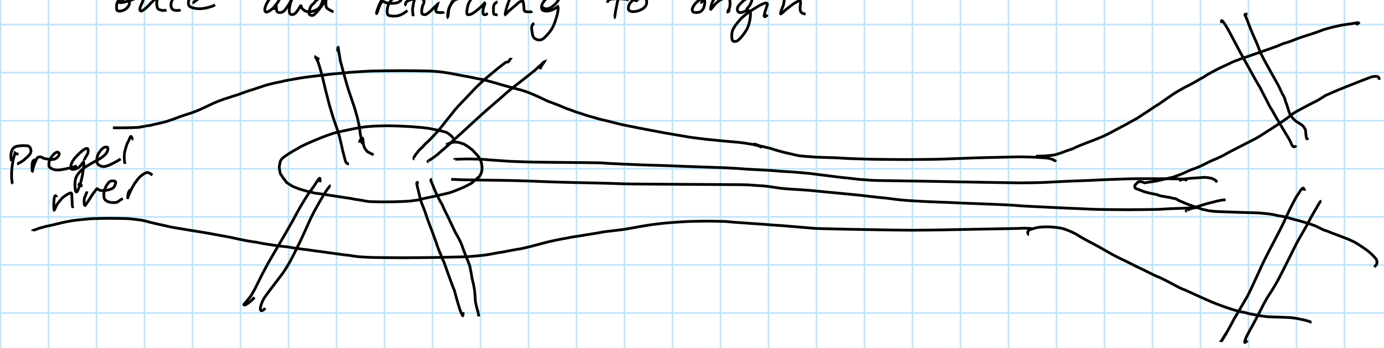
```
void Graph::dfs(Vertex v)
  {
    v.visited = true;
       :                        // e.g., output v
    for each Vertex w with (v,w) ∈ E { if (!w.visited) {dfs(w);}}
  }
```

applications:

- determine whether a given undirected (directed) graph is connected (strongly connected).
- finding "Euler circuits" in graphs (cycles using each edge exactly once)

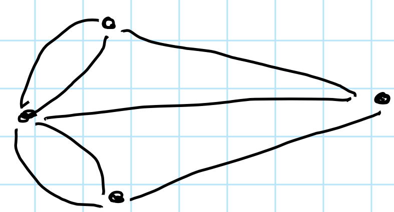
historical math problem: 7 Königsberg bridges

"walk through the city crossing each bridge exactly once and returning to origin"



L. Euler (1736) solved this: no such walk exists

Euler's translation: graph



needed is an Euler circuit, i.e., a cycle using each edge exactly once.

THEOREM: An undirected graph G has an Euler circuit if and only if G is connected and all its vertices have even degree

(degree of v = number of edges attached to it)

$\rightsquigarrow$ beginning of graph theory

5. Tractable, intractable, and unsolvable problems

5.1. The principle of reduction

One way of finding an alg. A solving a problem $\mathcal{P}$ is by reducing $\mathcal{P}$ to a problem $\mathcal{P}'$ for which we know an algorithm A', i.e.,

find a mapping <u>red</u> that transforms each problem instance x for $\mathcal{P}$ into a problem instance $red(x)$ for $\mathcal{P}'$ such that the solution to $red(x)$ produced by A' can be easily transformed to a solution to x.