Today: Greedy algorithms & Minimum Spanning Tree (MST)
- MST problem
- optimal substructure
- greedy-choice property
- Prim's algorithm
- Kruskal's algorithm

Recall: [Lecture 1]

Greedy algorithm: repeatedly make locally best choice/decision, ignoring effect on future
- saw greedy algorithm for scheduling problem
- Dijkstra's algorithm also ≈ greedy (cf. Bellman-Ford: incremental improvement)
- today: greedy algorithm for graph problem

Tree = connected graph with no cycles
Spanning tree of graph = subset of graph's edges that form a tree spanning (containing) all vertices
Minimum spanning tree (MST) problem:
given a graph $G = (V, E)$ & edge weights $w : E \to \mathbb{R}$,
find spanning tree $T \subseteq E$ of minimum weight:
$w(T) = \sum_{e \in T} w(e)$

Example:

Naïve algorithm: check all spanning trees
- exponential time

Greedy properties: problems amenable to greedy algorithms usually satisfy:

1. Optimal substructure: optimal solution to problem incorporate optimal solution(s) to subproblem(s)
   - essentially dynamic programming

2. Greedy-choice property: locally optimal choices lead to globally optimal solution
Optimal substructure for MST:
if \( e = (u, v) \) is an edge of some MST of \( G = (V, E, w) \):
- contract edge \( e \): merge vertices \( u \) & \( v \)
- if we get multiple copies of an edge, just keep lowest weight:

\[
\begin{align*}
G & \\
\text{contract} \\
\text{min} \{ w_1, w_2 \} \\
\end{align*}
\]

- if \( T' \) is an MST of \( G' = G/e \)
then \( T = T' \cup e \) is an MST of \( G \)
remap edges to decontract \( e \)

Proof:
- let \( T^* \) be an MST of \( G \) containing edge \( e \)
  \( \Rightarrow T^*/e \) is a spanning tree of \( G' \)
- \( T' \) is an MST of \( G' \)
  \( \Rightarrow w(T') \leq w(T^*/e) \)
  \( \Rightarrow w(T) = w(T') + w(e) \leq w(T^*/e) + w(e) = w(T^*) \)
Dynamic program attempt:
- guess an edge to put in MST
- contract to get new subproblem
- recurse
- decontract & add e

but # subproblems is exponential ::
greedy technique will make this polynomial! ::
Greedy-choice property for MST:

for any cut \((S, V\setminus S)\) in graph \(G = (V, E, w)\), any least-weight crossing edge \(e = \{u, v\}\) with \(u \in S \land v \in V\setminus S\) is in some MST of \(G\).

**Proof:** cut & paste argument

- consider an MST \(T\) of \(G\)
  - \(T\) has a path from \(u\) to \(v\)
  - \(u \in S \land v \in V\setminus S\), so the path has some edge \(e' = \{u', v'\}\) with \(u' \in S \land v' \in V\setminus S\)
  - then \(T' = T \setminus \{e'\} \cup \{e\}\) is a spanning tree of \(G\) & \(w(T') = w(T) - w(e') + w(e)\)
  - but \(e\) is a least-weight edge crossing \((S, V\setminus S)\)
    \[w(e) \leq w(e')\]
    \[w(T') \leq w(T)\]
  \[\Rightarrow T'\] is a MST too.

*modification only touches edge(s) crossing \((S, V\setminus S)\)*

Two algorithms based on different choices of cut \((S, V\setminus S)\).
Prim's algorithm: start with $|S|=1$ & grow from there
- maintain priority queue $Q$ on $V \setminus S$
  where $v.key = \min \{w(u,v) \mid u \in S\}$
- initially $Q$ stores $V \setminus \{S\}$
- $s.key = \emptyset$ for arbitrary start vertex $s \in V$
- for $v \in V \setminus \{S\}$: $v.key = \infty$
- until $Q$ empty:
  - $u = \text{Extract-Min}(Q)$ (add $u$ to $S$)
  - for $v \in \text{Adj}[u]$
    - if $v \in Q (v \notin S) \& w(u,v) < v.key$
      - $v.key = w(u,v)$ \textit{\& Decrease-Key}
      - $v.parent = u$
  - return $\{v \mid v.parent \notin S \} \setminus V \setminus \{S\}$

Correctness:
- invariant: $v \notin S \Rightarrow v.key = \min \{w(u,v) \mid u \in S\}$
- invariant: tree $T_s$ within $S \subseteq \text{MST of } G$
  - assume by induction: $\text{MST } T^* \supseteq T_s$
  - $S \rightarrow S' = S \cup \{e\}$
    where $e$ is a least-weight edge crossing cut $(S, V \setminus S)$
  - greedy cut & paste \textit{\&} can modify $T^*$
    to include $e$ without removing $T_s$
  - new $T^* \supseteq T_s = T_s \cup \{e\}$
Time: $\Theta(V) \cdot T_{\text{Extract-Min}} + \Theta(E) \cdot T_{\text{Decrease-Key}}$

$\frac{1}{\sqrt{v}} |\text{Adj}[v]| = \frac{1}{\sqrt{v}} \text{deg}(v) = 2 \cdot |E|$ (Handshaking Lemma)

<table>
<thead>
<tr>
<th>priority_queue</th>
<th>$T_{\text{Extract-Min}}$</th>
<th>$T_{\text{Decr.-Key}}$</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>array (nothing)</td>
<td>$O(V)$</td>
<td>$O(1)$</td>
<td>$O(V^2)$</td>
</tr>
<tr>
<td>binary heap</td>
<td>$O(lg V)$</td>
<td>$O(lg V)$</td>
<td>$O(E \cdot lg V)$</td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>$O(lg V)$</td>
<td>$O(1)$ [amortized]</td>
<td>$O(E + V \cdot lg V)$</td>
</tr>
</tbody>
</table>

(CLRS ch. 19)
Kruskal's algorithm: take globally lowest-weight edge & contract
- maintain connected components in MST-so-far \( T \)
in Union-Find structure \[ \text{[Recitation 3]} \]
- \( T = \emptyset \) \( \leftarrow \) will become MST
- for \( v \in V \): Make-Set \((v)\) \( \leftarrow \) initially, 1 vertex/comp.
- sort \( E \) by \( w \)
- for \( e = (u,v) \in E \) (in increasing weight order):
  - if \( \text{Find-Set}(u) \neq \text{Find-Set}(v) \): \( \leftarrow \) different components
  - add \( e \) to \( T \)
  - \( \text{Union}(u,v) \)

Correctness: invariant: tree \( T \) so far \( \subseteq \text{MST} \) \( T^* \)
- assume by induction \( T \subseteq T^* \)
- when adding \( e \) between components \( C_1 \) & \( C_2 \): use greedy-choice property on cut \((C_1 \cup V \setminus C_2)\)

Time: \( T_{\text{sort}}(E) + O(V) \cdot T_{\text{makeset}} + O(E) \cdot (T_{\text{find}} + T_{\text{union}}) \)
\( O(E \log E) \) tiny
\( O(E) \) e.g. if weights are integers \( \in [O, E^{O(1)}] \) \( \sim \) then can beat Prim

Best \( \text{MST} \) algorithm: \[ \text{[Karger, Klein, Tarjan 1993]} \]
\( O(V+E) \) expected time (randomized)