Math 182 Lecture 16

§8.4 Minimum spanning trees

Recall $G=(V,E)$ connected, undirected graph w/ weight function

$w: E \to \mathbb{R}$

Goal: find $T \subseteq E$ s.t. T tree, (V,T) connected (i.e., T "spanning tree") s.t.

$w(T) := \sum_{(u,v)\in T} w(u,v)$ is minimal.

Minimum-spanning-tree problem

We will use a greedy algorithm strategy to efficiently find a T.

Idea: Grow a set $A \subseteq E$ one edge at a time until we have a valid MST. We will maintain

(Loop Invariant) Prior to each iteration A is a subset of some MST.

Def: Given A satisfying LI, we call an edge $(u,v)\in E$ safe for A if $A \cup \{(u,v)\}$ also satisfies LI.

Prim's algorithm will follow general strategy:

GENERIC-MST(G, w)

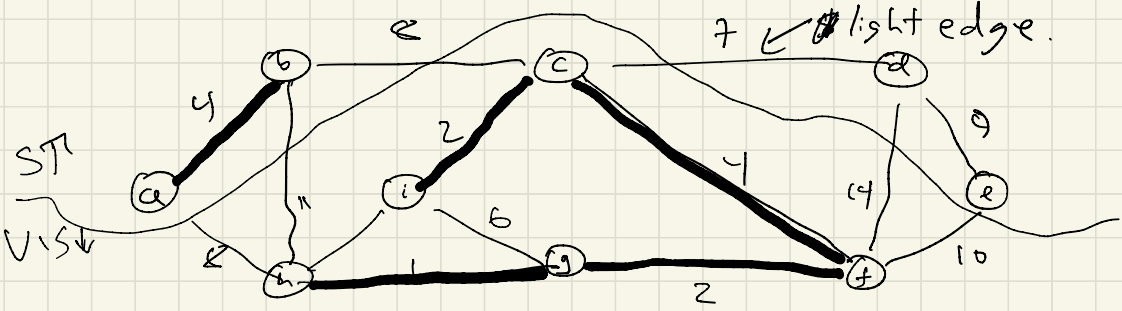
1 $A = \emptyset$ ← A satisfies LI
2 **while** A does not form a spanning tree
3 find an edge (u, v) that is safe for A
4 $A = A \cup \{(u, v)\}$ ← by def of "safe for A" A satisfies LI.
5 **return** A

run $|V|-1$ times → 3

This strategy preserves LI and returns a MST. Now will investigate safe edges.

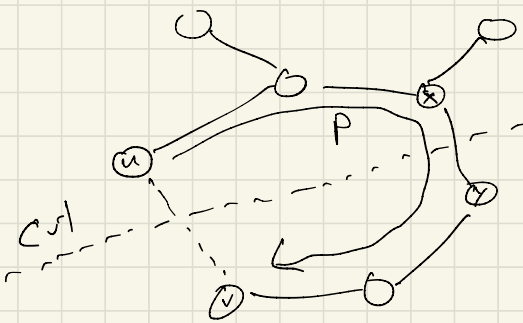
Defs • define a cut $(S, V \setminus S)$ of G to be a partition of V.

- We say edge (u,v) crosses the cut $(S, V \setminus S)$ if one endpoint is in S and other in $V \setminus S$.
- Given $A \subseteq E$, say cut respects A if no edge in A crosses the cut.
- Given a cut $(V, S \setminus V)$ call an edge (u,v) a light edge for the cut if it crosses the cut and has minimal weight among all such edges crossing cut.



Theorem 8.4.1. *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 $(S, V \setminus S)$ be any cut of G that respects A, and let (u, v) be a light edge crossing $(S, V \setminus S)$. Then, edge (u, v) is safe for A.*

Proof: (Exchange argument) Sps T is a MST s.t. $A \subseteq T$. Sps $(u,v) \in T$ then $A \cup \{(u,v)\} \subseteq T$ so (u,v) safe for A. O/w sps $(u,v) \notin T$. Adjoining (u,v) to T creates a cycle. u, v on opposite sides of crt. consider unique simple path p from u to v in T. p will contain edge (u,v) which crosses cut. cut respects A so $(x,y) \notin A$.



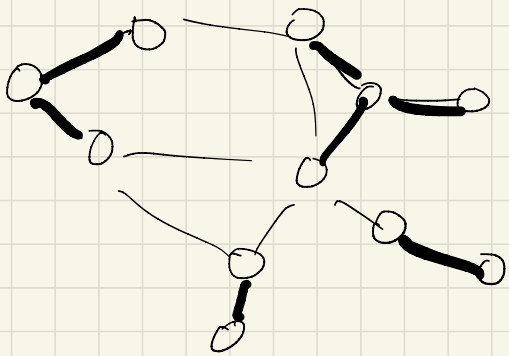
$w(u,v) \leq w(x,y)$

Consider $T' := T \setminus (x,y) \cup (u,v)$, then T is a spanning tree and $w(T') \leq w(T)$. Thus T' is a MST. ▩

Properties of Generic MST:

- A is acyclic always,
- Define $G_A = (V, A)$ is a forest and each connected component is a tree
- Adjoining safe edge connects two components of G_A, decreases # components by 1.

Corollary 8.4.2. *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, and let $C = (V_C, E_C)$ be a connected component (tree) in the forest $G_A = (V, A)$. If (u, v) is a light edge connecting C to some other component in G_A, then (u, v) is safe for A.*



Prim's algorithm gives efficient way to find safe edge.

- At all times A will form a single tree.
- Will use min-priority queue Q to keep track of light edge.
- Each v will have attribute
  $v.key$ = lightest edge from v to some vertex in A

and $v.\pi$ = vertex in A s.t. $(v.\pi, v)$ is lightest edge from v to A.

- The set A won't be explicitly mentioned, but can define $A = \{(v, v.\pi) : v \in V - \{r\} - Q\}$

When $Q = \emptyset$, $A = \{(v, v.\pi) : v \in V - \{r\}\}$ is MST.

first vertex in tree.

MST-PRIM(G, w, r)

```
1  for each u ∈ G.V
2      u.key = ∞
3      u.π = NIL
4  r.key = 0          ensures will process r first.
5  Q = G.V            put all vertices in min-priority queue
6  while Q ≠ ∅        |V|
7      u = EXTRACT-MIN(Q)      u has lightest edge to A.
8      for each v ∈ G.Adj[u]
9          if v ∈ Q and w(u, v) < v.key      update key values since A has grown.
10             v.π = u
11             v.key = w(u, v)
```

Correctness of MST-Prim uses following loop invariant:

At end of each time line 6 runs:

(1) $A = \{(v, v.\pi) : v \in V - \{r\} - Q\}$

(2) The vertices placed into the MST are those in $V - Q$

(3) For all $v \in Q$, if $v.\pi \neq NIL$, then $v.key < \infty$ and $v.key$ is minimum among all weights of edges v to A.

Running Time → Sps min-priority queue implemented w/ a heap.

Building Q takes $O(V)$ time
lines 1-5

In total: $O(V \lg V + E \lg V) = O(E \lg V)$
bcs $|E| \geq |V| - 1$ bcs G connected.

Chapter 9 Single-source shortest paths

§9.1 Single-source shortest paths

Idea: Want to know shortest driving distance from LA to all other cities.

Specifically: let $G=(V,E)$ be weighted directed graph. The weight $w(p)$ of a path $p = \langle v_0, v_1, \ldots, v_k \rangle$ is

$$w(p) := \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Define shortest-path weight $\delta(u,v)$

$$\delta(u,v) := \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if } \exists \text{ path} \\ \infty & \text{o/w} \end{cases}$$

If $\delta(u,v) < \infty$ define a shortest path to be any path $u \overset{p}{\rightsquigarrow} v$ s.t. $w(p) = \delta(u,v)$.

Finally, we're given source $s \in V$ and want to know all $\delta(s,v)$ and shortest paths.

Optimal substructure of a shortest path

Lemma 9.1.1 (Subpaths of shortest paths are shortest paths). *Given a weighted, directed graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$, let $p = \langle v_0, v_1, \ldots, v_k \rangle$ be a shortest path from vertex v_0 to vertex v_k and, for any i and j such that $0 \le i \le j \le k$, let $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ be the subpath of p from vertex v_i to vertex v_j. Then, p_{ij} is a shortest path from v_i to v_j.*

$$v_0 \rightsquigarrow v_i \rightsquigarrow v_j \rightsquigarrow v_k$$

Negative-weight edges

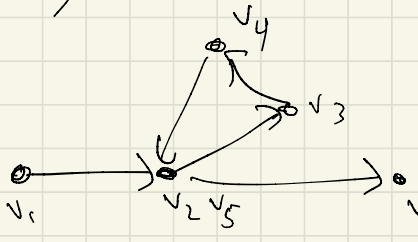
If a negative-weight cycle is reachable from s, then shortest path DNE. Define $\delta(s,v) = -\infty$ in this case. Won't allow negative-weight cycles.
O/w negative weights are o.k.

Cycles

No neg. weight cycles allowed.

Positive weight cycles cannot occur in shortest paths.

0-length cycles can occur, but can be removed.



$\langle v_1, v_2, v_3, v_4, v_5, v_6 \rangle$

Thus WLOG we shall assume shortest paths have no cycles.
thus length of shortest path $\leq |V| - 1$.

Representing shortest paths

In the algorithm, vertices will have attribute $v.\pi$ as before.

Have predecessor subgraph

$G_\pi = (V_\pi, E_\pi)$ $V_\pi = \{v \in V : v.\pi \neq NIL\} \cup \{s\}$

$E_\pi = \{(v.\pi, v) \in E : v \in V_\pi \setminus \{s\}\}$

G_π will converge to a shortest-path tree

by def is $G' = (V', E') \subseteq G$ s.t.

(1) V' = all reachable vertices from s

(2) G' is rooted tree w/ root s

(3) unique simple path from s to $v \in V'$ is a shortest path.

Relaxation