CS 25 - Algorithms

11/8/95

Last time (chap 23)
- Finish DFS
- Detecting cycles in U.G.

Today (chap 23, 24)
- DAGs / Detection
- Topological sort
- SCC
- MST

Announcements
- Exam Stats
- Midterm grades

Last time (chap23)
- DAGs: Detecting Cycles
- Topological Sort
- SCC

Today (chap 23, 24)
- Finish SCC
- MST

L23, P1

11/10/95

CS25-X94 Lecture 19 8/8/94

MST

Examples:

- wire together electronic components with least amt of wire
- make communication network that's fully connected w/ least cable
- also useful in approximations for NP-comp. problems, e.g., TSP.

Input: undirected $G = (V, E)$, connected
weight function $w: E \to \mathbb{R}$

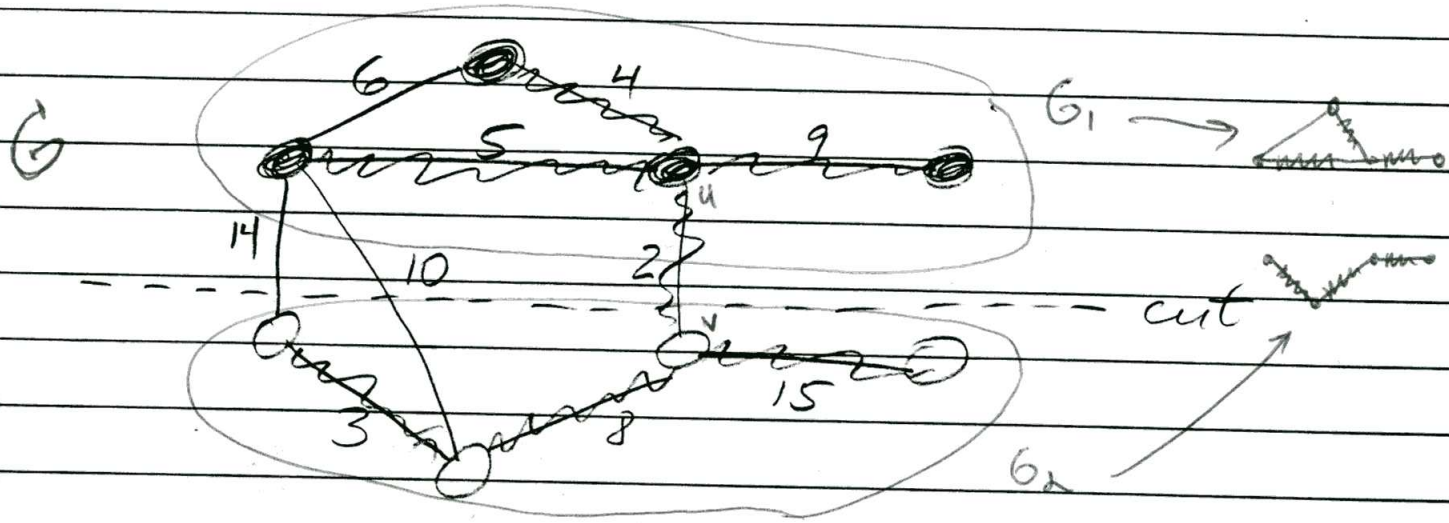
Output: acyclic subset $T \subseteq E$ connecting all vertices and whose total weight

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

is minimum.

T acyclic and T connects all vertices
$\Rightarrow$ T is a tree
$\Rightarrow |T| = |V| - 1$

We'll see a greedy approach to computing MST. It'll also be a little different in that we'll see a generic alg. then 2 implementations of it: Kruskal's alg & Prim's alg.



Optimal substructure

Let T be MST, (u,v) be any edge in T.
Removing (u,v) partitions T into T_1 and T_2.
Let $G_1 = (V_1, E_1)$, where

V_1 = vertices in T_1 incident from

$E_1 = \{(u,v) : u,v \in V_1\}$ edges in T_1

same for G_2.

Claim: T_1 is MST for G_1,
T_2 " " " G_2.

Proof: $w(T) = w(u,v) + w(T_1) + w(T_2)$.
If $\exists$ better tree for T_1 or T_2, could use it to get better tree than T. ⊠

Generic MST algorithm

- Grow MST one edge at a time.
- Grows set $A \subseteq MST$.

Def: If $A \subseteq$ some MST and $A \cup \{(u,v)\} \subseteq$ some MST, then (u,v) is a safe edge for A.

```
Generic-MST(V, E, w)
A ← ∅
while |A| < |V| − 1
    do find an edge (u,v) that is safe for A
        A ← A ∪ {(u,v)}
return A
```

How to find a safe edge?

- Cut $(S, V-S)$ is partition of V.
  Show example of cut from above example.
- Edge (u,v) crosses cut $(S, V-S)$ if one endpt in S and other in $V-S$.
- Cut respects set A of edges if no edge in A crosses the cut.
- Light edge crossing a cut has minimum weight of any edge crossing the cut.
  Note: can be > 1 light edge because of ties.

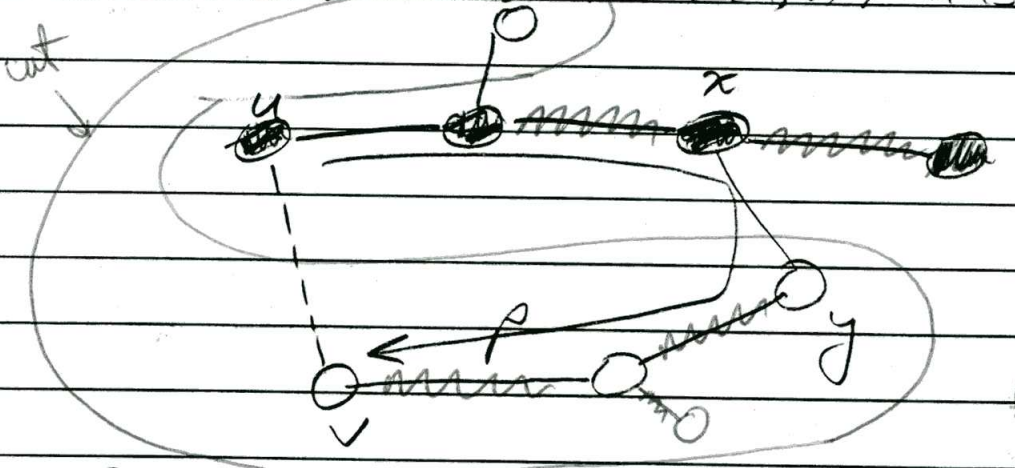
Min. weight edge crossing cut

Theorem: If ① $A \subseteq$ (some MST), ② cut $(S, V-S)$ respects A and ③ (u,v) is a light edge crossing cut $(S, V-S)$, then (u,v) is safe for A.

Proof: Let T be a MST, $A \subseteq T$. If $(u,v) \in T$, done. Otherwise, will show $\exists$ another MST T' s.t. $A \cup \{(u,v)\} \subseteq T'$.

Idea: cut & paste.

T is a tree $\Rightarrow T \cup \{(u,v)\}$ has a cycle.



black → S
white → V-S
edges → MST T
~~~ → A

Note: only tree edges & (u,v) shown; other edges may exist

- Cycle is $p \cup \{(u,v)\}$. $p \subseteq T$.
- u, v on opposite sides of $(S, V-S) \Rightarrow$ $\exists$ edge $(x,y) \in T$ on p crossing $(S, V-S)$.
- $(x,y) \notin A$ because $(S, V-S)$ respects A.
- (x,y) on unique path $u \rightsquigarrow v$ in $T \Rightarrow$ removing (x,y) breaks T into 2 components.
- Let ~~Define~~ $T' = T - \{(x,y)\} \cup \{(u,v)\}$. T' is a spanning tree.

(but is it minimum?)
~~~

Reductions to Graph Algs

3 steps
(Methodology)

1. Vertices – correspond to objects or subsolutions to problem
2. Edges – correspond to choices that can be made

Graph	Problem
path	$\Leftrightarrow$ valid solution

3. Edge Weights – correspond to cost of choice

Graph	Problem
weight of path	$\Leftrightarrow$ cost of valid solution

min weight path $\Leftrightarrow$ min cost solution ✓

Homework:
- Chess board
- rental car

In class:
- coin changing
- pretty printing

CS-25 Algorithms

11/13/95

Last time (chap 23,24)

- Faith SCC
- MST

Today (chap24)

- MST - Kruskal

Last time (chap24)

- MST - Kruskal

Today (chap24,25)

- MST - Prim
- SSSP - BF
  - Dykstra

Handin

- ME solution

L25, P1

11/15/95

Review

CS25-X94 L19 P3

Generic MST algorithm

Grow MST one edge at a time.

Grows set $A \subseteq MST$.

If $A \subseteq$ some MST and $A \cup \{(u,v)\} \subseteq$ some MST, then (u,v) is a safe edge for A.

```
Generic-MST(V, E, w)
A ← ∅
while |A| < |V| − 1
    do find an edge (u,v) that is safe for A
        A ← A ∪ {(u,v)}
return A
```

How to find a safe edge?

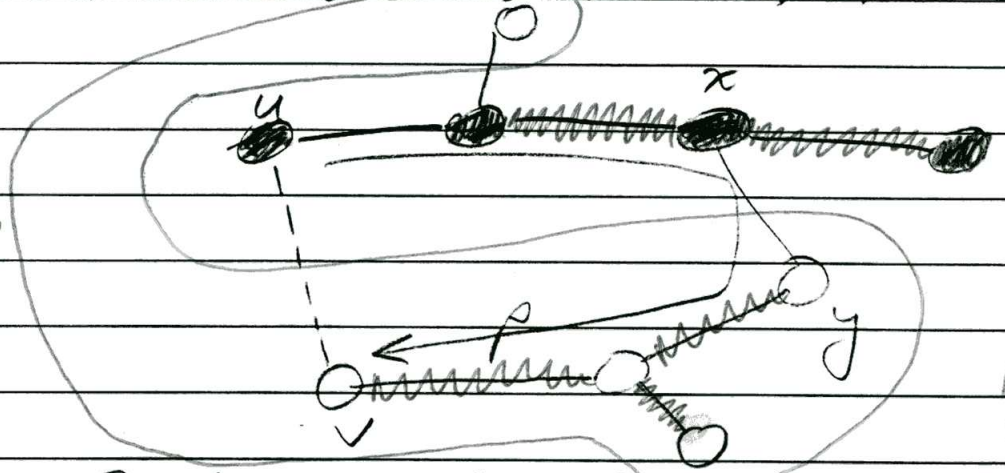
- Cut $(S, V-S)$ is partition of V.
  Show example of cut from above example.
- Edge (u,v) crosses cut $(S, V-S)$ if one endpt in S and other in $V-S$.
- Cut respects set A of edges if no edge in A crosses the cut.
- Light edge crossing a cut has minimum weight of any edge crossing the cut.
  Note: can be > 1 light edge because of ties.
  minimum weight edge crossing the cut

Theorem: If ① $A \subseteq$ (some MST), ② cut $(S, V-S)$ respects A and ③ (u,v) is a light edge crossing cut $(S, V-S)$, then (u,v) is safe for A.

Proof: Let T be a MST, $A \subseteq T$. If $(u,v) \in T$, done. Otherwise, will show $\exists$ another MST T' s.t. $A \cup \{(u,v)\} \subseteq T'$.

Idea: cut & paste.

T is a tree $\Rightarrow T \cup \{(u,v)\}$ has a cycle.



- Cycle is $p \cup \{(u,v)\}$. $p \subseteq T$.
- u, v on opposite sides of $(S, V-S) \Rightarrow$ $\exists$ edge (x,y) on p crossing $(S, V-S)$.
- $(x,y) \notin A$ because $(S, V-S)$ respects A.
- (x,y) on unique path $u \rightsquigarrow v$ in $T \Rightarrow$ removing (x,y) breaks T into 2 components.
- Let $T' = T - \{(x,y)\} \cup \{(u,v)\}$. T' is a spanning tree. (but is it minimum?)

Claim: T' is an MST.

Proof of claim: (u,v) is light edge crossing $(S, V-S)$ and (x,y) crosses $(S, V-S)$ $\Rightarrow$ $w(u,v) \leq w(x,y)$.

$w(T') = w(T) - w(x,y) + w(u,v)$

$\leq w(T)$.

In fact, if T is MST, then $w(u,v) = w(x,y)$.

T is MST $\Rightarrow$ T' is MST. ☒ claim

Finally, show that (u,v) is safe for A by showing $A \cup \{(u,v)\} \subseteq T'$.

$A \subseteq T$ and $(x,y) \notin A \Rightarrow A \subseteq T' \Rightarrow A \cup \{(u,v)\} \subseteq T'$

$\Rightarrow$ (u,v) is safe for A. ☒

* Generic alg adds an edge at a time.
  Maintains a forest, with some # of trees.
  How many depends on the implementation.

* Corollary: Let $A \subseteq$ some MST and C be a connected component in the forest $G_A = (V, A)$. If (u,v) is a light edge connecting C to some other comp. in G_A, then (u,v) is safe for A.

Proof: $(C, V-C)$ respects A and (u,v) is light for $(C, V-C)$. ☒

Kruskal's algorithm

Chooses as the safe edge the min-weight edge connecting 2 distinct trees in the forest.

(u,v) has min weight over all (u,v) s.t. $u \in$ some C and $v \notin C$.

Problems:
- Given vertex u, how to determine which CC it's in?
- When adding (u,v) to A, we combine 2 CC's into 1. How?

Answer: Disjoint-set union DS

```
Kruskal(V, E, w)
A ← ∅
for each vertex v ∈ V
    do Make-Set(v)
sort edges of E by nondecreasing weight w
for each edge (u,v) ∈ E, in order by nondec. weight
    do if Find-Set(u) ≠ Find-Set(v)
          then A ← A ∪ {(u,v)}
               Union(u,v)
return A
```

Run on example.

Time: $O(V)$ to initialize
$O(E \lg E)$ to sort
$O(E \lg^* V)$ to run loop
Total: $O(E \lg E) = O(E \lg V)$ since $E = O(V^2)$

abuse of notation:
A - set of edges
A - set of vertices incident on those edges

Prim's algorithm
A is always a single tree.
Add lightest edge connecting A to V−A.

How to find lightest such edge?
Maintain priority queue Q for all vertices in V−A.
key[v] = min weight of any edge connecting v to some $u \in A$.
key[v] = ∞ if no such edge.

Output: a rooted tree, where $\pi[v] = u$ if u is v's parent. $\pi[root] = NIL$.

```
Prim(V,E,w,r)  ▷ r is arbitrary root
Q ← empty
for each u ∈ V − {r}
    do key[u] ← ∞
       Insert(Q, u)
key[r] ← 0
Insert(Q, r)
π[r] ← NIL
while Q not empty
    do u ← Extract-Min(Q)
       for each v ∈ Adj[u]
           do if v ∈ Q and w(u,v) < key[v]
                 then π[v] ← u
                      Decrease-Key(Q, v, w(u,v))
```

and output? No, just check vertices later to see which parent it has

just need a boolean variable to check this.

GOT THIS FAR

Do example.

Time: Depends on implementation of priority que

Extract-Min: $|V|$ times Insert: $|V|$ times

Decrease-Key: $\leq |E|$ times

Time $= O(V \cdot T_{\text{extract-min}} + E \cdot T_{\text{decrease-key}} + V \cdot T_{\text{insert}})$

Q	Insert	$T_{\text{extract-min}}$	$T_{\text{decrease-key}}$	Total time
unsorted array	$O(1)$	$O(V)$	$O(1)$	$O(V^2)$
binary heap / (just heap)	$O(\lg V)$	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
F-heap	$O(1)$	$O(\lg V)$ amort.	$O(1)$ amort	$O(V \lg V + E)$ worst case

Compare Kruskal to Prim $O(E \log V)$ vs. $O(V \log V + E)$

- Prim w/ F-heap seems to win, but...

But

	Kruskal	Prim array	Prim binary/binomial heap	Prim F-heap	
sparse graph $E = O(V)$	$O(V \log V)$	$O(V^2)$	$O(V \log V)$	$O(V \log V)$	probably go with Kruskal or Prim/binary heap
dense graph $E = \Omega(V^2)$	$O(V^2 \log V)$	$O(V^2)$	$O(V^2 \log V)$	$O(V^2)$	Probably go with Prim/array
$E = \Theta(V \log V)$	$O(V \log^2 V)$	$O(V^2)$	$O(V \log^2 V)$	$O(V \log V)$	

↑ due to constants

if you know that graph is sparse or dense