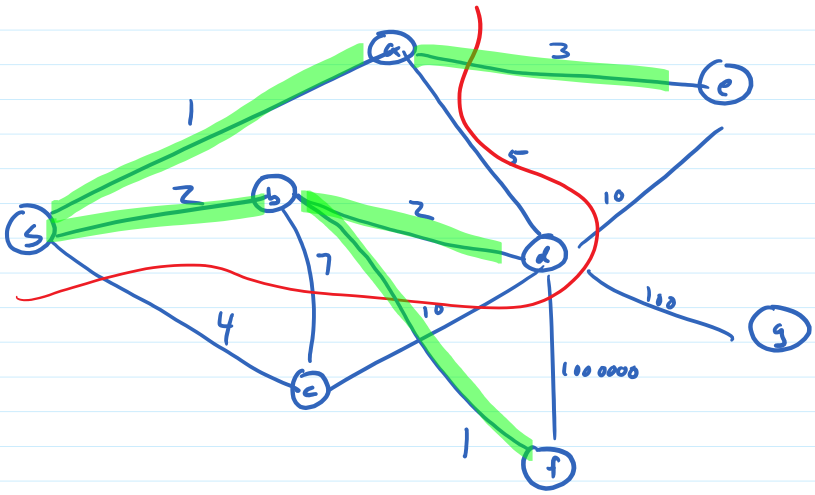
Prim's Algorithm



Prim

$\cancel{\text{Dijkstra}}(G, \ell)$

  $S \leftarrow \{s\}$

  $d(s) \leftarrow 0$

  while $S \neq V$

    choose $v \notin S$ to minimize $d'(v) = \min\limits_{\substack{u \in S \\ (u,v) \in E}} \cancel{d(u) +} \ell(u,v)$

$Q \leftarrow \emptyset$

$S \leftarrow \{s\}$

$\cancel{d[s] \leftarrow 0}$

$\pi[s] \leftarrow NIL$

for $v \in V, v \neq s$

  if $(s,v) \in E$

    $d'[v] \leftarrow \ell(s,v)$

    $\pi[v] \leftarrow v$

  else

    $d'[v] \leftarrow \infty$

    $\pi[v] \leftarrow NIL$

  $Q.\text{enqueue}(v, d'[v])$

while $Q \neq \emptyset$

  $v = Q.\text{extractMin}()$

  $\cancel{d[v] \leftarrow d'[v]}$ $A \leftarrow A \cup \{(\pi[v], v)\}$

cut $= S, Q$ $S \leftarrow S \cup \{v\}$

  for $(v,w) \in E$ where $w \in Q$

    if $\cancel{d[v]} + \ell(v,w) < d'(w)$

      $d'[w] = \cancel{d[v]} + \ell(v,w)$

      $\pi[w] = v$

      $Q.\text{decreasePriority}(w, d'[w])$

return $\cancel{(d, \pi)}$ A

INV: A is a proto-MST

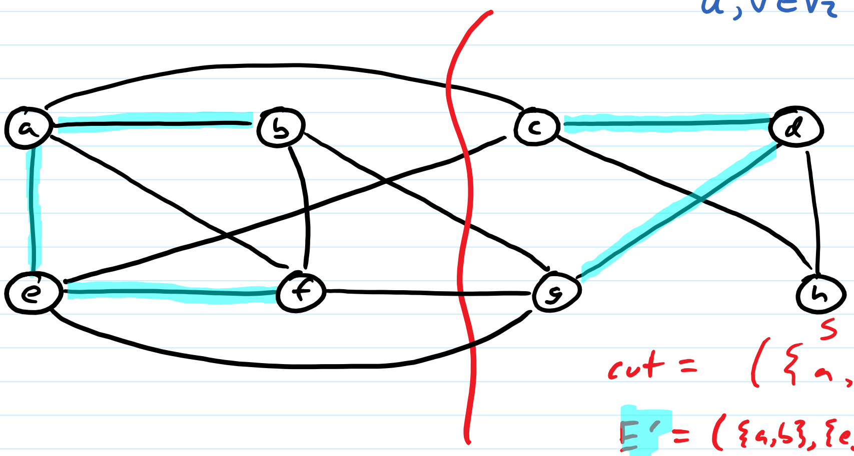
A connects S

for $w \in Q$, $(\pi[w], w)$ is min-weight across cut going to w, and $d'[w]$ is its weight

For an undirected graph $G = (V, E)$

a cut partition of V into S, $V-S$

a subset of edges E' respects cut (V_1, V_2) if, for all $(u,v) \in E'$
$u, v \in V_1$ or
$u, v \in V_2$

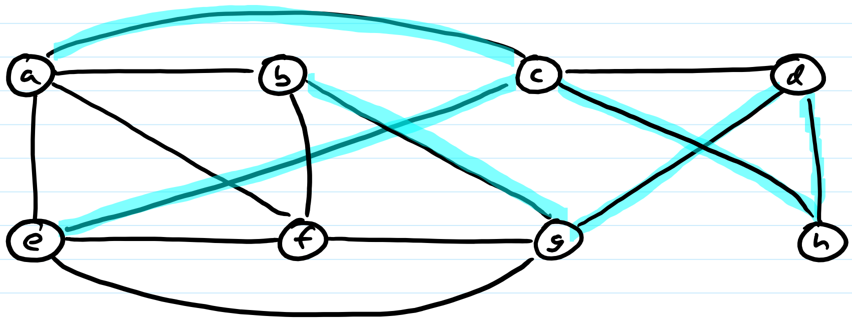


S

cut $= (\{a,b,e,f\}, \{c,d,g,h\})$

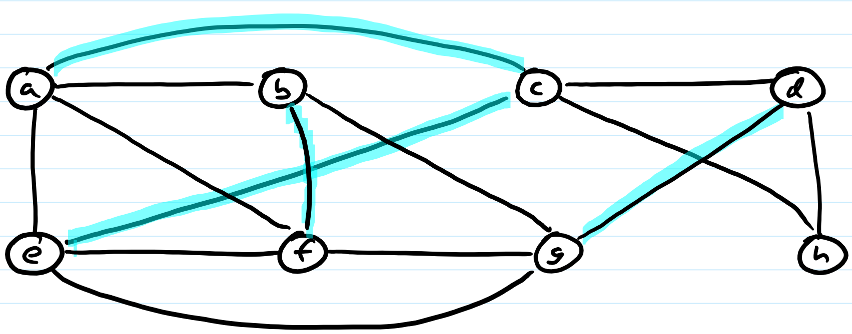
$E' = (\{a,b\}, \{e,f\}, \{a,e\}, \{c,d\}, \{d,g\})$

E' respects the cut



Does E' respect cut $(\{a,d,g\}, \{b,c,e,f,h\})$?

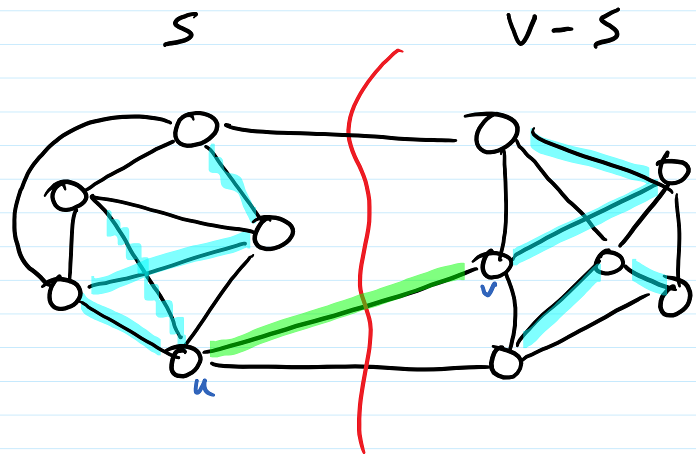
NO (a, c)



Cut Property / Light Edge Thm

Thm: If
1) $G = (V, E)$ is an undirected, weighted graph with $w(u,v) \geq 0$ for all $(u,v) \in E$
2) $(S, V-S)$ is a cut of G
3) $A \subseteq E$ is an acyclic subset of E that respects $(S, V-S)$
4) A is a proto-MST (A is a subset of some MST T)
5) (u,v) is min-weight edge that crosses $(V, V-S)$

then $A \cup \{(u,v)\}$ is a proto-MST



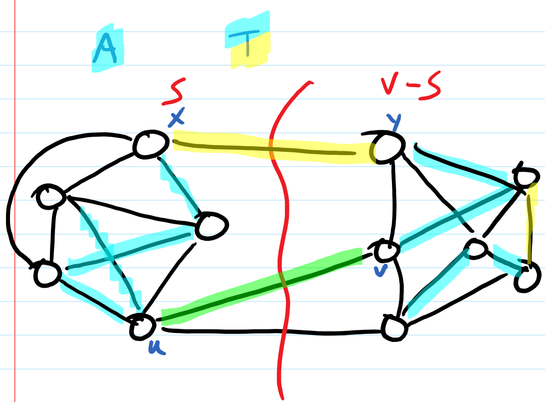
Proof: Suppose $G, S, A, \underline{(u,v)}$ satisfy 1-5 above.
light edge across $(S, V-S)$; A respects S

Find MST T so that $A \subseteq T$

Two cases: 1) $(u,v) \in T$. Then $\underbrace{A \cup \{(u,v)\}}_{\text{this is a proto MST}} \subseteq T$

2) $(u,v) \notin T$

Then there is some edge (x,y) on path $u \rightsquigarrow v$ in T so that $x \in S$, $y \in V-S$



Let $T' = \underbrace{T - \{(x,y)\}}\, \cup \{(u,v)\}$

↳ $T - \{(x,y)\}$ has 2 connected components: that of x and u and that of y and v

adding $\{(u,v)\}$ connects those connected components

So T' connects V

And T' is acyclic

And T' is spanning tree

Furthermore, $w(T) \leq w(T')$ since T is MST

$= w(T) - \ell(x,y) + \ell(u,v)$

$= w(T) + \underbrace{(\ell(u,v) - \ell(x,y))}_{\leq 0 \text{ since } (u,v) \text{ is min-weight across cut}}$

$\leq w(T)$ squeeze - all $\leq$ are =

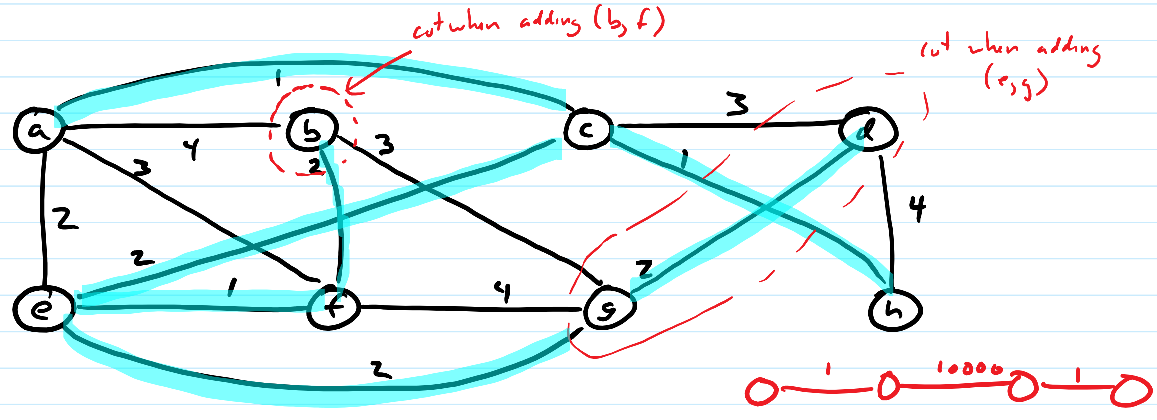
$w(T') = w(T)$ so T' is an MST

and $\underbrace{A \cup \{(u,v)\}}_{\text{proto-MST}} \subseteq T'$

INVARIANT: T is proto-MST

GENERIC-MST
$T \leftarrow \emptyset$
while $|T| < n-1$
find min-weight edge (u,v) across some cut that T respects
$T \leftarrow T \cup \{(u,v)\}$
return T

Kruskal's Algorithm: consider edges in order of ↑ weight
add edge if connects two different components of proto-MST



{e,f} {c,h} {c,a} {d,g} {c,e} {b,f} {~~a,e~~} {e,g} {a,f} {b,g} {c,d} {a,d} {d,h}

connected components {d,g} {a, c, e, f, b, h}

Disjoint Set Forest (partition): add(v): adds {v} to partition
find(v): returns elt of part of partition v is in; same elt returned for all input elts in that part
representative
union(u,v): merges u, v's part of the partition

MST-KRUSKAL(G) precondition: G is connected, no neg-weight edges

sort edges in order of nondecreasing weight $O(m \log n)$ $(= O(m \log m))$

b/c $n-1 \le m \le n^2$

$\log n-1 \le \log m \le 2 \log n$

$A \leftarrow \emptyset$

for each $v \in V$

n $P.add(v)$ $\}$ $O(n)$ total

for each edge (u,v) in order of sort

m if $P.find(u) \ne P.find(v)$ ← $O(m)$ total

$A \leftarrow A \cup \{(u,v)\}$

n-1 $P.union(u,v)$ $O(n \log n)$ total

Disjoint Set Data Structure

ADD (u) : add {u} to partition

FIND-SET (u) : return representative of u's part of partition

UNION (u,v) : merge u, v's parts of partition

UNION (e,f)
UNION (c,h)
UNION (c,a)
UNION (d,g)
UNION (c,e)
UNION (b,f)
UNION (e,g)

a	b	c	d	e	f	g	h
0	1	~~2~~	3	~~4~~	~~5~~	~~6~~	~~7~~
					4		~~2~~
		0					0
						3	
				0			

array comp[v] for index of v's component
rep[i] for representative of component i

FIND(v): return rep[comp[v]] $O(1)$:)
UNION(u,v): change all occurrences of comp[v] to comp[u] $O(n)$:(