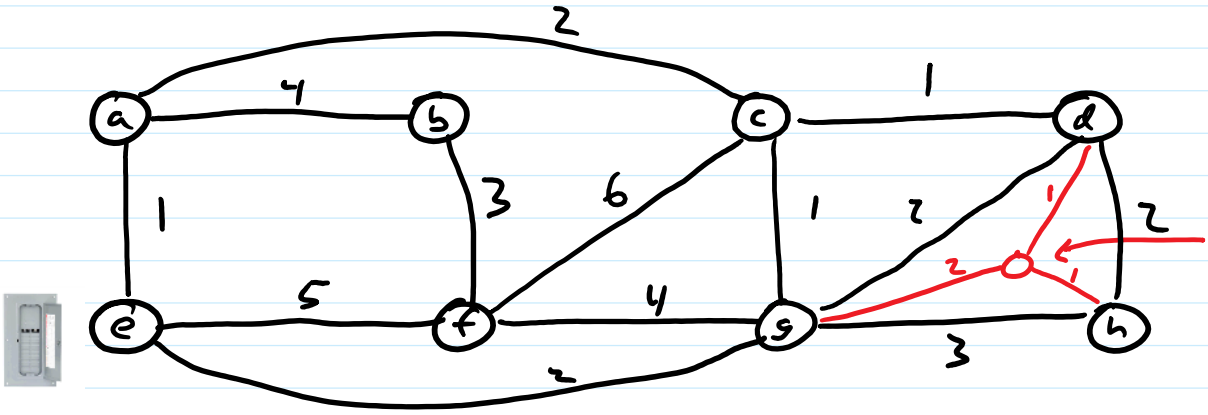
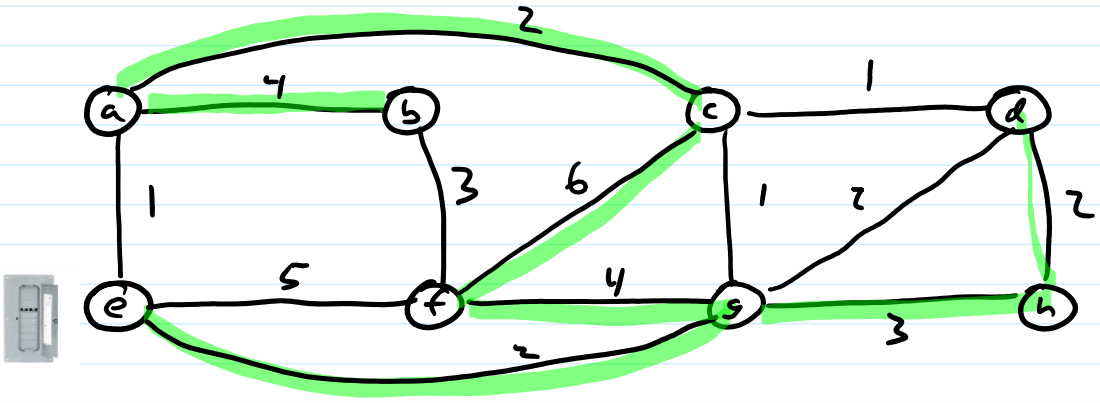
Minimum Spanning Tree



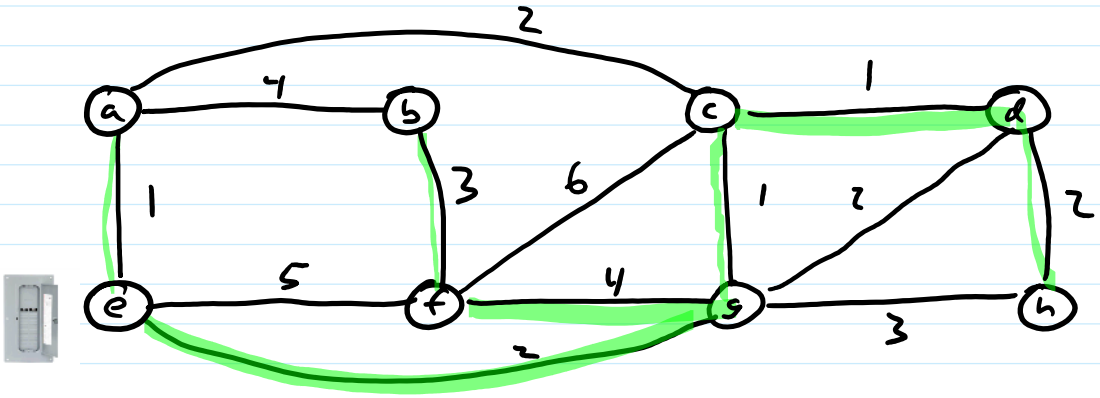
Minimum Spanning Tree



4+2+6+4+2+3+2=23

spanning tree: a subset of edges that forms a tree and connects all vertices

minimum spanning tree: a spanning tree of minimum total weight



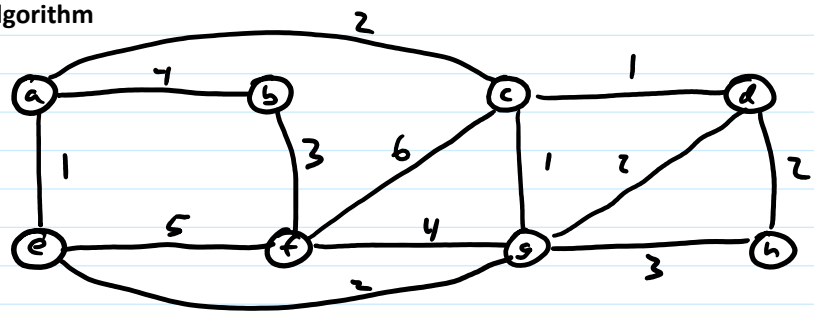
4+2+6+4+2+3+2=23

1+2+1+4+3+1+2=14

spanning tree: a subset of edges that forms a tree and connects all vertices

minimum spanning tree: a spanning tree of minimum total weight

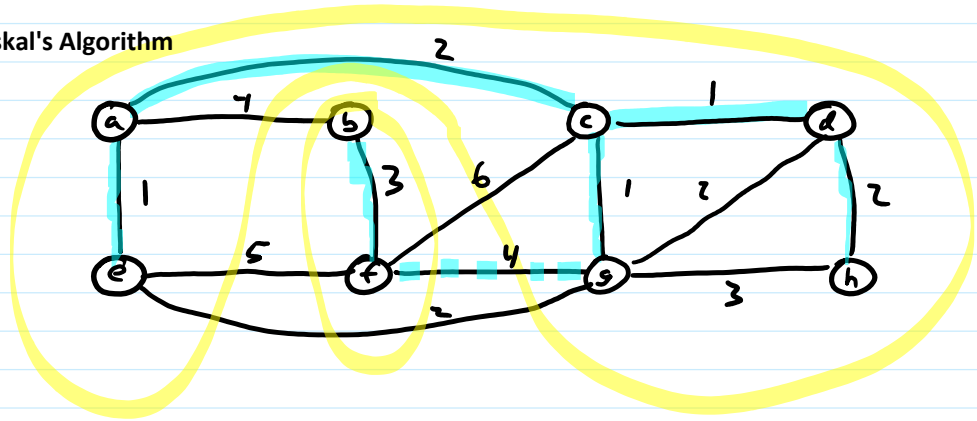
Kruskal's Algorithm



sort edges in order of increasing weight (a,e)(c,g)(c,d)(d,g)(a,c)(d,h)(e,g)(g,h)(b,f)(f,g)(a,b)(c,f)

for each edge (u,v) in order of sort
    if u and v aren't connected using previously selected edges
        add (u,v)

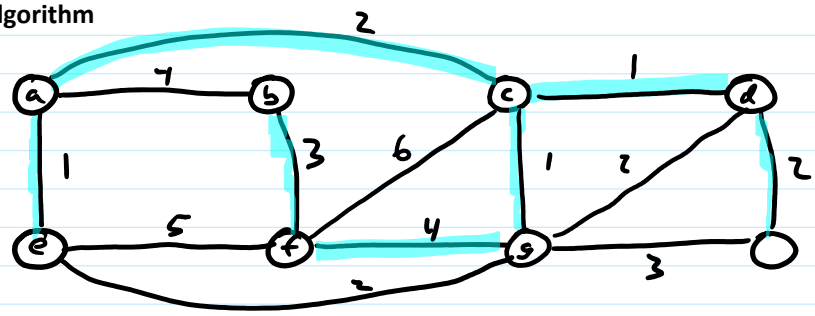
Kruskal's Algorithm



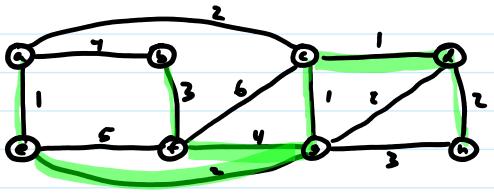
sort edges in order of increasing weight (a,e)(c,g)(c,d)(d,g)(a,c)(d,h)(e,g)(g,h)(b,f)(f,g)(a,b)(c,f)

for each edge (u,v) in order of sort
    if u and v are connected using previously selected edges
        add (u,v)

Kruskal's Algorithm



1+2+1+4+3+1+2 = 14

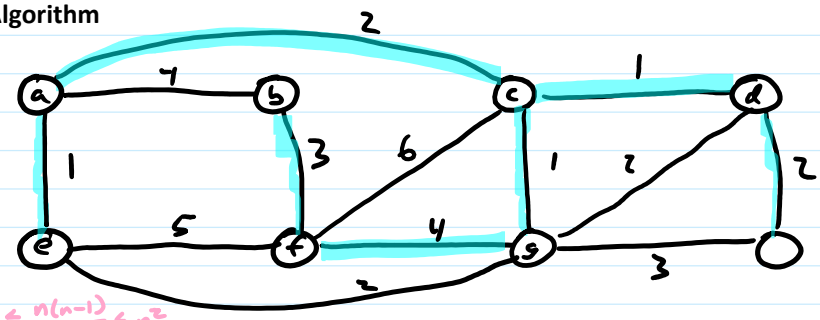


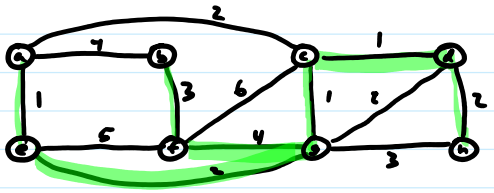
1+2+1+4+3+1+2 = 14

sort edges in order of increasing weight (a,e)(c,g)(c,d)(d,g)(a,c)(d,h)(e,g)(g,h)(b,f)(f,g)(a,b)(c,f)

for each edge (u,v) in order of sort
    if u and v are connected using previously selected edges
        add (u,v)

Kruskal's Algorithm





$n-1 \le m \le \frac{n(n-1)}{2} \le n^2$

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

$\log m \in \Theta(\log n)$

$1+2+1+4+3+1+2 = 14$

$1+2+1+4+3+1+2 = 14$

PRE: input is a connected, undirected graph with positive weights

POST: output is an MST

sort edges in order of increasing weight (a,e)(c,g)(c,d)(d,g)(a,c)(d,h)(e,g)(g,h)(b,f)(f,g)(a,b)(c,f)

→ worst case $\Theta(m \log n)$

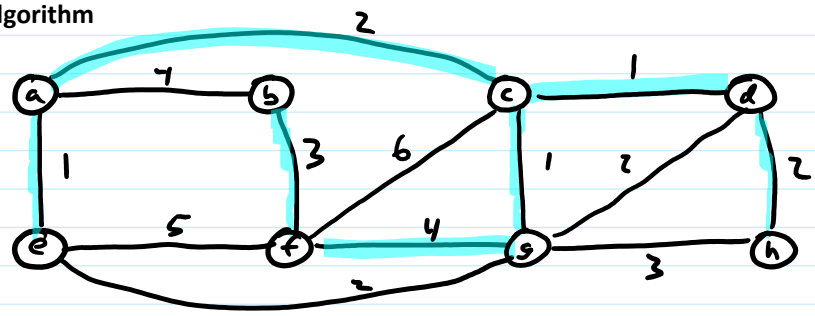
for each edge (u,v) in order of sort — worst case m iterations

if u and v are connected using previously selected edges — BFS/DFS: worst case $\Theta(n)$ ($\le n$ edges selected)

add (u,v)

total: worst-case $\Theta(nm)$

Kruskal's Algorithm



connected components

e a c g d h b f

initialize (V)

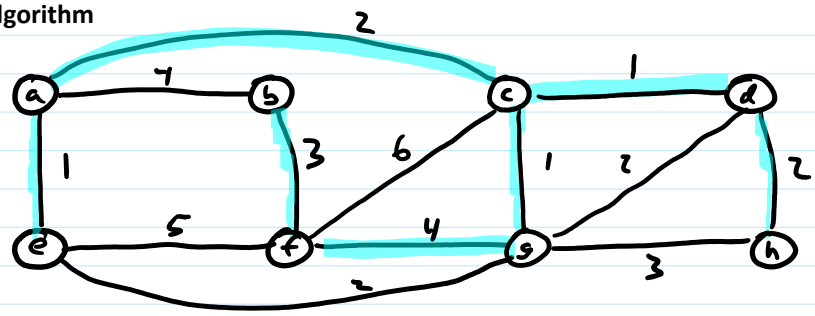
sort edges in order of increasing weight (a,e) (c,g) (c,d) (d,g) (a,c) (d,h) (e,g) (g,h) (b,f) (f,g) (a,b) (c,f)

```
for each edge (u,v) in order of sort
        if  find-set(u) ≠ find-set(v)
              add (u,v)
              union(u,v)
```

worst-case $\theta(m \log n)$

UNION/FIND ADT (or disjoint set, or partition):
initialize (S): make singletons for each set S
union(u,v): merge sets containing u, v
find-set(u): find representative of set containing u

Kruskal's Algorithm



connected components

e a c g d h b f

sort edges in order of increasing weight (a,e)(c,g)(c,d)(d,g)(a,c)(d,h)(e,g)(g,h)(b,f)(f,g)(a,b)(c,f)

```
for each edge (u,v) in order of sort
        if  find-set(u) ≠ find-set(v)
              add (u,v)
              union(u,v)
```