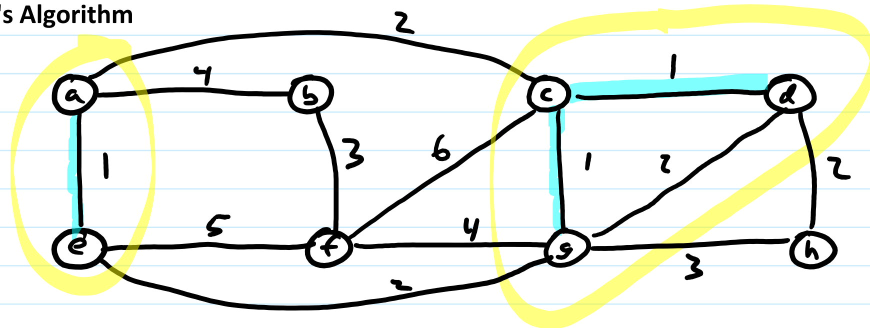
Kruskal's Algorithm



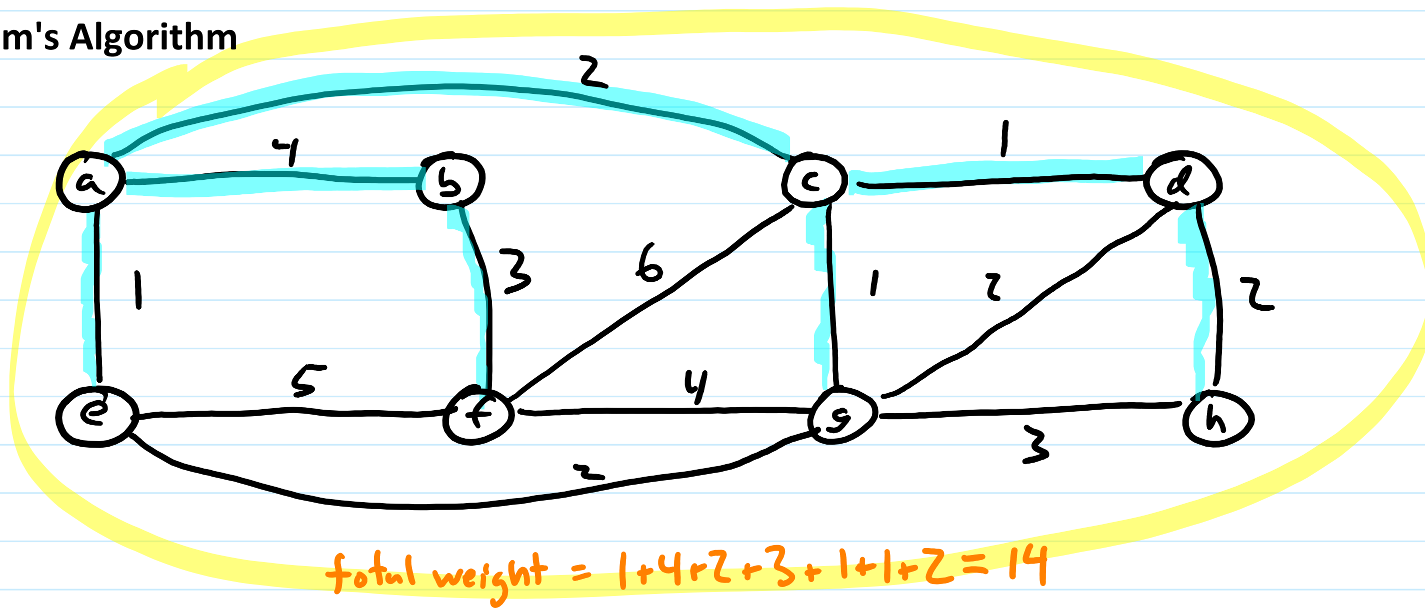
connected components

e a
b
c d g
f
h

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

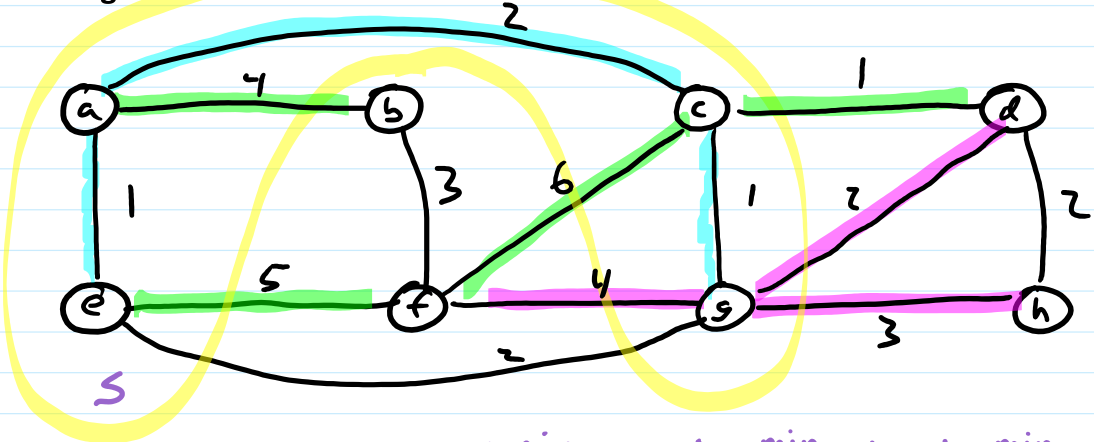
for each edge (u,v) in order of sort
    if find-set(u) ≠ find-set(v)
        add (u,v)
        union(u,v)

Prim's Algorithm



total weight = 1+4+2+3+1+1+2 = 14

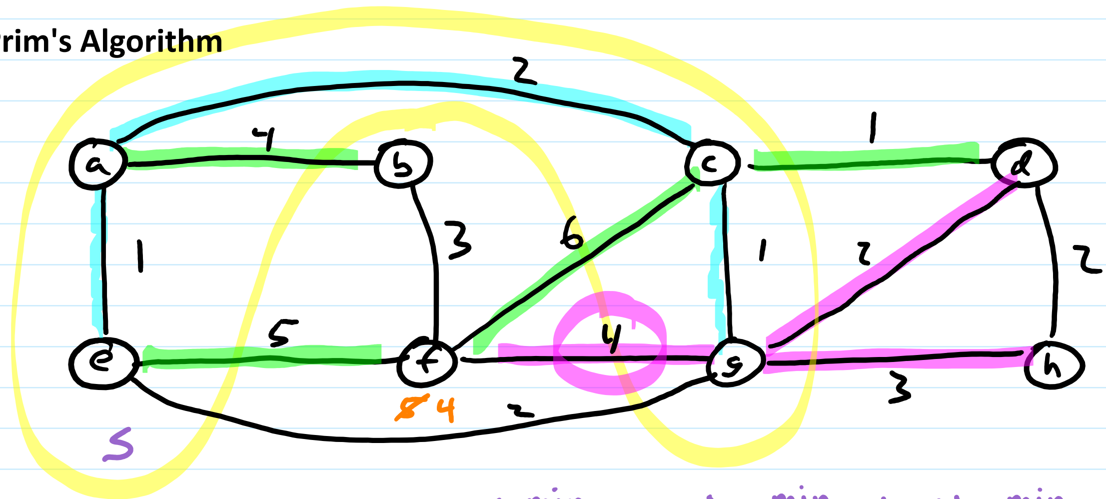
Prim's Algorithm



$$\min_{\substack{(u,v)\\u\in S\\v\notin S}} w(u,v) = \min\left(\min_{\substack{(u,v)\\u\in S}} w(u,b), \min_{\substack{(u,v)\\u\in S}} w(u,f), \min_{\substack{(u,v)\\u\in S}} w(u,d), \min_{\substack{(u,v)\\u\in S}} w(u,h)\right)$$

$$\min_{\substack{(u,f)\\u\in S}} w(u,f) = \min\left(\min_{\substack{(u,f)\\u\in\{a,e,c\}}} w(u,f) , w(g,f)\right)$$

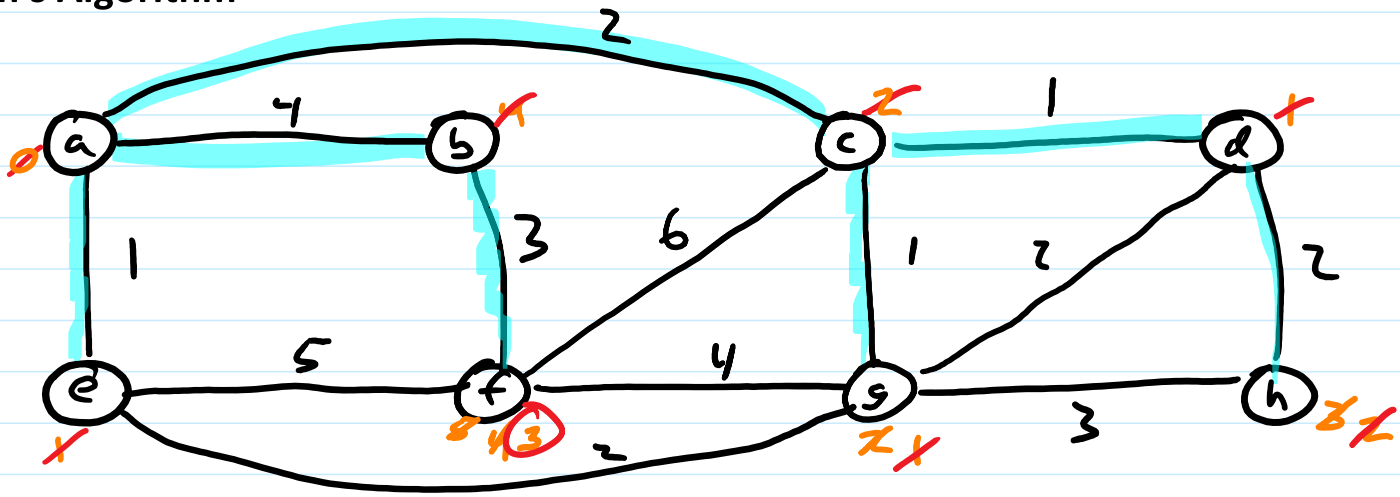
Prim's Algorithm



$$\min_{\substack{(u,v)\\ u\in S\\ v\notin S}} w(u,v) = \min\left(\min_{\substack{(u,v)\\ u\in S}} w(u,b), \min_{\substack{(u,v)\\ u\in S}} w(u,f), \min_{\substack{(u,v)\\ u\in S}} w(u,d), \min_{\substack{(u,v)\\ u\in S}} w(u,h)\right)$$

$$\min_{\substack{(u,f)\\ u\in S}} w(u,f) = \min\left(\min_{\substack{(u,f)\\ u\in\{u,e,c\}}} w(u,f), w(g,f)\right)$$

Prim's Algorithm



~~Dijkstra~~'s Algorithm

Prim

PRE: no negative weight edges

POST: T is a minimum spanning tree

```
for each v
  color[v], pred[v], d[v] ← IN_QUEUE, NIL, ∞
d[s] ← 0
S ← Ø

Q ← new PriorityQueue(d)

while Q not empty
  u ← dequeue(Q)
  S ← S U {u}
  for each outneighbor v of u
    if color[v] = IN_QUEUE and d[v] > d[u] + w(u, v)
      change_priority(Q, v, d[u] + w(u, v))
      d[v] ← d[u] + w(u, v)
      pred[v] ← u
  color[u] ← DONE
T ← {(u, pred[u]) | pred[u] ≠ NIL}
```

using adjacency list + unsorted array worst case $\Theta(n^2)$

binary heap worst case $\Theta(m \log n)$

Fibonacci heap worst case $O(m + n \log n)$

dequeue amortized $O(\log n)$

change_priority to make priority ↓ amortized $O(1)$