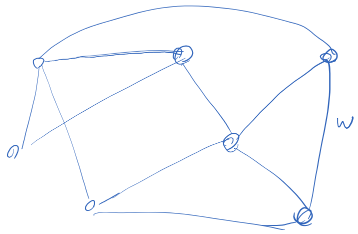
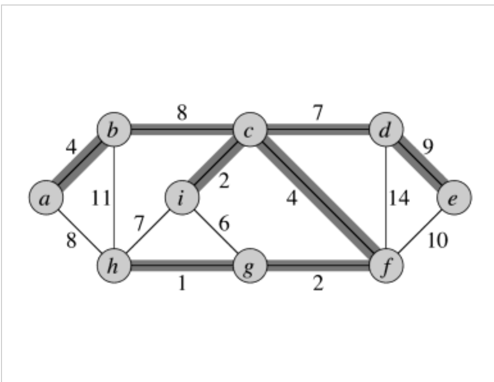
csi503-s21-lecture10

Thursday, April 1, 2021

Minimum Spanning Trees (MST)



Tree, connecting all vertices with minimum weight



✓ MSTs are not necessarily unique

✓ $|E| = |V| - 1$

✓ Tree → no cycle

Build a set of edges A

$A = \{\}$

Let's add edges to A that maintain a loop invariant:

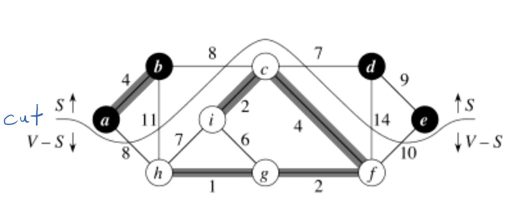
A to be subset of an MST

GENERIC-MST(G, w)

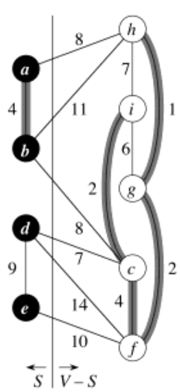
$A = \emptyset$
while A is not a spanning tree
  find an edge (u, v) that is safe for A
  $A = A \cup \{(u, v)\}$
return A

safe (u,v) iff $A \cup \{(u,v)\}$ is also a subset of some MST

How to find safe edge?



(a)



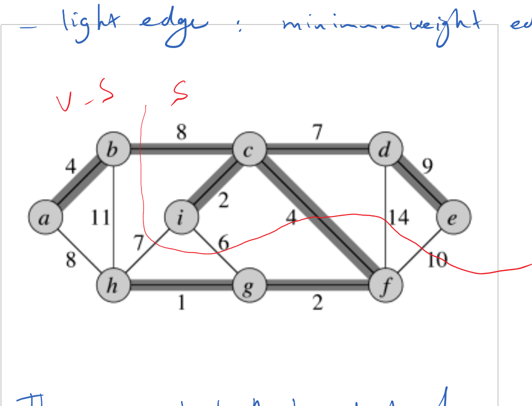
(b)

- S & V−S are disjoint

- S & V - S are disjoint

- cut (S, V-S) respects A iff no edge in A crosses the cut.

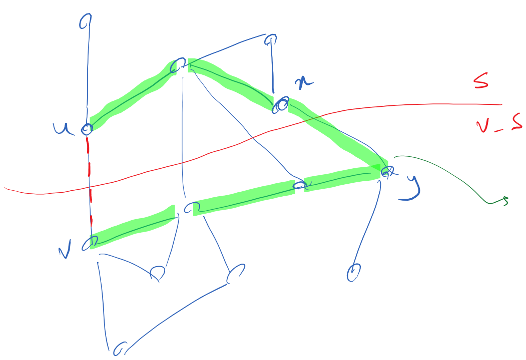
- light edge : minimum weight edge crossing the cut



Theorem : Let A be subset of some MST,
(S, V-S) be a cut that respects A,
(u,v) be light edge crossing (S, V-S)

$\Longrightarrow$ (u,v) is safe for A

proof : Let T be MST containing A . Assume (u,v) not in T .



path in MST T connecting u and v

(u,v) is light edge for cut (S, V-S) $\Longrightarrow$ $w(u,v) \leq w(x,y)$

Let's form another MST, T' :

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

$$w(T') \leq w(T)$$

$\Longrightarrow$ T' is an MST

$$A \cup \{(u,v)\} \subseteq T'$$

→ Kruskal

KRUSKAL(G, w)
  $A = \emptyset$
  for each vertex $v \in G.V$
    MAKE-SET(v)
  sort the edges of $G.E$ into nondecreasing order by weight w
  for each (u, v) taken from the sorted list
    if FIND-SET$(u) \neq$ FIND-SET(v)
      $A = A \cup \{(u, v)\}$
      UNION(u, v)
  return A

$\to |V|$ Time(Make-Set)

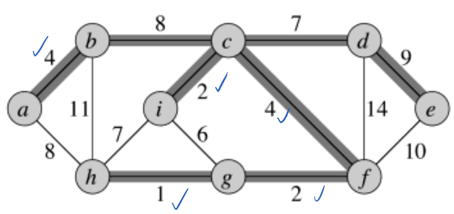
$\to O(E \lg E)$

$O(E)$ (Time (Find-Set) + Time (Union))

disjoint-sets DS : $O(m\,\alpha(n))$

number of ops. → m; number of elements → n

↳ very slow growing function



(h, g) ✓
(c, i) ✓
(f, g) ✓
(a, b) ✓
(c, f) ✓
(g, i) ✗
⋮

$O((V+E)\,\alpha(V)) + O(E \lg E)$

$\alpha(|V|) = O(\lg V) = O(\lg E)$

total : $O(E \lg E)$

$= O(E \lg V)$

$|E| \leq |V|^2$

$O(\lg E) = O(2 \lg V)$

Prim's Algorithm

- pick a node as the root, maintain tree represented by A

- at each step, choose light edge crossing $(V_A, V - V_A)$

use a priority queue to maintain candidates for adding to A

u.key : represents minimum weight for edge (v, u) where $v \in V_A$

```
PRIM(G, w, r)
  Q = ∅
  for each u ∈ G.V
      u.key = ∞
      u.π = NIL
      INSERT(Q, u)
  DECREASE-KEY(Q, r, 0)      // r.key = 0
  while Q ≠ ∅
      u = EXTRACT-MIN(Q)
      for each v ∈ G.Adj[u]
          if v ∈ Q and w(u, v) < v.key
              v.π = u
              DECREASE-KEY(Q, v, w(u, v))
```

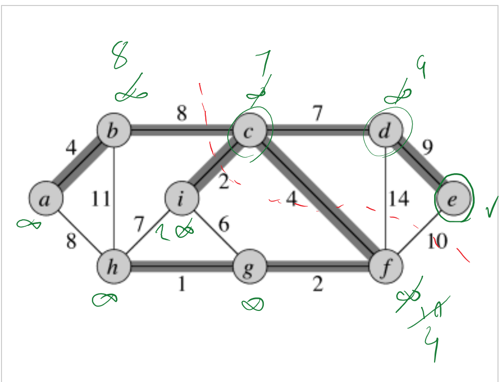
$\rightarrow O(V \lg V)$

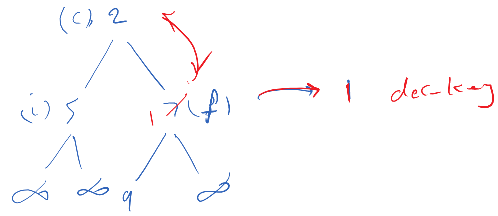
$O(\lg V)$

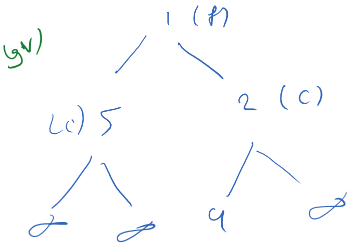
$\rightarrow O(V \lg V)$

$O(E \lg V)$

Total Cost







MST : e, d, c