muddytown: two greedy MST/F algorithms

- Prim/Dijkstra(/Boruvka)

```
tree <- any vertex
do n-1 times
  add min cost edge which grows tree
```

- Kruskal(/Boruvka)

```
forest <- all vertices, no edges
do n-1 times
  add min cost edge which grows forest
```

- correctness uses graph theory
- different complexities, depend on imp'n

Prim/Dijkstra MST

```
E:  edges of tree   ... so far
T: ‘tree’   vertices... so far
F: ‘fringe’ vertices... not in T and see some t in T
U: ‘unseen’ vertices... not in T or F

T <- any v    F <- N(v)    U<- V-N(v)-v
while F not empty do
  e <- any min weight tree-to-fringe edge (t,f)
  add e to E
  move f from F to T
  move N(f) intersect U from U to F
endwhile
```

Prim/Dijkstra MST algorithm correctness

- first, equivalent properties for a connected graph with n vertices:
  1. the graph is a tree (namely connected and acyclic)
  2. the graph is connected and has $n-1$ edges
  3. for each x, y there is a unique x, y-path

  proof sketch:
  - $1 \Longrightarrow 2$: in a tree, the ends of a maximal path have degree one; remove such a vertex, and argue by induction
  - $1 \Longrightarrow 3$: suppose not; find a cycle
  - $3 \Longrightarrow 1$: suppose not; find two paths
  - $2 \Longrightarrow 1$: a connected graph with $m = n - 1$ has no vertex with degree 0 (connected) so some vertex with degree 1 (sum the degrees); remove such a vertex, and argue by induction
- next, properties of MST's
  - spanning tree has *MST property* if adding any edge not in tree creates cycle in which added edge is heaviest
  - any two trees with MST property have same total weight
  - spanning tree: MST $\Leftrightarrow$ has MST property
  - at all times, each tree in P/D and K alg'ms has MST property

detailed proof that 1 $\Rightarrow$ 2

- proofs on next page
- how many proofs can you do?

- *maximal* path: not a proper subpath
- *leaf*: degree 1 vertex
- lemma 1: in acyclic graph, end of maximal path has degree ≤ 1
- corollary: acyclic graph has vertex of degree 0 or 1
- lemma 2: tree with degree 0 vertex has only 1 vertex
- lemma 3: removing a leaf from a tree leaves a tree
- theorem: tree with n vertices has exactly $n-1$ edges

detailed proofs of MST properties in text

- proof of lemma 1: if the end vertex of a path is adjacent to a vertex not in the path, the path is not maximal; if the end vertex p_0 of a path $(p_0, p_1, \ldots)$ is adjacent to a vertex p_j different from p_1, then $(p_0, p_1, \ldots, p_j, p_0)$ is a cycle.

- proof of lemma 2: in a connected graph there is a path between any two vertices, so if there is a vertex of degree 0 in a connected graph, it must be the only vertex in the graph.

- proof of lemma 3: Let T be a tree. Since T is acyclic, $T - v$ is acyclic.

  Let w be the neighbour of v in T. Let x, y be any two vertices of $T - v$. Since T is connected, there is a path $P = (x, \ldots, y)$ in T. Since every vertex in $P - \{x, y\}$ is adjacent to at least 2 path vertices, v is not in $P - \{x, y\}$, so v is not in P, so P is a path in $T - v$, so $T - v$ is connected.

- proof of theorem (by induction):

  base case: the graph with 1 vertex is a tree and has 0 edges

  inductive case: Fix $k \geq 1$ and assume that all trees with k vertices have $k - 1$ edges. Consider a tree T with $k + 1$ vertices. We want to show that T has k edges.

  Since $k \geq 1$, T has at least two vertices, so by lemmas 1 and 2, T has some vertex v with degree 1. By lemma 3, $T - v$ is a tree. $T - v$ has k vertices and (by inductive assumption) $k - 1$ edges. Since v is a leaf, T has exactly one more edge and one more vertex than $T - v$. Thus T has $1 + k - 1 = k$ edges.

analysis of Prim/Dijkstra MST algorithm

- "the devil is in the details"
- imp'n 1: keep list of all fringe edges
- time:
  - dominated by finding min of unsorted list with size $O(m)$ $n-1$ times so $O(mn)$
  - e.g. complete graph, during step j, tree has size j, fringe has size $n-j$, so $j(n-j)$ fringe edges, so time proportional to

$$
\begin{aligned}
& \sum_{j=1}^{n-1} j(n-j) \\
= \; & n \sum_{j=1}^{n-1} j \; - \; \sum_{j=1}^{n-1} j^2 \\
= \; & n \frac{(n-1)n}{2} \; - \; \frac{2(n-1)^3 + 3(n-1)^2 + (n-1)}{6} \\
\approx \; & n^3/2 - n^3/3 \\
\approx \; & n^3/6 \\
\in \; & \Theta(n^3)
\end{aligned}
$$

- imp'n 2: for each fringe vertex, keep best fringe edge

```
proc MST(v)      (*assume graph connected*)
  for all x do
    S[x] <- U            (*status unseen*)
    B[x] <- infinity  (*best fringe to x*)
  S[v] <- T              (*status intree*)
  t <- v
  loop n-1 times:
    for all nbrs x of t do
      if (S[x]=U) or (S[x]=F and wt(t,x)< B[x]) then
        S[x] <- F (*fringe*)
        P[x] <- t (*parent*)
        B[x] <- wt(t,x)  (*update best*)
    t <- fringe vertex x with min B[x]
    S[t] <- T         (*changed from F*)
```

- except for finding t, time is $\Theta(n+m)$
- finding t: n searches, WC size $\Theta(n)$, so time $\Theta(n^2)$
- total WC time: $\Theta(n^2)$

- imp'n 3: imp'n 2 with priority queue ... time $O(n + m \lg n)$

analysis of Kruskal MST algorithm

- assume $m \geq n$ (usually true; analysis simpler)
- imp'n 1:
  - adjacency list rep'n of graph
  - forest: for each vertex, store component number
  - sort edges: $\Theta(m \log m)$
  - check whether edge creates cycle: $\Theta(1)$, up to $\Theta(m)$ times
  - add edge to forest: (merge components: change label of every vertex in one component) $\Theta(n)$, $n-1$ times
  - total $\Theta(m \log m + n^2)$
- imp'n 2: use union-find data structure

union-find data structure

- motivation: repeatedly
  - identify name of component containing vertex
  - merge components
- maintain set partition, supporting
  - find(x): return name of set containing x
  - union(x,y): merge sets find(x) and find(y)
- each set stored as rooted tree
- find(x): travels from x to root of tree and returns root
- cFind(x): find(x) with path compression
  (make every node on path from x to root a child of root)
- union(x,y): x,y both roots; make one child of other
- wUnion(x,y): weighted union
  (make root of smaller tree child of other)

```
for v <- 1 to n do   (*initialize*)
  P[v] <- 0  (*parent of v*) (*v starts as root*)
  S[v] <- 1  (*size of subtree rooted at v*)

cFind(x)
  if P[x] = 0
    return x
  else
    r <- x
    while P[r]>0 do
      r<- P[r]
    y <- x           (*path compression*)
    while P[y]<>r do (*                *)
      newy <- P[y]   (*                *)
      P[y] <- r      (*                *)
      y    <- newy   (*                *)
    return r

wUnion(x,y) (*x,y both roots*)
  if S[x] <= S[y] then
    P[x] <- y    S[y] <- S[x]+S[y]
  else
    P[y] <- x    S[x] <- S[x]+S[y]
```

union-find analysis

- $\lg^*(n)$: smallest t so that $2^{2^{2^{\cdots}}} \geq n$, where t is number of 2's
- $\lg^*(n) \leq 5$ for $n \leq 2^{65536}$ virtually constant

```
    n     2  ...4  ...16  ...65536  ...2^{65536}
  lg*n    1    2     3       4            5
```

- m U/F's on an n-set: WC $\Theta(n + m \log n)$ time p288
- m wU/cF's on an n-set: WC $O((n+m)\lg^* n)$ time p293

WC analysis of Kruskal MST algorithm

- sort edges $\Theta(m \log m)$ time
- union-finds (assume m in $\Omega(n)$)
  - about $2m$ cFinds
  - $n-1$ wUnions
  - $\Theta(m)$ cF/wU op'ns with n-set $O(m \lg^* n)$ time
- total $\Theta(m \log m)$ time