Minimum spanning trees

CIS 315
Kruskal’s Method (p 631)

1) A = ∅
2) for each v ∈ V
3) makeSet(v)
4) sort E by weight
5) for each (u,v) ∈ E
6) if findSet(u) ≠ findSet(v)
7) then A = A ∪ {(u,v)}
8) union(u, v)
9) return A

Timing:
lines 2-3: O(V)
line 4: O(E lg E) -- faster if small edge weights (counting sort)?
lines 5-8: E calls to 3 union-find operations, each O(lg*V) amortized
lines 5-8: total O(E lg* V)
overall total: O(E lgE)
aside: disjoint sets

Figure 5.5 A directed-tree representation of two sets \{B, E\} and \{A, C, D, F, G, H\}.

from Dasgupta-Papadimitriou-Vazirani (also our chap 21)
union-find by rank with path compression

```
procedure makeset(x)
    \( \pi(x) = x \)
    rank(x) = 0

function find(x)
    while x \neq \pi(x): x = \pi(x)
    return x
```

```
procedure union(x, y)
    \( r_x = \text{find}(x) \)
    \( r_y = \text{find}(y) \)
    if \( r_x = r_y \): return
    if rank(\( r_x \)) > rank(\( r_y \)):
        \( \pi(r_y) = r_x \)
    else:
        \( \pi(r_x) = r_y \)
        if rank(\( r_x \)) = rank(\( r_y \)):
            rank(\( r_y \)) = rank(\( r_y \)) + 1
```

Any sequence of \( m \) operations, \( n \) of which are makeset, takes time \( O(m \lg^* n) \)
- \( \lg^* n \) is minimum \( k \) such that \( \lg \lg \lg \ldots n \leq 1 \) (\( k \) iterations)
- actually better -- \( O(m \alpha(n)) \) -- \( \alpha(n) \) is inverse Ackermann function
- both \( \lg^* n \) and \( \alpha(n) \) are very very slow growing, essentially constant
Prim’s method (p 634)

for each $u \in V$
  $u$.key = $\infty$
  $u$.prev = nil

$r$.key = 0 -- start point

priority queue $Q \leftarrow V$ -- insert all of $V$ into $Q$

while $Q$ not empty
  $u =$ $Q$.extractMin
  for each $v \in \text{adj}[u]$
    if $v \in Q$ and $W[u,v] < v$.key
      then
        $v$.prev = $u$
        $v$.key = $W[u,v]$ -- use heap decreaseKey operation
time for Prim

• there is one buildHeap
• V extractMin operations
• E decreaseKey operations
• time using binary heap
  \[O((V+E) \lg V)\]
• time using Fibonacci heap
  \[O(V \lg V + E)\]
generic MST proof with loop invariant!

A = ∅
while A not yet spanning tree
  choose a safe edge \((u,v)\) for A
  add \((u,v)\) to A

Definition: Suppose A is a subset of a MST of the graph G. A safe edge for A is an edge \((u,v)\) such that \(A \cup \{(u,v)\}\) is also a subset of a MST of G.

- so our algorithm is trivially correct (think about initialization, maintenance, and termination)
- still need to fill it out
safe edges and cuts

- Prim and Kruskal choose safe edges by means of cuts
- Let $G=(V,E)$ be the (weighted) graph, and let $A \subseteq E$ be a set of edges
- The idea is that $A$ is a subset of a MST
- A cut that respects $A$ is a proper subset of vertices $S \subseteq V, \ldots$, so $(S,V-S)$ partitions the vertices
- ... and no edge of $A$ is allowed to cross $(S,V-S)$
light edge

• a light edge for a cut \((S, V-S)\) is a minimum weight edge crossing the cut

• main theorem: for any cut \((S, V-S)\) respecting \(A\), a light edge for the cut is safe for \(A\)

• both Prim and Kruskal pick light edges for some cut

• therefore, they are both correct
the dual to a cut is a cycle

input: graph G=(V,E), with weights

T=E
while T has a cycle
    pick a cycle C in T
    find a max weight edge (u,v) in T
    remove edge (u,v) from T

• does this work?
• can it be proved correct loop invariantly?
• efficiency?