Minimum spanning trees

CIS 315
Kruskal’s Method

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
union-find by rank with path compression

procedure makeset($x$)
\[
\pi(x) = x \\
\text{rank}(x) = 0
\]

function find($x$)
\[
\text{while } x \neq \pi(x): x = \pi(x) \\
\text{return } x
\]

procedure union($x$, $y$)
\[
\pi(x) = \text{find}(x) \\
\pi(y) = \text{find}(y) \\
\text{if } \pi(x) = \pi(y): \text{return} \\
\text{if rank}(\pi(x)) > \text{rank}(\pi(y)):
\quad \pi(\pi(y)) = \pi(x) \\
\text{else:}
\quad \pi(\pi(x)) = \pi(y) \\
\quad \text{if rank}(\pi(x)) = \text{rank}(\pi(y)):
\quad \text{rank}(\pi(y)) = \text{rank}(\pi(y)) + 1
\]

Any sequence of $m$ operations, $n$ of which are makeset, takes time $O(m \lg^* n)$
- $\lg^* n$ is minimum $k$ such that $\lg \lg \lg n \leq 1$ ($k$ 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

for each u ∈ V
    u.key = ∞
    u.prev = nil
r.key = 0 -- start point

priority queue Q ← V -- insert all of V into Q

while Q not empty
    u = Q.extractMin
    for each v ∈ adj[u]
        if v ∈ 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!

\[ A = \emptyset \]
while A not yet spanning tree
\[ \text{choose a safe edge } (u,v) \text{ for } A \]
\[ \text{add } (u,v) \text{ to } A \]

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 AU{(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

\begin{itemize}
\item does this work?
\item can it be proved correct loop invariantly?
\item efficiency?
\end{itemize}

\textbf{input:} graph $G=(V,E)$, with weights

$T=E$

while $T$ has a cycle

\hspace{1cm} \text{pick a cycle } C \text{ in } T

\hspace{1cm} \text{find a max weight edge } (u,v) \text{ in } T

\hspace{1cm} \text{remove edge } (u,v) \text{ from } T
the greedy algorithm

• red rule
  – Let C be a cycle with no red edges.
  – Select an uncolored edge of C of max cost and color it red

• blue rule
  – Let D be a cutset with no blue edges.
  – Select an uncolored edge in D of min cost and color it blue.

• greedy algorithm
  – Apply the red and blue rules (nondeterministically!) until all edges are colored. The blue edges form a MST.
  – Note: can stop once $n - 1$ edges colored blue.