Algorithm Design Techniques

- Induction
- Divide and Conquer
- Dynamic Programming
- **Greedy**
- Reduction

Kruskal's Algorithm Running Time – O(m lg m + m(?))

Let $e_1, e_2, \ldots, e_m$ be the edges of G sorted by increasing weight.

F=ϕ

For i=1 to m

If F+{e_i} is acyclic F=F+{e_i}.

Return(F)

Disconnected(G,u,v)

- Mark each vertex unvisited
- DFS(u) /* Marks each node connected to u as visited */
- If v marked visited then return NO
- Else return YES

Kruskal's Algorithm Running Time – O(m lg m + m(n+m))

Let $e_1, e_2, \ldots, e_m$ be the edges of G sorted by increasing weight.

F=ϕ

For i=1 to m

If F+{e_i} is acyclic then F=F+{e_i}.

Return(F)

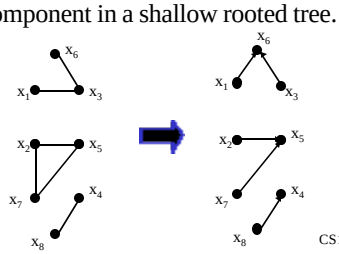
Using a naïve algorithm O(n+m)

Can we do better?

- Data structure to describe the connected components of F:
  - Operations
    - Join(i,j)
      - join the components containing vertices x_i and x_j
    - Is Comp(i)=Comp(j)?
      - Are x_i and x_j in the same connected component?
  - Each operation takes O(log n)

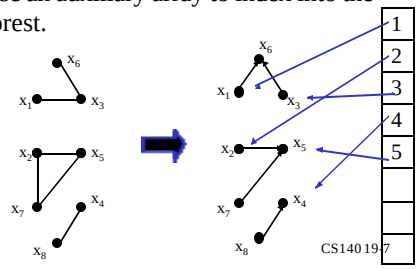
Connected Components Data Structure

- Store the vertices of each connected component in a shallow rooted tree.



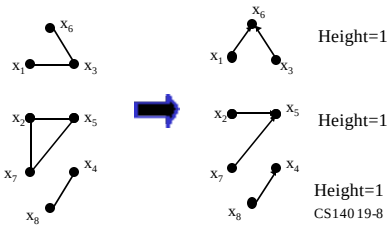
Connected Components Data Structure

- Use an auxiliary array to index into the forest.



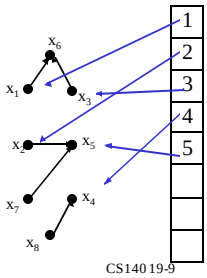
Connected Components Data Structure

- Plus remember the height



Connected Components Data Structure

- How can we answer Comp(i)=Comp(j)?
- Yes iff their roots are the same
- Time $O(h_{max})$



Connected Components Data Structure

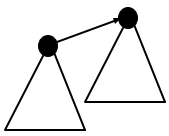
- Join(i,j)
  - Find root(i) and root(j)
  - If root(i)=root(j) we're done.
  - Otherwise add edge between root(i) and root(j) so as to minimize height.
- Time $O(h_{max})$

Join cont.

The join tree has height

1+max(h(i),h(j)) if h(i)=h(j)

max(h(i),h(j)) otherwise



Claim

- Let T be a tree in the data structure containing k nodes. Then $h(T) \leq \lg k$.

Proof of Claim

- Let T be a tree in the data structure containing k nodes. Then $h(T) \leq \lg k$.
- Initially each tree consists of a single node so the claim is true.

Proof of Claim cont.

- Let T be a tree in the data structure containing k nodes. Then $h(T) \leq \lg k$.
- Suppose T_1 and T_2 are combined in a join operation to produce T.
  - Case: $h(T_1) = h(T_2)$
  - Case: $h(T_1) \neq h(T_2)$

Proof of Claim cont. Case $h(T_1)=h(T_2)$

If $h(T_1) = h(T_2)$ then $h(T)=1+\max(h(T_1),h(T_2))$

Thus $h(T) = 1 + h(T_1) \leq 1 + \lg(k_1)$ AND
$h(T) = 1 + h(T_2) \leq 1 + \lg(k_2)$

So $h(T) \leq 1 + \lg(\min(k_1,k_2))$
$\leq \lg(2\min(k_1,k_2))$
$\leq \lg(k)$

Proof of Claim cont. Case $h(T_1)\neq h(T_2)$

If $h(T_1) \neq h(T_2)$ then $h(T)=\max(h(T_1),h(T_2))$
$\leq \max(\lg(k_1),\lg(k_2))$
$\leq \lg(k)$

Claim Proven

- Let T be a tree in the data structure containing k nodes. Then $h(T) \leq \lg k$.

Kruskal's Algorithm Running Time – O(m lg m + m(?))

Let $e_1, e_2, \ldots, e_m$ be the edges of G sorted by increasing weight.

$F=\phi$

For i=1 to m

If $F+\{e_i\}$ is acyclic then $F=F+\{e_i\}$.

Return(F)

Query and Join in time O(lg n)

Kruskal's Algorithm

Running time on an input graph with n nodes and m edges:

O(m lg m)

(Note: this is also O(m lg n))