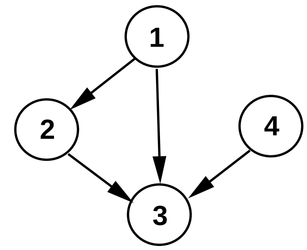
Lecture 14: April 5, 2001

Today:

- Graph Representations & Algorithms
- Greedy Algorithms
  - Minimum Spanning Tree
    - ∗ Kruskal's Algorithm
    - ∗ Prim's Algorithm

Graph Representation

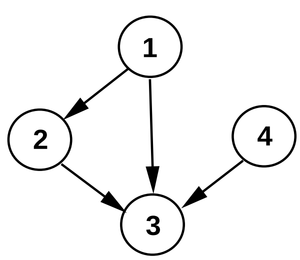
Graph $G = (V, E)$
V = set of vertices
E = set of edges = subset of $V \times V$
If G is connected, then
$|E| \geq |V| - 1 \Rightarrow \lg |E| = \Theta(\lg V)$



(Note: drop | | inside asymptotic notation)

Graph Representation

Graph can be *directed* or *undirected*



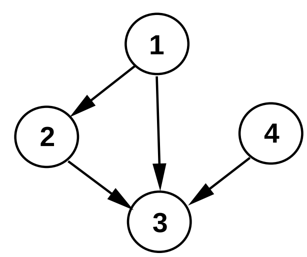
Assume vertices $V = \{1, 2, \ldots, n\}$
Define "Adjacency matrix" A[1 .. n, 1..n]

$$A[i,j] = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ if } (i,j) \notin E. \end{cases}$$

```
   | j=1 2 3 4
 --|-----------
i=1 |   0 1 1 0
  2 |   0 0 1 0
  3 |   0 0 0 0
  4 |   0 0 1 0
```

Adjacency matrix:
$\Theta(V^2)$ storage → **dense** representation

Sparse Graph Representation



Adjacency **list**: $\Theta(V + E)$ storage
→ **sparse** representation
For each vertex v, keep a list Adj[v] of vertices adjacent to v

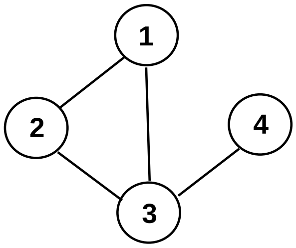
```
Adj[1] = {2,3}
Adj[2] = {3}
Adj[3] = {}
Adj[4] = {3}
```

Adj[v] = degree(v) [for undirected graphs]
Adj[v] = out-degree(v) [for digraphs]

Handshaking Lemma

For undirected graphs,

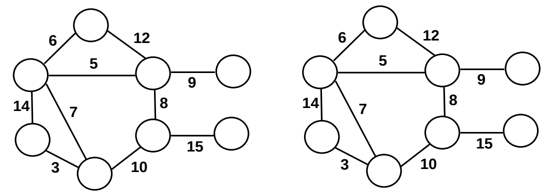
$$\Sigma \deg(v) = 2|E|$$



Thus adjacency list uses $\Theta(V + E)$ storage

Minimum Spanning Tree

Undirected, connected graph $G = (V, E)$
Weight function $W : E \to \Re$
(Here, assume edge weights distinct)



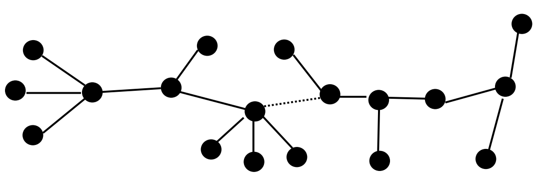
Spanning Tree: Tree that connects all vertices
What is ST of above graph?
Minimum Spanning Tree:
Tree that connects all vertices, and minimizes

$$w(T) = \sum_{(u,v)\in T} w(u, v)$$

What is MST of above graph?

Optimal Substructure

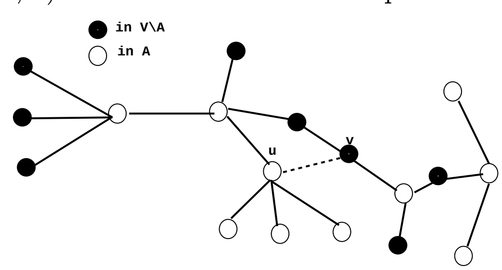
T (Note: some other edges omitted):



Removing (u, v) partitions T into T_1 and T_2.
Claim: T_1 is MST of $G_1 = (V_1, E_1)$, the subgraph of G **induced** by vertices in T_1.
V_1 = vertices in T_1
$E_1 = \{(x, y) \in E : x, y \in V_1\}$
$T2$ is MST of $G2$.
$w(T) = w(u, v) + w(T1) + w(T2)$
Can't be a tree better than T_1 or T_2, or T would be suboptimal!
(Overlapping subproblems? Dynamic prog? Yes, but...)

Greedy Choice

Greedy choice property:
Locally optimal (greedy) choice yields a globally optimal solution!
Theorem: Let T be MST of G, and let $A \subseteq V$.
Let (u, v) be min weight edge in G connecting A to $V - A$.
Then, $(u, v) \in T$. **Proof:** "cut and paste"



Suppose $(u, v) \notin T$
Look at path from u to v in T
Swap (u, v) with first edge on path from u to v in T that crosses from A to $V - A$.
This improves T!

Kruskal's algorithm for MST

Disjoint-set data structure
Sets $S = \{S_i\}, S_i$ intersects S_j = empty set
Operations:

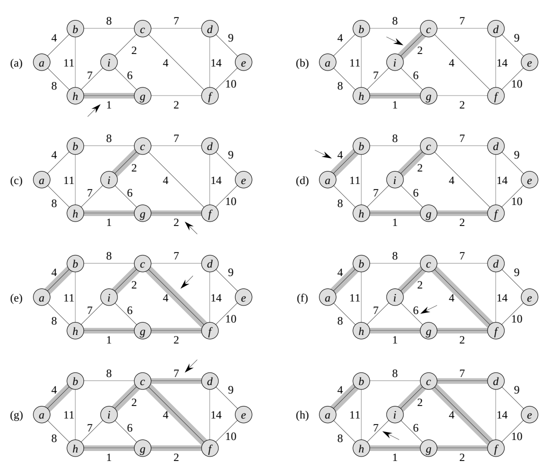
- Insert(x): $S \leftarrow \S \cup \{\{x\}\}$
- Union(S_i, S_j): $S \leftarrow S - \{S_i, S_j\} \cup \{S_i \cup S_j\}$
- FindSet(x): returns unique $S_i \in S$ where $x \in S_i$

Kruskal's algorithm for MST

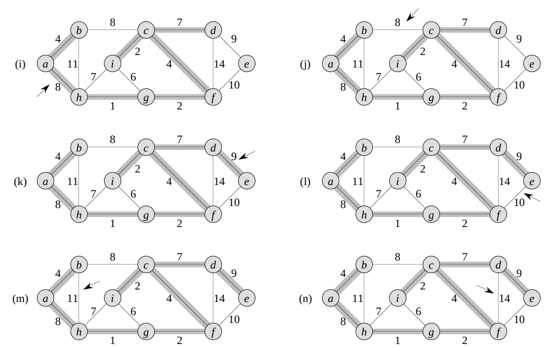
$T \leftarrow$ empty set
for each $v \in V$
  do Insert (v)
Sort E by edge weight
for each edge $(u, v) \in E$
  do if $FindSet(u) \neq FindSet(v)$
    then $T \leftarrow T \cup \{(u, v)\}$
      Union($FindSet(u), FindSet(v)$)

That is, adds cheapest edge that connects two trees of "forest".
Why is this algorithm correct?

Example



Example (cont.)



Kruskal's Running Time

Sort: $\Theta(E \times \lg E) = \Theta(E \times \lg V))$, since $|E| \leq |V^2|$
$\Theta(V)$ calls to Insert
$\Theta(E)$ calls to FindSet
$\Theta(V)$ calls to Union
Thus latter part takes $\Theta(E \times \alpha(E, V))$ time using best data structure (CLR, Ch. 22)
m operations on n sets require $\Theta(m \cdot \alpha(m, n))$ time.
$\alpha(m, n)$ is "functional inverse" of Ackermann's fn.
$\alpha(m, n) \leq 4$ even for $m, n = 10^{80}$ (atoms in the universe)
$\alpha(m, n)$ grows very slowly, but not constant!

Overall running time $\Theta(E \log E)$.
Best MST to date: 1993 Karger, Klein, Tarjan
$\Theta(E)$ time randomized (uses fast MST test)

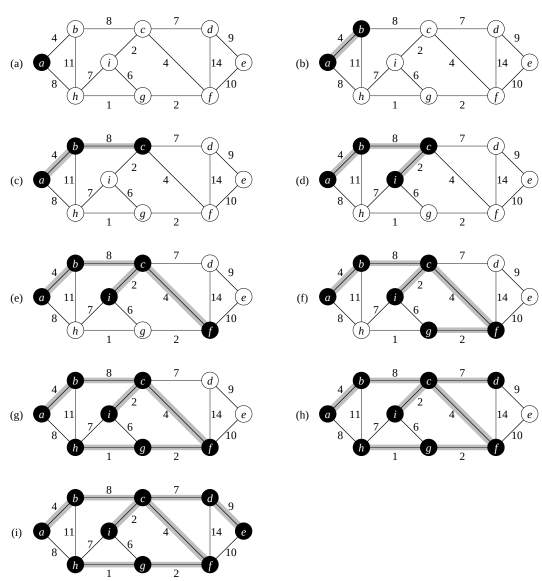
Prim's Algorithm

Keep $V - A$ in priority queue Q, sorted by weight of lightest edge connecting to A

$Q \leftarrow$ V
$key[v] \leftarrow \infty$, $\forall v \in V$
$key[s] \leftarrow 0$, for arbitrary $s \in V$
while $Q \neq \emptyset$
  do $u \leftarrow Extract - Min(Q)$
    for each $v \in Adj[u]$
      do if $v \in Q$ and $w(u, v) \leq key[v]$
        then $key[v] \leftarrow w(u, v)$
          $\pi[v] \leftarrow u$

At end, $\{(v, \pi[v]\}$ forms MST.

Example



Time $= |V| \times T(Extract - Min) +$
$\Theta(E) \times T(Decrease - Key)$

Q	T(Extract-Min)	T(Decrease-Key)	
array	$\Theta(V)$	$\Theta(1)$	1
binary heap	$\Theta(\lg V)$	$\Theta(\lg V)$	1
Fibonacci heap	$\Theta(\lg V)$	$\Theta(1)$	$\Theta(V \times \lg V + E)$