Warm up:
Show that the sum of degrees of all nodes in any undirected graph is even

Show that for any graph $G = (V, E)$, $\sum_{v \in V} \deg(v)$ is even
\[ \sum_{\nu \in V} \deg(\nu) \text{ is always even} \]

- \( \deg(\nu) \) counts the number of edges incident \( \nu \)
- Consider any edge \( e \in E \)
- This edge is incident 2 vertices (on each end)
- This means \( 2 \cdot |E| = \sum_{\nu \in V} \deg(\nu) \)
- Therefore \( \sum_{\nu \in V} \deg(\nu) \text{ is even} \)
Today’s Keywords

• Greedy Algorithms
• Choice Function
• Graphs
• Minimum Spanning Tree
• Kruskal’s Algorithm
• Prim’s Algorithm
• Cut Theorem
CLRS Readings

• Chapter 22
• Chapter 23
Homeworks

• HW7 Due **Tuesday** April 16 @11pm
  – Written (use latex)
  – Graphs
  – Released tonight
ARPANET
We need to connect together all these places into a network.

We have feasible wires to run, plus the cost of each wire.

Find the cheapest set of wires to run to connect all places.

Find a Minimum Spanning Tree.
Graphs

Definition: $G = (V, E)$

$w(e) = \text{weight of edge } e$

$V = \{A, B, C, D, E, F, G, H, I\}$

$E = \{(A, B), (A, C), (B, C), \ldots\}$
Tradeoffs
Space: $V + E$
Time to list neighbors: $\text{Degree}(A)$
Time to check edge $(A, B) : \text{Degree}(A)$
Tradeoffs
Space: $V^2$
Time to list neighbors: $V$
Time to check edge $(A, B): O(1)$
Definition: Path

A sequence of nodes \((v_1, v_2, ..., v_k)\)

s.t. \(\forall 1 \leq i \leq k - 1, (v_i, v_{i+1}) \in E\)

Simple Path:
A path in which each node appears at most once

Cycle:
A path of > 2 nodes in which \(v_1 = v_k\)
Definition: Connected Graph

A Graph $G = (V, E)$ s.t. for any pair of nodes $v_1, v_2 \in V$ there is a path from $v_1$ to $v_2$
Definition: Tree

A connected graph with no cycles
Definition: Spanning Tree

A Tree $T = (V_T, E_T)$ which connects (“spans”) all the nodes in a graph $G = (V, E)$

How many edges does $T$ have? $V - 1$
Definition: Minimum Spanning Tree

A Tree $T = (V_T, E_T)$ which connects (“spans”) all the nodes in a graph $G = (V, E)$, that has minimal cost

$\text{Cost}(T) = \sum_{e \in E_T} w(e)$

How many edges does $T$ have?

$V - 1$
Greedy Algorithms

• Require **Optimal Substructure**
  – Solution to larger problem contains the solution to a smaller one
  – Only one subproblem to consider!

• Idea:
  1. Identify a greedy choice property
     • How to make a choice guaranteed to be included in some optimal solution
  2. Repeatedly apply the choice property until no subproblems remain
Kruskal’s Algorithm

Start with an empty tree $A$
Add to $A$ the lowest-weight edge that does not create a cycle
Kruskal’s Algorithm

Start with an empty tree $A$
Add to $A$ the lowest-weight edge that does not create a cycle
Kruskal’s Algorithm

Start with an empty tree $A$
Add to $A$ the lowest-weight edge that does not create a cycle
Kruskal’s Algorithm

Start with an empty tree $A$

Add to $A$ the lowest-weight edge that does not create a cycle
Kruskal’s Algorithm

Start with an empty tree $A$
Add to $A$ the lowest-weight edge that does not create a cycle
Definition: Cut

A Cut of graph $G = (V, E)$ is a partition of the nodes into two sets, $S$ and $V - S$.

Edge $(v_1, v_2) \in E$ crosses a cut if $v_1 \in S$ and $v_2 \in V - S$ (or opposite), e.g. $(A, C)$.

A set of edges $R$ respects a cut if no edges cross the cut, e.g. $R = \{(A, B), (E, G), (F, G)\}$.
Exchange argument

• Shows correctness of a greedy algorithm

• Idea:
  – Show exchanging an item from an arbitrary optimal solution with your greedy choice makes the new solution no worse
  – How to show my sandwich is at least as good as yours:
    • Show: “I can remove any item from your sandwich, and it would be no worse by replacing it with the same item from my sandwich”
Cut Theorem

If a set of edges $A$ is a subset of a minimum spanning tree $T$, let $(S, V - S)$ be any cut which $A$ respects. Let $e$ be the least-weight edge which crosses $(S, V - S)$. $A \cup \{e\}$ is also a subset of a minimum spanning tree.
Proof of Cut Theorem

Claim: If $A$ is a subset of a MST $T$, and $e$ is the least-weight edge which crosses cut $(S, V - S)$ (which $A$ respects) then $A \cup \{e\}$ is also a subset of a MST.

Consider some MST $T$, Case 1: (the easy case)
If $e \in T$ Then claim holds
Proof of Cut Theorem

Claim: If $A$ is a subset of a MST $T$, and $e$ is the least-weight edge which crosses cut $(S, V - S)$ (which $A$ respects) then $A \cup \{e\}$ is also a subset of a MST.

Consider some MST $T$,

Case 2:

Consider if $e = (v_1, v_2) \notin T$

Since $T$ is a MST, there is some path from $v_1$ to $v_2$.

Let $e'$ be the first edge on this path which crosses the cut

Build tree $T'$ by exchanging $e'$ for $e$
Proof of Cut Theorem

Claim: If $A$ is a subset of a MST $T$, and $e$ is the least-weight edge which crosses cut $(S, V - S)$ (which $A$ respects) then $A \cup \{e\}$ is also a subset of a MST.

Consider some MST $T$,

Case 2:

if $e = (v_1, v_2) \notin T$

$T' = T$ with edge $e$ instead of $e'$

We assumed $w(e) \leq w(e')$

$w(T') = w(T) - w(e') + w(e)$

$w(T') \leq w(T)$

So $T'$ is also a MST!

Thus the claim holds
Kruskal’s Algorithm

Start with an empty tree $A$
Repeat $V - 1$ times:
Add the min-weight edge that doesn’t cause a cycle

Keep edges in a Disjoint-set data structure (very fancy)
$O(E \log V)$
General MST Algorithm

Start with an empty tree $A$
Repeat $V - 1$ times:
    Pick a cut $(S, V - S)$ which $A$ respects
    Add the min-weight edge which crosses $(S, V - S)$
Prim’s Algorithm

Start with an empty tree $A$
Repeat $V - 1$ times:
   Pick a cut $(S, V - S)$ which $A$ respects
   Add the min-weight edge which crosses $(S, V - S)$

$S$ is all endpoint of edges in $A$
$e$ is the min-weight edge that grows the tree
Prim’s Algorithm

Start with an empty tree $A$

Pick a **start node**

Repeat $|V| - 1$ times:

Add the min-weight edge which connects to node $A$ with a node not in $A$
Prim’s Algorithm

Start with an empty tree $A$
Pick a start node
Repeat $V - 1$ times:
Add the min-weight edge which connects to node in $A$ with a node not in $A$
Prim’s Algorithm

Start with an empty tree $A$
Pick a start node
Repeat $V - 1$ times:
  Add the min-weight edge which connects to node in $A$ with a node not in $A$
Prim’s Algorithm

Start with an empty tree A
Pick a start node
Repeat $V - 1$ times:
Add the min-weight edge which connects to node in $A$ with a node not in $A$
Prim’s Algorithm

Start with an empty tree $A$.

Pick a start node.

Repeat $V - 1$ times:

Add the min-weight edge which connects to node in $A$ with a node not in $A$.

Keep edges in a Heap $O(E \log V)$. 
Summary of MST results

• Fredman-Tarjan ‘84: \( \Theta(E + V \log V) \)
• Gabow et al ‘86: \( \Theta(E \log \log^* V) \)
• Chazelle ‘00: \( \Theta(E\alpha(V)) \)
• Pettie-Ramachandran ‘02: \( \Theta(?) \) (optimal)
• Karger-Klein-Tarjan ‘95: \( \Theta(E) \) (randomized)

[read and summarize any/all for EC]