Announcements

- Scholarship apps are in main office (for sophomores/juniors)
- HW due Tuesday after break (by 4pm) (but don't wait until after break!!)
- Midterm is Friday after break
Independent Sets in a graph

**Def:** Given a graph $G = (V, E)$, a set $S \subseteq V$ is an independent set if no two vertices of $S$ have an edge between them, i.e., $\forall x, y \in S, \{x, y\} \notin E$.

Goal: Compute a maximum independent set.
How can we approach this problem recursively?
(Hint: Consider any vertex \( v \). What do we know?)

\[ \text{max Ind Set}(G) \]

* \( v \) is either in the max ind set
  What can we recurse on?
  \[ 1 + \text{max Ind Set}(G - v - N(v)) \]

* \( v \) is not in ind set
  What can we recurse on?
  \[ \text{max Ind Set}(G - v) \]
Pseudo code:

```
MAXIMUMINDSETSIZE(G):
    if G = Ø
        return 0
    v ← any node in G
    withv ← 1 + MAXIMUMINDSETSIZE(G \ N(v))
    withoutv ← MAXIMUMINDSETSIZE(G \ {v})
    return max{withv, withoutv}.
```

In worst case, this looks at all possible V subsets of V, \( \geq 2^n \)

Runtime? leads to \( \Omega(2^n \text{ poly } n) \)

Can't directly use dynamic program.
Actually, this can't really be improved.

This problem is **NP-Complete** (which we'll define formally later).

Basically, this is a set of problems for which no sub-exponential algorithms are known.

Many people believe these problems are intrinsically hard and have no possible polynomial time solution.

P vs NP issue
Independent sets in trees

If we remove a vertex from a tree, what do we get? A set of trees?

What if we remove a vertex and its neighbors? Still get a set of trees?
So: give T a "root":
(make each child a root of its own subtree)

\[
\begin{align*}
\text{If I include } r: & \\
1 & + \sum \max \text{IndSet}(x) \\
\text{grandchildren } x & \\
\text{If I don't include } r: & \\
\leq & \sum \max \text{IndSet}(x) \\
\text{children } x & \\
\text{If } r \text{ is a leaf:} & \\
& \text{return 1} \\
\text{or if } G \text{ is empty, return 0}
\end{align*}
\]
Smart recursion

We need to store our answers so we don't do extra recursive calls.

Data structure?

use The tree!

with each node store value of best indep set in subtree rooted at that node.
Pseudocode (where $x.MIS$ is size of max ind. set in subtree rooted at $x$)

```
MAXIMUMINDSETSIZE(v):
    withoutv ← 0
    for each child $w$ of $v$
        withoutv ← withoutv + MAXIMUMINDSETSIZE(w)
    withv ← 1
    for each grandchild $x$ of $v$
        withv ← withv + $x.MIS$
    $v.MIS$ ← max{withv, withoutv}
    return $v.MIS$
```
Another version:

At node $x$, store 2 values. One is max. ind. set if $x$ is included, other is max. ind. set if $x$ is not included.

```
MaximumIndSetSize($v$):
    $v.MISno \leftarrow 0$
    $v.MISyes \leftarrow 1$
    for each child $w$ of $v$
        $v.MISno \leftarrow v.MISno + MaximumIndSetSize(w)$
        $v.MISyes \leftarrow v.MISyes + w.MISno$
    return max\{v.MISyes, v.MISno\}
```
Run time:

To fill in node $v$ we look at all children & grandchildren.

Think about how many times $v$ is accessed.

Each node is only accessed twice - once for parent, once for grandparent.

$\Rightarrow O(n)$  ($+$ $O(n)$ space)