an independent set of a graph \( G = (V, E) \): a subset \( I \subseteq V \) of vertices such that no pair of vertices in \( I \) share an edge.

Given a (rooted) tree \( T \) on \( n \) vertices, find a maximum cardinality independent set.
if we don't take root, take max. ind. sets from each child subtree

if we do take root, take max. ind. sets from each grandchild subtree

$\text{MIS}(v)$: size of MIS of $v$'s subtree

include $v$?
Dynamic Programming:

subproblems: vertices \( v \) of \( T \)
memoization: \( v, MIS \) for each node \( v \) of \( T \)

dependencies: children and grand children

eval order: post-order traversal

space: \( O(n) \)

time: \( O(\# \text{times the nodes are children or grandchildren}) = O(n) \)
for each node $v$ in post-order

compute $v$ in MIS

return $v$. MIS

Can MIS($G$) be is real

of $T$.
\( \text{MISyes}(v) \): size of MIS in \( v \)'s subtree that \underline{must} include \( v \)

\( \text{MISno}(v) \): same, but must \underline{not} include \( v \).

\[
\text{MISyes}(v) = 1 + \sum_{u \in v} \text{MISyes}(u) = 1 + \sum_{u \in v} \text{MISno}(u) = 1 + \sum_{u \in v} \text{MIS}(u)
\]

\[
\text{MISno}(v) = \overline{\text{MISno}(v)} \cap \overline{\text{MISyes}(v)} = \overline{\text{MISno}(v)} \cap \overline{\text{MISyes}(v)}
\]

<table>
<thead>
<tr>
<th>MIS(v):</th>
<th>MIS(v):</th>
<th>MIS(v):</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v.\text{MISno} \leftarrow 0 )</td>
<td>( v.\text{MISno} \leftarrow 0 )</td>
<td>( v.\text{MISno} \leftarrow 0 )</td>
</tr>
<tr>
<td>( v.\text{MISyes} \leftarrow 1 )</td>
<td>( v.\text{MISyes} \leftarrow 1 )</td>
<td>( v.\text{MISyes} \leftarrow 1 )</td>
</tr>
<tr>
<td>for each child ( v ) of ( \text{MISno} + \text{MISyes} )</td>
<td>for each child ( v ) of ( \text{MISno} + \text{MISyes} )</td>
<td>for each child ( v ) of ( \text{MISno} + \text{MISyes} )</td>
</tr>
</tbody>
</table>
Class Scheduling

Given \( S[1..n] \) of start times \( F[1..n] \) of finish times.

\[ 0 \leq S[i] < F[i] \quad \forall i \]

Want a maximal conflict-free schedule: max size subset \( X \subseteq \{1, \ldots, n \} \)

s.t., for each \( i, j \in X \), \( i \neq j \)

either \( S[i] > F[j] \)

or \( S[j] > F[i] \)
(max of non-overlapping intervals)

Lemma: Optimal schedule includes the class that finishes first.

Proof: Let \( f \in E \) be a class that finishes earliest. Let \( X \) be an optimal schedule. If \( f \in X \), we are done.
Let $g$ be the first class of $X$ to finish. If $f$ finishes before $g$, then $f$ does not conflict with $X \setminus \{g, \emptyset\} \cup \{f\}$. So let $X' = (X \setminus \{g\}) \cup \{f\}$. $X'$ is conflict free, and $|X'| \geq |X|$.
Lemma: Correct to take class finishing first & recurse.

Proof: We proved we can take that class $f$.
We can take any conflict free subset that does not conflict with

```
fold(S[1..n]) := fold(S[1..n])
if F and permute S to match F and permute S to match F
   count := 1
   X[count] := 1
   for i := 2 to n
      if S[i] > F[X[count]]
         count := count + 1
   if count < n
      return S[1..n]
   return S[1..n]
```

So we find the biggest one by induction.
Greedy Algorithms:

backtracking without backtracking

Can commit to best choice before recursing

First choice uses an exchange argument:

1) Start with a hypothetical optimal solution.
2) Do an exchange so new solution agrees with greedy choice.
3) Argue new solution is still optimal.

Usually want dynamic programming instead.