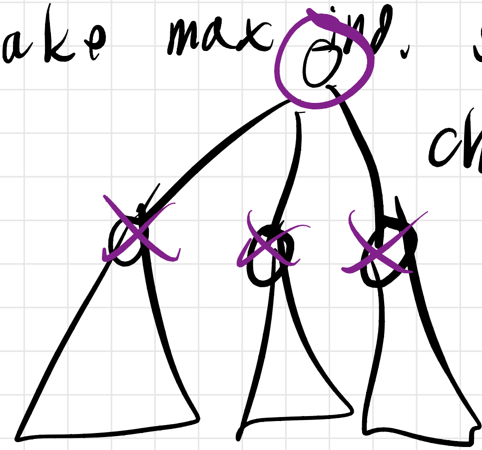
an <u>independent set</u>, of a graph $G=(V,E)$: a subset $I \subseteq V$ of vertices s.t. no pair of vertices in I share an edge.

Given a (rooted) tree T on n vertices, find a max 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

MIS(v): size of MIS of v's subtree

include v?

$$MIS(v) = \max \left\{ \sum_{\underset{\text{children } w}{w \downarrow v}} MIS(\overset{w}{\cancel{x}}),\ 1 + \sum_{w \downarrow v} \sum_{\underset{\text{grandchildren } x}{x \downarrow w}} MIS(x) \right\}$$

Dynamic Programming:

subproblems: vertices v of T

memoization: v.MIS for each node v of T

dependencies: children & grandchildren

eval order: post-order traversal

space: $O(n)$

time: $O($# times the nodes are children / grandchildren$)$

$= O(n)$

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

returns MIS(v) + sets v.MIS

Call MIS(r) if r is root of T.

-or-

for each node v in post-order
  «compute v.MIS»
return r.MIS

$MIS_{yes}(v)$: size of MIS in v's subtree that <u>must</u> include v

$MISno(v)$: same, but must <u>not</u> include v.

$$MISyes(v) = 1 + \sum_{w \downarrow v} MISno(w)$$

$$MISno(v) = \sum_{w \downarrow v} \max\{MISyes(w), MISno(w)\}$$

```
MIS(v):
  v.MISno ← 0
  v.MISyes ← 1
  for each child w of v
      v.MISno ← v.MISno + MIS(w)
      v.MISyes ← v.MISyes + w.MISno
  return max{v.MISyes, v.MISno}
```

Class Scheduling

Given $S[1..n]$ of start times

$F[1..n]$ of finish times.

$0 \le S[i] < F[i] \ \forall i$

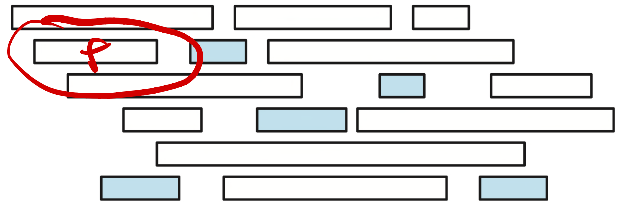
Want a $\underbrace{\text{maximal conflict-free schedule}}$: max size

subset $X \subseteq \{1, \dots, n\}$

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

either $S[i] > F[j]$

or $S[j] > F[i]$



(max # non-overlapping intervals)

Lemma: ~~⊗~~ an optimal schedule includes ~~the~~ a class that finishes first

Proof: Let $f \in \{1, \dots, n\}$ be a class that finishes earliest.

Let X be an optimal schedule.

If $f \in X$, we are done.

o.w...

Let g be the first class of X to finish.

f finishes before g.

$\Rightarrow$ f does not conflict with $X \setminus \{g\}$ $\leftarrow (X - g)$

So let $X' := (X \setminus \{g\}) \cup \{f\}$

X' is conflict free.

$|X'| = |X|$

Lemma: Correct to take class finishing first & recurse.

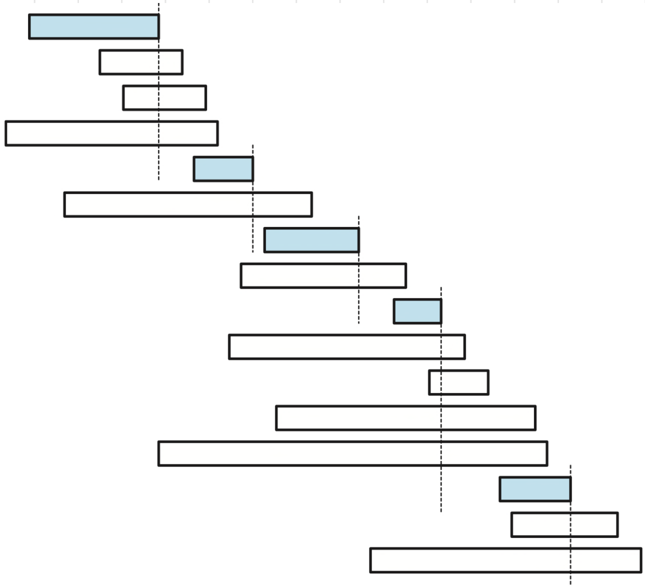
Proof: We proved we can take that class f.

We can take any conflict free subset that does not conflict with

```
GREEDYSCHEDULE(S[1..n], F[1..n]):
    sort F and permute S to match
    count ← 1
    X[count] ← 1
    for i ← 2 to n
        if S[i] > F[X[count]]
            count ← count + 1
            X[count] ← i
    return X[1..count]
```

f.

& we find the biggest one by induction.



Greedy Algorithms:

backtracking without <u>back</u>tracking

can commit to best choice before recursing

First choice uses an <u>exchange argument</u>:

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.