CS 314 - Dynamic Programming on Trees

Note Title 3/3/2010

Announcements

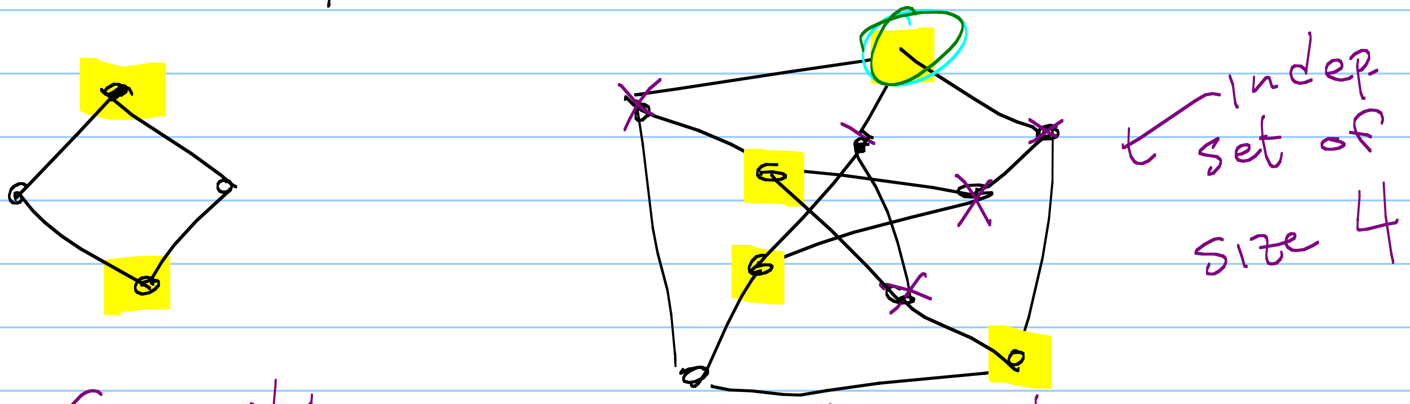
★ - Scholarship apps are in main office (for sophomores/juniors)

- HW due Tuesday after break (by 4pm) (but don't wait until after break!!)
- Midterm is Friday after break

Independent Sets in a graph

Dfn: 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,

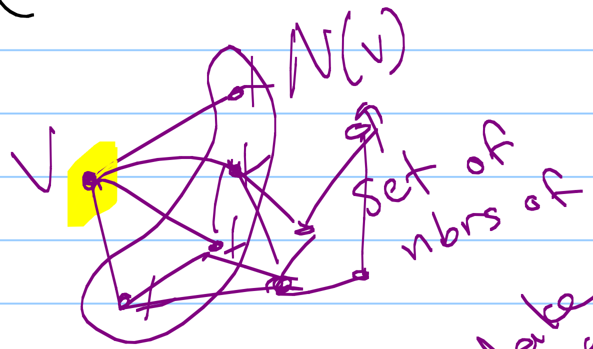
→ ie $\forall x,y \in S$, $\{x,y\} \notin E$.

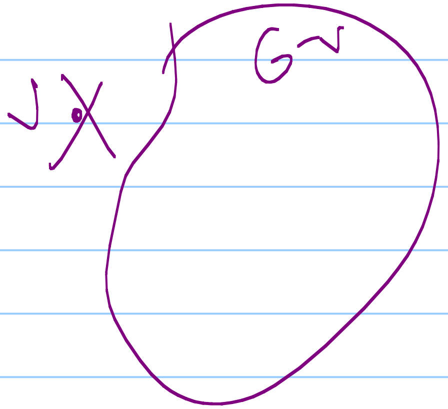


★ Goal: Compute a maximum indep set.

How can we approach this problem recursively?

(Hint: Consider any vertex v. What do we know?)





take max

- v is either in the max ind. set

  What can we recurse on?

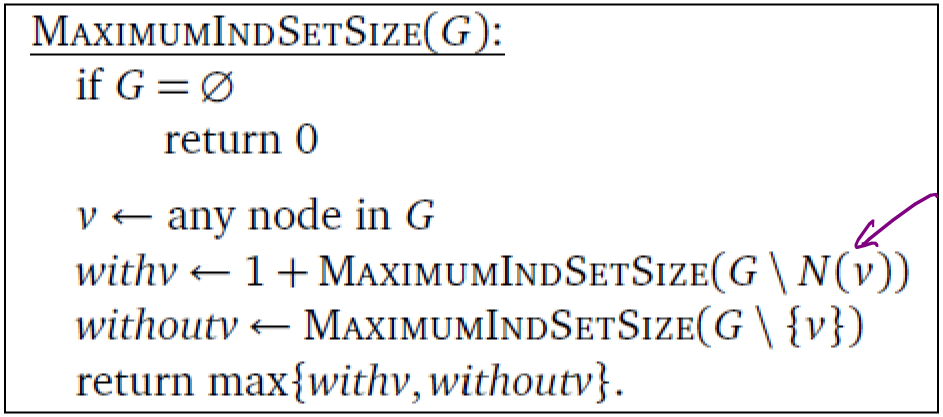
  $1 + \text{maxIndSet}(G - v - N(v))$

- v is not in ind. set

  What can we recurse on?

  $\text{maxIndSet}(G-v)$

Pseudocode:



```
MaximumIndSetSize(G):
    if $G = \emptyset$
        return 0

    $v \leftarrow$ any node in $G$
    $withv \leftarrow 1 + \text{MaximumIndSetSize}(G \setminus N(v))$
    $withoutv \leftarrow \text{MaximumIndSetSize}(G \setminus \{v\})$
    return $\max\{withv, withoutv\}$.
```

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

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

Can't directly use dynamic program.

Actually, this can't really be improved.

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

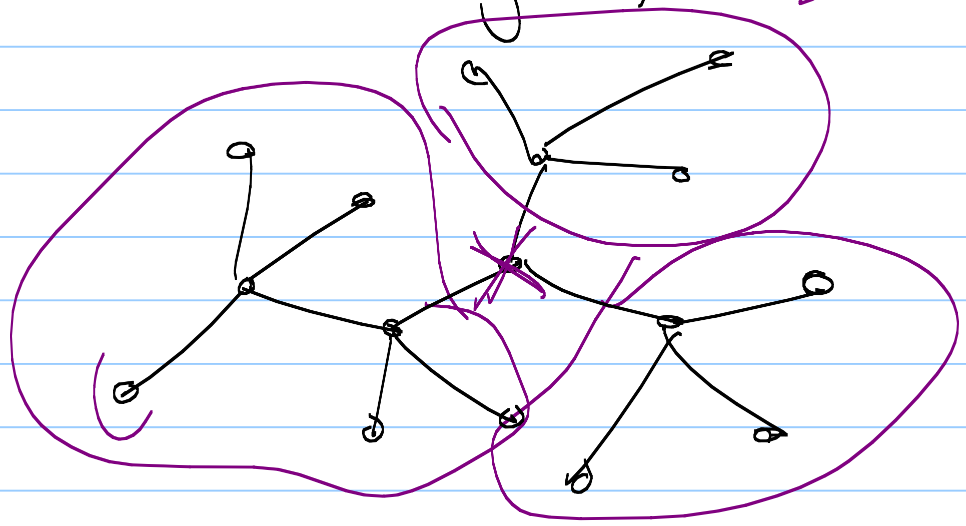
Basically, this is a set of problems for which no sub-exponential algorithms are known.

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

P vs NP issue

Indepenent sets in trees

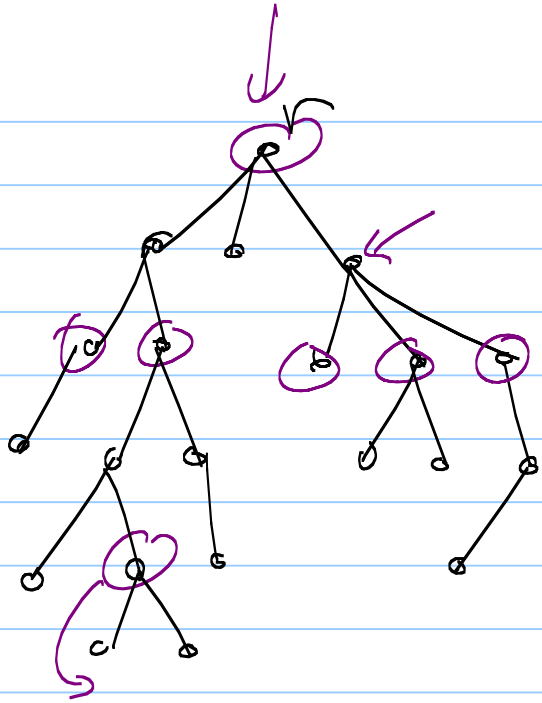
If we remove a vertex from a tree, what do we get? set of trees



What if we revove a vertex + its neighbors?

still get a set of tree

So: give T a "root" : (& make each child a root of its own subtree)



max

If I include r:

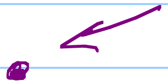
$$1 + \sum_{\substack{\text{grandchildren } x \\ \text{of } r}} \text{maxIndSet}(x)$$

If I don't include r:

$$\sum_{\text{children } x} \text{maxIndSet}(x)$$

If r is a leaf:

return 1



or if G is empty, return 0

Smart recursion

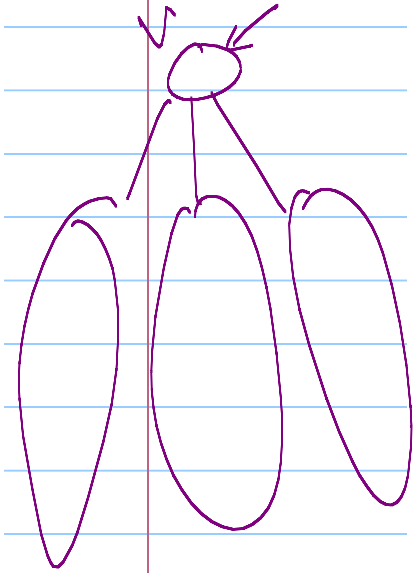
We need to store our answers so we
don't do extra recursive calls.

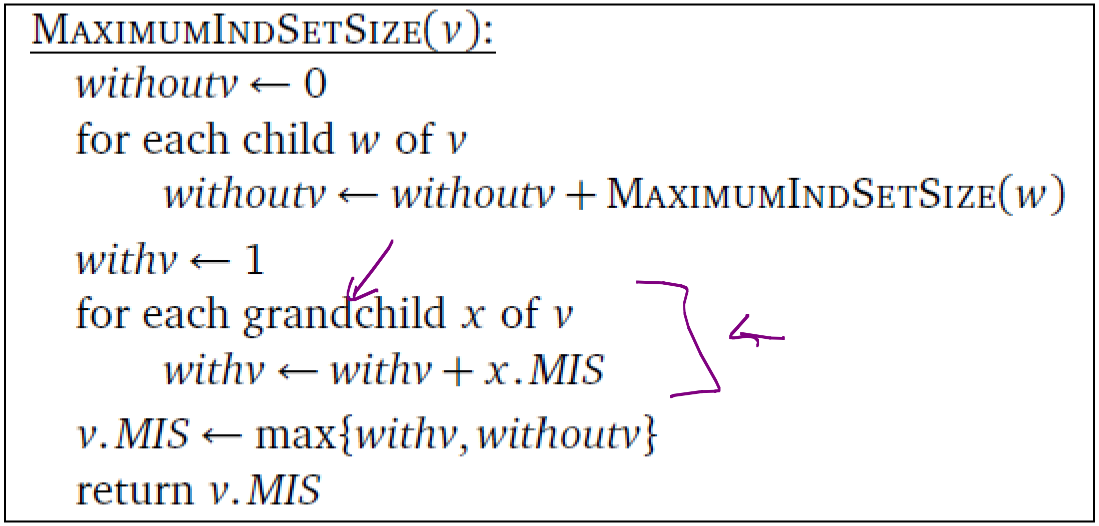
Data structure?

use the tree!

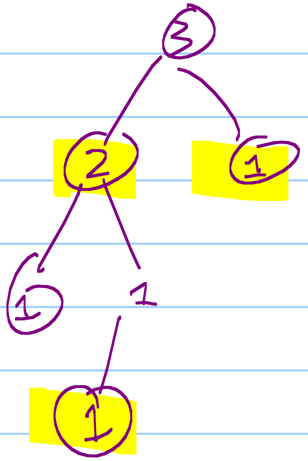
with each node, store value of
best indep set in subtree rooted
at that node.

Pseudocode (where x.MIS is size of max ind. set in subtree rooted at x)



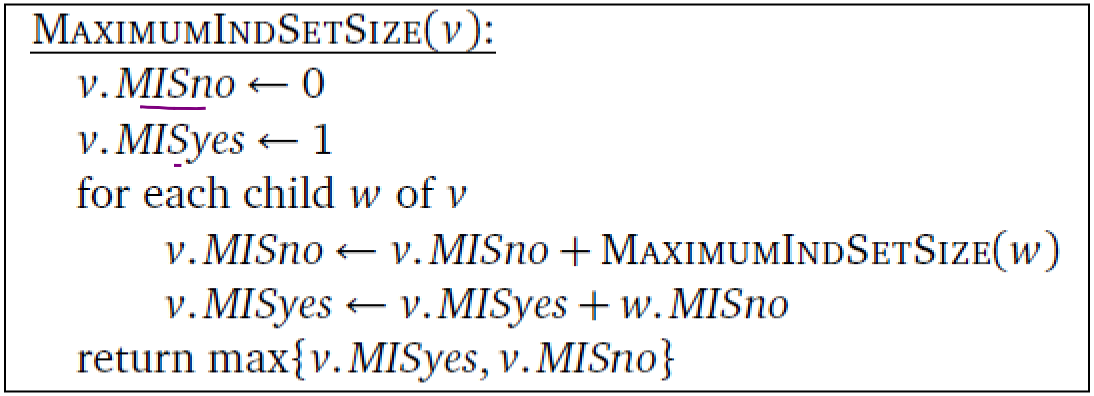


```
MaximumIndSetSize($v$):
    $withoutv \leftarrow 0$
    for each child $w$ of $v$
        $withoutv \leftarrow withoutv + \text{MaximumIndSetSize}(w)$
    $withv \leftarrow 1$
    for each grandchild $x$ of $v$
        $withv \leftarrow withv + x.MIS$
    $v.MIS \leftarrow \max\{withv, withoutv\}$
    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 $v$
        $v.MISno \leftarrow v.MISno + \text{MaximumIndSetSize}(w)$
        $v.MISyes \leftarrow v.MISyes + w.MISno$
    return $\max\{v.MISyes, v.MISno\}$
```

(slightly worse space)
2n versus n

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.

⇒ $O(n)$ (+ $O(n)$ space)