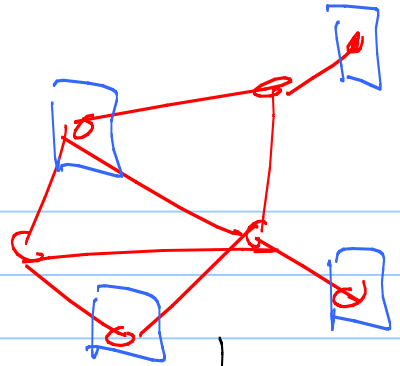
CS314 – More dynamic programming

Note Title 9/16/2013

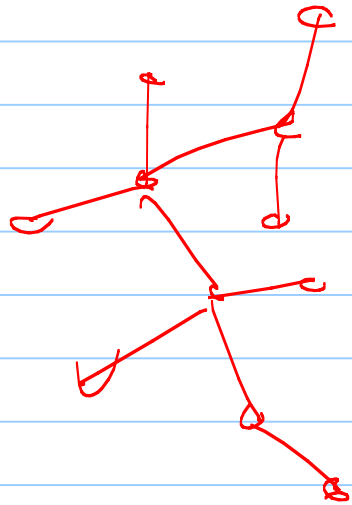
Today

- Sign-ups for oral grading (everyone!)

Dynamic Programming on Trees



Dfn: An independent set in a graph is a subset that have no edges between them.



First - why?

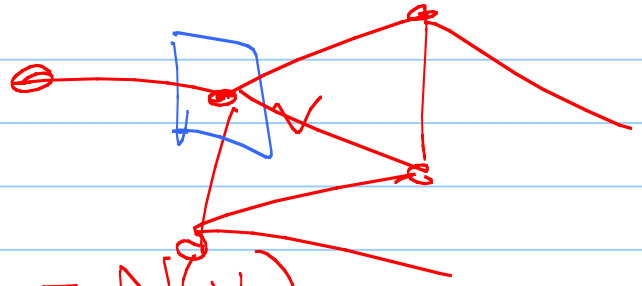
Graphs model everything.

Recursion

Goal: Compute largest indep. set.

Consider a node v.

The options?



- include v

  recurse on $G - N(v)$

- not include v

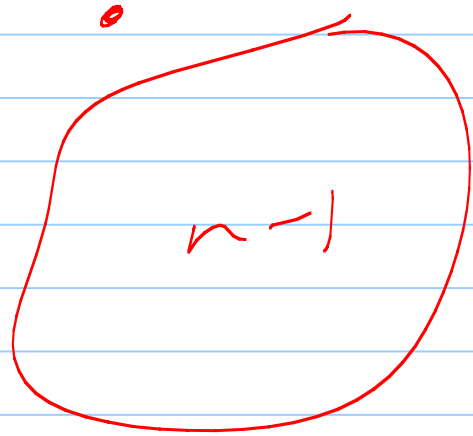
  recurse on $G - v$

v: list of nbrs

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

O(1)

O(1)



Runtime: $T(n) = 2T(n-1) + \text{poly}(n)$

$= O(\text{poly}(n)\, 2^n)$

poly(n): depends on d.s.

Aside: Can actually do a bit better.

How big will those recursive calls be?

$T(n) \leq T(n-1) + T(n-2) + \text{poly}(n)$

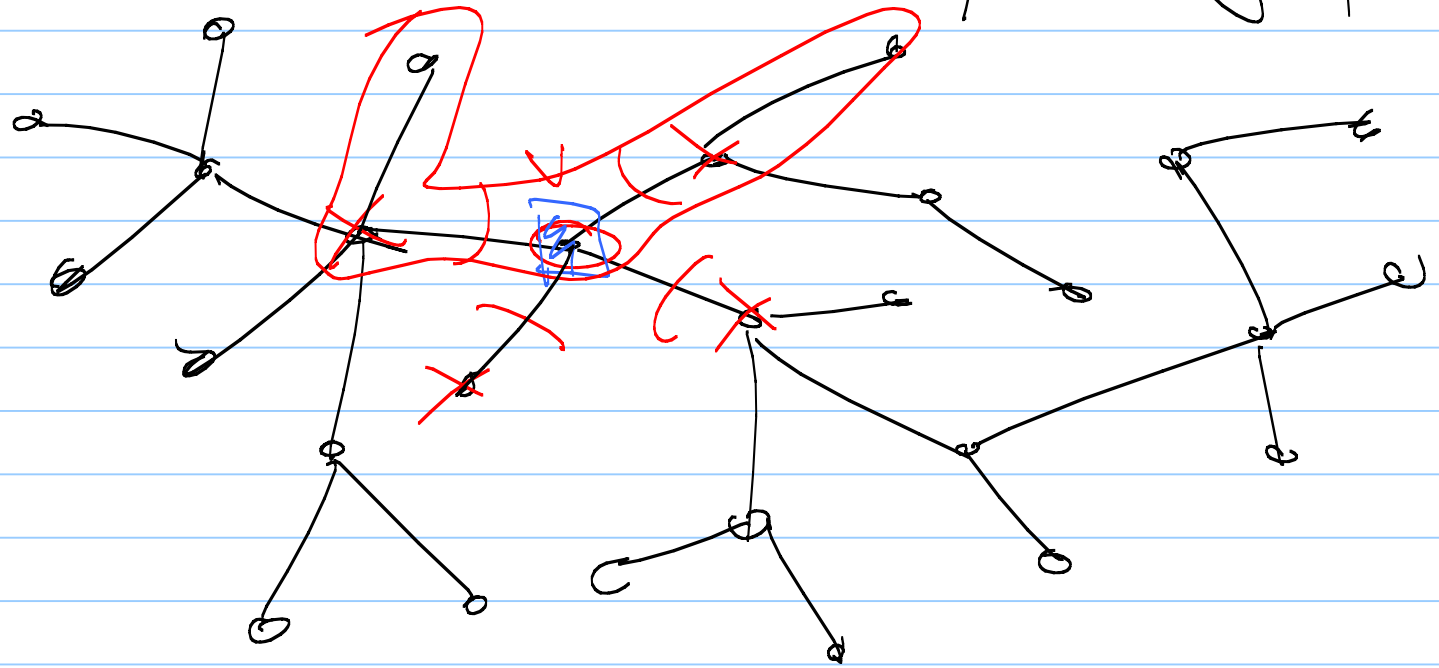
Fibonacci #'s

$\hookrightarrow \leq (\text{poly}(n)\,\phi^n)$

$\phi = \frac{1+\sqrt{5}}{2} < 2$

Ok - back to dynamic programming.

Consider a tree: an acyclic graph.



This limits the recursion!

Each piece of $T-v$ or $T-N(v)$ is another tree itself.

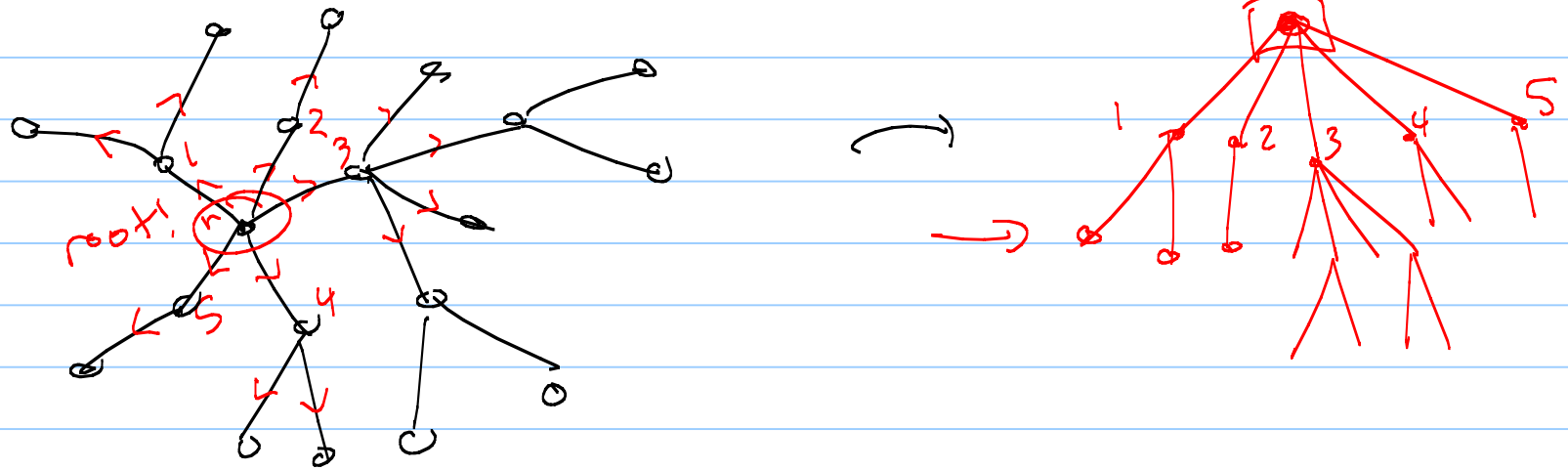
Let's memoize:

- each call considers a subtree of T.
- unfortunately, still an exponential number of those...

```
MaximumIndSetSize(T):
  if T = ∅
      return 0
  v ← any node in T
  withv ← 1
  for each tree T' in T \ N(v)
      withv ← withv + MaximumIndSetSize(T')
  withoutv ← 0
  for each tree T' in T \ {v}
      withoutv ← withoutv + MaximumIndSetSize(T')
  return max{withv, withoutv}.
```

The key: we pick v.

Call our first v the root, & impose an ordering:

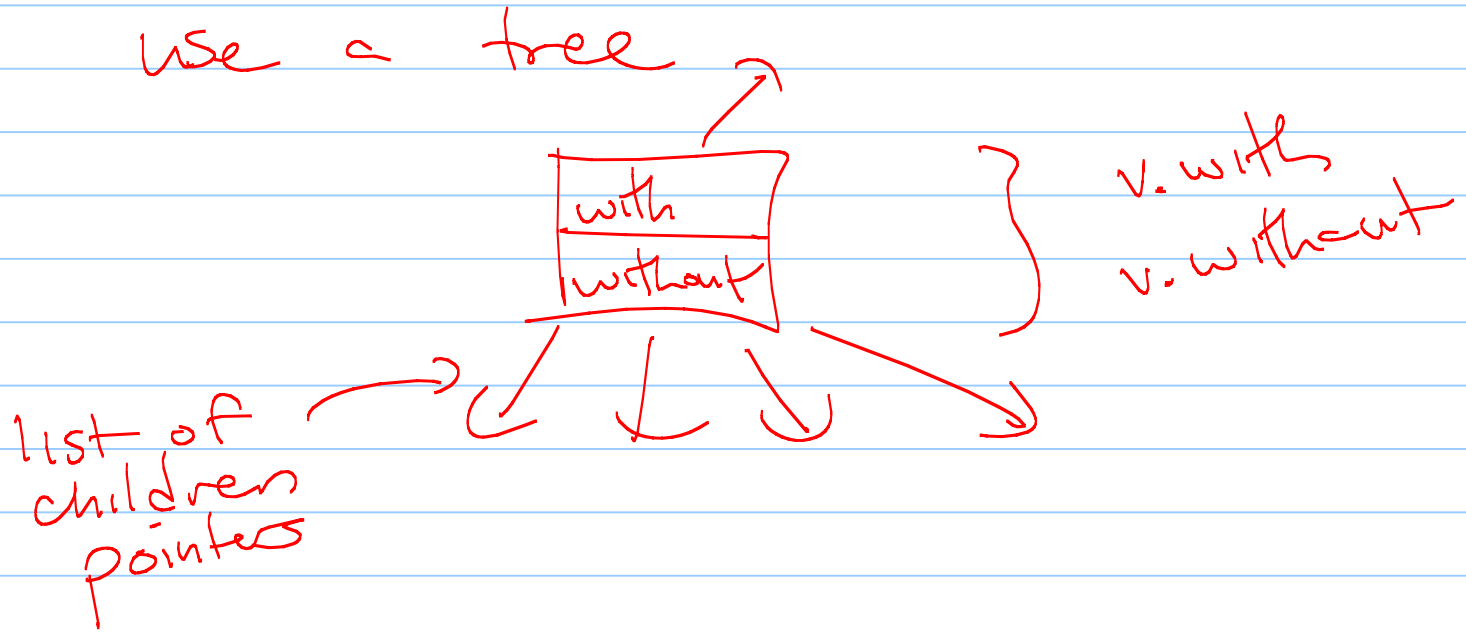


Now, for a rooted tree:

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

Note: base case?

Memoize: How to store + fill in?

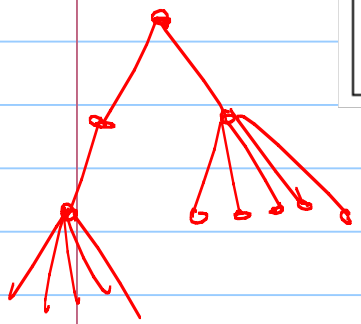


Pseudocode:

```
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
```

recording rooted

Max Ind set at x



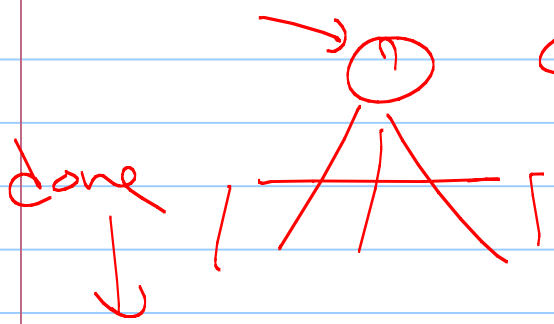
Alternative

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

Runtimes?

Each node is accessed twice
+ it accesses its children
+ grandchildren once.

$$d(v) + \sum_{w \text{ child of } v} d(w)$$



Any tree has $n-1$ edges.

$\Rightarrow O(n)$

Correctness:

We try all possibilities