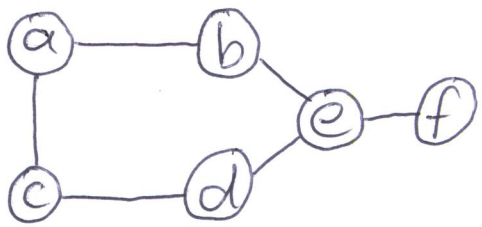
Maximum Independent Set

Given a graph $G=(V,E)$, a set of vertices $S \subseteq V$ is an *independent set* if there are no edges between two vertices in S.

Example:



$\{a, e\}$ is an independent set

$\{a, d, e\}$ is not: $(d, e) \in E$.

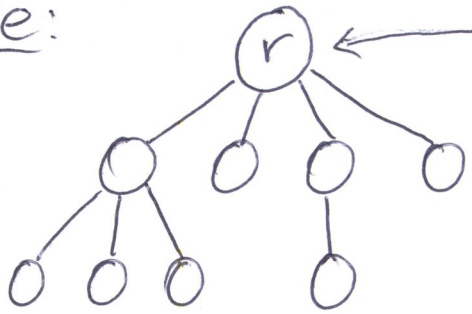
$\{a, d, f\}$ is a *maximum* independent set

Finding the maximum independent set is really hard. The current best algorithm takes $\Omega(1.2^n)$ time.
In fact, it is NP-hard. Informally, this means that we think no polynomial algorithm exists. More about this later.

When a problem is really hard, we often simplify it by placing restrictions on the input.

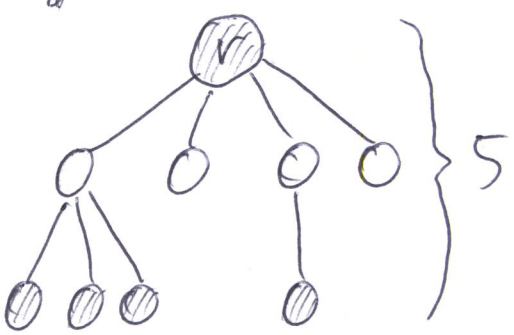
What if G is a tree? (Undirected acyclic connected graph)

Example:

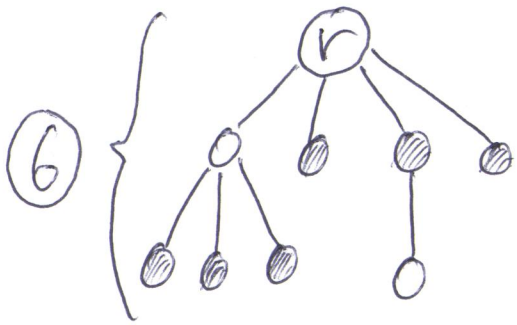
$G =$ 

pick some vertex to be the root

What is the maximum independent set of G? ~~If we include r then:~~



Include r



Don't include r

① Structure of the optimal Solution

Consider the maximum independent set I. ~~There are~~ Either $r \in I$ or $r \notin I$.

Case $r \notin I$: Then we're free to include all children of r. Each subtree rooted at a child of r is independent, so

$$MIS = \sum_{u \in \text{children}(r)} MIS \text{ of the subtree rooted at } u.$$

Case $r \in I$: Then we cannot take any children of r. But we can take all of r's grandchildren.

$$MIS = \underset{\vee}{1}_{r} + \sum_{u \in \text{grandchildren}(r)} MIS \text{ of the subtree rooted at } u$$

② Recurrence

Let $I(v)$ be the size of the maximum independent set of the subtree rooted at v.

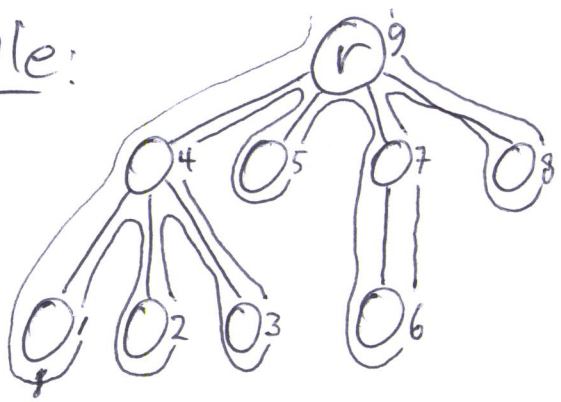
$$I(v) = \begin{cases} 1 & \text{if } v \text{ is a leaf} \\ \max\left\{ \sum_{u \in \text{children}(v)} I(u),\ 1 + \sum_{u \in \text{grandchildren}(v)} I(u) \right\} & \text{otherwise} \end{cases}$$

③ Solve bottom-up

We need to make sure that all of v's children and grandchildren (if any) have been processed before we can compute $I(v)$.

This is a post-order traversal of G.

Example:



Algorithm ComputeMIS(v, I):

```
1. if v is a leaf then
2:   I[v] ← 1
3. else
4.   for each child u of v do
5.     ComputeMIS(u, I)
6:   I⁺ ← 1; I⁻ ← 0
7.   for each child u of v do
8:     I⁻ ← I⁻ + I[u]
9.     for each child w of u do
10:      I⁺ ← I⁺ + I[w]
11:  I[v] ← max(I⁺, I⁻)
```

- Correct? Yes, by ① and ②.
- Terminates? Yes, subtrees get strictly smaller.
- Efficient?

  ComputeMIS is called once for each vertex. Thus, lines 1-3, 6, and 11 are executed $\leq n$ times. Lines 4-5 and 7-8 are executed once for every parent-child pair. Since every vertex has at most one parent, these execute $\leq n$ times. Lines 9-10 execute once for every grandparent-grandchild pair. Every vertex has at most one grandparent, so these also execute $\leq n$ times. Since every line takes $O(1)$ work and executes $\leq n$ times, the running time is $O(n)$.