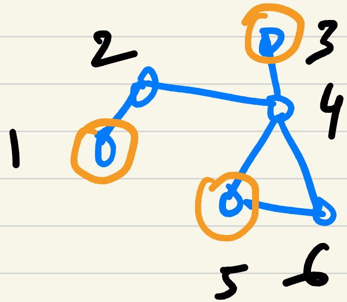
# Lecture 06

Date: 18 Jan 2024

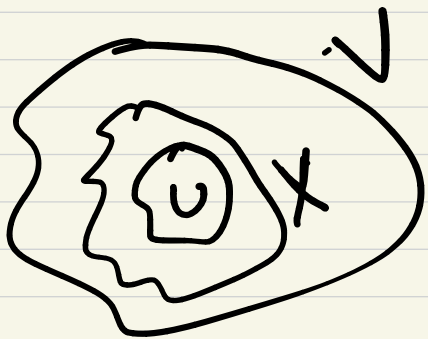
**Defn.:-** Let $G = (V, E)$ be a graph. We say that $U \subseteq V$ is an independent set (IS) if $\forall\, x, y \in U.\ \{x, y\} \notin E$.

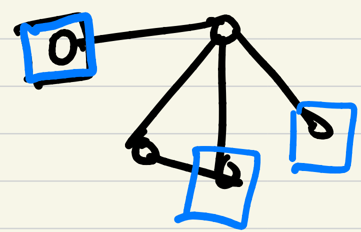


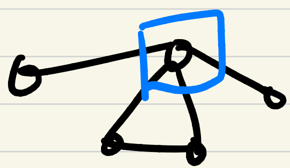

$U = \{1, 3, 5\}$

$G=(V,E)$

<u>Def$^n$</u>:- $U$ is called a maximal IS if for all superset $X$ of $U$, $X$ is not an IS.





<u>Input</u>:- $G=(V,E)$

<u>Compute</u>:- Find an IS with maximum size. MaxIS

* Algorithm to find a <u>maximal</u> IS

$I \leftarrow \{v_1\}$

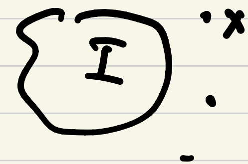
Pick $v_2$ s.t $\{v_2, v_1\} \in E$

$I \leftarrow I \cup \{v_2\}$

$\vdots$

Pick $x \leftarrow V \setminus I$

s.t $\{x, u\} \notin E \ \forall u \in I$.




**HW**:- Find the runtime.

$V = \{1, \ldots, n\}$

$I \subseteq V$

Take an array $X$

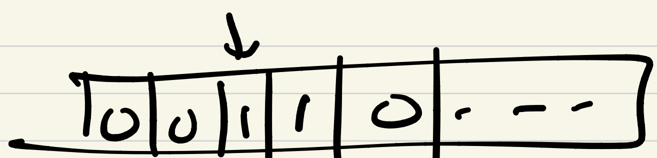
$$X[i] = \begin{cases} 1 & \text{if } i \in I \\ 0 & \text{o/w.} \end{cases}$$

We can generate all possible such $X$'s.

0 0 0 0

1 0 0 0

0 1 0 0



Max IS can be solved in time

$$O(2^n \cdot n^2).$$

$$2^{O(n)} \qquad 2^{\log n^2}$$

$$O(2^{n+\log n^2}) = 2^{O(n)}.$$

$$O((1.4..)^n)$$

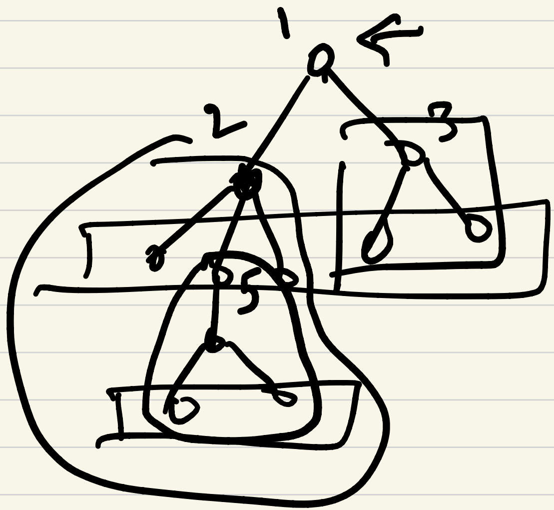
$$2^{O(n)}$$

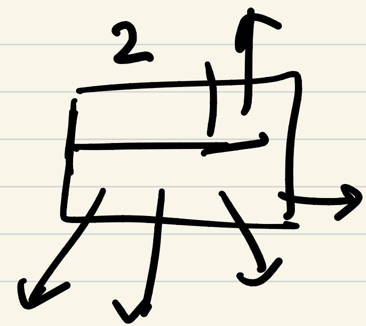
NP problems.

Given an $I$ it is "easy" to check if $I$ is an IS.

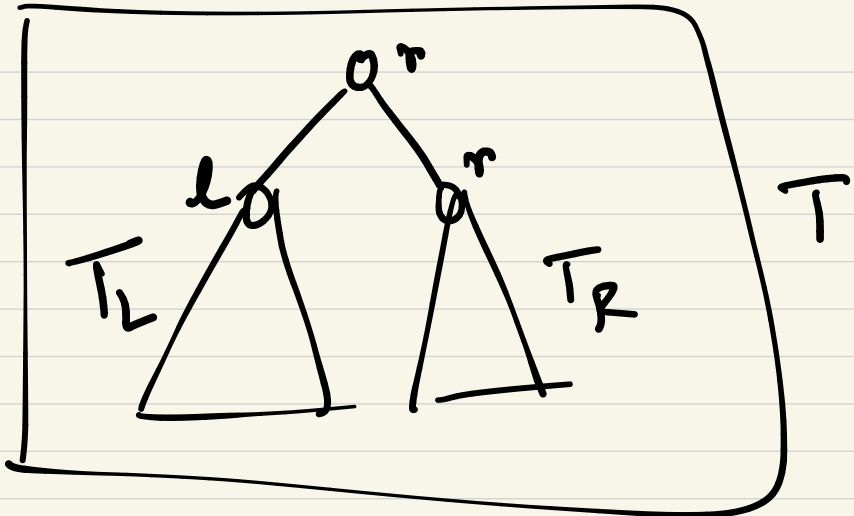
# MaxIS in Trees.

Assume the input tree is a rooted tree



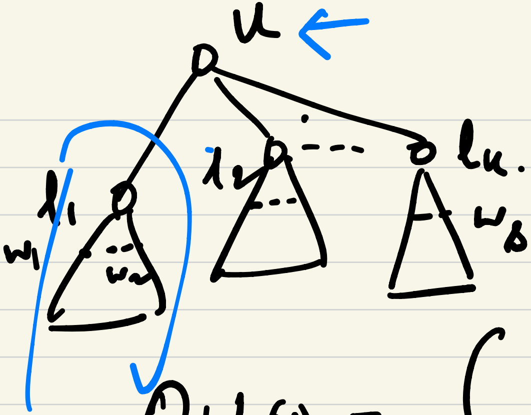

list of pointer





$Opt(x)$ = size of the optimal sol$^n$ to the tree rooted at $x$.

$Opt(x) = 1$ if $x$ is a leaf.



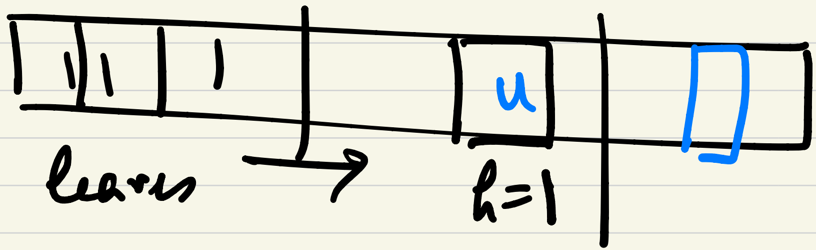

$f(x) = \begin{cases} 1 \text{ if} \end{cases}$

$$Opt(u) = \max \begin{cases} Opt(l_1) + \cdots + Opt(l_k), & u \text{ is not picked} \\ 1 + \sum_{w \text{ is a grandchild of } u} Opt(w) & u \text{ is picked.} \end{cases}$$

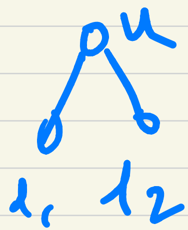
$l_1 \; l_2 \cdots l_k$

Sort the vertices by their heights

Opt




∘ $h(x)$ = length of the longest path from $x$ to a leaf avoiding root ($x \neq$ root)




Runtime $O(n)$

# Greedy Algorithm

$$S = \{\underline{a}_1, \ldots, \underline{a}_k\}$$

$$a_i = [s_i, t_i)$$

$$0 \le s_i < t_i$$