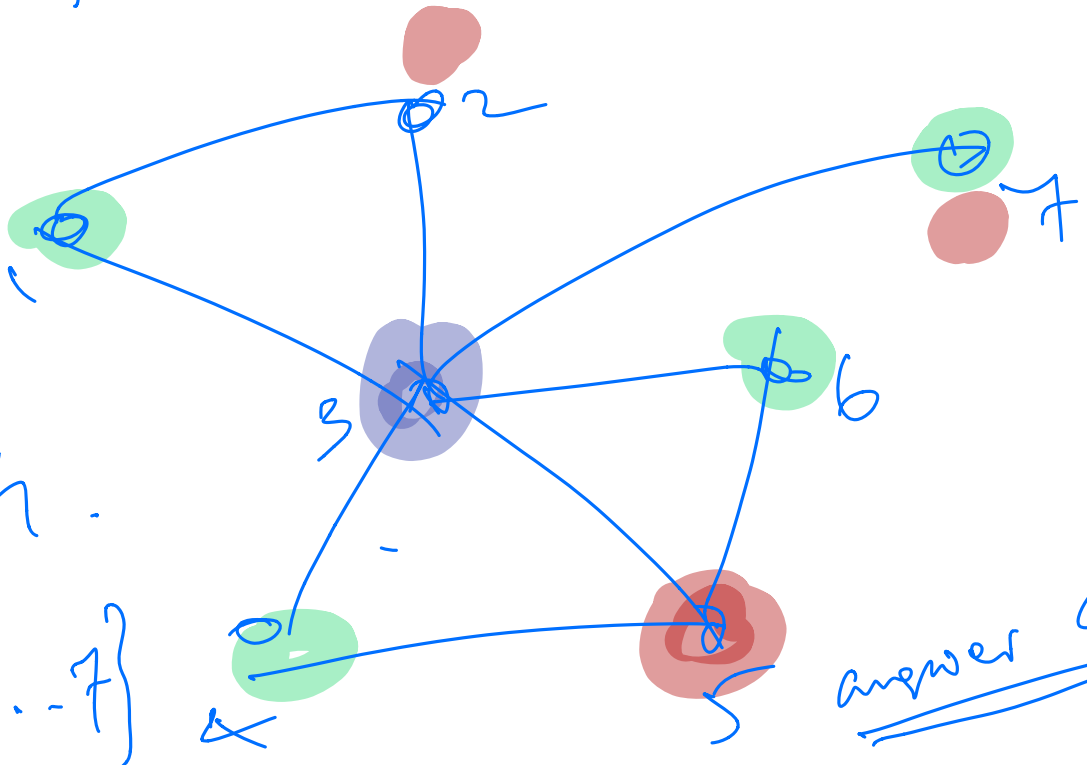
Dynamic Programming

- Writing recurrence
- Find out evaluation order
- Find out auxiliary Data structure

Independent Set



G.

$V(G) = \{1, \ldots 7\}$ &

answer 4.

$S \subseteq V(G)$ is an independent set if for any two vertices $x, y \in S$, $\{x, y\} \notin E(G)$

IND-SET

Input: A graph G

Output: The size of a max independent set in G.

$n = |V(G)|$, $m = |E(G)| \leq n^2$

2^n ← brute-force algorithm

Polynomial time -
polynomial in input length
- input length $N = n+m$
$\leq 2n^2$

$T(n)$ is upperbounded by $f(N)$ where f is polynomial

f is a polynomial such that

$O(N)$ or $O(N^c)$

where c is a constant "independent" on N.

$\downarrow O(1)$

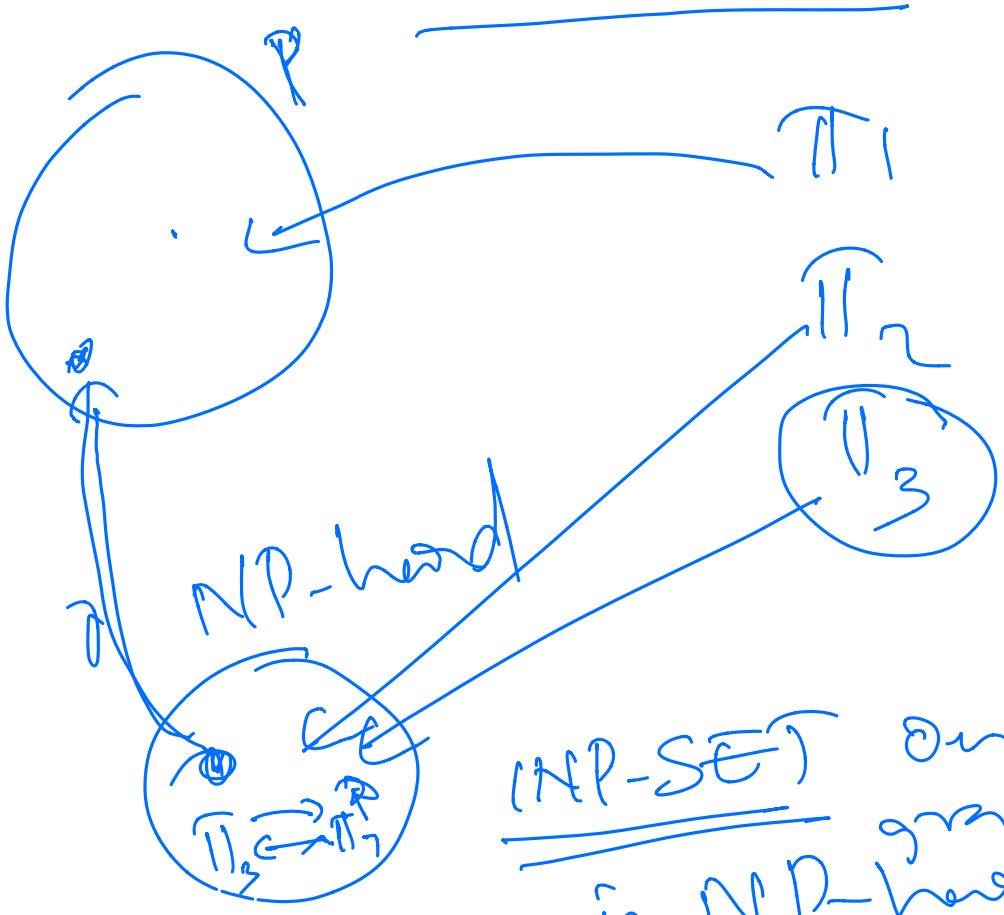
N

$2^N \rightarrow n^2$ ✓

P - the class of problems which are solvable in polynomial time

a problem $\Pi \in P$, if $\exists$ an algorithm for Π $O(1)$ time N,

that solves in
where N is the input length



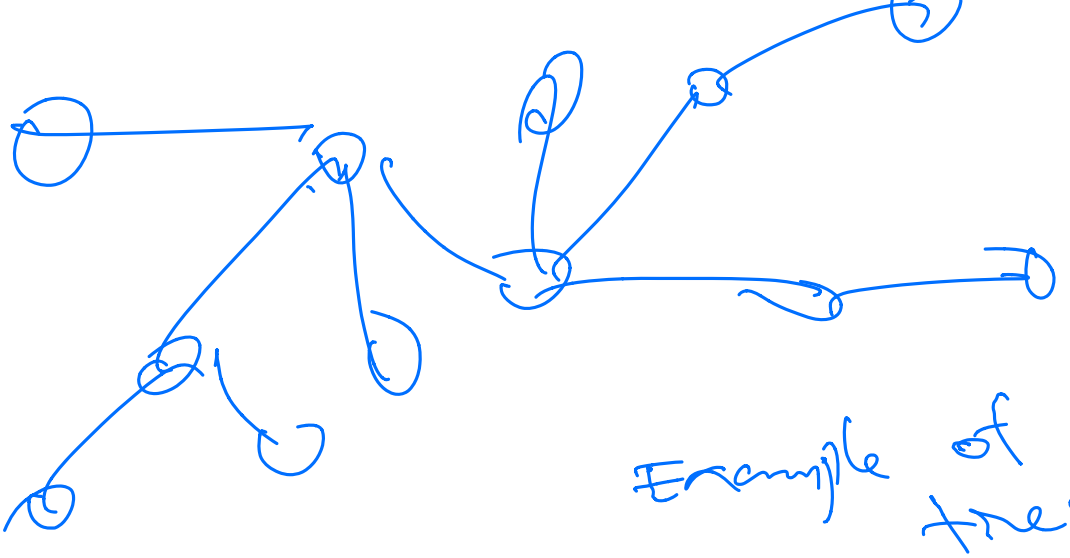
(NP-SET on general graphs is NP-hard.

+ more in the final module

IND-SET-TREES.

Def: Tree

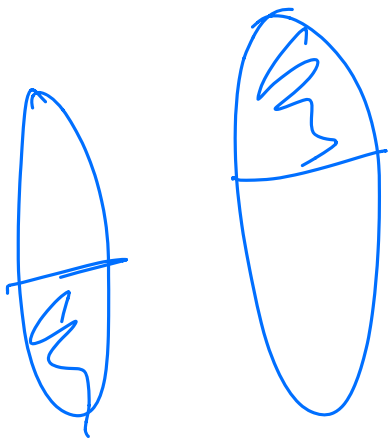
An acyclic connected graph.

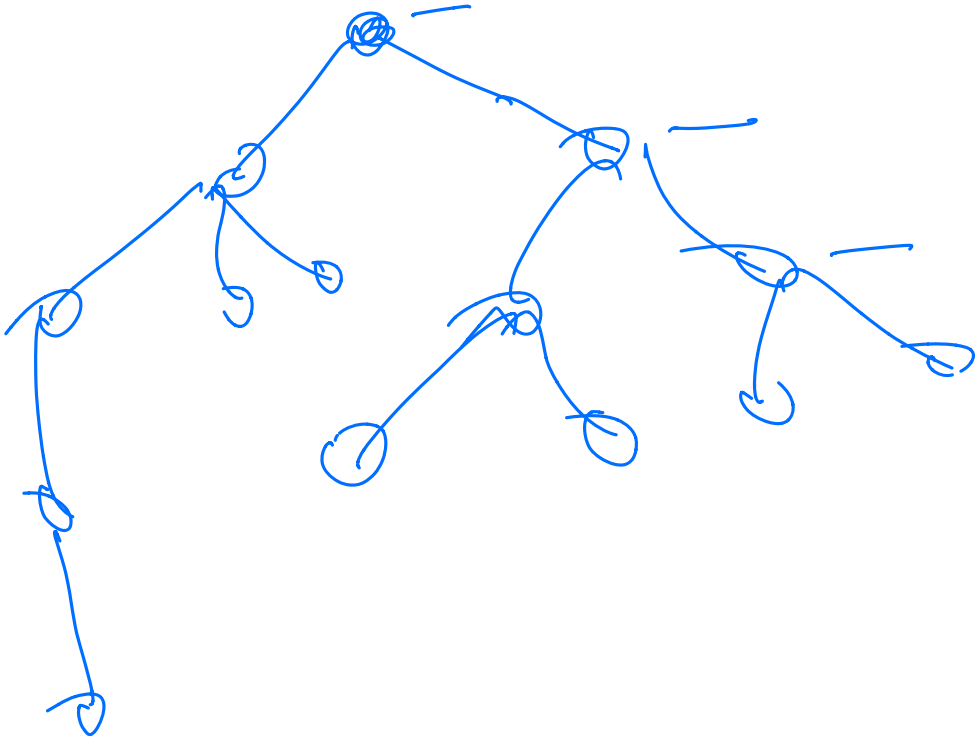


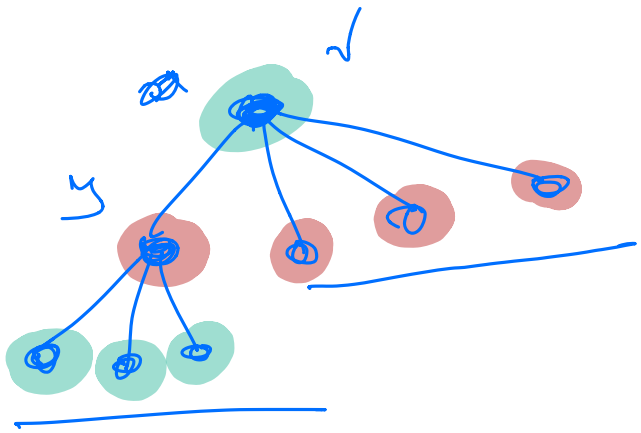
Example of a tree.

Input: A tree. T

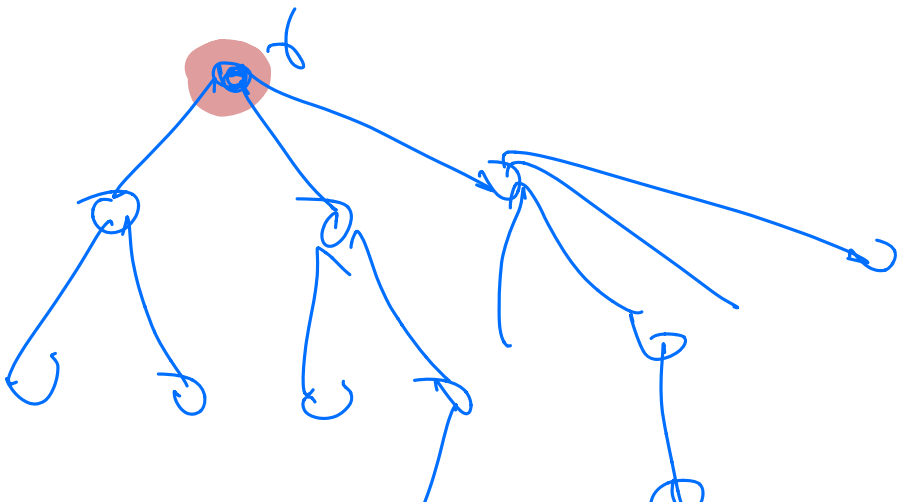
Output: The cardinality of a largest indepdt set in T.







$|Green| = 5$



S^*

$|S^*|$

Case 1 $r \in S^*$

Case $r \notin S^*$

$MIS(r) = 1 + \sum_{g \text{ is grad child of } r} MIS(g)$

$MIS(r) = \sum_{c \text{ is a child of } r} MIS(c)$

$MIS(g_1) + MIS(g_2) + MIS(g_3)$

$\leq$ $\geq$

$MIS(r) = \max$ { $1 + \sum_{g \text{ is grad child of } r} MIS(g)$, $\sum_{c \text{ is a child of } r} MIS(c)$ }

$\geq Q$ //

MIS(v) =
the cardinality of a
max. size independent
set of the subtree of
T, rooted at v.

$MIS(\sigma) \geq \max\{ \; , \; \}$

$|S^*|$

induction on size of the tree.
= depth of the rooted tree.

$MIS(\sigma) \leq$

$\exists S^* \quad |S^*| = MIS(\sigma)$.

Case 1 $v \notin S^*$ $\quad v \in S^*$ $\quad$ by induction

$$|S^* \cap \gamma(T_{q_1})| \leq mis(q_1)$$

$$|S^* \cap \gamma(T_{q_2})| \leq mis(q_2)$$

$$|S^*| = 1 + |S^* \cap \gamma(T_{q_1})| + |S^* \cap \gamma(T_{q_2})| + \cdots$$

$$\leq 1 + \sum_{\substack{q \text{ is grand} \\ \text{child of } v}} mis(q).$$

$$|S^*| \leq \sum_{\substack{q \text{ is graph} \\ \text{of } I}} mis(q)$$

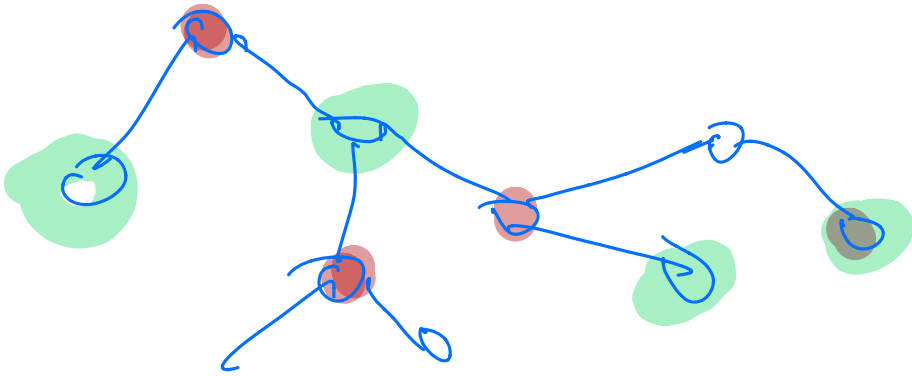
Same interpolation strategy

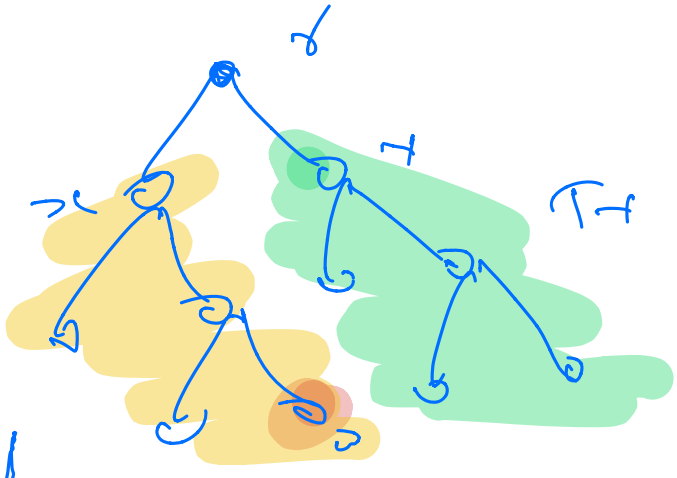
24/2/21

Ind-Set-Trees

Input: A tree T

Output: The cardinality of a maximum independent set in T





T

T_v - the subtree rooted at v - $T_r = T$

T_{vc}

MIS(x) = the cardinality of a max independent set in T_x

$T_0 = T$; Our objective is to find MIS(r)

$$MIS(v) = \max\left\{\sum_{w \downarrow v} MIS(w),\ 1 + \sum_{w \downarrow v}\sum_{x \downarrow w} MIS(x)\right\} = Q$$

Q_1 Q_2

$w \downarrow v$:= w is a child of v

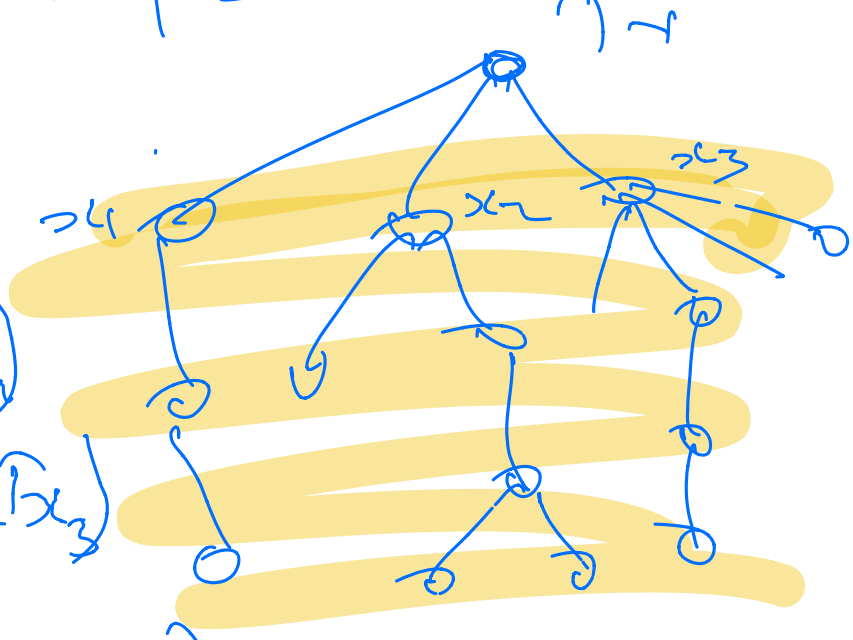
MIS(v) $\geq Q$: induction on the size of the tree / depth of the rooted tree

MIS(r) $\leq Q$

Let S^* be a max indep. set in T_r

Case 1 $r \in S^*$

Case 2: $v \notin S^*$



$S_1^* = S^* \cap V(T_{x_1})$

$S_2^* = S^* \cap V(T_{x_2})$

$S_3^* = S^* \cap V(T_{x_3})$

$\vdots$

$S_\ell^* = S^* \cap V(T_{x_\ell})$

- $S_1^*, S_2^*, \ldots, S_\ell^*$ are pairwise disjoint

- $\bigcup_{i=1}^{\ell} S_i^* = S^*$

$\Rightarrow |S^*| = \sum_{i=1}^{\ell} |S_i^*|$

$\leq \sum_{i=1}^{\ell} MIS(x_i)$

by induction hypothesis

because

$|T_{2i}| < |T_{2i}|$

$\leq Q_1$

$\leq \max\{Q_1, Q_2\}$

$\leq Q^{2i}$

$\leq Q^{2i}$

$mIS(4) \leq Q^{2i}$

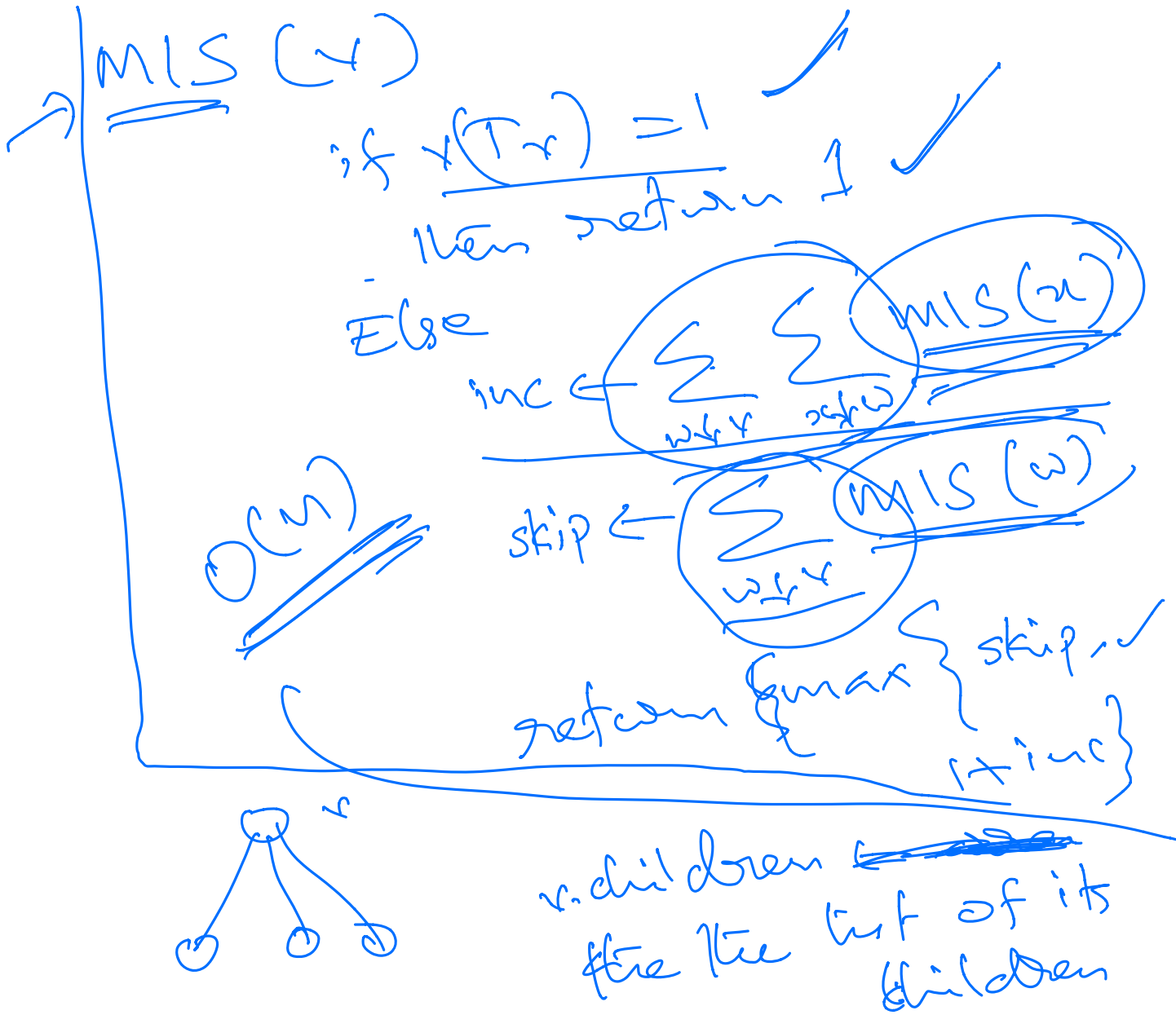
Case 1: When $v \in S^*$

$mIS(4) \leq Q_2$

$\leq \max\{Q_1, Q_2\}$

$\leq Q^{2i}$

$mIS(\alpha)$

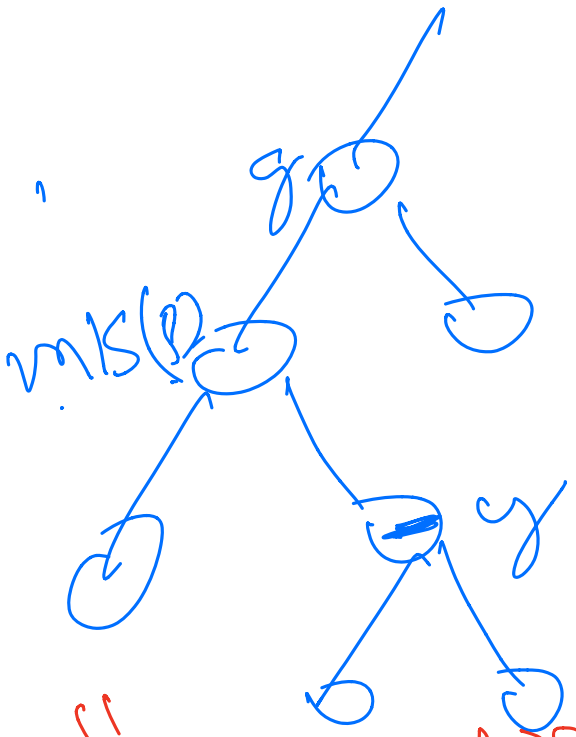


T is the input:

$|V(T)| = n$; $|E(T)| = n-1$

$n_1 + n_2 + \dots = n-1$

$$T(n) = \underbrace{T(n_1) + T(n_2) + \dots}_{O(n)} + T(n_1) + T(n_2) + \dots$$



"this is wrong"

Wrong!! All together you make almost 2n calls to the function MIS

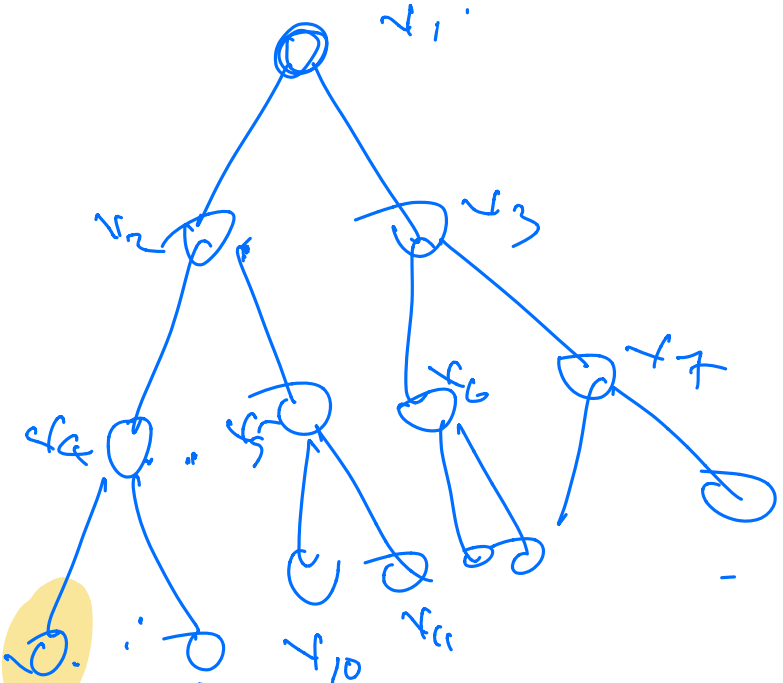
With in function MIS, the no. of operation (other than the one to get output from recursive calls) is at most O(1).

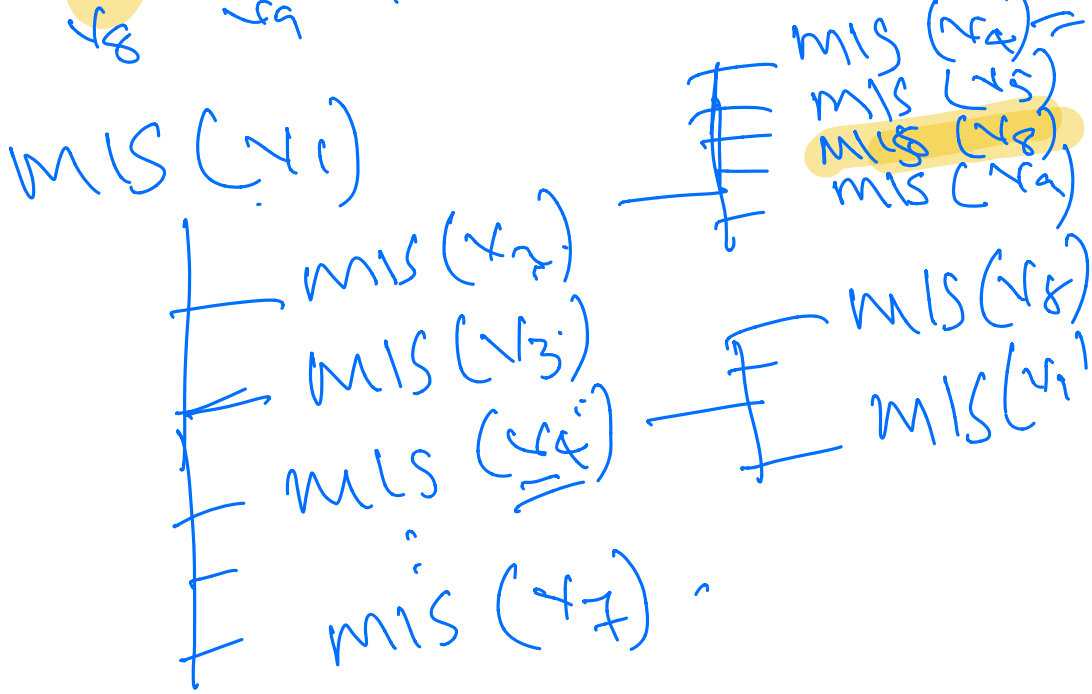
Worst case run time is $O(n^2)$

is polynomial

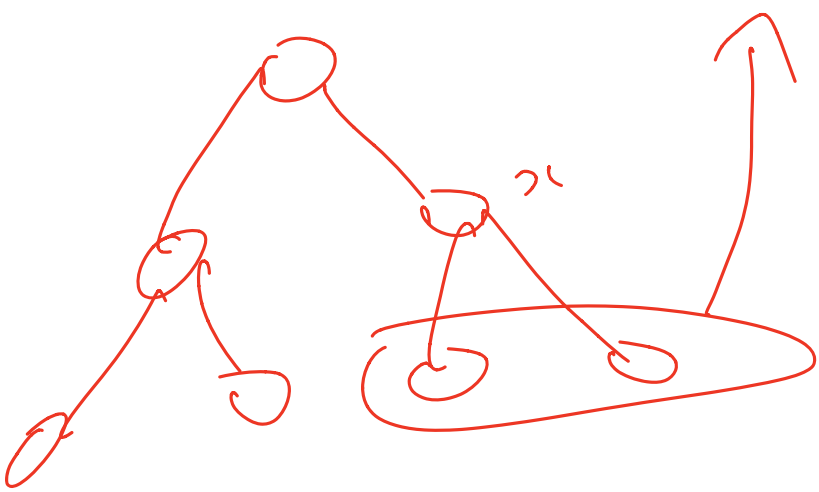
- What is the order in which you compute them
- What is D.S you use:

→ compute it for the children before the parent.

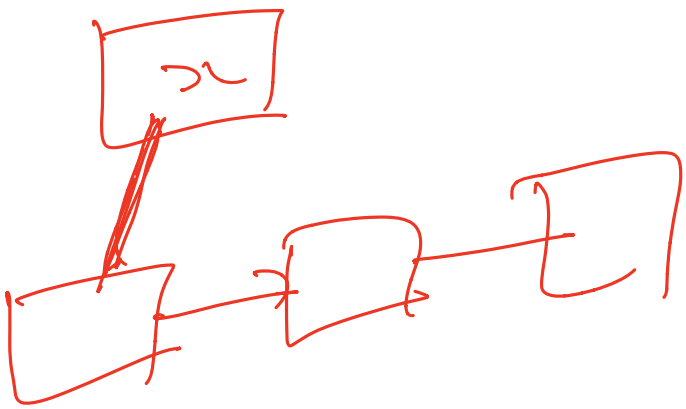


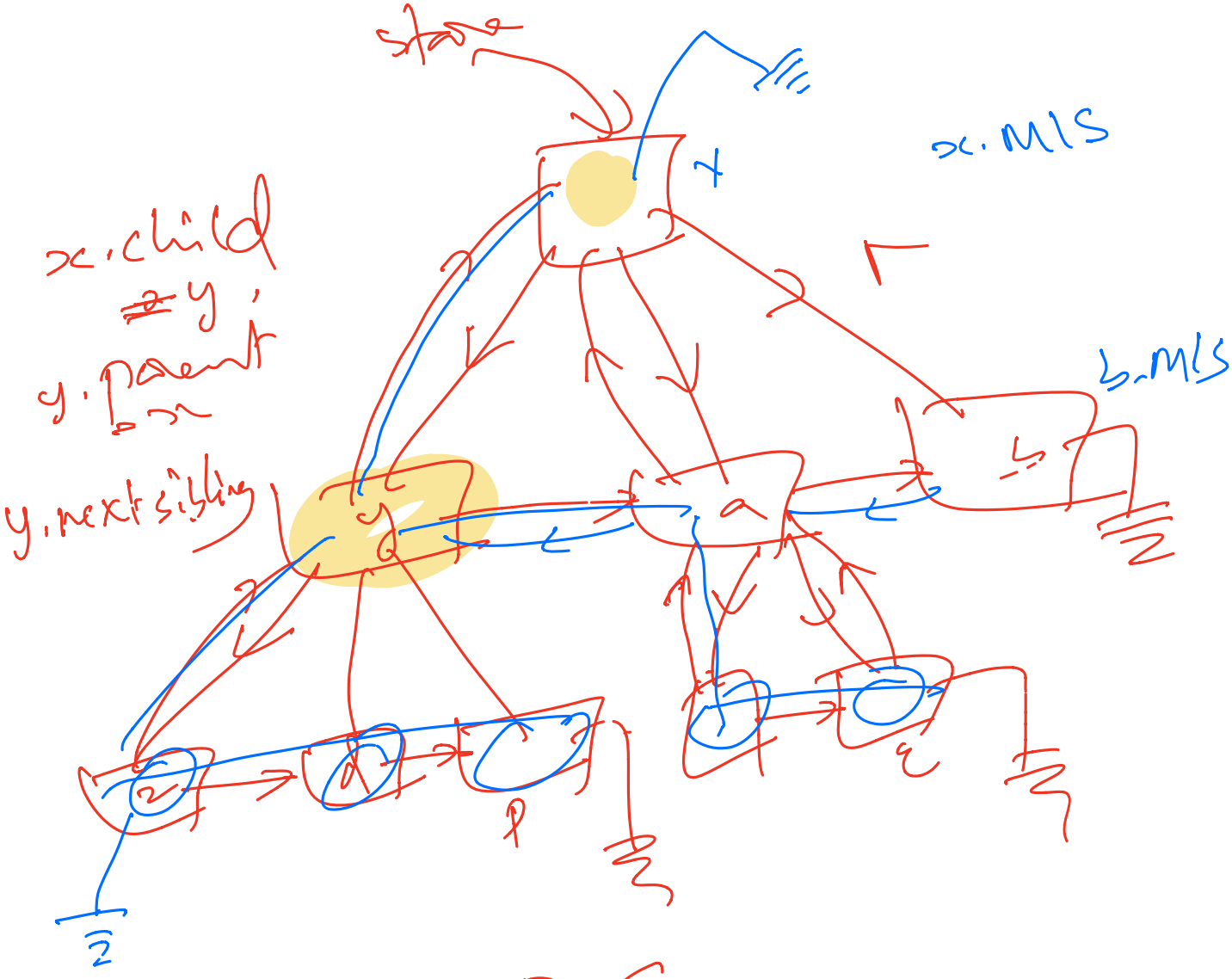


n tree d.s:



x.value



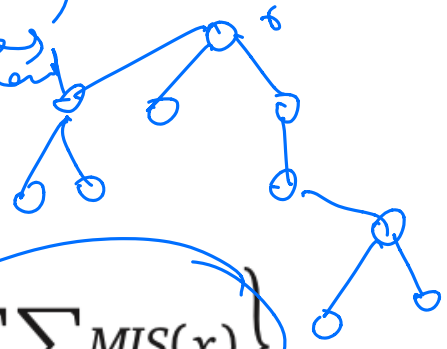


Tree D.S.

Similar to doubly linked list but more involved

25/2/2021

f compute values for child for parent tree - D.S.



$$MIS(v) = \max\left\{\sum_{w \downarrow v} MIS(w),\ 1 + \sum_{w \downarrow v}\sum_{x \downarrow w} MIS(x)\right\}$$

MIS(v) : The cardinality of a max ind. set in the subtree rooted at v.

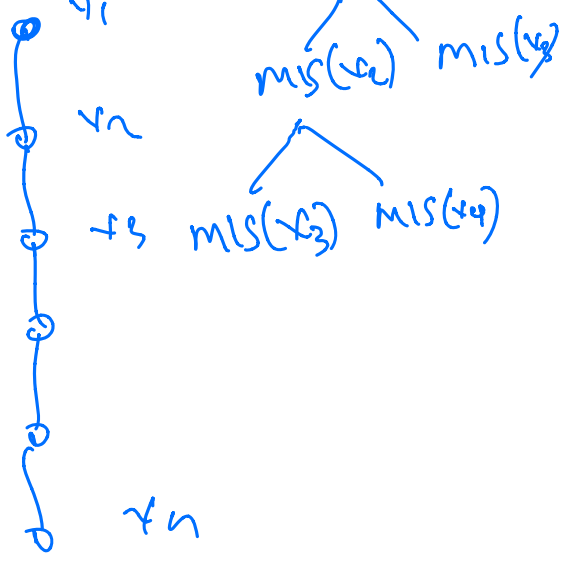
$w \downarrow v$ = w is a child of v.

Naive recursive implementation

$$T(n) = T(n-1) + T(n-2) + c$$

$$T(1) = 1$$

$$T(n) = O(\quad)$$



$T(n) \geq 2^n$

$2^n = 2^{n-1} + 2^{n-2}$

$x^2 = x + 1$

$x^2 - x - 1 = 0$

$\frac{-b \pm \sqrt{b^2-4ac}}{2a}$

$x = \frac{1+\sqrt{5}}{2}$

≤ 1.62 ✓

Use induction

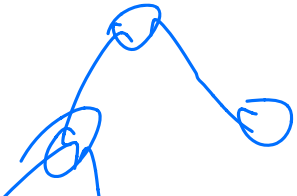
$T(n) \leq (1.62)^n$

$v.MIS \leftarrow$

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

(Not a recursive call; just a lookup in the D.S.

TREEMIS(v) v is a leaf node



Total no. of calls to the function TREEMIS : no. of children + no. of grand children $\leq n$.

$O(n^2)$

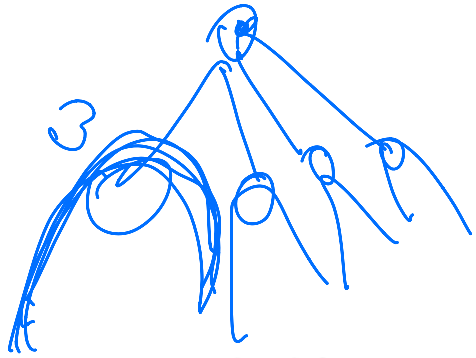
Is this a tight analysis?

— ~~Yes~~ Not tight.

the total no. of ~~calls~~ elties steps in all execution of the function TREEMIS is $O(n)$.

∴ The worst-case running

time = $O(n)$.



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

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

$MISyes(v)$ = the size of the max indep. set in T_v containing v.

$MISno(v)$ = the size of the max indep. set in T_v excluding v.

Dynamic Programming.

- recursive formula
- bottom-up order to compute values
- select suitable D.S.
- write a pseudocode

```
TREEMIS2(v):
    v.MISno ← 0
    v.MISyes ← 1
    for each child w of v
        v.MISno ← v.MISno + TREEMIS2(w)
        v.MISyes ← v.MISyes + w.MISno
    return max{v.MISyes, v.MISno}
```