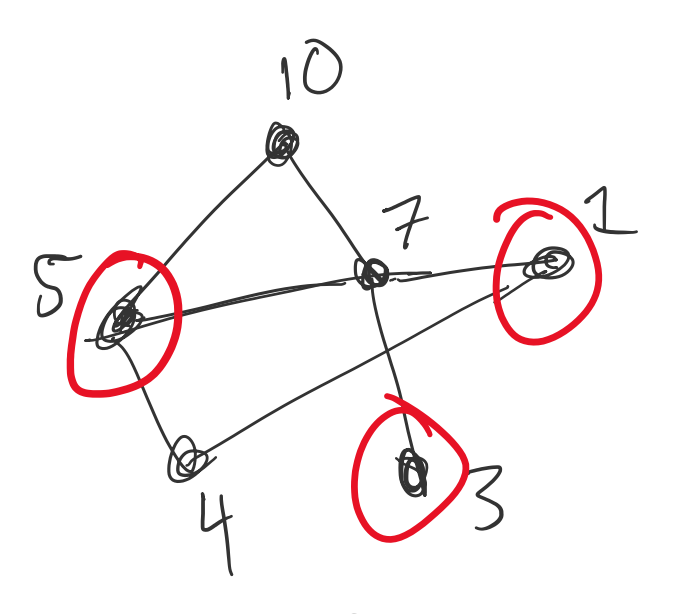
4s. MWIS (Dynamic Programming)

Friday, March 12, 2021
11:44 AM

Max Weight Independent Set Problem (MWIS)

Input: Graph: $G = (V, E)$ (← set vertices, ← set of edges)
Weights: $w : V \to \mathbb{Z}^+$ ($\mathbb{N}$)



Output: $S \subseteq V$ s.t.

Independent set ⇒ • If $\{u, v\} \in E$, $u \notin S$ or $v \notin S$

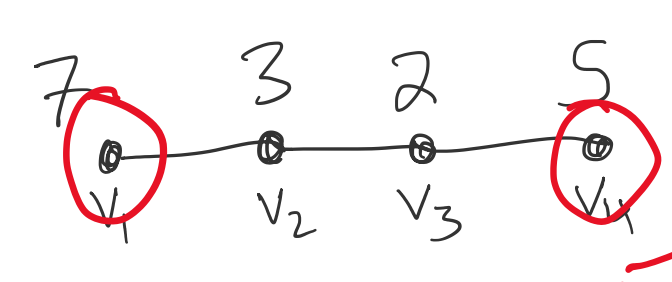
Max weight ⇒ • $W(S) = \sum_{v \in S} w(v)$ is maximized (objective function)

Applications
- Cell Tower Transmission
- Choose Franchise Location
- Party Invite
- Scheduling

☆ General graph → very hard problem

☆ Only look at line graph

What is $W(S)$ for MWIS S of



$S = \{v_1, v_4\}$

~~A) 8~~ ~~B) 9~~ **C) 12** ~~D) 17~~

v_1 v_2

00
01
10
11

Ind Set	Weight
$\emptyset$	0
$\{v_1\}$	7
$\{v_1, v_3\}$	9
~~$\{v_2, v_3\}$~~	⋮

n-vertex graph

For each $S \subseteq V$ ⇐ $O(2^n)$
  If Ind. Set: ⇐ $O(n^2)$
    Check Weight ⇐ $O(n)$
    Store if largest seen ⇐ $O(1)$

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

Settling in? Change in tutoring/office hours? Talk today! WICS++ Wed! Hacking Thurs!

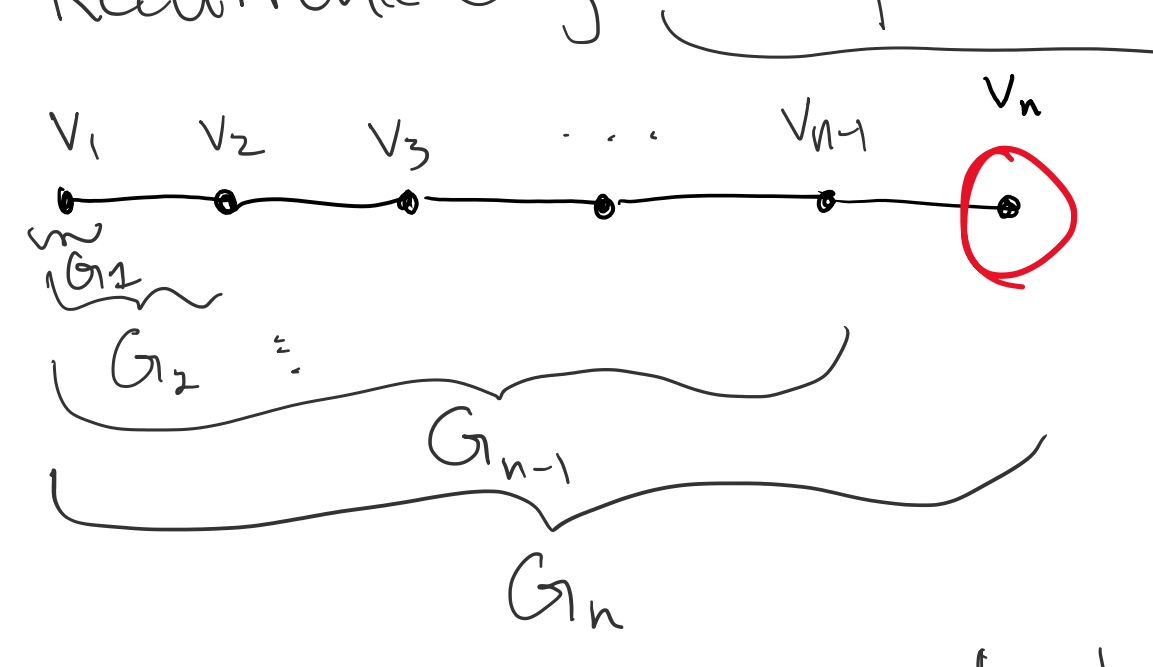
Goals:
- Design a dynamic programming algorithm for MWIS

Qs: Big idea with greedy proof. Why "greedy"? More than two vertices?

Divide + Conquer... better but not best

Designing a Dynamic Programming Alg.

0. Find series of increasingly smaller similar subproblems
   Recurrence Object: Optimal output of each subproblem.



$S_i = \text{MWIS } G_i$

1. Think about cases for final element of recurrence obj.

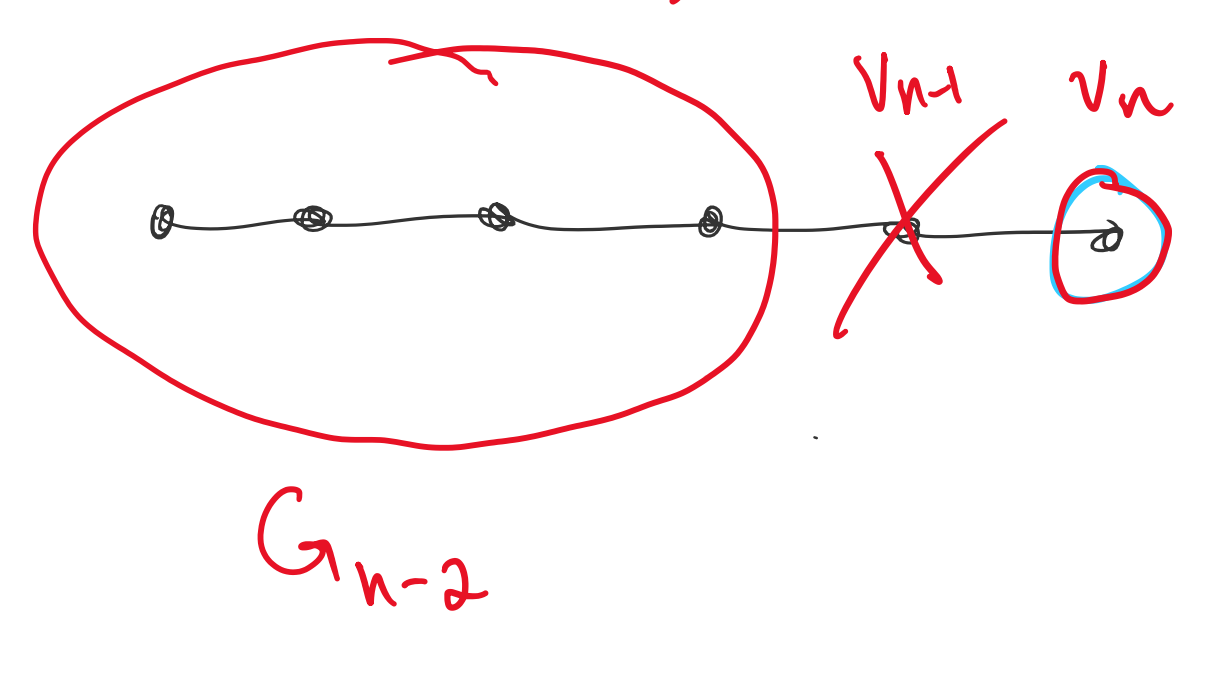
   S_n
   i) $v_n \notin S_n$
   ii) $v_n \in S_n$

2. For each case, create a recurrence

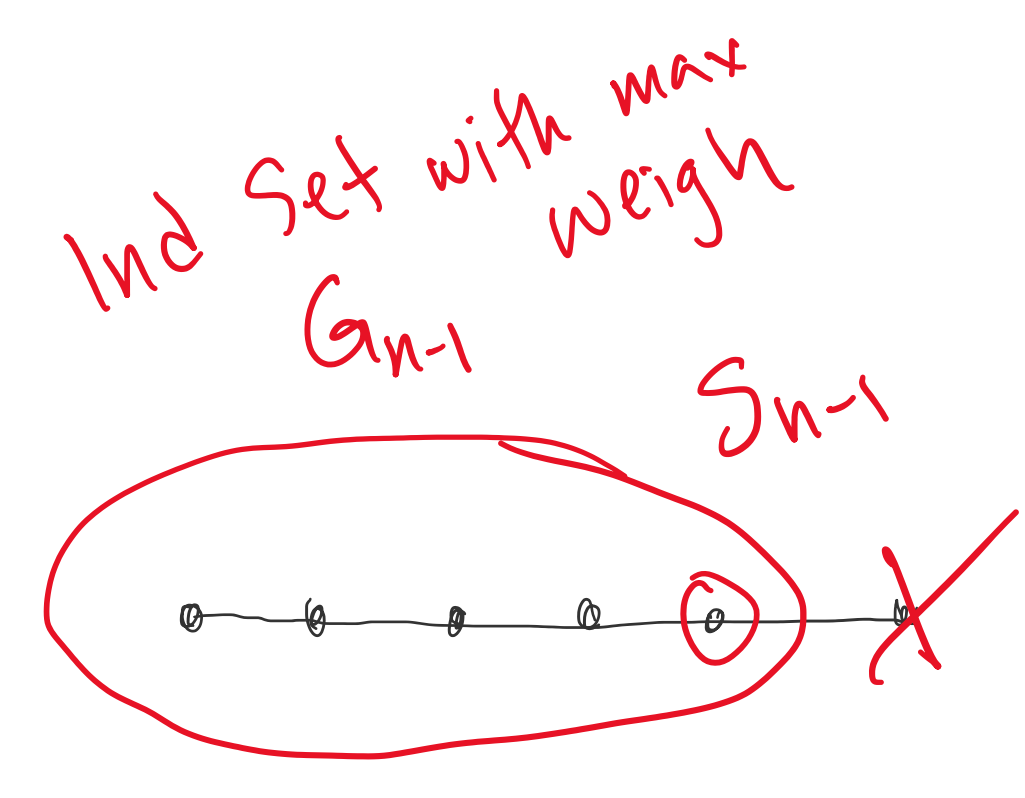
   i) If $v_n \notin S_n$, $S_n = S_{n-1}$
   ii) If $v_n \in S_n$, $S_n = S_{n-2} \cup \{v_n\}$

   Options:
   S_{n-1}
   S_{n-2}
   $S_{n-1} \cup \{v_n\}$
   $S_{n-2} \cup \{v_{n-1}\}$
   $S_{n-2} \cup \{v_n\}$

ii)



i)

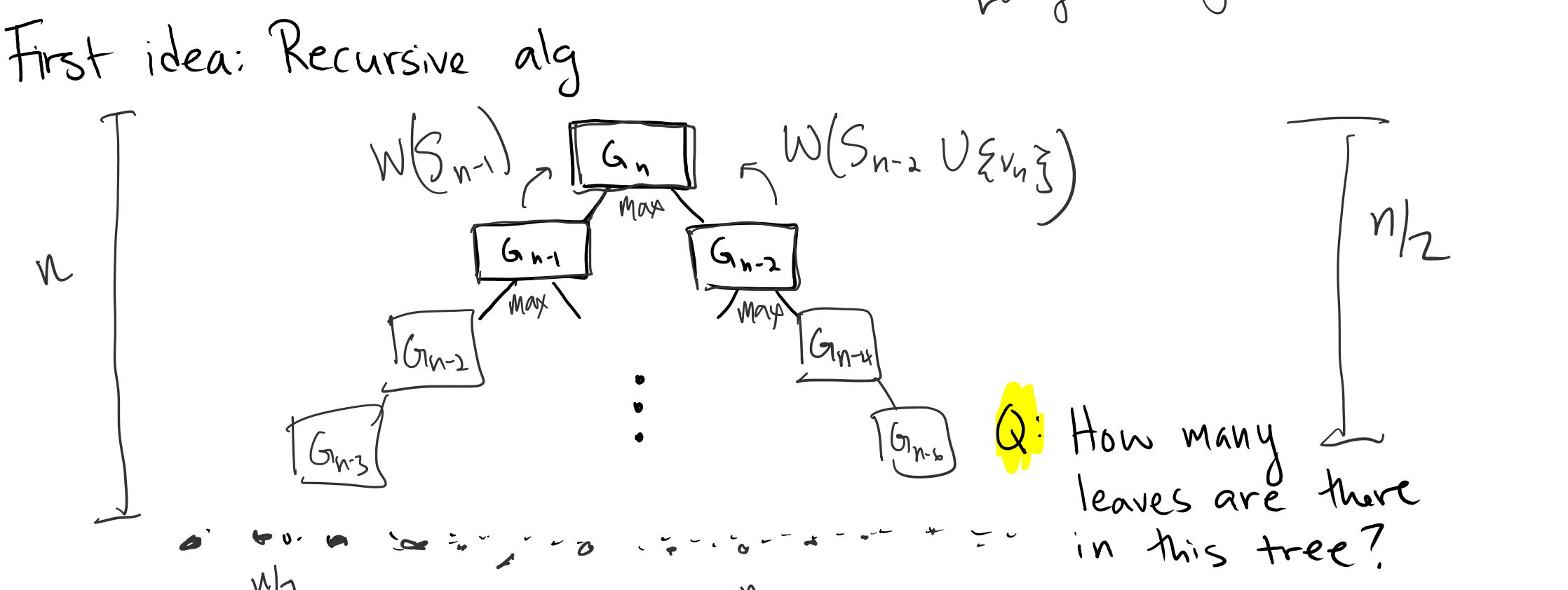


Conclusion

$$S_n = \begin{cases} S_{n-1} \\ \text{or} \\ S_{n-2} \cup \{v_n\} \end{cases}$$

ONLY Possibilities
Evaluate both + Take better option
⇓
Larger weight

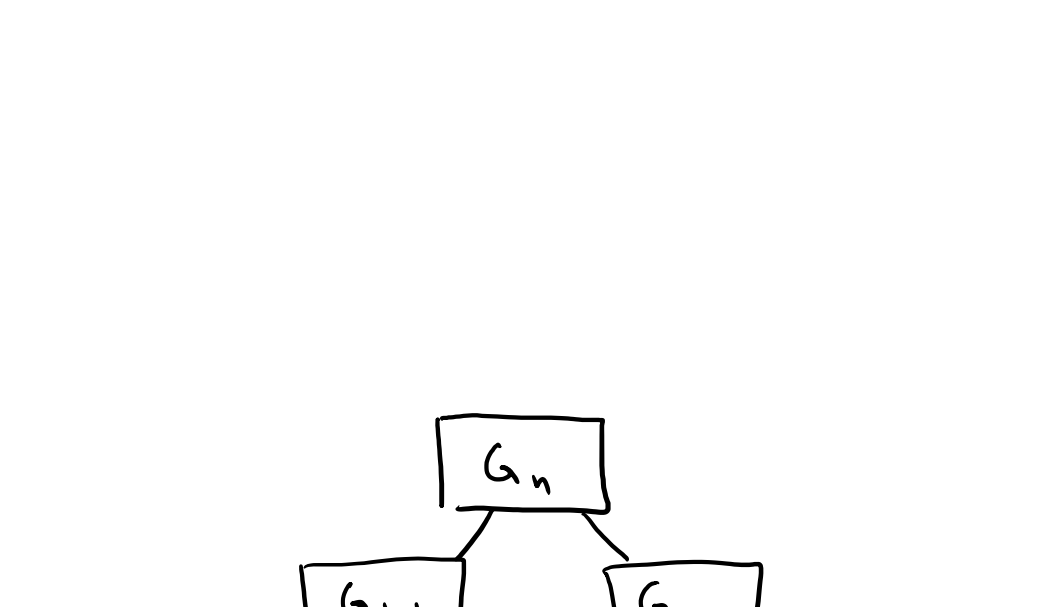
First idea: Recursive alg



Q: How many leaves are there in this tree?

$2^{n/2} <$ #leaves $< 2^n$

A) $O(1)$ B) $O(n)$ C) $O(n^2)$ **D) $O(2^n)$**



☆ Actually solving same problems over and over!

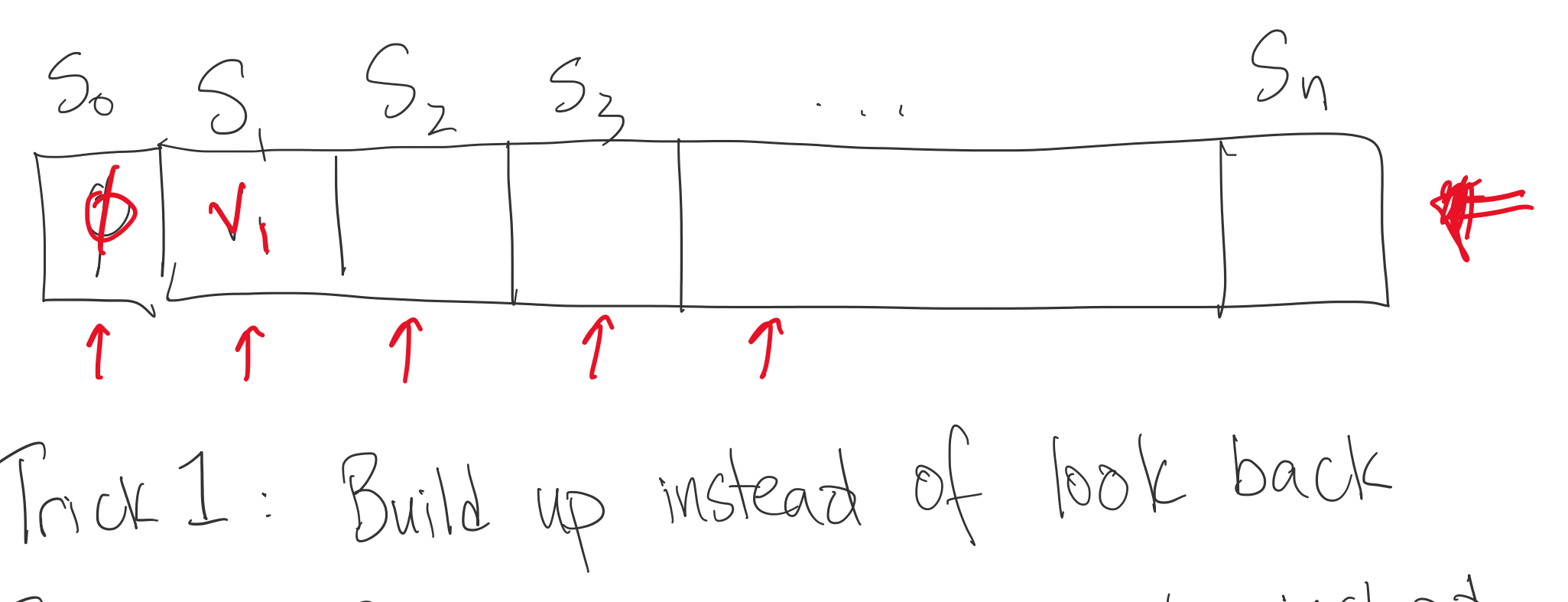
Q. How many distinct subproblems are there?

A) $O(1)$ **B) $O(n)$** C) $O(n^2)$ D) $O(2^n)$

$G_n, G_{n-1}, G_{n-2} \dots G_1$

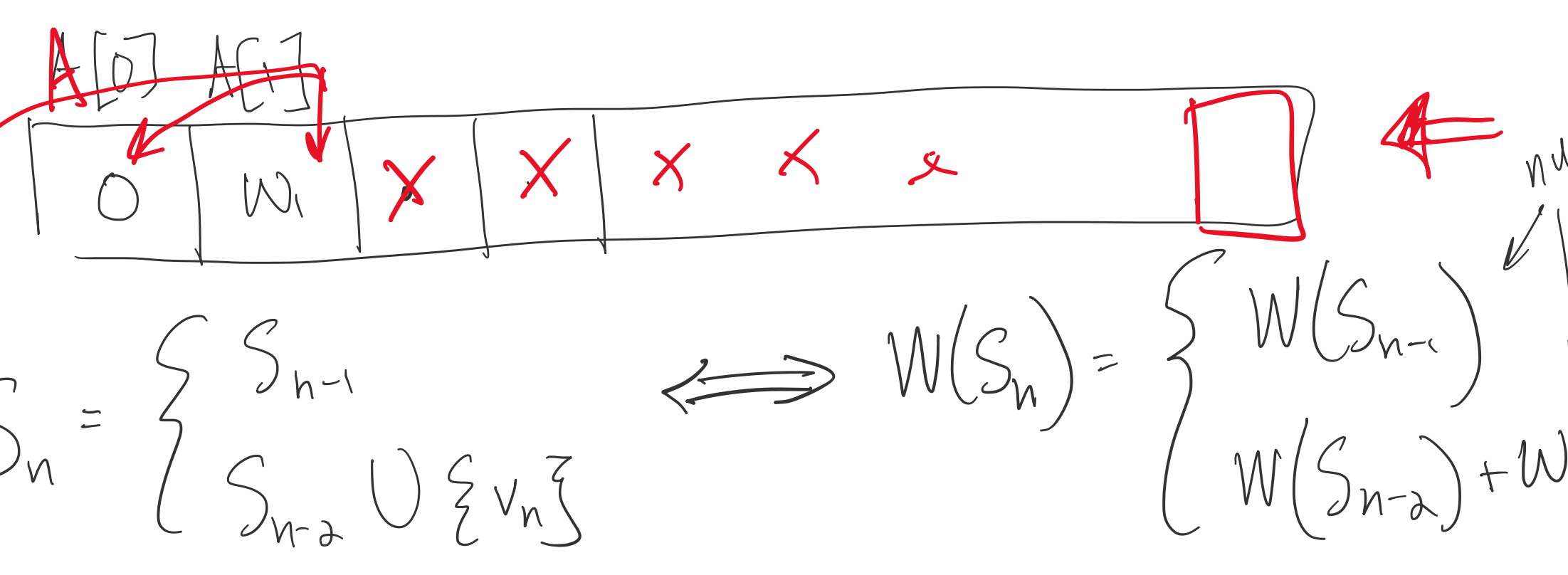
Idea: Instead of solving recursively, store solutions in an array, look up.

G_0 G_1 (v_1)



Trick 1: Build up instead of look back

Trick 2: Store objective function value instead of set/strategy object (faster)



$$S_n = \begin{cases} S_{n-1} \\ S_{n-2} \cup \{v_n\} \end{cases} \iff W(S_n) = \begin{cases} W(S_{n-1}) \\ W(S_{n-2}) + W(v_n) \end{cases}$$

MWIS on Line

// Determine $W(S_i)$'s:

1. $A[0] \leftarrow 0$ } Base cases
2. $A[1] \leftarrow w_1$
3. For $i = 2$ to n:
   $A[i] \leftarrow \max\{A[i-1], A[i-2] + w(v_i)\}$