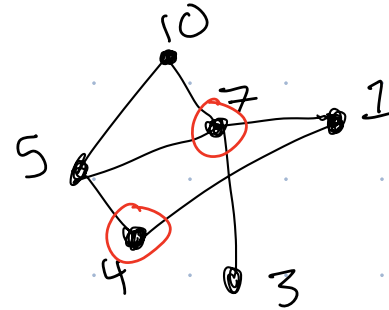


Max Weight Independent Set

Optimization
Problem

Input: Graph $G = (V, E)$
weights $w: V \rightarrow \mathbb{Z}^+$



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

• If $\{u, v\} \in E \quad \neg (u \in S \wedge v \in S)$ } Constraint

• Maximizes $W(S) = \sum_{v \in S} w(v)$

Objective function

Applications:

- Cell tower transmission
- Choose franchise locations
- Party invite
- Scheduling
- ~~House robbing~~

General Graph: HARD

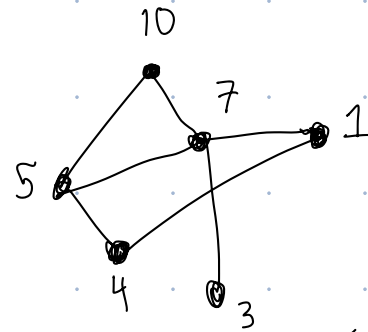
Line Graph: Easy, with



Dynamic
Programming

Max Weight Independent Set (MWIS)

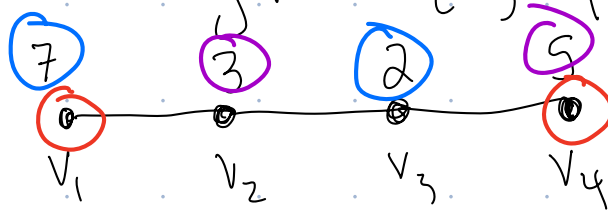
Input: Graph $G = (V, E)$
weights $w: V \rightarrow \mathbb{Z}^+$



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

- If $\{u, v\} \in E$, $\neg (u \in S \wedge v \in S)$ $\Leftarrow$ "Independent Set Condition"
not $\downarrow$ and
- Maximizes $W(S) = \sum_{v \in S} w(v)$ $\Leftarrow$ "Weight of S"
"constraint"

What is max weight $W(S)$ for MWIS of this line graph:



A) 0

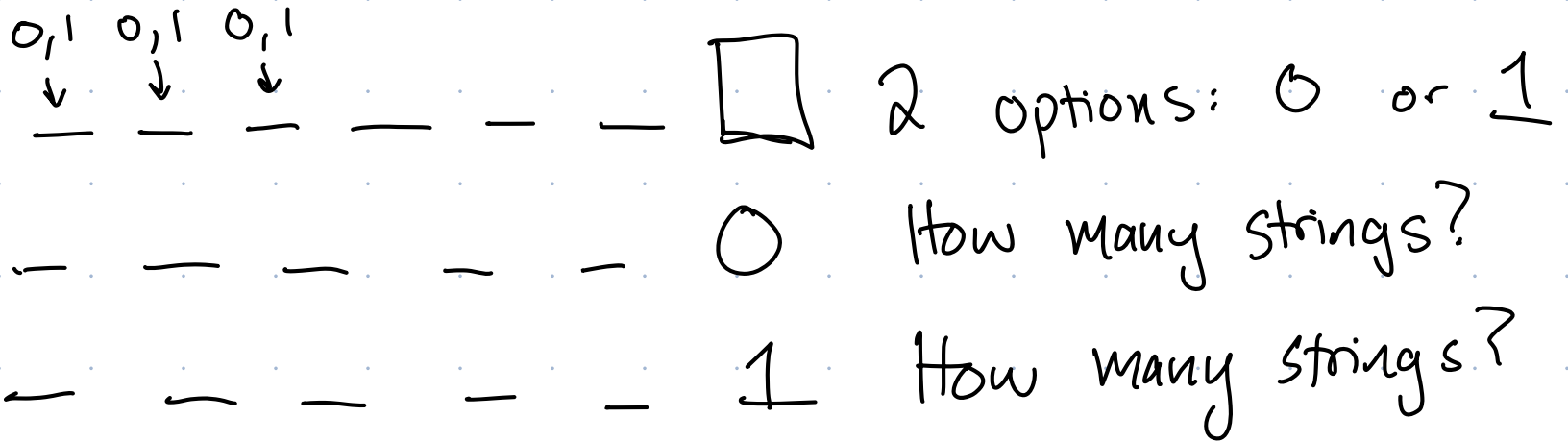
B) 8

C) 9

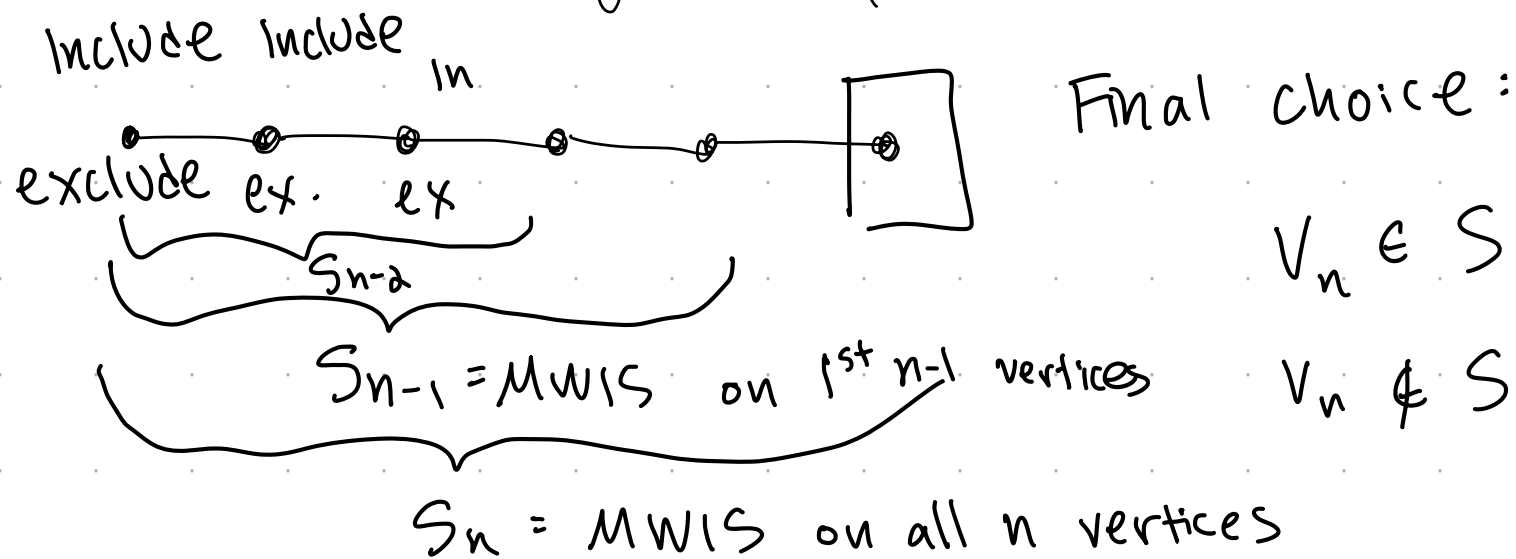
D) 12

Dynamic Prog. Approach

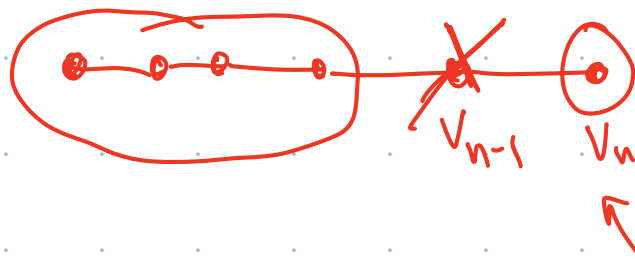
Recall: # of n bit strings with 2 consecutive ones



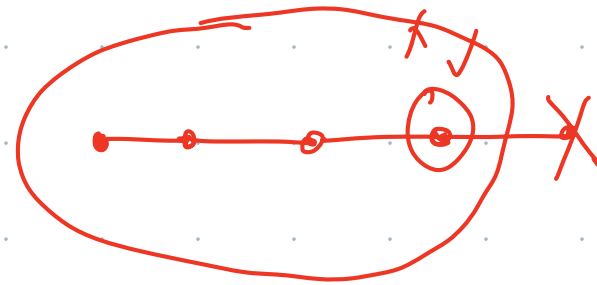
To create a DP alg, (often) need to conceptualize the optimal solution as a sequence of choices



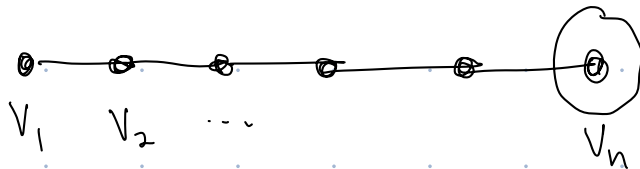
Dynamic Prog. Approach



or



MWIS



2 cases : $v_n \in S$ or $v_n \notin S$

Let's call S_i the MWIS of first i vertices

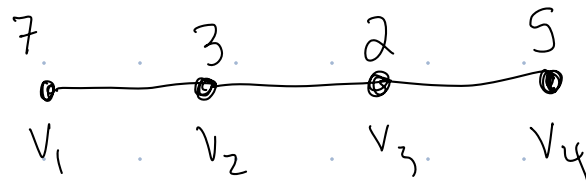
- $v_n \in S$, then $S_n = \underline{S_{n-2}} \cup \{v_n\}$ Options: S_{n-1} , S_{n-2} , $S_{n-1} \cup \{v_n\}$
- $v_n \notin S$, then $S_n = \underline{S_{n-1}}$ $S_{n-2} \cup \{v_n\}$, $S_{n-2} \cup \{v_{n-1}\}$

Name, pronouns, best thing watched/listened to recently

Write pseudocode for brute force approach, analyze runtime

Brainstorm greedy, divide + conquer approaches

Brute Force $O(n2^n)$



MWIS($G=(V,E), w$)

max $S \leftarrow \emptyset$

max $W \leftarrow 0$

For each set $S \subseteq V$: $O(2^n)$

 If S is I.S.:

$w \leftarrow W(S)$

 if $w > \text{max } W$ then

 max $S \leftarrow S$

 max $W \leftarrow w$

Return max S

0110

0001

0010

S is I.S.:

For $i \leftarrow 1$ to $n-1$:

 If $v_i \in S$ and $v_{i+1} \in S$:

 Return False

Return true

$O(n)$

$W(S)$:

$W \leftarrow 0$

For $i \leftarrow 1$ to n

 If $v_i \in S$

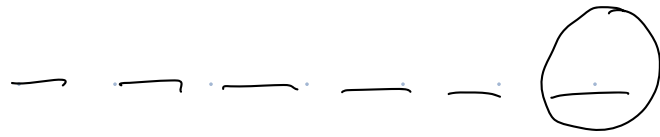
$W \leftarrow W + w(v_i)$

return W

$O(n)$

Dynamic Prog. Approach

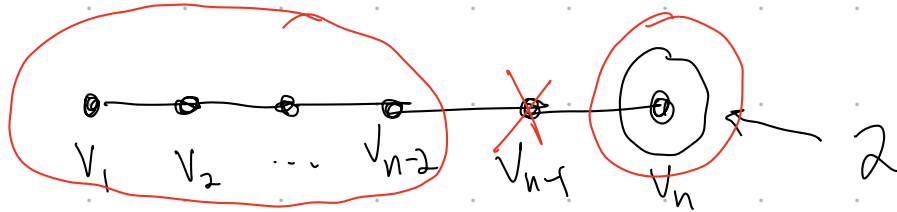
Recall: # of n bit strings with 2 consecutive ones



← 2 cases, 0 or 1

↓
 $T(n-1)$ $T(n-2)$

MWIS



2 cases: $v_n \in S$ or $v_n \notin S$

Let's call S_i the MWIS of first i vertices

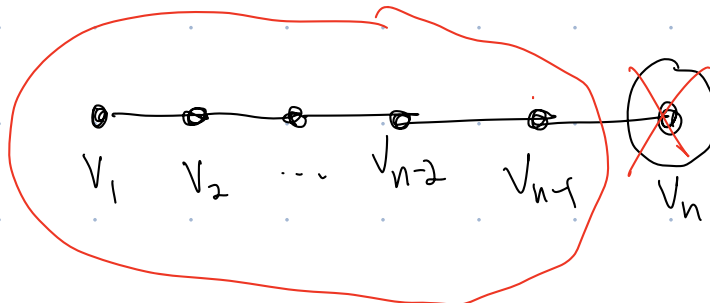
• $v_n \in S$, then $S_n = S_{n-2} \cup \{v_n\}$

• $v_n \notin S$, then $S_n = S_{n-1}$

Options:

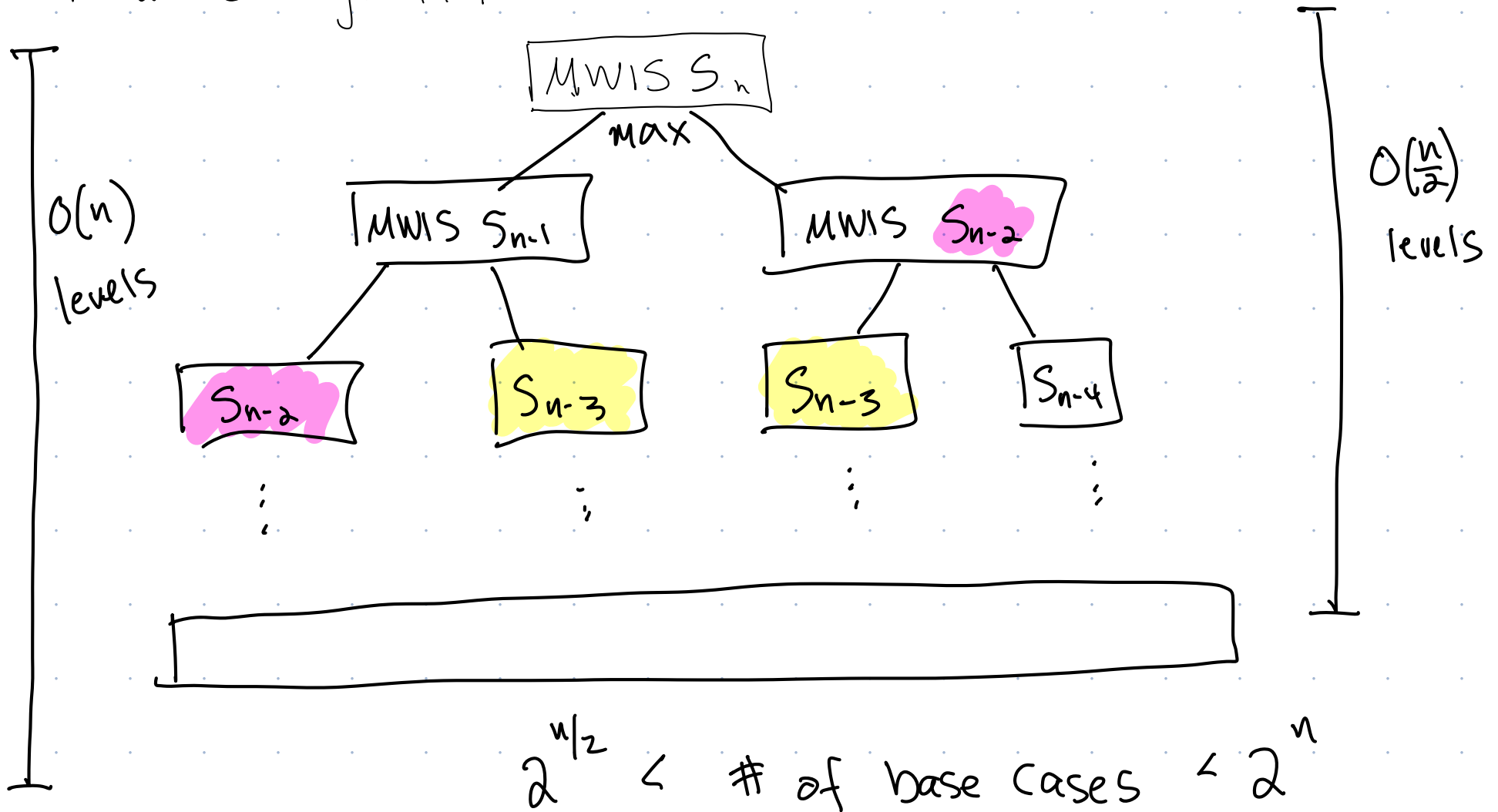
S_{n-1} , $S_{n-2} \cup \{v_n\}$

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



$$S_n = \begin{cases} \text{max Weigh} \begin{cases} S_{n-1} \\ S_{n-2} \cup \{v_n\} \end{cases} \\ \text{Base case} \end{cases}$$

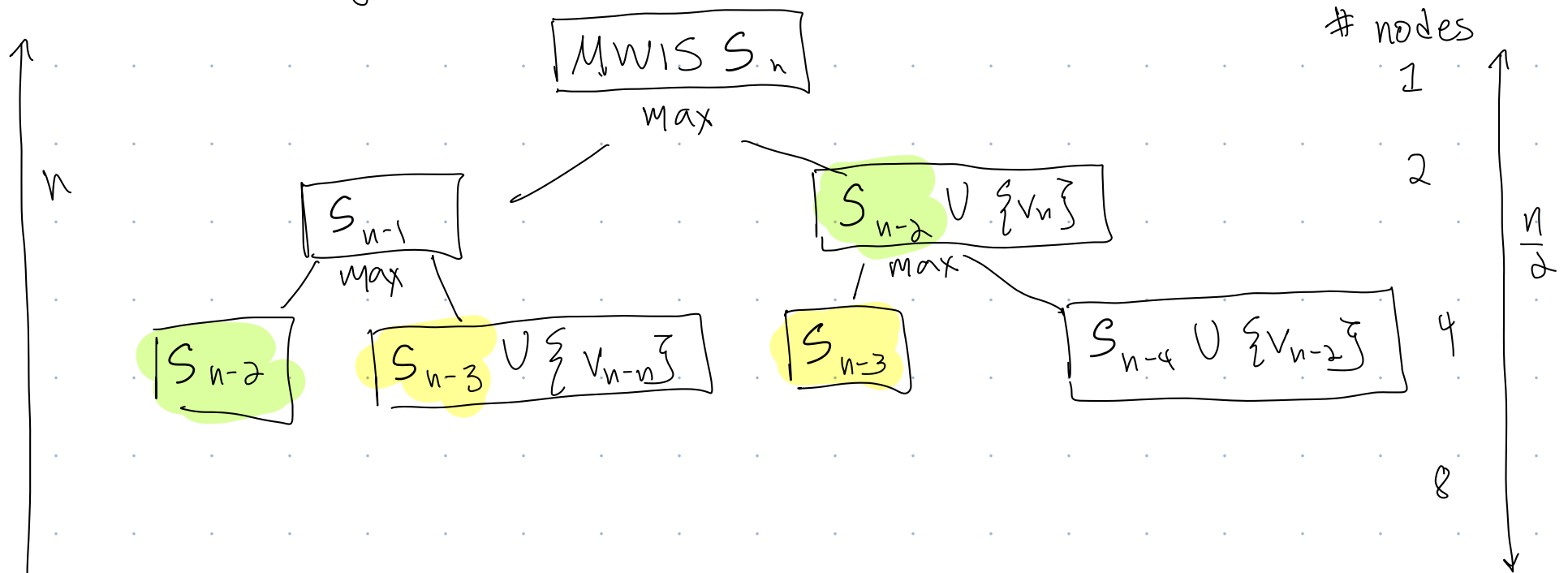
Recursive Algorithm:



$$S_n = \begin{cases} \max_{\text{weight}} \left\{ \begin{array}{l} S_{n-1} \\ S_{n-2} \cup \{v_n\} \end{array} \right\} & \text{Only 2 possible options, take larger!} \\ \text{base case} & \end{cases}$$

★ And base case! Later..

Recursive Algorithm:



$$2^n > \# \text{ of nodes in tree} > 2^{n/2}$$

Each node requires $\Omega(1)$ time $\rightarrow \Omega(2^{n/2})$ algorithm
 ↑ asymptotic lower bound ↑
Bad!

How many unique subproblems are there?

A) $\sqrt{n}$

B) $\frac{n}{2}$

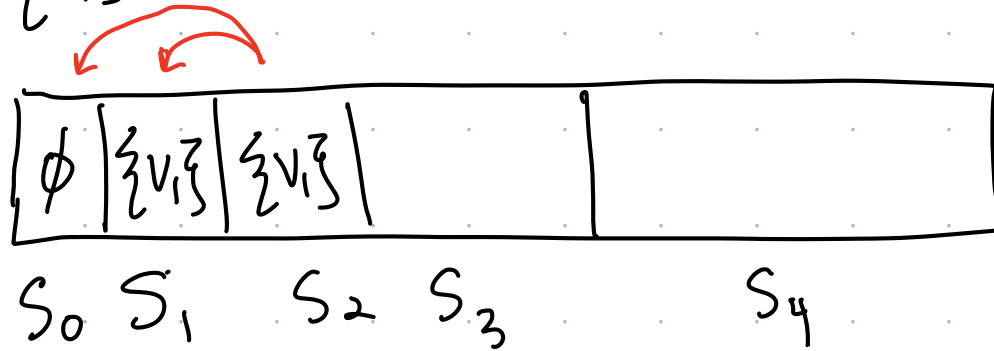
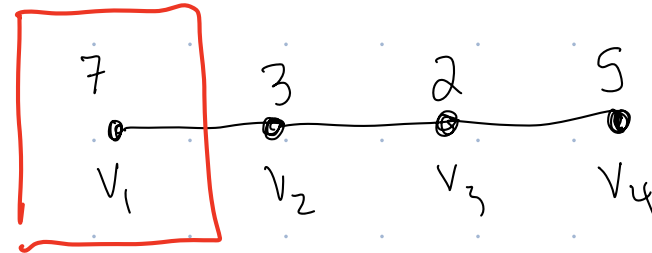
C) n

D) n^2

$S_n, S_{n-1}, S_{n-2} \dots S_1$

Dynamic Programming Idea: Store subproblems in an array + look up

$$S_n = \begin{cases} \max_{wt} \{ S_{n-1} \\ S_{n-2} \cup \{v_n\} \} \\ \emptyset \text{ if } n=0 \\ \{v_1\} \text{ if } n=1 \end{cases}$$



Trick: Start at smallest subproblem and go up

Trick: Create a size 0 problem