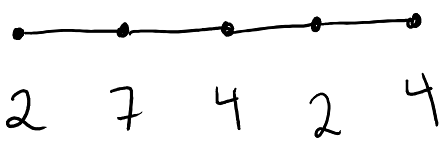
S.KIMMEL

Goals

- Create MWIS algorithm
- Describe recursive relations of MWIS
- Test greedy algorithms

Quiz: upload solution to Canvas. Have scanning device ready.
Check it is readable.

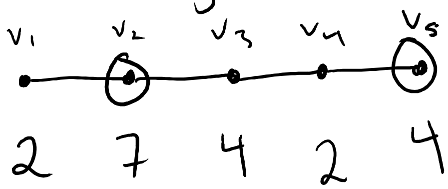
Q: What is weight of MWIS of



A) 9 B) 10 C) 11 D) 13

S.KIMMEL

Q: What is weight of MWIS of

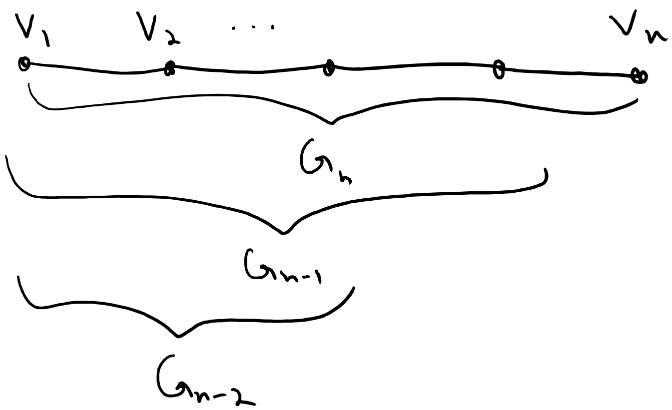


Optimal set $= \{v_2, v_5\}$

A) 9   B) 10   C) 11   D) 13

⇑ weight = objective function

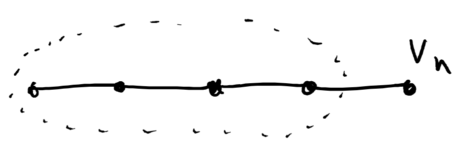
Recall



2 options for S, the MWIS on G_n:

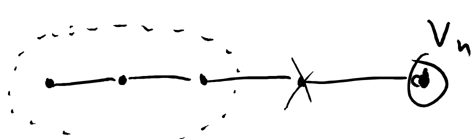
- $v_n \in S$
- $v_n \notin S$

$v_n \notin S$:



Best we can do for S is MWIS on G_{n-1}

$v_n \in S$:



Best we can do for S is $[\text{MWIS on } G_{n-2}] + v_n$

S.KIMMEL

Conclusion

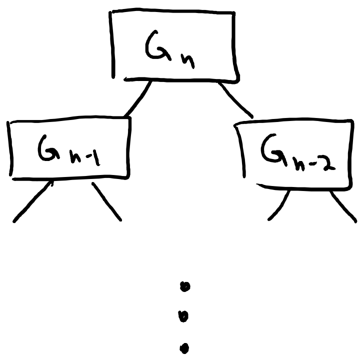
MWIS on G_n is:

i) [MWIS on G_{n-1}]

OR

ii) [MWIS on G_{n-2}] + $\{v_n\}$

$\Bigg\}$ Only possibilities. Take whichever is better



Q: How many leaves are there in this tree?

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

S. KIMMEL

Conclusion

MWIS on G_n is:

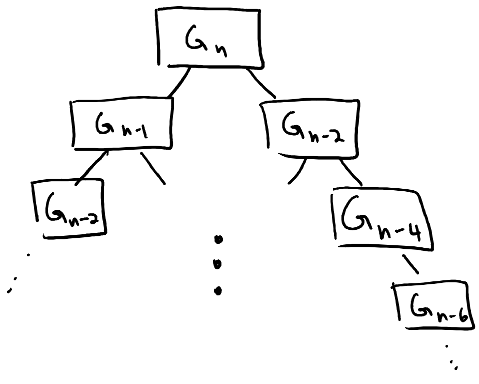
i) [MWIS on G_{n-1}]

OR

ii) [MWIS on G_{n-2}] + $\{v_n\}$

Only possibilities. Take whichever is better

at most, depth n



at least depth $n/2$

Q: How many leaves are there in this tree?

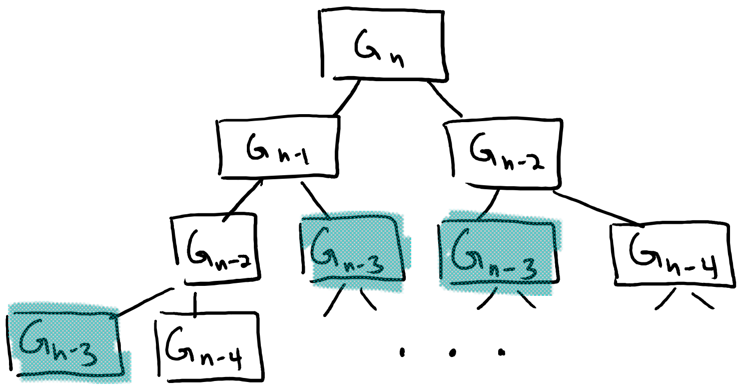
A) $O(1)$ B) $O(n)$ C) $O(n^2)$ D) $O(2^n)$

$$2^{n/2} \leq \#\text{ Leaves} \leq 2^n$$

This is bad. Since need to do at least 1 operation to solve base case, if solve recursively, use time $O(2^n)$.

S.KIMMEL

But, let's look more carefully:



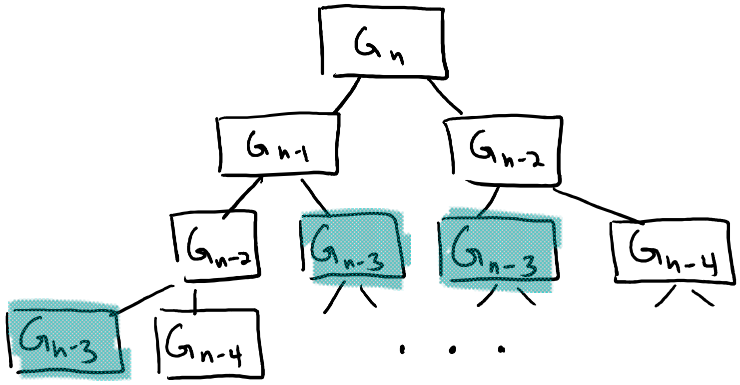
* Actually solving same problems over and over!

Q. How many distinct subproblems are there?

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

S. KIMMEL

But, let's look more carefully:



* 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) $O(2^n)$

$\{G_1, G_2, \ldots G_n\}$

Idea: Instead of solving recursively, create an array containing solutions. Look up subproblems in array.

S. KIMMEL

Next: use recurrence relation for <u>sets</u> to create recurrence relation for optimal <u>weight</u>.

- Let $A[i]$ be max weight of MWIS on G_i.
  - Create recurrence relation for A

$$A[i] = \begin{cases} \boxed{} & \text{if } v_i \notin \text{MWIS on } G_i \\ \boxed{} & \text{if } v_i \in \text{MWIS on } G_i \end{cases}$$

+ Base Case

  - Create code that fills in A using a For Loop:

Base Case for A

1. $\boxed{}$

2. for $i = \boxed{}$ to $\boxed{}$:

   $A[i] = \boxed{}$

Recurrence relation for A

(very similar to our recurrence relation for sets!)

S. KIMMEL

$$A[i] = \begin{cases} A[i-1] & \text{if } v_i \notin \text{MWIS on } G_i \\ A[i-2] + w(v_i) & \text{if } v_i \in \text{MWIS on } G_i \end{cases}$$

+ Base Case: $A[0] = 0$ $A[1] = w(v_1)$

Q: Write pseudocode to fill in array A:

```
A[0] = 0
A[1] = w(v_1)
for i = 2 to n:
    A[i] = max{A[i-1], A[i-2] + w(v_i)}
```

Run Time: $O(n)$

Correctness: Loop Invariants

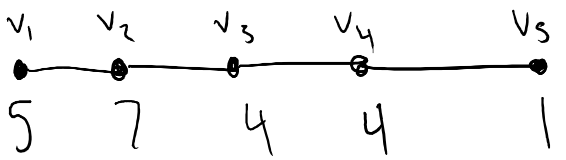
MWIS on G_n is:

i) [MWIS on G_{n-1}]

OR

ii) [MWIS on G_{n-2}] + $\{v_n\}$

Only possibilities. Take whichever is better. Take Max ex:



using code from previous page

A

0	1	2	3	4	5
0	5	7	9	11	11

$S = \emptyset$

Figuring out S

Case (i): $A[5] = A[4]$ * $v_5 \notin S$
Case (ii): $A[5] = A[3] + 1$

Case (i) $A[4] = A[3]$
case (ii) $A[4] = A[2] + 4$ * $v_4 \in S$

Pseudocode to extract S, given A:

```
S = ∅
i = n
while i ≥ 0:
    if A[i] = A[i-2] + w(v_i):
        S = S + i
        i = i - 2
    else:   i = i - 1
```

S.KIMMEL

Final Algorithm

// Construct A

$A[0] = \boxed{0}$

$A[1] = \boxed{1}$

for $i = \boxed{2 \text{ to } n:}$

$\quad A[i] = \boxed{\max\{A[i-1],\ A[i-2] + w(v_i)\}}$

O(n) runtime

// Construct S

$S = \emptyset$

$i = n$

while $i \geq 0$:

$\quad$ if $A[i] = A[i-2] + w(v_i)$:

$\quad\quad S = S + i$

$\quad\quad i = i - 2$

$\quad$ else: $\quad i = i - 1$

O(n) runtime

$\boxed{\begin{array}{l} O(n) \text{ runtime} \\ \text{vs } O(2^n) \text{ runtime with recursion!} \end{array}}$