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.

What is weight of MWIS of

2 7 4 2 4

A) 9    B) 10    C) 11    D) 13
Q: What is weight of MWIS of

\[ v_1, v_2, v_3, v_4, v_5 \]

2 7 4 2 4

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 \( G_n \):

- \( v_n \in S \)
- \( v_n \notin S \)

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

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

MWIS on $G_n$ is:

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

OR

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

Only possibilities. Take whichever is better.

\[ \text{G}_n \]

\[ \text{G}_{n-1} \]

\[ \text{G}_{n-2} \]

\[ \vdots \]

Q: How many leaves are there in this tree?

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

MWIS on $G_n$ is:

\[ \begin{cases} 
\text{i) MWIS on } G_{n-1} \\
\text{OR} \\
\text{ii) MWIS on } G_{n-2} + \{v_n\} 
\end{cases} \]

Only possibilities. Take whichever is better.

\[
\begin{array}{c}
\text{at most, depth } n \\
\downarrow \\
G_n \\
\downarrow \\
G_{n-1} \\
\downarrow \\
G_{n-2} \\
\downarrow \\
\vdots \\
G_1 \\
\downarrow \\
G_0 \\
\downarrow \\
G_{n-4} \\
\downarrow \\
\vdots \\
G_2 \\
\downarrow \\
G_{n-4} \\
\downarrow \\
\vdots \\
G_4 \\
\downarrow \\
G_{n-4} \\
\downarrow \\
\vdots \\
G_6 \\
\downarrow \\
\vdots \\
G_{n-4} \\
\downarrow \\
\vdots \\
G_8 \\
\downarrow \\
\vdots \\
G_{n-4} \\
\downarrow \\
\vdots \\
G_{n-2} \\
\downarrow \\
\vdots \\
G_{n-1} \\
\downarrow \\
\vdots \\
G_n \\
\end{array}
\]

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 4 operations to solve base case, if solve recursively, use time $O(2^n)$. 

Max Weight Independent Set Page 4
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) \( O(2^n) \)
But let's look more carefully:

Actually solving same problems over and over!

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.
Next: Use recurrence relation for \textit{sets} to create recurrence relation for optimal weight.

- Let $A[i]$ be max weight of MWIS on $G_i$.

Create recurrence relation for $A$

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

Base Case

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

1. 

2. for $i = \square$ to $\square$:

   $A[i] = \square$

   

   Recurrence relation for $A$

   (very similar to our recurrence relation for sets!)
\[ A[i] = \begin{cases} 
A[i-1] & \text{if } v_i \notin \text{MWIS on } G_i \\
A[i-2] + w(v_i) & \text{if } v_i \in \text{MWIS on } G_i
\end{cases} \]

**Base Case:** \( A[0] = 0 \) \( A[1] = w(v_1) \)

Q: Write pseudocode to fill in array \( A \):

\[
\begin{align*}
A[0] &= 0 \\
A[1] &= w(v_1) \\
\text{for } i &= 2 \text{ to } n: \\
A[i] &= \max \{A[i-1], A[i-2] + w(v_i)\}
\end{align*}
\]

**Run Time:** \( O(n) \)

**Correctness:** Loop Invariants

MWIS on \( G_n \) is:

\[
\begin{align*}
i) \text{[MWIS on } G_{n-1}] \\
o) \text{or} \\
ii) \text{[MWIS on } G_{n-2}] + \{v_n\}
\end{align*}
\]

Only possibilities. Take whichever is better. Take \( \max \)
EX:

v_1 v_2 v_3 v_4 v_5
5 7 4 4 1

Using code from previous page:

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

S = \emptyset

Figuring out S

\begin{cases}
\end{cases}

Pseudocode to extract S, given A:

\begin{align*}
S &= \emptyset \\
i &= n \\
\text{while } i \geq 0: \\
&\quad \text{if } A[i] = A[i-2] + w(v_i): \\
&\quad \quad S = S + i \\
&\quad \quad i = i - 2 \\
&\quad \text{else: } i = i - 1
\end{align*}
Final Algorithm
// Construct A
\[ A[0] = 0 \]
\[ A[1] = 1 \]
for \( i = 2 \) to \( n \):
\[ A[i] = \max \{ A[i-1], A[i-2] + w(v_i) \} \]

// Construct S
\[ S = \emptyset \]
\[ i = n \]
while \( i > 0 \):
    if \( A[i] = A[i-2] + w(v_i) \):
        \[ S = S + i \]
        \[ i = i - 2 \]
    else:
        \[ i = i - 1 \]

\( O(n) \) runtime

vs \( O(2^n) \) runtime with recursion!