Dynamic Programming

Text Segmentation

Decisions: what is the first word?
Subproblems: suffixes
Eval order: [ ]

LIS:
Decisions: [ ]
Subproblems: sentinel + suffix:

Woodcutter's problem:
Subproblems = intervals \([i...j]\)

Maximum Independent Set

NP-hard
Fast algo for some special cases
TREES!
Tree = Connected acyclic graph.

Choose a root node

Direct edges away from root

Rooted tree = node with a set of rooted (sub) trees.

Subproblem = vertex

Define \( MIS(v, p) \), the size of the largest independent set in the subtree rooted at \( v \) if parent of \( v \) is included (\( p=\text{True} \)) or excluded (\( p=\text{False} \)).

Define \( MIS_{\text{yes}}(v) = \) size of largest ind. set in subtree rooted at \( v \) that includes \( v \)

\( MIS_{\text{no}}(v) = \) size of largest ind. set in subtree rooted at \( v \) that excludes \( v \)

\[
MIS_{\text{yes}}(v) = 1 + \sum_{w\in v} MIS_{\text{no}}(w)
\]

\[
MIS_{\text{no}}(v) = \sum_{w\in v} \max \{ MIS_{\text{yes}}(w), MIS_{\text{no}}(w) \}
\]

We want \( \max \{ MIS_{\text{yes}}(\text{root}), MIS_{\text{no}}(\text{root}) \} \).
Evaluate children before parents:

1. Reverse BFS: level by level
2. DFS at root postorder

Memoize into the given tree data structure
v. MIS yes
v. MIS no

O(n) # nodes

Recurrence defines a dependency graph
nodes = subproblems edges = recursive calls
MUST be acyclic

**Memoize***(x):**
if value[x] is undefined
initialize value[x]

for all subproblems y of x
**Memoize**(y)
update value[x] based on value[y]
finalize value[x]

**DFS***(v):**
if v is unmarked
mark v
**PRE**VISIT*(x)*
for all edges v→w
**DFS**(w)
**POST**VISIT*(x)*

**DYNAMIC**PROGRAMMING**(G):**
for all subproblems x in postorder
initialize value[x]
for all subproblems y of x
update value[x] based on value[y]
finalize value[x]

Longest Path in DAG
LLP(v) = length of longest path s→v