Admin:
HW1 revision #2
(a) \rightarrow (b) \leq correctness:
  greedy notes

DP rubric

3: English
  incl. top level call

4: recurrence

5: iterative

pseudocode
Dynamic Programming Trees = Postorder traversal

Depth-First search

Memoize(x):
if value[x] is undefined
    initialize value[x]
for all subproblems of x
    Memoize(y)
    update value[x] based on value[y]
finalize value[x]
    return value[x]

DepthFirstSearch(v):
if v is unmarked
    mark v
    PreVisit(v)
    for each edge v → w
        DFS(w)
    PostVisit(v)

Graph is defined implicitly by a recursive function:

- Vertices = subproblems
- Edges = dependencies

Given an explicit directed graph

Proper recurrence ↔ dag ↔ correctness
Memoization ↔ marking ↔ efficiency
Evaluation order ↔ (rev) topological order = postorder
\textbf{Longest Path} \((G, s, t)\) \(G\) is a dag
Length of the longest path from \(s\) to \(t\) in \(G\)

Seq of decisions: Where next?

Real recursive prob:
\(LP(v) = \text{length of longest path from } v \text{ to } t \) in \(G\)

Recurrence:

\[
LP(v) = \begin{cases} 
0 & \text{if } v = t \\
\max \{LP(w) + 1 \mid w \rightarrow v \in E\} & \text{otherwise}
\end{cases}
\]

\(\max \phi = \emptyset - \infty\)

Memoize:
add a field \(v, LP\) to every vertex record.
If \(G\) is adjacency list/adj matrix \(\Rightarrow\text{AICRAY!}\)
Evaluation order:

- Top order
- DFS + reindex

Equivalently: POSTORDER by DFS

Time = $O(V + E)$

1. Recurrence + dyn prog
2. DAG + longest path