SO FAR...

Divide & Conquer

Connectivity & Shortest Paths

Greedy Algs
Dynamic Programming

- Powerful & widely applicable "recipe" for algorithm design

Examples:
1) Maximum Increasing Subsequence
2) Knapsack
3) Edit Distance
4) All-Pairs Shortest Paths
5) Hamiltonian Cycle
6) Independent Sets in Trees

Many more . . . .
**Dynamic Programming Example No. 1:**

**Longest Path in a DAG**

**Input:** A DAG (directed acyclic graph) \( G = (V, E) \)

**Goal:** Find the length of longest path.

[Assume: 1, 2, ..., n is a topological sort of graph]
**Step 1: Define “Subproblems”**

Let “\( L[i] = \text{length of longest path ending at vertex } i \)”

\[ L(1), L(2), \ldots, L(n) \rightarrow n \text{ subproblems} \]

The longest path = maximum \( \{ L(1), L(2), \ldots, L(n) \} \)

in the DAG
**STEP 2:** Write a recurrence relation among subproblems

\[ L[i] = \text{“length of longest path ending at } i \text{”} \]

for \( i = 1 \ldots n \)

\[ L[7] = \text{maximum} \left\{ \begin{array}{l}
L[3] + 1 \\
L[6] + 1 \\
L[5] + 1
\end{array} \right. \]

length of longest path ending at 7
Step 2: Write a recurrence relation among subproblems

\[ L[i] = \text{"length of longest path ending at } i \text{"} \]
for \( i = 1 \ldots n \).

\[ L(i) = \max_{j \rightarrow i} \{ L(j) + 1 \} \]
**Step 3:** Use the recurrence relation to solve subproblems

\[
L[i] \quad \text{for } i = 1 \ldots n
\]

- **Initialise:** \( L[i] = 0 \quad \forall \ i = 1 \ldots n \)
- **for** \( i = 1 \) **to** \( n \)
  - \( L[i] = \max \left\{ L[j] + 1 \mid j \in \text{prev}[i], j < i \right\} \)
- \( \text{prev}[i] = j \) for which \( L[i] = \max \left( L[j] + 1, L[i] \right) \)

\[|E| + |V| \]

\[
\text{return} \quad \text{Maximum } \left\{ L[1], L[2], \ldots, L[n] \right\}
\]
**Longest Increasing Subsequence (LIS)**

**Input:** Array of numbers $A[1], \ldots, A[n]$

**Goal:** Find the LIS

$\text{longest path in the OAH} = \text{longest increasing subsequence}$

$A := \begin{bmatrix} 11 & 100 & 1 & 7 & 18 & 23 & 25 & 10 \end{bmatrix}$

$\#\text{edges} = O(n^2)$