SO FAR...

DIVIDE & CONQUER

CONNECTIVITY & SHORTEST PATHS

GREEDY ALGS

DIVIDE
&
CONQUER

CONNECTIVITY
&
SHORTEST PATHS

GREEDY
ALGS

COMING UP

DYNAMIC
PROGRAMMING

LINEAR
PROGRAMMING

DYNAMIC PROGRAMMING

- Powerful & widely applicable "recipe" for algorithm design

EXAMPLES:

1) MAXIMUM INCREASING SUBSEQUENCE
2) KNAP SACK
3) EDIT DISTANCE
4) ALL-PAIRS SHORTEST PATHS
5) HAMILTONIAN CYCLE
6) INDEPENDENT SETS IN TREES

Many more . . . . .

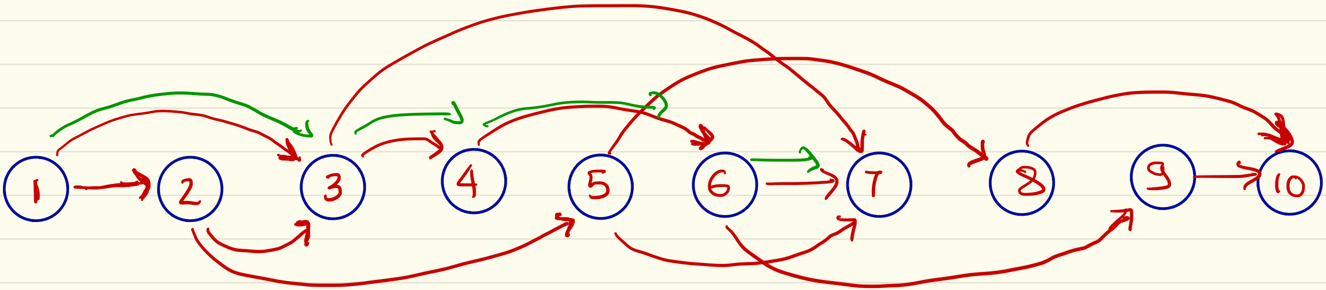
DYNAMIC PROGRAMMING EXAMPLE NO. 1:

LONGEST PATH IN A DAG

INPUT: A unweighted DAG (directed acyclic graph) $G = (\{1 \ldots n\}, E)$

GOAL: Find the length of longest path.

[Assume: 1, 2, .. n is a topological sort of graph]

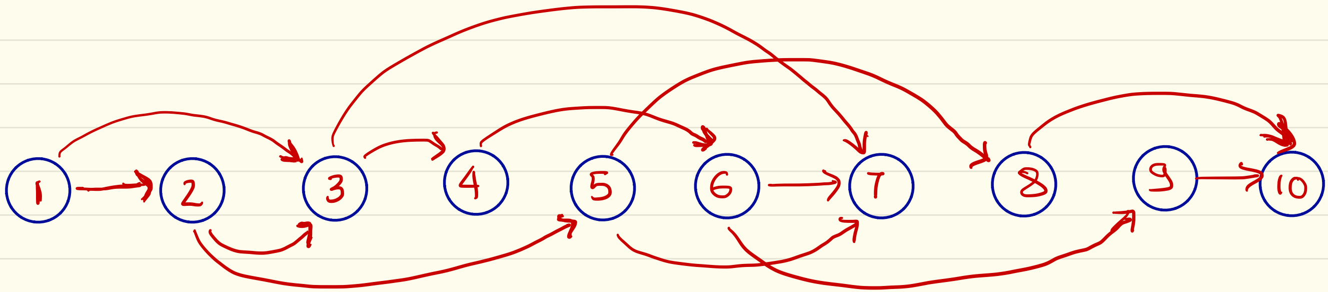


STEP 1: DEFINE "SUBPROBLEMS"

Let "$L[i]$ = length of longest path ending at vertex i"

$L[1], L[2] \ldots L[n] \rightarrow n$ subproblems

longest path in the DAG = maximum $\{L[1], L[2], \ldots L[n]\}$

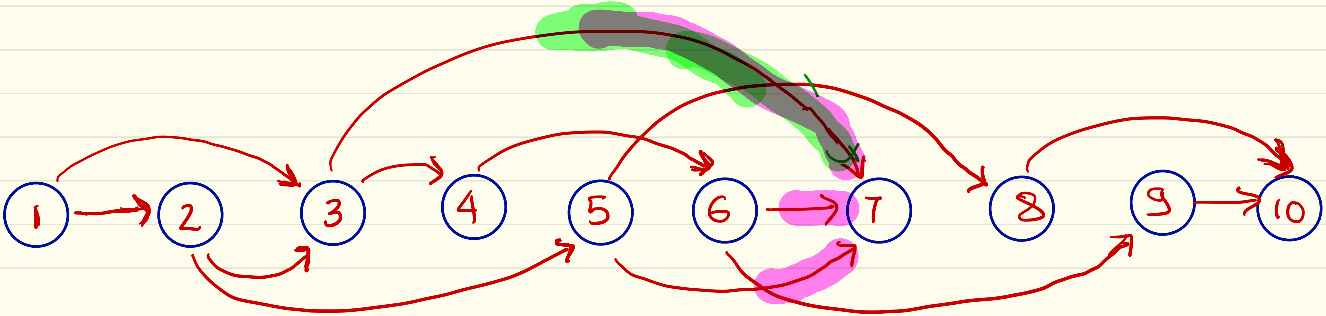


STEP 2: WRITE A RECURRENCE RELATION AMONG SUBPROBLEMS

$L[i]$ = "length of longest path ending at i" for $i = 1 \ldots n$

$$L[7] = \text{maximum} \begin{cases} \text{from } 3 & L[3]+1 \\ \text{from } 6 & L[6]+1 \\ \text{from } 5 & L[5]+1 \end{cases}$$

length of longest path ending at 7



STEP 2: WRITE A RECURRENCE RELATION AMONG SUBPROBLEMS

$L[i]$ = "length of longest path ending at i"

for $i = 1..n$

$$L[i] = \max_{j \to i} \left\{ L[j] + 1 \right\}$$

STEP 3: USE THE RECURRENCE RELATION TO SOLVE SUBPROBLEMS

$L[1] \; . \; . \; . \qquad L[n]$

$$L[i] = \max_{\substack{j \to i \\ j < i}} \{ L[j] + 1 \}$$

topologically sorted on $\{1 .. n\}$

Initialise: $L[i] = 0 \quad \forall \; i = 1 .. n$

for $i = 1$ to n

$L[i] = \underset{\substack{\text{over } j \to i \\ j < i}}{\text{maximum}} \{ L[j] + 1 \}$

$|E| + |V|$

for all $j \to i$ edges

$L[i] = \max(L[j]+1, L[i])$

$prev[i] = j$ for which max is attained

RETURN Maximum $\{ L[1], L[2] \; . . \; L[n] \}$.

Longest Increasing Subsequence (LIS)

INPUT: Array of numbers $A[1] \ldots A[n]$ [DAG with $\{1..n\}$ vertices]

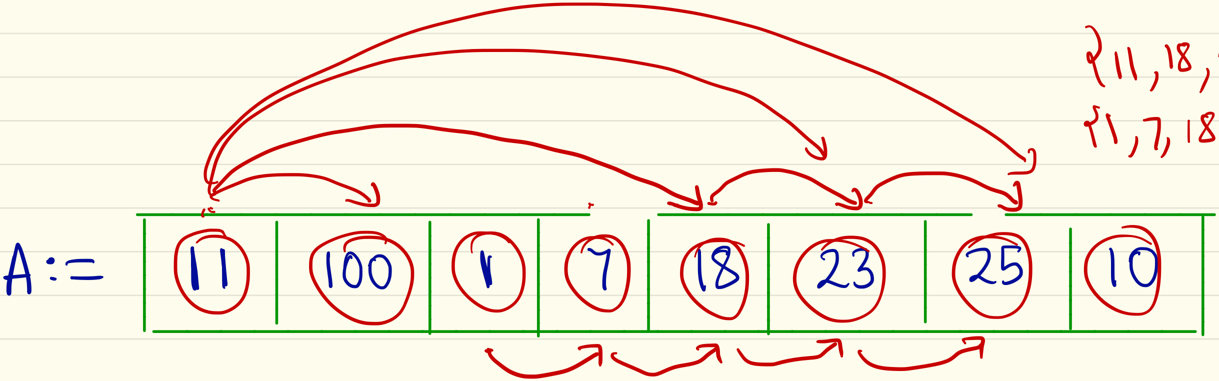
$i \to j$ if $A[i] < A[j]$, $i < j$

GOAL: Find the LIS

longest path in the DAG = longest increasing subsequence

$\{11, 18, 23, 25\}$

$\{1, 7, 18, 23, 25\}$



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