Dynamic Programming

Algorithms: Design and Analysis, Part II

WIS in Path Graphs: A Linear-Time Algorithm

The Story So Far

Upshot: If we knew whether or not v_n is in the max-weight IS, then could recursively compute the max-weight IS of G' or G'' and be done.

Proposed algorithm:

- Recursively compute $S_1 =$ max-weight IS of G'
- Recursively compute $S_2 =$ max-weight IS of G''
- Return S_1 or $S_2 \cup \{v_n\}$, whichever is better.

Good news: Correct. [Optional exercise - prove formally by induction]

Bad news: Exponential time.

The \$64,000 Question

Important question: How many distinct subproblems ever get solved by this algorithm?

A) $\Theta(1)$

B) $\boxed{\Theta(n)}$

C) $\Theta(n^2)$

D) $\Theta(2^n)$

Only 1 for each "prefix" of the graph!

[Recursion only plucks vertices off from the right]

Eliminating Redundancy

Obvious fix: The first time you solve a subproblem cache its solution in a global table for $O(1)$-time lookup later on. ["memoization"]

Even better: Reformulate as a bottom-up iterative algorithm. Let G_i = 1st i vertices of G.

Plan: Populate array A left to right with $A[i]$ = value of max-weight IS of G_i.

Initialization: $A[0] = 0, A[1] = w_1$

Main loop: For $i = 2, 3, \ldots, n$:

$$A[i] = \max\{ \boxed{A[i-1]} , \boxed{A[i-2] + w_i} \}$$

Case 1 - max-wt IS of G_{i-1}

Case 2 - max-wt IS of $G_{i-2} + \{v_n\}$

Run time: Obviously $O(n)$, Correctness: Same as recursive version.