Dynamic Programming

WIS in Path Graphs: A Linear-Time Algorithm

Algorithms: Design and Analysis, Part II

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-wt IS of G'
- " S_2 = max-wt 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) $\Theta(n)$ — only 1 for each "prefix" of the graph!

[recursion only plucks vertices off from the right]

(C) $\Theta(n^2)$

(D) $\Theta(2^n)$

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-wt IS of G_i.

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

Main loop: For $i = 2, 3, 4, \ldots, n$:

$$A[i] = \max\{\underbrace{A[i-1]}_{\text{case 1 - max wt IS of } G_{i-1}}, \underbrace{A[i-2] + w_i}_{\text{case 2 - max wt IS of } G_{i-2} + \{v_n\}}\}$$

Running Time

obviously $O(n)$

Correctness

same as recursive version