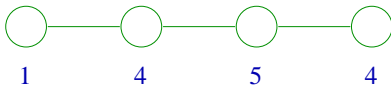# Dynamic Programming

Algorithms: Design and Analysis, Part II

Introduction, and WIS in Path Graphs

Problem Statement

Input: A path graph $G = (V, E)$ with nonnegative weights on vertices.

Desired output: Subset of nonadjacent vertices – an <u>independent set</u> – of maximum total weight.

Next: Iterate through our algorithm design principles.

Brute-force search: Requires exponential time.

Tim Roughgarden

A Greedy Approach

Greedy: Iteratively choose the max-weight vertex not adjacent to any previously chosen vertex.
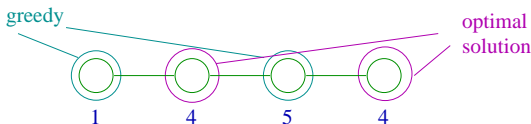
Question: In the example, what is the value of the max-weight independent set, and that of the output of our greedy algorithm.

A) 14 and 10
B) 8 and 6
C) 8 and 8
D) 9 and 8

A Divide & Conquer Approach

Idea: Recursively compute the max-weight IS of 1st half, ditto for 2nd half, then combine the solutions.
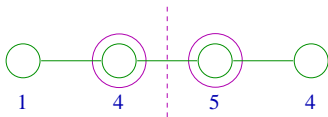
Problem: What if recursive sub-solutions conflict?
$\Rightarrow$ Not clear how to quickly fix.

Dynamic Programming

Algorithms: Design and Analysis, Part II

WIS in Path Graphs: Optimal Substructure

Optimal Substructure

Critical step: Reason about structure of an optimal solution.
[In terms of optimal solutions of smaller subproblems]

Motivation: This thought experiment narrows down the set of candidates for the optimal solution; can search through the small set using brute-force-search.

Notation: Let $S \subseteq V$ be a max-weight independent set (IS). Let v_n = last vertex of path.
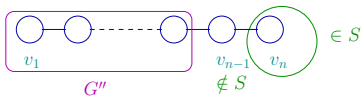
A Case Analysis

Case 1: Suppose $v_n \notin S$. Let $G' = G$ with v_n deleted.

Note: S also an IS of G'.

Note: S must be a max-weight IS of G' – if S^* was better, it would also be better than S in G. [contradiction]

Case 2: Suppose $v_n \in S$.

Note: Previous vertex $v_{n-1} \notin S$ [by definition of an IS]. Let $G'' = G$ with v_{n-1}, v_n deleted.

Note: $S - \{v_n\}$ is an IS of G''.

Note: Must in fact be a max-weight IS of G'' – if S^* is better than S in G'', then $S^* \cup \{v_n\}$ is better than S in G. [contradiction]

Toward an Algorithm

Upshot: A max-weight IS <u>must be</u> either
(i) a max-weight IS of G' or
(ii) v_n + a max-weight IS of G''

Corollary: <u>If we knew</u> whether or not v_n was in the max-weight IS, could recursively compute the max-weight IS of G'' and be done.

(Crazy?) idea: Try both possibilities + return the better solution.

Dynamic Programming

Algorithms: Design and Analysis, Part II

WIS in Path Graphs:
A Linear-Time Algorithm

The Story So Far

Upshot: <u>If we knew</u> 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 <u>distinct</u> subproblems ever get solved by this algorithm?

A) $\Theta(1)$

B) $\Theta(n)$

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 $i = 2, 3, \ldots, n$:
$$A[i] = \max\{ \underbrace{A[i-1]}, \underbrace{A[i-2] + w_i} \}$$

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.

Dynamic Programming

Algorithms: Design and Analysis, Part II

WIS in Path Graphs:

A Reconstruction Algorithm

Optimal Value vs. Optimal Solution

Recall: $A[0] = 0, A[1] = w_1$, for $i = 2, 3, \ldots, n$, $A[i] := \max\{A[i-1], A[i-2] + w_i\}$.

Note: Algorithm computes the <u>value</u> of a max-weight IS, not such an IS itself.

Correct but not ideal: Store optimal IS of each G_i in the array in addition to its value.

Better: Trace back through filled-in array to reconstruct optimal solution.

Key point: We know that a vertex v_i belongs to a max-weight IS of G_i $\iff$ w_i + max-weight IS of $G_{i-2} \geq$ max-weight IS of G_{i-1}. (Follows from correctness of our algorithm!)

Tim Roughgarden

A Reconstruction Algorithm

Let A = filled-in array:

0	4	4	7	...	184
0	1	2	3	...	n

- Let $S = \emptyset$
- While $i \geq 1$ [scan through array from right to left]
    - If $A[i-1] \geq A[i-2] + w_i$ [i.e. case 1 wins]
        - Decrease i by 1
    - Else [i.e., case 2 wins]
        - Add v_i to S, decrease i by 2
- Return S

Claim: [By induction + our case analysis] Final output S is a max-weight IS of G.

Running time: $O(n)$

Tim Roughgarden

Dynamic Programming

Algorithms: Design and Analysis, Part II

Principles of Dynamic Programming

Principles of Dynamic Programming

Fact: Our WIS algorithm is a dynamic programming algorithm!

Key ingredients of dynamic programming:
(1) Identify a <u>small</u> number of subproblems
[e.g., compute the max-weight IS of G_i for $i = 0, 1, \ldots, n$]

(2) Can quickly+correctly solve "larger" subproblems given the solutions to "smaller subproblems"
[usually via a recurrence such as
$A[i] = \max\{A[i-1], A[i-2] + w_i\}$]

(3) After solving all subproblems, can quickly compute the final solution
[usually, it's just the answer to the "biggest" subproblem]

Tim Roughgarden

Why "Dynamic Programming"?

"The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research. Im not using the term lightly; Im using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making,. . . But planning, is not a good word for various reasons. I decided therefore to use the word, "programming". . . [Dynamic] has a very interesting property as an adjective, and that is its impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. Its impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activites."

Richard Bellman, "Eye of the Hurricane: an autobiography" 1984.