Exact Algorithms for NP-Complete Problems

A Dynamic Programming Algorithm for TSP

Algorithms: Design and Analysis, Part II

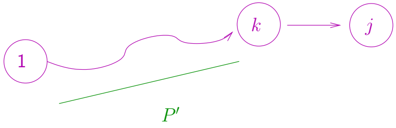
The Subproblems

Moral of last video: To enforce constraint that each vertex visited exactly once, need to remember the identities of vertices visited in a subproblem. [But not the order in which they're visited]

Subproblems: For every destination $j \in \{1, 2, \ldots, n\}$, every subset $S \subseteq \{1, 2, \ldots, n\}$ that contains 1 and j, let
$L_{S,j}$ = minimum length of a path from 1 to j that visits precisely the vertices of S [exactly once each]

Optimal Substructure

Optimal Substructure Lemma: Let P be a shortest path from 1 to j that visits the vertices S (assume $|S| \geq 2$) [exactly once each]. If last hop of P is (k, j), then P' is a shortest path from 1 to k that visits every vertex of $S - \{j\}$ exactly once. [Proof = straightforward "cut+paste"]



Corresponding recurrence:

$L_{S,j} = \min_{k \in S, k \neq j} \{L_{S-\{j\},k} + c_{kj}\}$

["size" of subproblem $= |S|$]

A Dynamic Programming Algorithm

Let A = 2-D array, indexed by subsets $S \subseteq \{1, 2, \ldots, n\}$ that contain 1 and destinations $j \in \{1, 2, \ldots, n\}$

Base case:

$$A[S, 1] = \begin{cases} 0 \text{ if } S = \{1\} \\ +\infty \text{ otherwise} \end{cases}$$ [no way to avoid visiting vertex (twice)]

For $m = 2, 3, \ldots, n$ [m = subproblem size]

  For each set $S \subseteq \{1, 2, \ldots, n\}$ of size m that contains 1

    For each $j \in S, j \neq 1$

      $A[S, j] = \min_{k \in S, k \neq j} \{A[S - \{j\}, k] + c_{kj}\}$ [same as recurrence]

Return $\min_{j=2,\ldots,n}\{ \underbrace{A[\{1, 2, \ldots, n\}, j]}_{\text{min cost from 1 to } j \text{ visiting everybody once}} + \underbrace{c_{j1}}_{\text{cost of final hop of tour}} \}$

Running time: $O(\underbrace{n}_{\text{choices of } j} \cdot \underbrace{2^n}_{\text{choices of } S}) \underbrace{O(n)}_{\text{work per subproblem}} = O(n^2 2^n)$

choices of j $\cdot$ choices of S = # of subproblems