Algorithms: Design and Analysis, Part II

Exact Algorithms for NP-Complete Problems

A Dynamic Programming Algorithm for TSP

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]

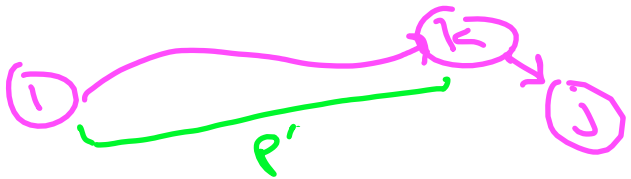
Subproblems: For every destination $j \in \{1,2,\ldots,n\}$, every subset $S \subseteq \{1,2,\ldots,n\}$ that contains 1 and j, let

$L_{S,j}$ = 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 [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.

S → assume $|S| \geq 2$



[proof = straightforward "cut+paste"]

Corresponding Recurrence:

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

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

A Dynamic Programming Algorithm

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

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

For $m = 2,3,4,\dots,n$: [m = subproblem size]
  For each set $S \subseteq \{1,2,\dots,n\}$ of size m that contains 1:
    For each $j \in S, j \neq 1$:

$$A[S,j] = \min_{\substack{k \in S \\ k \neq j}} \{A[S-\{j\},k] + C_{kj}\}$$

[same as recurrence]

Return $\min_{j=2} \{A[\{1,2,3,\dots,n\},j] + C_{j1}\}$

[min cost from 1 to j, visiting everybody once] [cost of final hop of tour]

Running Time

$O(n2^n) \cdot O(n)$

[choices of j] [choices of S] [work per subproblem] [# of subproblems] [subproblem]

$= O(n^2 2^n)$