Section 21.1: A Dynamic Programming Algorithm for the TSP

The Baseline: Exhaustive Search

TSP: given complete graph $G=(V,E)$ with real-valued edge costs c_e compute a tour T (cycle visiting every vertex exactly once) with minimum-possible total cost $\sum_{e \in T} c_e$.

Exhaustive search: $O(n!)$ time.

Stirling's Approximation

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Dynamic Programming

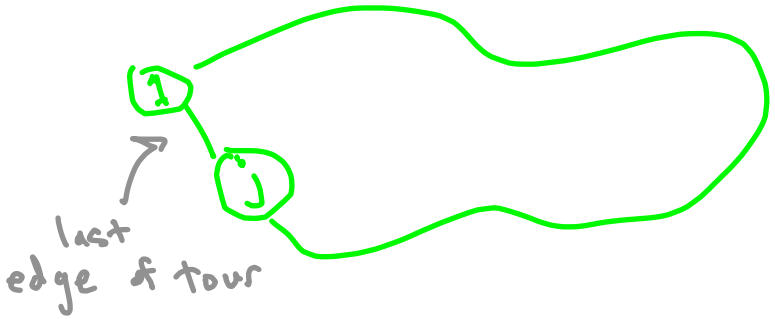
(1) Identify relatively small collection of subproblems. $f(n)$

(2) Quickly + correctly solve "larger" subproblems given the solutions to "smaller" ones. $g(n)$

(3) Quickly + correctly infer the final solution from the subproblem solutions. $h(n)$

Overall running time: $O(f(n) \cdot g(n) + h(n))$

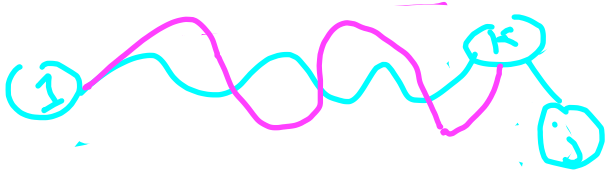
Optimal Substructure



$$\text{optimal tour cost} = \min_{j=2}^{n} \left(C_{j1} + \begin{array}{l}\text{min-cost of cycle-free} \\ \text{1-j paths that visits} \\ \text{every vertex}\end{array} \right)$$

exhaustive search over j

how to compute?

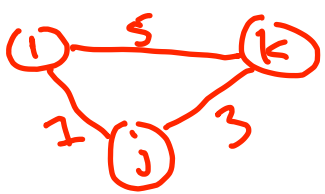


Quiz

Let P be a minimum-cost cycle-free path from 1 to j that visits every vertex, with final hop (k, j). $P' := P - (k, j)$.

Which of the following are true? (Choose all that apply.)

- (a) P' is a cycle-free path from 1 to k that visits every vertex of $V - \{j\}$.
- ~~(b)~~ P' is a min-cost path of the type in (a).
- (c) P' is a cycle-free path from 1 to k that visits every vertex of $V - \{j\}$ and does not visit vertex j.
- (d) P' is a min-cost path of the type in (d).



Recurrence and Subproblems

Notation: $C_{S,j}$ = Minimum cost of a cycle-free path from 1 to j that visits exactly the vertices of S.

Recurrence: $C_{V,j} = \min_{\substack{k \in V \\ k \neq 1,j}} \left(C_{V-\{j\},k} + c_{kj} \right)$ [k = penultimate vertex]

In general: $C_{S,j} = \min_{\substack{k \in S \\ k \neq 1,j}} \left(C_{S-\{j\},k} + c_{kj} \right)$

Subproblems: Compute $C_{S,j}$ for each subset $S \subseteq \{1,2,\ldots,n\}$ with $|S| \geq 2$ and $1 \in S$, and each choice of $j \in S - \{1\}$.

Final solution: $\min_{j \geq 2} \left(C_{V,j} + c_{j1} \right)$

The Bellman-Held-Karp Algorithm

$A :=$ $(2^{n-1} - 1) \times (n-1)$ 2-D array [subproblem solutions]

For $j = 2$ to n:
  $A[\{1,j\}][j] := c_{1j}$ [base cases]

For $s = 3$ to n: [s = subproblem size]
  For each S with $|S| = s$ and $1 \in S$:
    For each $j \in S - \{1\}$:
      $A[S][j] := \min_{\substack{k \in S \\ k \neq 1,j}} (A[S - \{j\}][k] + c_{kj})$ [as per recurrence]

(already computed, $O(1)$ to look up)

Return $\min_{j=2}^{n} (A[V][j] + c_{j1})$ [final solution]

Properties of Bellman-Held-Karp

Correctness: follows by induction on subproblem size (+fact that recurrence is correct, due to optimal substructure).

Running time: $f(n)$ [# of subproblems] $= O(n2^n)$.

$g(n)$ [time per subproblem] $= O(n)$

$h(n)$ [post processing] $= O(n)$

→ can also reconstruct an optimal tour in $O(n)$ time

$\Rightarrow$ Overall: $O(f(n) \cdot g(n) + h(n))$

$= \underline{O(n^2 2^n)}$