Best cycle that visits all vertices?

Brute Force algorithm?

Brute Force runtime?
The Problem?

The sub problems?

The recurrence?

The initialization?

The processing order?

Given:

\[ G = (V, E) \]  # complete graph

\[ l(i, j) \]  # edge weights

Find minimum cost of complete tour.

Does Bellman-Ford or Floyd-Warshall give any inspiration?
The problem: 
\[ \min_{c \in V-\{1\}} \sum_{i} A[v, i] + l(i, 1) \]

The subproblems: 
\[ A[S, j] = \text{minimum path cost from 1 to } j, \text{ visiting each vertex in } S \text{ exactly once.} \]

The recurrence: 
\[ A[S, j] = \min_{c \in S-\{j\}} \sum_{i} A[S-\{j\}, i] + l(i, j) \]

The initialization: 
\[ A[\{1\}, 1] = 0 \]
\[ A[S, 1] = \infty \quad \# \text{all other } S. \]
The processing order:

from smaller to larger set sizes.

\[ A[S, j] \] depends on \[ A[S-\{j\}, i] \]
def TSP(G, l):
    A[ε13, 1] = 0

    for s = 2..n:
        for all S' ⊆ V - ε13, size s-1:
            S = S' ∪ ε13
            A[S, 1] = ∞  # can't end at 1

            for j ∈ S - ε13:
                A[S, j] =
                \min \left\{ A[S - εj3, i] + l(i, j) \right\}
                i ∈ S - εj3

    return \min \left\{ A[V, i] + l(i, 1) \right\}
    i ∈ V - ε13
TSP Algorithm

Runtime?

Data requirements?

Real data requirements?
TSP Algorithm

Runtime: \(2^n\) sets.

- \(n\) set members
- \(n\) predecessors per member

\(O(n^2 2^n)\)

Memory:
- \(2^n\) sets
- \(2^n\) members
- \(A[S, i]\)

\(O(n^2)\)

Real Memory: only need size \(S\) and \(S-1\) at any given time.

\(O(n^2)\)