Design and Analysis of Algorithms

L31: Traveling Salesman Problem

Dynamic Programming

Dr. Ram P Rustagi
Sem IV (2020-Even)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

Resources

- Text book 2: Horowitz
  – Sec 5.9
- R1: Introduction to Algorithms
  - Cormen et al.
- https://www.youtube.com/watch?v=-JjA4BLQyqE
- https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_travelling_salesman_problem.htm

Travelling Salesman Problem

- Known as Held-Karp algorithm
  - Proposed in 1962 to solve TSP
- TSP problem:
  - Find a tour of all cities in a country (assuming all cities are reachable)
    - The tour should visit each city only once
    - Tour should end at starting city, and
    - Tour should be of minimum distance. (cost)

Example 1: TSP problems

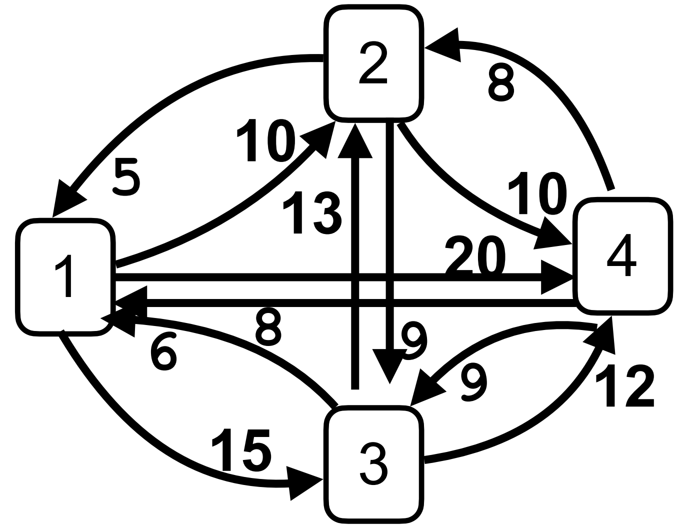
- You are organizing a function at your home and you would like to invite your friends for the same.
  - Starting from your home, you need to visit each friend's house to personally invite.
  - The route/distance from one house to another house is known.
  - The up and down time taken to travel between two houses is not same i.e. depends upon travel direction
    - e.g. some one way roads, pot-holed roads etc
- Goal: Find the shortest (time) route.

Example 2: TSP problems

- A robotic arm needs to tighten the screw/bolts on a machine.
  - There are different points where screw/bolts needs to be tightened
  - Robotic arm can reach from any screw/bolt position to another screw/bolt position.
  - The time taken to tighten to a bolt is constant so can be ignored. Time taken by robotic arm varies.
    - Interested in time taken by robotic arm when moving
- Goal: Find the optimal path for robot arm to tighten all the bolts and return to its start point.

TSP Problem

- Given directed graph $\texttt{G=(V,E)}$ with $\texttt{n>1}$ edges,
  - Cost of each directed edge $\texttt{(i,j)}$ is given as $\texttt{c}_{\texttt{ij}}\geq 0$
  - Cost is considered as ∞ when edge is not defined
  - A tour of G is a directed simple cycle that includes every vertex in the graph
  - The cost of a tour is the sum of cost of edges on the tour.
  - **T**raveling **S**alesman **P**roblem is to find the tour of minimum cost.
- For simplicity, we assume tour starts at $\texttt{s=1}$



TSP Problem

- Brute force approach
  - Enumerate all permutations of n nodes
  - Compute the cost corresponding to each permutation
  - Find the permuation with minimum cost.
  - Time complexity: $O(n!)$
- TSP is an NP-Hard problem
  - Can we do better though still exponential, e.g. $O(2^n)$

    $O(n^n) > O(n!) > O(k^n, _{n>k}) > O(2^n)$
    - Subset problems are easier compared to permutations
  - k^n is always better than $n!$ (for $n>k$).
  - Subset problem leads to dynamic programming approach

TSP Problem: Dynamic Programming

- Let start vertex `s=1`, and thus tour ends at `1`.
- Every tour consists of
  - An edge e_{1k}, for some $k \in V-\{1\}$, and
  - A path from `k` to `1` going thru each vertex exactly once other than `k` and `1`
    - i.e. $v \in V-\{1,k\}$.
  - Optimal tour is minimu of all such tours
- Using Optimality principle:
  - The tour is optimal, when
    - Path from `k` to `1` must be a shortest path going thru all vertices in $V-\{1,k\}$.

TSP Sub-Problem: Dynamic Programming

- What is appropriate subproblem for TSP?
  - Subproblem refers to partial solution
- Most obvious partial solution
  - Initial portion of a tour
- Starting at vertex `1`
  - Consider we visited few cities, and currently at city `i`.
  - What we need to do to extend this tour?
    - Need to know `i`
      - So as to know which cities to visit next
    - Need to know cities (i.e. subset S) visited so far
      - So as not to revisit any of them again

TSP Problem: Dynamic Programming

- To solve using DP, we need to identify recurrence relation
- Let $g(i,S)$ denotes the length of shortest path
  - Starting from vertex i,
  - Going thru all vertices in $S-\{i\}$, and
  - Terminating at vertex 1.
    - Note: we return to vertex 1, even though start from i
- Goal: compute $g(1,V-\{1\})$
  - Denotes the length of optimal TSP tour
- Recurrence relation using Principle of Optimality:

$$g(1,V-\{1\})=\min_{2\le k\le n}\{c_{1k}+g(k,V-\{1,k\})\} \quad \ldots\ldots (1)$$

- Generalizing above for $i \notin S$

$$g(i,S)=\min_{j\in S}\{c_{ij}+g(j,S-\{j\})\} \quad \ldots\ldots (2)$$

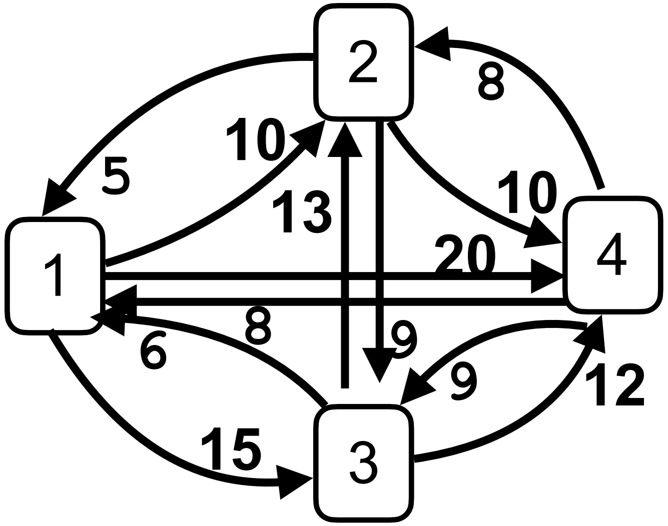
- Thus, solving $g(1,V-\{1\})$ requires to solving

  $g(k,V-\{1,k\})$ for all $k\neq 1$

TSP Problem: Dynamic Programming

- Computing $g(i,S)$ where $|S|=0$
- $g(i,\emptyset)$ implies shortest path from node i to 1
  - Going thru an empty set ($\emptyset$)of vertices i.e.
  - Without going thru any vertex i.e. direct edge $i \rightarrow 1$
    - Thus, $g(i,\emptyset)=c_{i1}$, $1 \leq i \leq n$.
- Next, compute $g(i,S)$ for all S of size 1 i.e. $\forall S$, $|S|=1$
- Thus, then we compute $g(i,S)$ for all S of size 2
  - i.e. $\forall S$, $|S|=2$, and so on
- When, $|S|<n-1$, then the values of i and S for which $g(i,S)$ is needed are such that $i \neq 1$, $1 \notin S$, and $i \notin S$.
- Tour construction requires that we maintain node j that
  - Minimizes $g(i,S)$ i.e. $\min_{j \in S}\{c_{ij}+g(j,S-\{j\})\}$
  - Let $J(i,S)$ denote this node

TSP Example: Computation



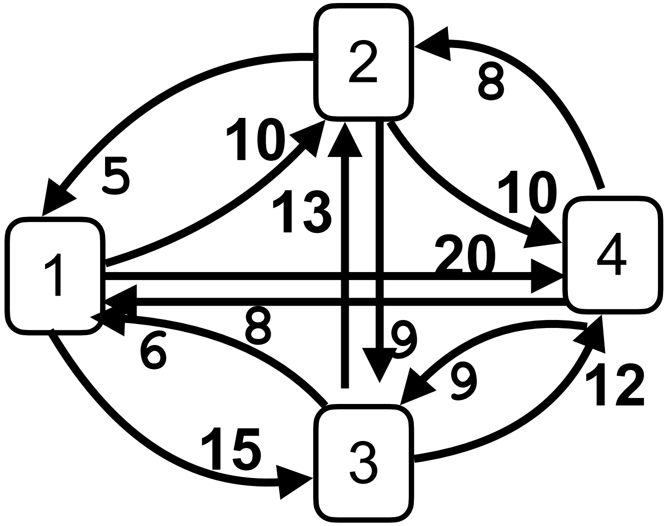
	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

- Goal : `g(1,V-{1})`
  `=g(1,{2,3,4})`

$$g(1,\{2,3,4\}) = \min\{c_{12}+g(2,\{3,4\}),\ c_{13}+g(3,\{2,4\}),\ c_{14}+g(4,\{2,3\})\}$$

- Process: compute bottom up
  - i.e. `g(i,|S|)`, for `|S|=0,1,`...

TSP Example: Computation



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

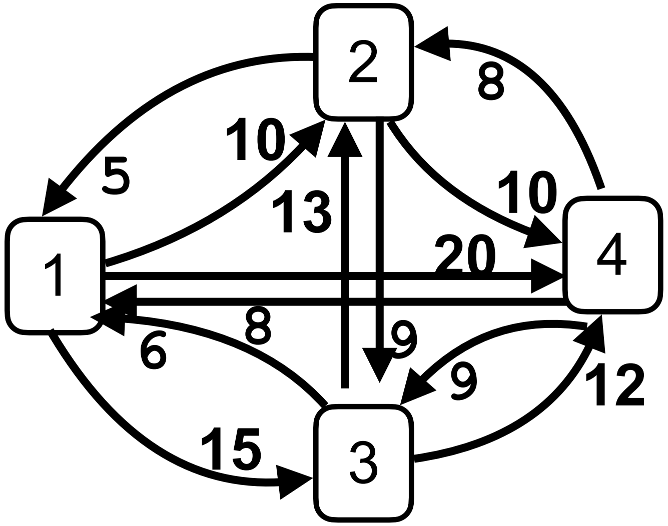
- Goal: $g(1, V-\{1\})$
  $= g(1, \{2,3,4\})$
- Power set of $\{2,3,4\}$
  $\emptyset, \{2\}, \{3\}, \{4\},$
  $\{2,3\}, \{2,4\}, \{3,4\}$
  $\{2,3,4\}$

  $g(1,\emptyset) = c_{11} = 0$
  $g(2,\emptyset) = c_{21} = 5$
  $g(3,\emptyset) = c_{31} = 6$

Compute $g(i,S), \forall |S|=1, i \in \{2,3,4\}$

$g(2,\{3\}) = c_{23} + g(3,\emptyset) = 9+6 = 15$
$g(2,\{4\}) = c_{24} + g(4,\emptyset) = 10+8 = 18$
$J(2,\{3\}) = 3, J(2,\{4\}) = 4$
$g(3,\{2\}) = c_{32} + g(2,\emptyset) = 13+5 = 18$
$g(3,\{4\}) = c_{34} + g(4,\emptyset) = 12+8 = 20$
$J\{3,\{2\} = 2, \quad J(3,\{4\}) = 4$
$g(4,\{2\}) = c_{42} + g(2,\emptyset) = 8+5 = 13$
$g(4,\{3\}) = c_{43} + g(3,\emptyset) = 9+6 = 15$
$J(4,\{2\}) = 2, J(4,\{3\} = 3$

TSP Example: Computation



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

$g(2,\{3\})=15,\ g(2,\{4\})=18, g(3,\{2\})=18,$
$g(3,\{4\})=20,\ g(4,\{2\})=13,\ g(4,\{3\})=15$

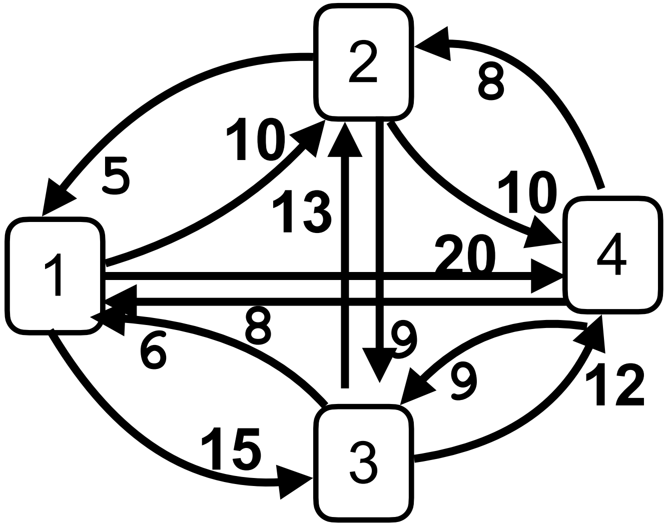
Compute $g(i,S)$, for $|S|=2$

$g(2,\{3,4\})=\min\{c_{23}+g(3,\{4\}), c_{24}+g(4,\{3\})$
$=\min\{9+20,\ 10+15\} = 25;$ $J(2,\{3,4\})=4$

$g(3,\{2,4\})=\min\{c_{32}+g(2,\{4\}), c_{34}+g(4,\{2\})$
$=\min\{13+18,\ 12+13\} = 25;$ $J(3,\{2,4\})=4$

$g(4,\{2,3\})=\min\{c_{42}+g(2,\{3\}), c_{43}+g(3,\{2\})$
$=\min\{8+15,\ 9+18\} = 23;$ $J(2,\{3,4\})=2$

TSP Example: Computation



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

g(2,{3,4})=25, g(3,{2,4})=25, g(4,{2,3})=23,

Compute g(i,S), for |S|=3

g(1,{2,3,4})=

min{c_{12}+g(2,{3,4}),c_{13}+g(3,{2,4}),c_{14}+g(4,{2,3})}

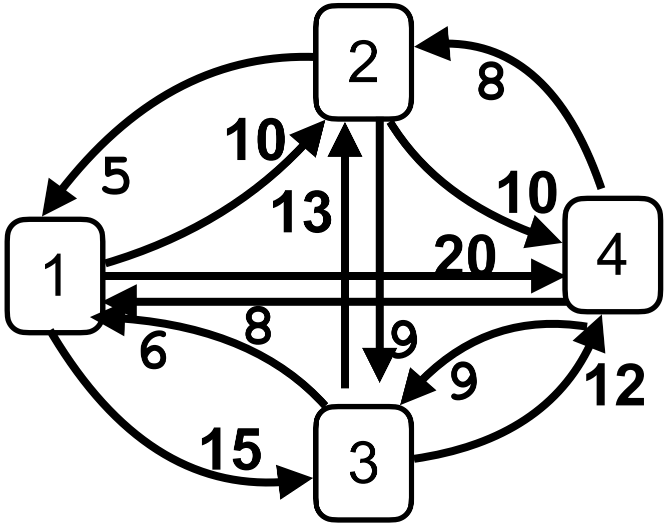
=min{10+25, 15+25, 20+23}

=35

J(1,{2,3,4})=2

- Thus, the optimal tour has length 35.

TSP Example: Tour Construction



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

Knowing `J(1,{2,3,4})=2`,
`J(2,{3,4})=4`, and
`J(4,{3})=3`

The optimal tour is `1,2,4,3,1.`

Complexity Analysis (Book)

- Number of g(i,S) that have to be computed
- For each value of |S|=k,
  - There are n-k-1 choices for i
    - Exclude vertex 1 from remaining n-k nodes
  - The number of subsets of size k excluding 1, and i
    $^{n-2}C_k$
- Thus, total number of g(i,S) to be computed

$$\sum_{k=0}^{n-2} (n-k-1)(^{n-2}C_k) \geq \sum_{k=0}^{n-2} (n-1)(^{n-2}C_k)$$

$$= (n-1)(1+1)^{n-2} = (n-1)2^{n-2}$$

- Computation of g(i,S) for each |S|=k requires
  - k-1 comparisons (min of k terms)
- Taking highest value of k as n-1
- Thus, total time complexity = $(n-1)*(n-1)2^{n-2}=O(n^2 2^n)$

Complexity Analysis (Other Look)

- For the n vertices in the graph,
  - There are 2^n subsets.
- For each subset, two kind of work is done
  - Addition (costs),
  - Comparison (to find minimum).
- Computation for each subset
  - Go thru each vertex once to find the min cost path
    - $O(n)$
  - For each vertex, check which is the right vertex before it.
    - $O(n)$
  - Thus, work done $O(n) * O(n) = O(n^2)$
- Total time complexity: $O(n^2) * O(2^n) = O(n^2 2^n)$

Summary

- Understanding TSP problem
- Application of Dynamic Programming
- Complexity analysis