Traveling Salesman Problem
&
linear/integer Programming

Traveling Salesman Problem

- Congrats! You got hired as the new super delivery person at the USPS! Your first job is to deliver your letters across the country.

- **Goal**: In what order should you visit all of the cities to minimize the total distance of your trip?
  - You’ll hear this also called the “Traveling Salesman Problem” or TSP for short.

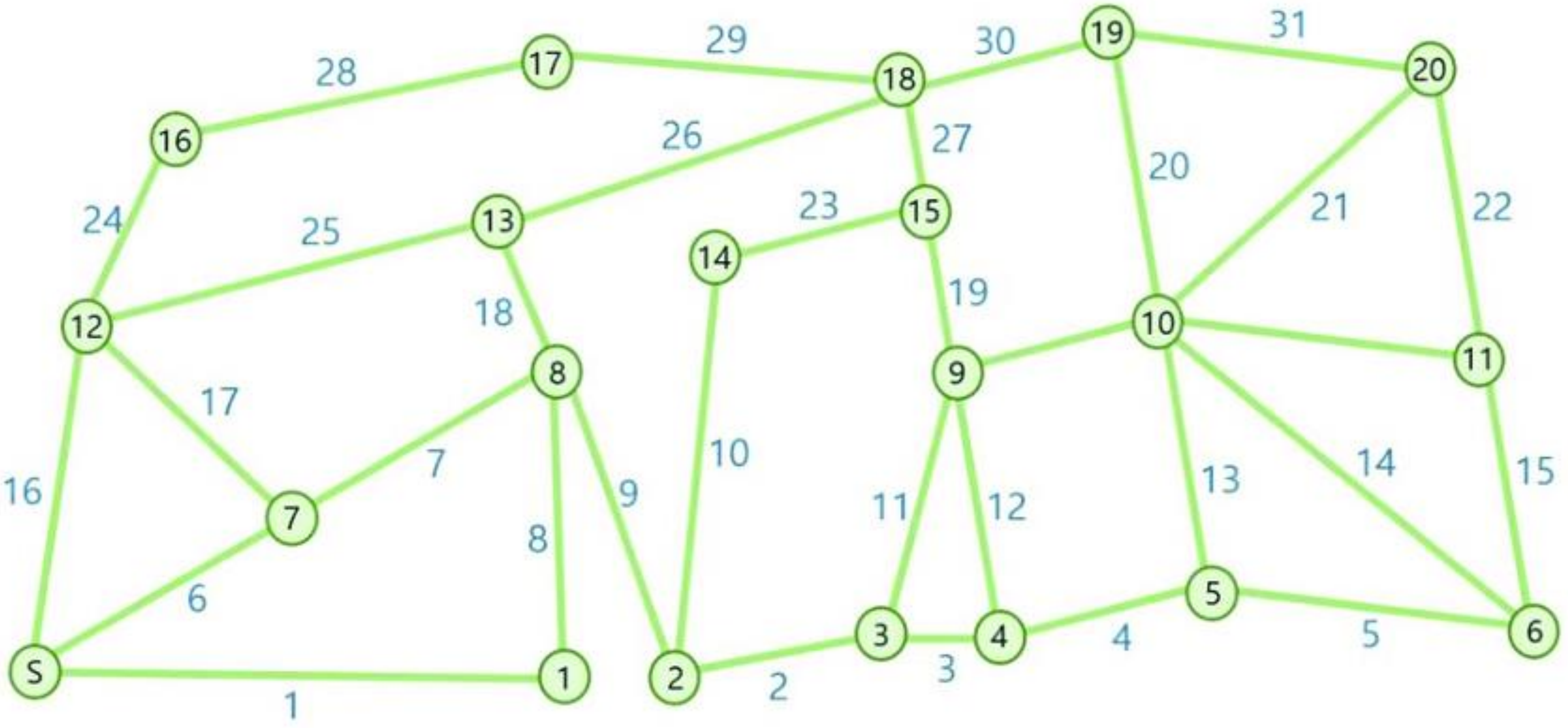
- *Problem*: There are a LOT of possible paths

Traveling Salesman Problem

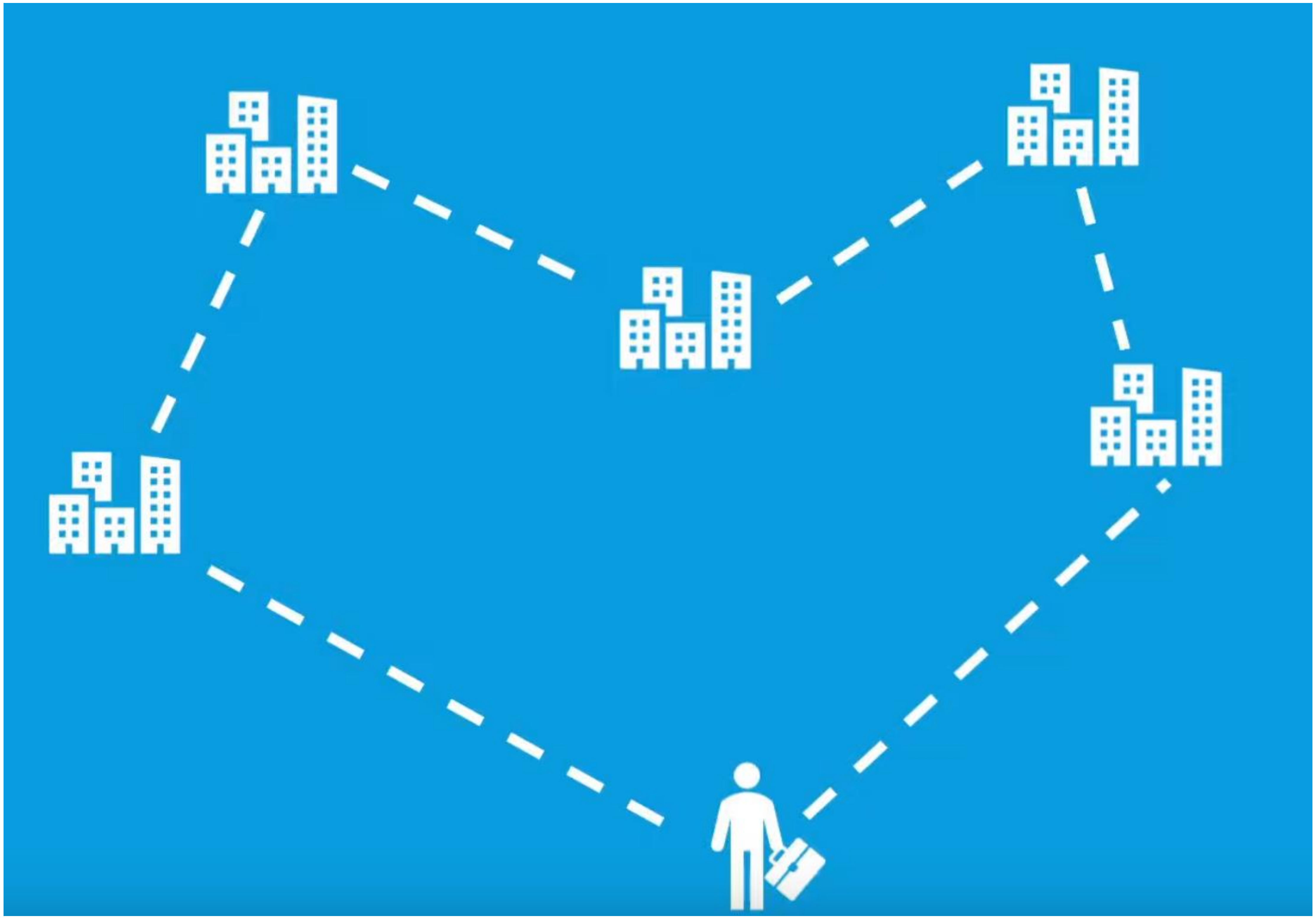
Given a weighted graph on n nodes, find a tour that visits every node once.

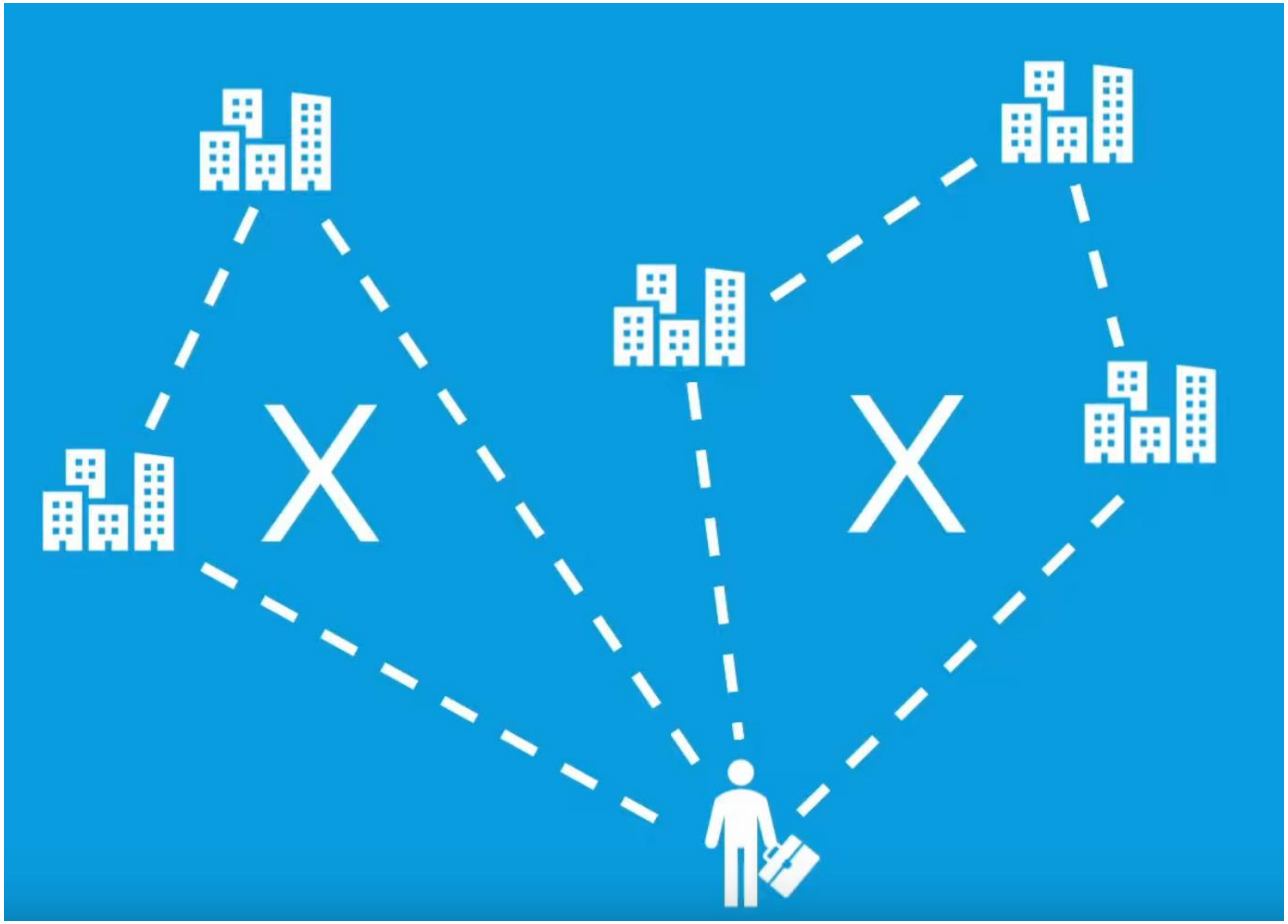
Though this problem is easy enough to explain, it is very difficult to solve.

Try it Out

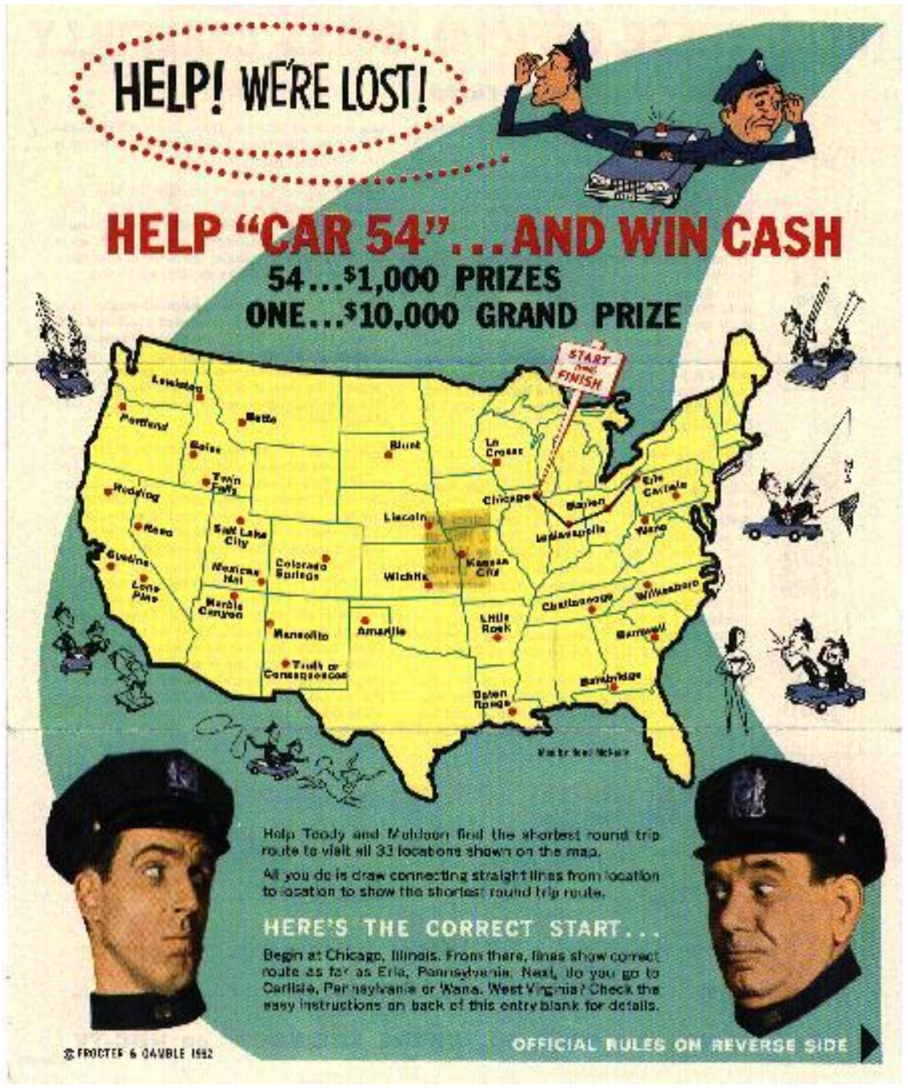

S
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
1
2
3
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

Example




X
X

TSP contest (1962)


HELP! WE'RE LOST!
HELP "CAR 54"...AND WIN CASH
54...$1,000 PRIZES
ONE...$10,000 GRAND PRIZE
START AND FINISH
HERE'S THE CORRECT START...
OFFICIAL RULES ON REVERSE SIDE
© PROCTER & GAMBLE 1962

Help Toody and Muldoon find the shortest round trip route to visit all 33 locations shown on the map.

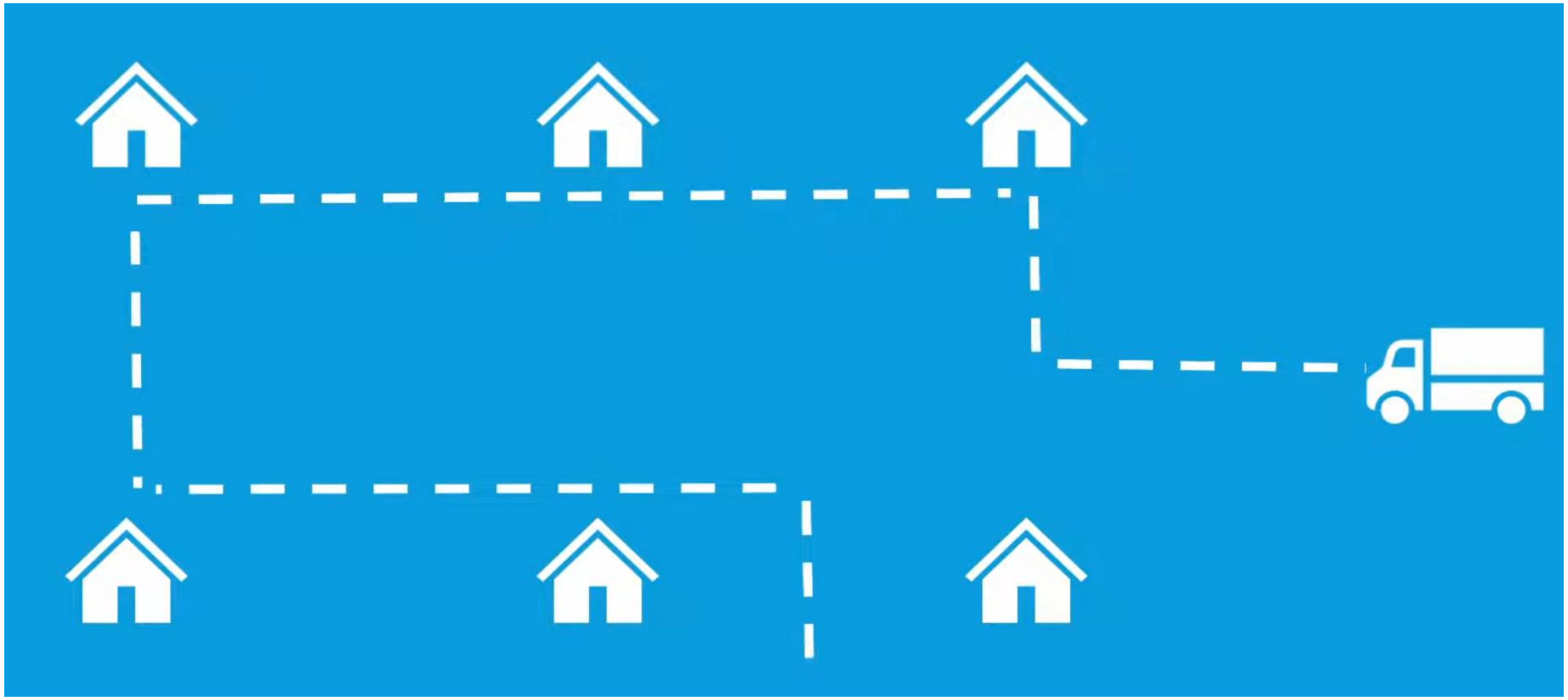
All you do is draw connecting straight lines from location to location to show the shortest round trip route.

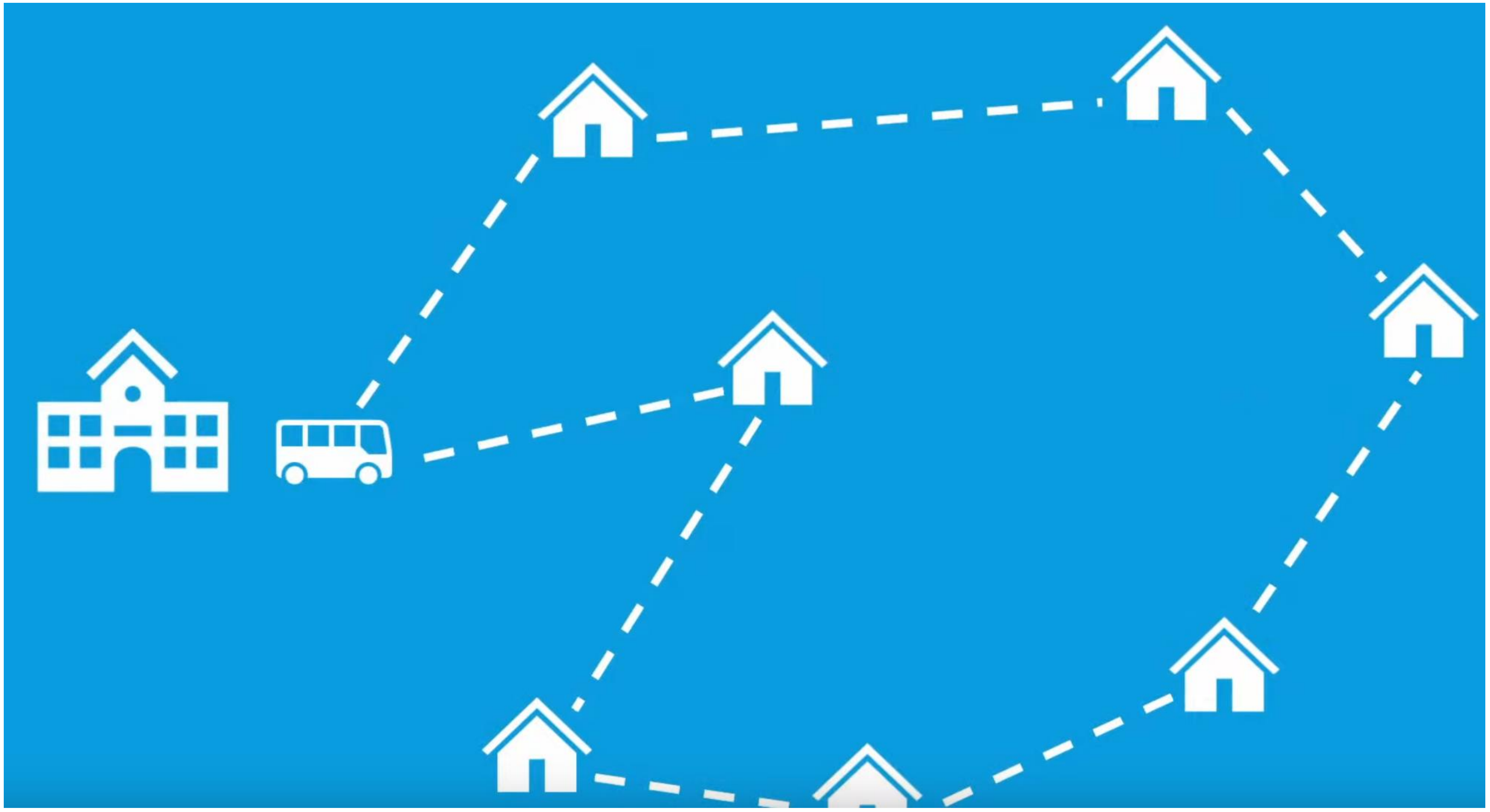
Begin at Chicago, Illinois. From there, lines show correct route as far as Erie, Pennsylvania. Next, do you go to Carlisle, Pennsylvania or Wana, West Virginia? Check the easy instructions on back of this entry blank for details.

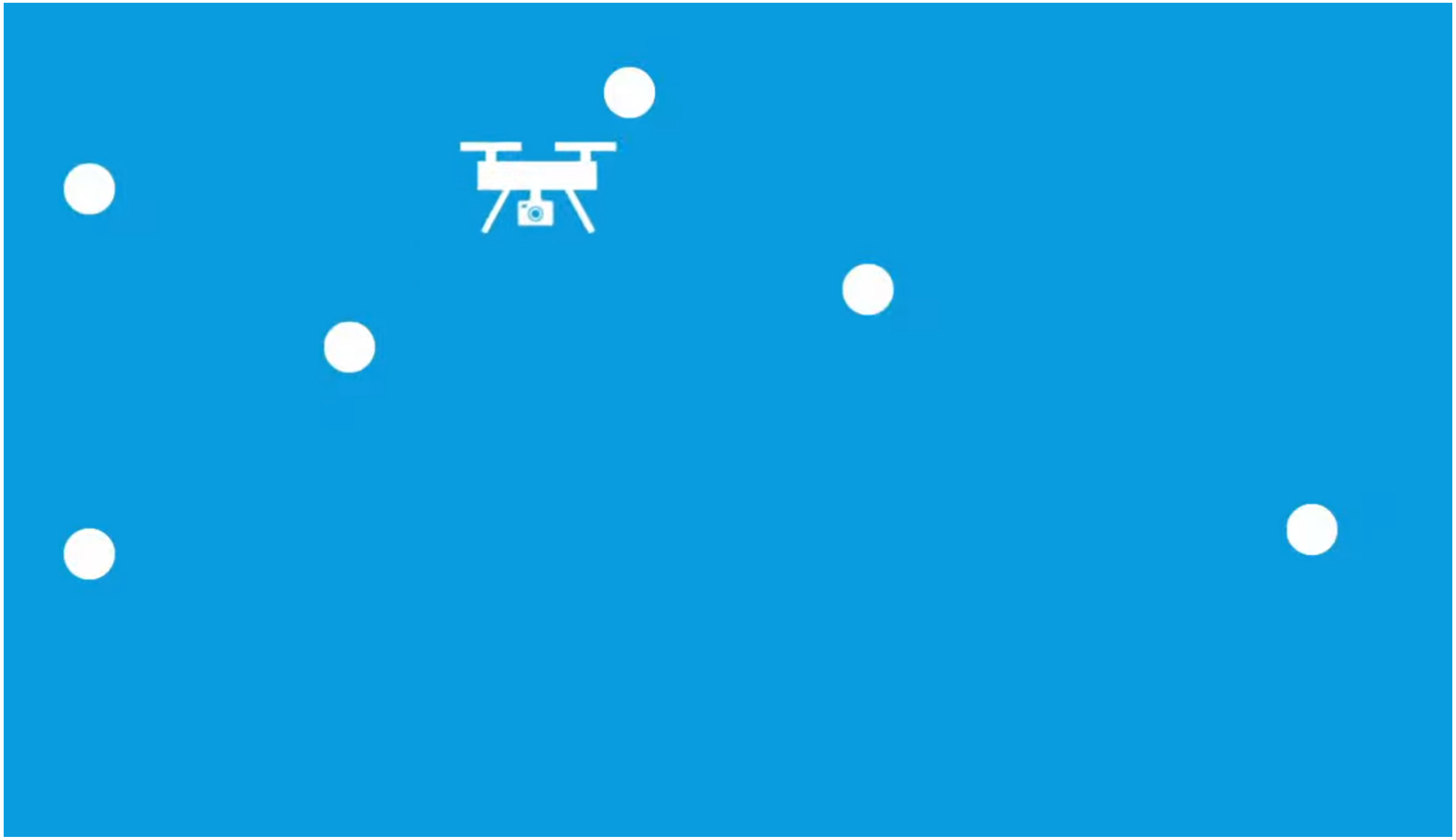
Why bother?

The TSP has many practical applications

-data transferring between different centers.
-manufacturing
-plane routing
-telephone routing
-networks
-traveling salespeople
-structure of crystals







Why bother?

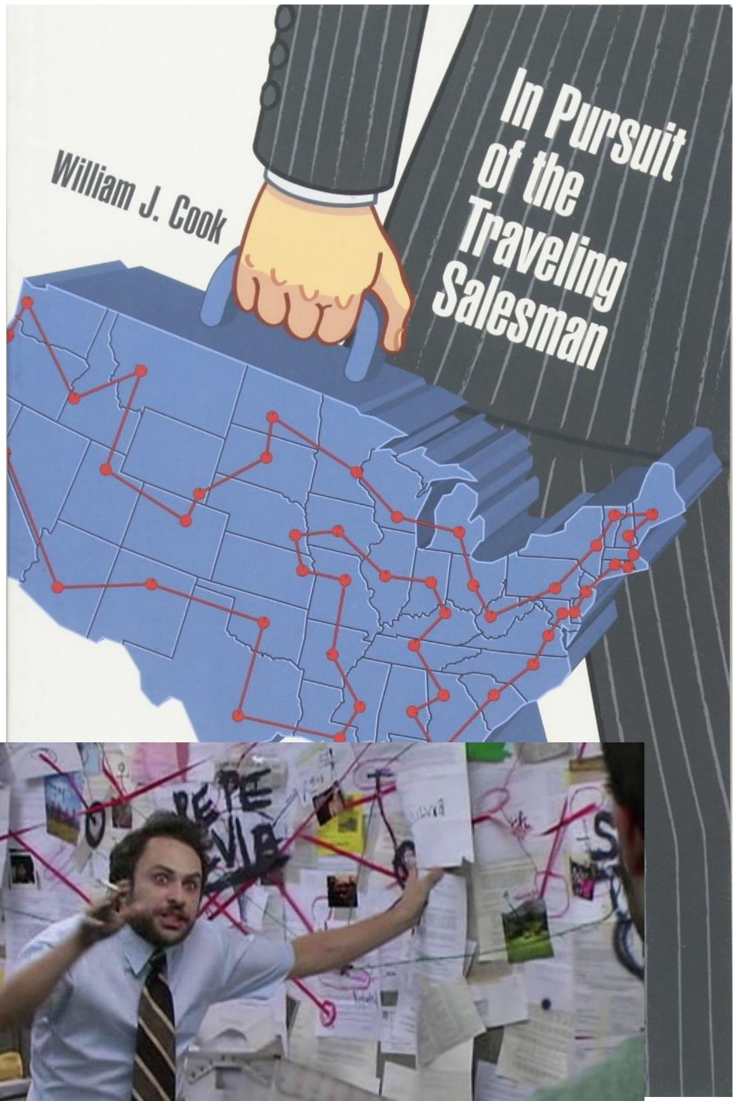
The TSP is interesting because it is in the class that is called NP-complete. If the TSP is found to have an algorithm that does not take exponential time, the other problems that are NP-complete will also be solved, and vice-a-versa.

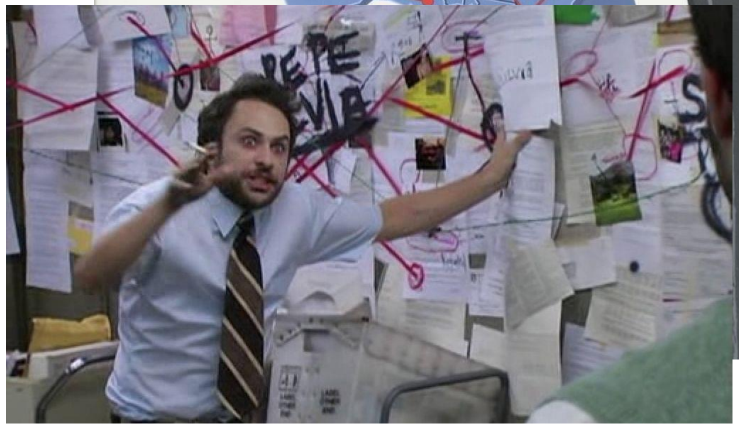
Why bother?

The TSP has quite a history.

-In the Odyssey by Homer, Ulysses has to travel to 16 cities. 653,837,184,000 distinct routes are possible.

-One of the first TSP papers was published in the 1920s.


William J. Cook
In Pursuit of the Traveling Salesman



Input: A undirected, weighted graph G**.**

- Commonly assume G is complete (pair-wise connections)

Output: A path of minimum total cost from some start vertex that visits every node in the graph exactly once.

- Start vertex will be visited twice, once at beginning, and once at end
- Also called a “Hamiltonian cycle of minimum weight”

Aside Metric TSP

Many variations of this problem exist. Today, we will focus on one called “Metric TSP” which assumes the graph represents something like a bunch of cities and the edge weights are distances between them.

The “Metric” requirement means that the distances should behave normally:

- All the edge-weights are non-negative (0 if an edge from a node to itself).
- The edge weights follow the “triangle inequality”

$$d(A,C) \leq d(A,B) + d(B,C)$$

- Intuition: “The hypotenuse is shorter than the sum of its sides”

Possible solution???

Possible solution???

Finding shortest paths in a graph? I know! Let’s use Dijkstra’s!

Algorithm:

- Pick a start vertex *v*
- Run Dijkstra’s from *v* to get shortest path tree of graph
- ???
- ???

Not clear how to turn this shortest path tree into a solution! It gives an idea of where to start, but once you explore one path, you won’t know how to finish the path that visits the rest of the graph.

An Exact Solution

The simplest algorithm is to try all possible paths

- Takes $\mathcal{O}(|V|!)$ time

So far, the only algorithms we know of to solve TSP exactly involve some kind of “choose-explore-unchoose” pattern of recursive backtracking.

- Some of the best algorithms take something like $\mathcal{O}\left(2^{|V|}\right)$

Facts

Most people think that the best we can hope for is exponential.

- This means this problem isn’t tractable for even moderately sized inputs. With *only* 85k cities, would take something like 130 years to compute an exact solution on a fast computer (Applegate et al. (2006)).
- More on why computer scientists are pessimistic on finding a better solution on Monday! It has to do with this problem called “P vs. NP.”

Integer Programming?

Integer Programming?

$$
\begin{array}{ll}
\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} & \\
0 \leq x_{ij} \leq 1 & i, j = 0, \cdots, n \\
u_i \in \mathbf{Z} & i = 0, \cdots, n \\
\sum_{i=0, i \neq j}^{n} x_{ij} = 1 & j = 0, \cdots, n \\
\sum_{j=0, j \neq i}^{n} x_{ij} = 1 & i = 0, \cdots, n \\
u_i - u_j + n x_{ij} \leq n - 1 & 1 \leq i \neq j \leq n
\end{array}
$$

Approximate solution???

If we don’t think we can solve the problem exactly, is there hope of finding an efficient algorithm that yields an answer that is “close enough”?

This is the idea behind **approximation algorithms.** We give up on exactness in the hope for efficiency gains.

- For example, if the true TSP solution was a 100-mile journey, is it that big of a deal if our approximation reports and answer that is 110 miles long?

Approximation Ratios

Define the **approximation ratio** of our proposed approximate solution *P* when compared to the optimal solution *P*.* We call an approximation algorithm that has an approximation ratio of α an **α-approximation.**

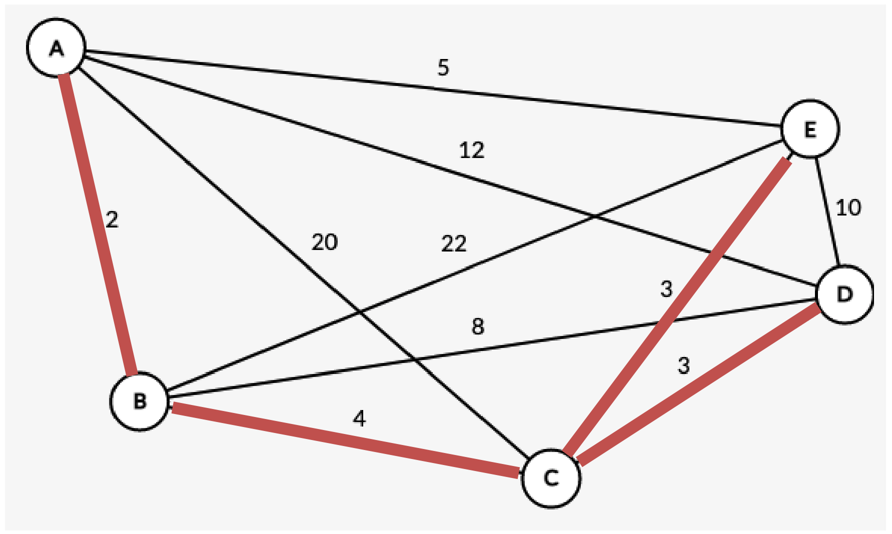
$$cost(P) \leq \alpha \cdot cost(P^*)$$

This is a property that constrains the gap in quality between your approximation and the globally optimal solution. The lower α, the better your approximation will be!

Examples:
2-approximation: If the true solution is 100 miles, we will guarantee an approximate solution with cost at most 200 miles.
1.1-approximation: If the true solution is 100 miles, we will guarantee an approximate solution with cost at most 110 miles.

There is an approximation algorithm to TSP that uses a MST.

1. Find an MST.
2. "Trace" a DFS on the MST to explore graph.
   - Build up path as you visit
   - Trace lists out nodes both times you visit it.
3. Skip vertices already visited. Use edges not included in MST when skipping.



Trace of DFS: A, B, C, D, C, E, C, B, A

Solution: A, B, C, E, A with total cost 24.
In this case, the optimal solution is A,B,D,C,E,A with cost 21

MST (minimum spanning tree)

Yes! This MST approach is a 2-approximation of the TSP problem!

- No matter what the optimal solution is, this algorithm will find one that’s no worse than twice as bad.

Intuition: In most cases, there should be a lot of overlap with an MST and a solution to the TSP (using some of the smallest edges in the graph).

Natural question then is: Can we do better than a 2-approximation?

References

http://iris.gmu.edu/~khoffman/papers/trav_salesman.html

http://www.densis.fee.unicamp.br/~moscato/TSPBIB_home.html

http://riceinfo.rice.edu/projects/reno/m/19980625/tsp.html

http://www.pcug.org.au/dakin/tspbb.htm

The Traveling Salesman Problem Edited by Lawler, Lenstra, Rinnoy Kan, Shmoys