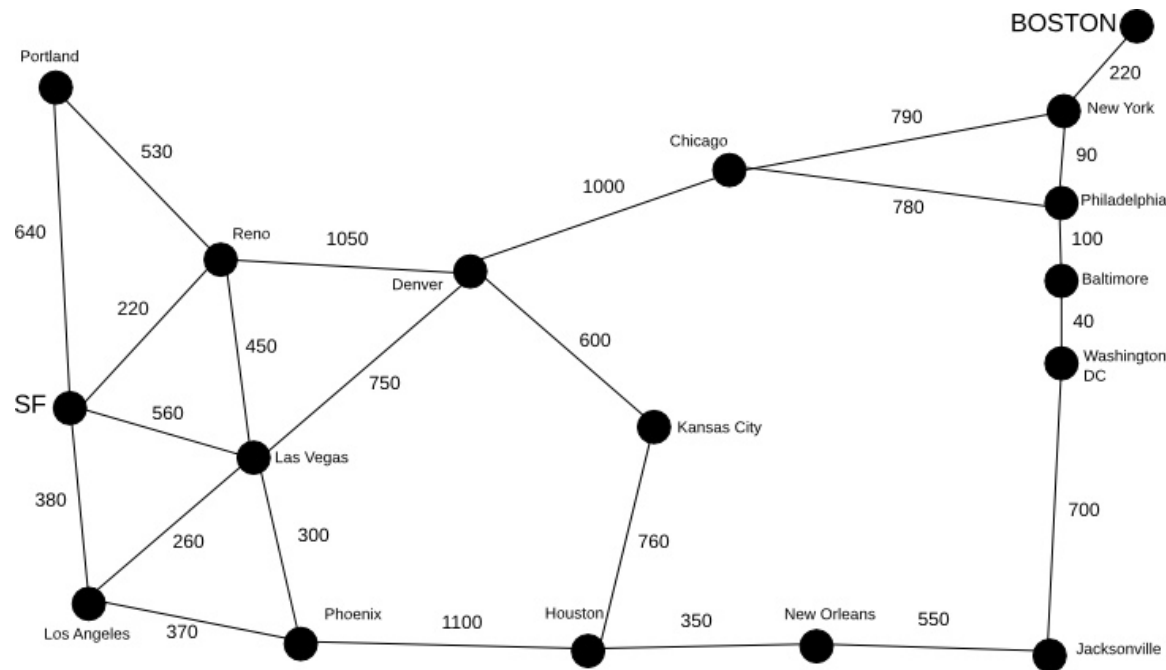


# WEIGHTED GRAPHS

A **weighted graph** is a graph  $G(V,E)$  together with a function  $w: E \rightarrow [0, \infty)$

For  $e \in E$ , the number  $w(e)$  is the **weight** of  $e$ .



# WEIGHTED GRAPHS

Graph	Vertices	Edges	Weights
communication	computers	fiberoptic cables	response time
air travel	airports	flights	flight times
car travel	street corners	streets	distances
Kevin Bacon	actors	common movies	1
stock market	stocks	transactions (directed edges)	cost
operations research	projects	dependencies (directed edges)	times

# DISTANCE PROBLEMS

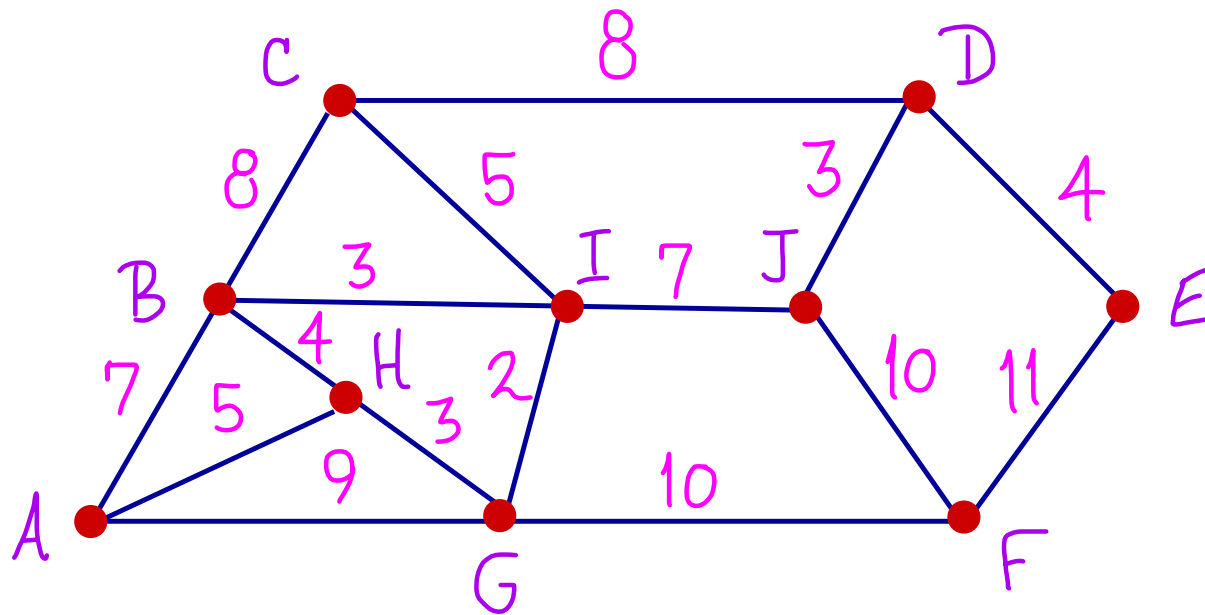
**TRAVELING SALESMAN PROBLEM.** Given a list of cities to visit, what is the minimum distance you need to travel?

TSP is really a question about weighted graphs.

**EASIER PROBLEM.** Given two vertices in a weighted graph, what is their "distance."

# EXAMPLE

PROBLEM. Find the distance between A and E.



How to find the shortest path in general?

# DIJKSTRA'S ALGORITHM

To find the distances from a given vertex  $A$  in a weighted graph to all other vertices, do the following.

First, give  $A$  the permanent label 0, and give all other vertices the temporary label  $\infty$ .

Then repeat the following step:

Find the vertex  $v$  with the newest permanent label.

For each vertex  $v'$  adjacent to  $v$  with a temporary label, check if

$$\text{label of } v + w(vv') \leq \text{label of } v'$$

If so, change the temporary label of  $v'$ .

Make the smallest temporary label permanent.

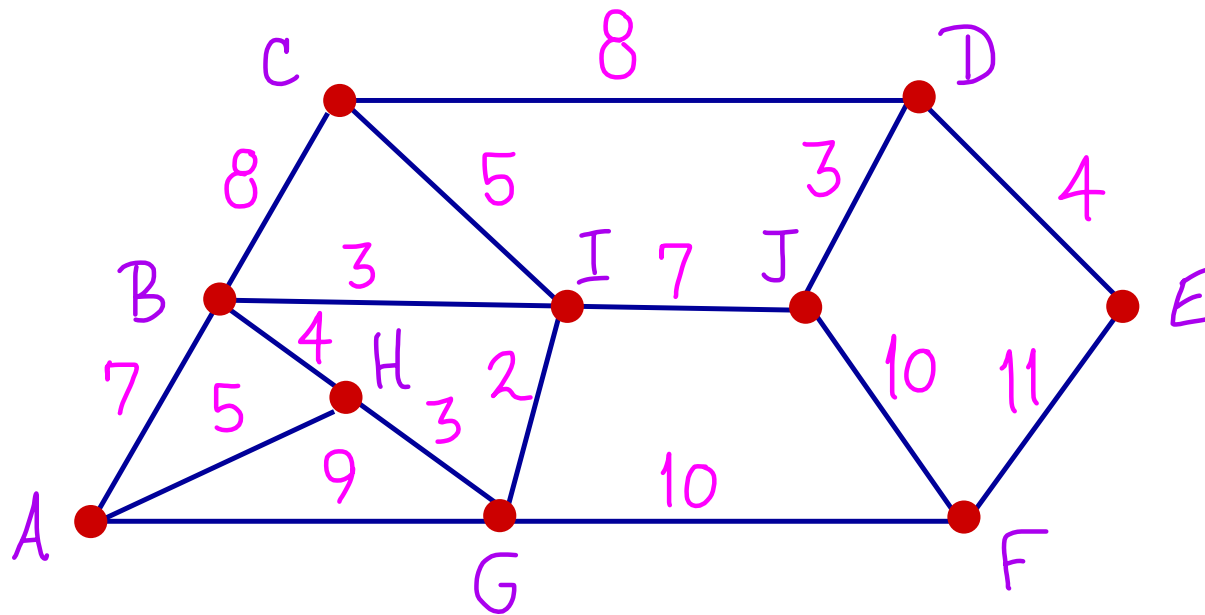


Edsger Dijkstra

Permanent labels are the distances from  $A$ .

# DIJKSTRA'S ALGORITHM

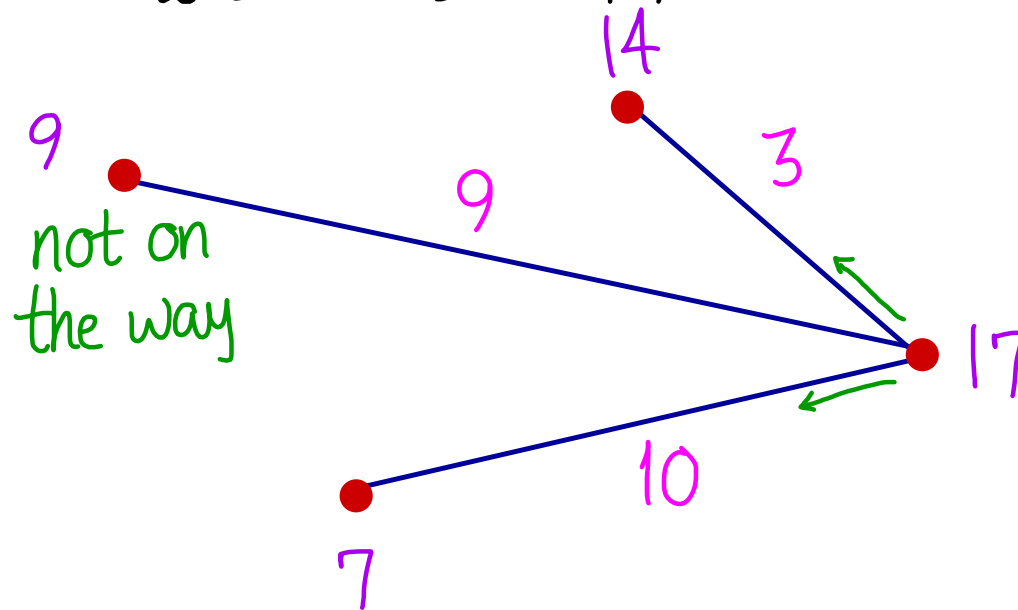
Find the distance from A to each other vertex.



# DIJKSTRA'S ALGORITHM

What if we further want to find a walk between two vertices with the shortest length (not just the distance between the two vertices)?

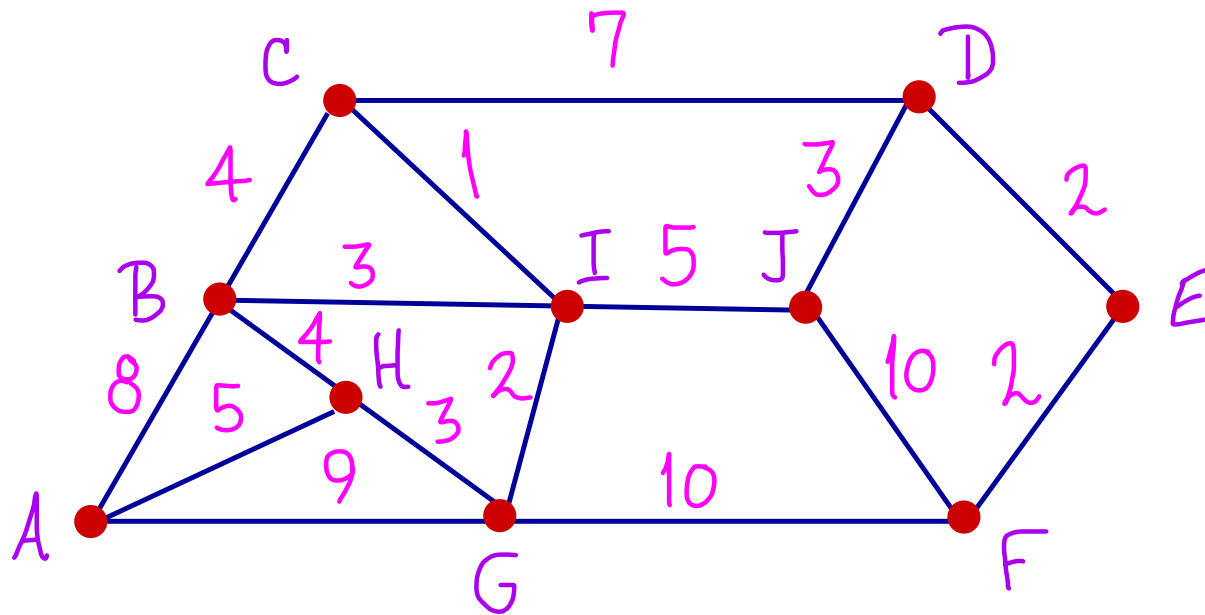
Idea: Every time we make a label permanent, draw a little arrow from that vertex to all other vertices that are "en route" to the home vertex A



Then, follow the arrows to find all shortest walks home.

# DIJKSTRA'S ALGORITHM

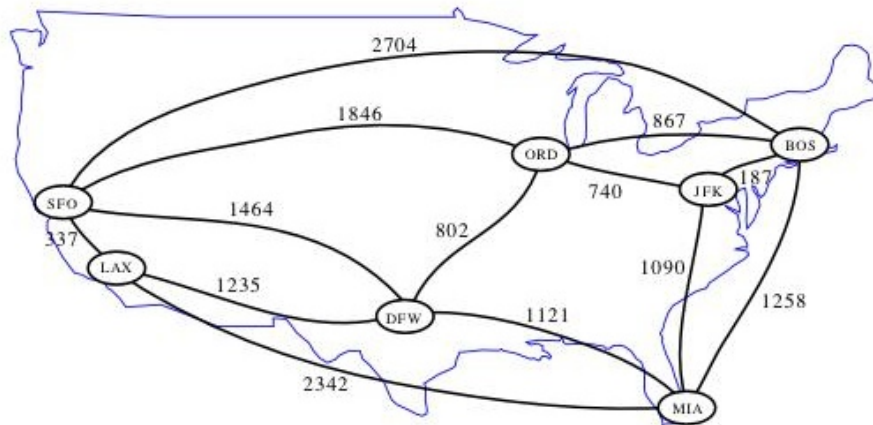
Find all shortest paths from A to E.



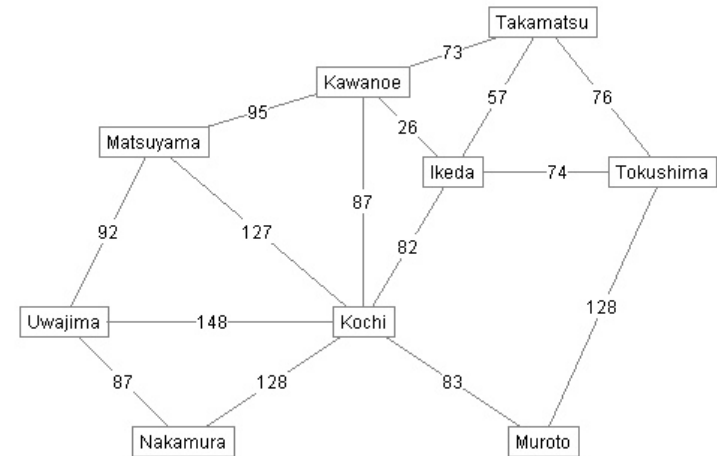


# DIJKSTRA'S ALGORITHM

Find the shortest paths...



from LAX to JFK



from Nakamura to Tokushima

# DIJKSTRA'S ALGORITHM

What is the complexity of Dijkstra's algorithm, if size is measured in the number of vertices and cost is measured in terms of number of operations (=additions and comparisons)?

# FLOYD-WARSHALL ALGORITHM

Idea: Number the vertices  $v_1, \dots, v_n$ .

Step  $k$ : Find the shortest path from  $v_i$  to  $v_j$  if you are only allowed to use  $v_1, \dots, v_k$  as intermediate vertices (= pit stops).

Can write this info in a matrix  $M_k$ .

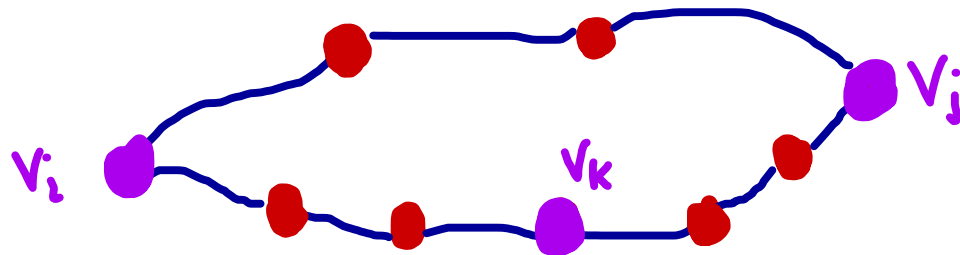
Write  $\infty$  if there is no path.

Do this for  $k=0, \dots, n$ . (At Step 0, no pit stops allowed.)

The  $ij$ -entry of  $M_n$  is the distance from  $v_i$  to  $v_j$ .

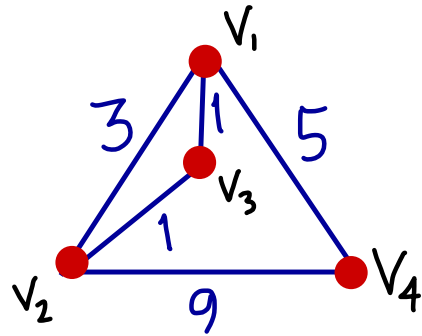
Key observation. For  $k \geq 1$ :

$$M_k(i, j) = \min \{ M_{k-1}(i, j), M_{k-1}(i, k) + M_{k-1}(k, j) \}$$



# FLOYD-WARSHALL ALGORITHM

EXAMPLE.



$$M_0 = \begin{pmatrix} 0 & 3 & 1 & 5 \\ & 0 & 1 & 9 \\ & & 0 & \infty \\ & & & 0 \end{pmatrix}$$

$$M_1 = \begin{pmatrix} 0 & 3 & 1 & 5 \\ & 0 & 1 & 8 \\ & & 0 & 6 \\ & & & 0 \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 0 & 3 & 1 & 5 \\ & 0 & 1 & 8 \\ & & 0 & 6 \\ & & & 0 \end{pmatrix}$$

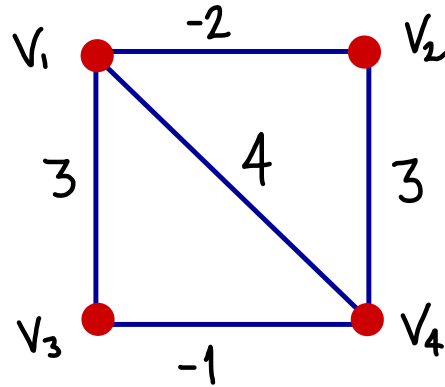
$$M_3 = \begin{pmatrix} 0 & 2 & 1 & 5 \\ & 0 & 1 & 7 \\ & & 0 & 6 \\ & & & 0 \end{pmatrix}$$

$$M_4 = \begin{pmatrix} 0 & 2 & 1 & 5 \\ & 0 & 1 & 7 \\ & & 0 & 6 \\ & & & 0 \end{pmatrix}$$

Note:  $M_k$  has same row/col  $k$  as  $M_{k-1}$ .

# FLOYD-WARSHALL ALGORITHM

Find all distances using the Floyd-Warshall algorithm.



# DIJKSTRA vs FLOYD-WARSHALL

To find distances for all pairs of vertices, we need to run Dijkstra's algorithm  $n$  times  $\leadsto O(n^3)$ .

Floyd-Warshall is also  $O(n^3)$ , but is quicker for large graphs.

One advantage to Floyd-Warshall is that it even works with negative edge weights.