Lecture 15: Chapter 3 Part 3

Dynamic Programming

CS3310

The Principle of Optimality

- With an **optimization problem**, the goal is to determine an optimal series of choices from a set of possible choices.

- We have discussed several optimization problems, such as finding a shortest path from every vertex to every other vertex in a graph.

- It may seem like any optimization problem can be solved using dynamic programming. However, this isn't always true.

- The **principle of optimality** must apply in the problem.

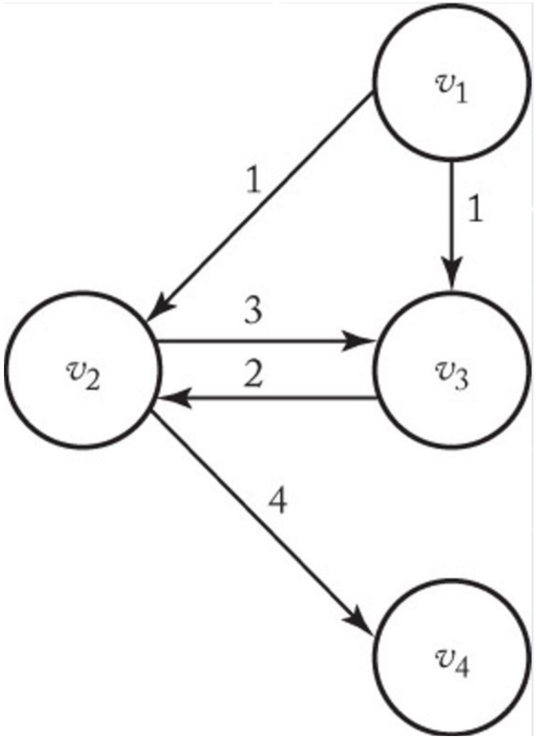
The Principle of Optimality

The **principle of optimality** applies in a problem if an optimal solution to an instance of that problem always contains optimal solutions to all of its sub-instances.

What does this mean?

The Principle of Optimality

Consider this graph. What is the optimal path from v_1 to v_4?



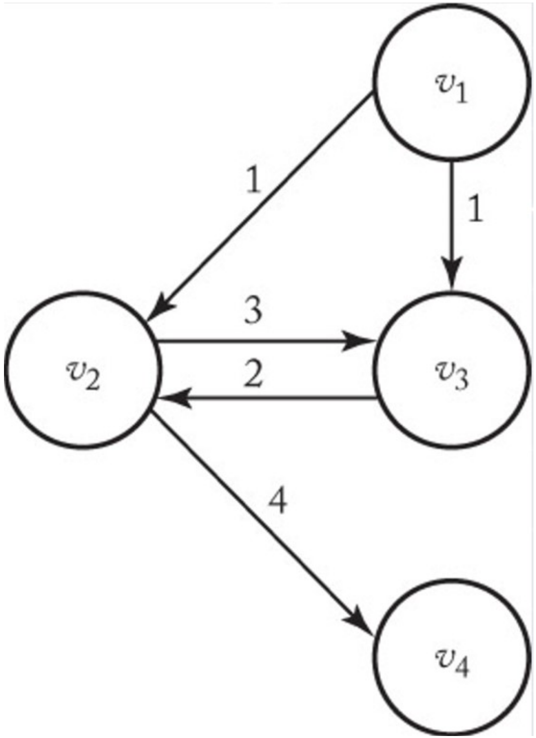
The Principle of Optimality

Consider this graph. What is the optimal path from v_1 to v_4? **[v_1, v_2, v_4]**

Note: the *subpaths* of the optimal path are also optimal solutions to their respective sub-instances.

- i.e. [v_1, v_2] is the optimal path from v_1 to v_2 and [v_2, v_4] is the optimal path from v_2 to v_4.

The **principle of optimality** applies in a problem if an optimal solution to an instance of that problem always contains optimal solutions to all of its sub-instances.

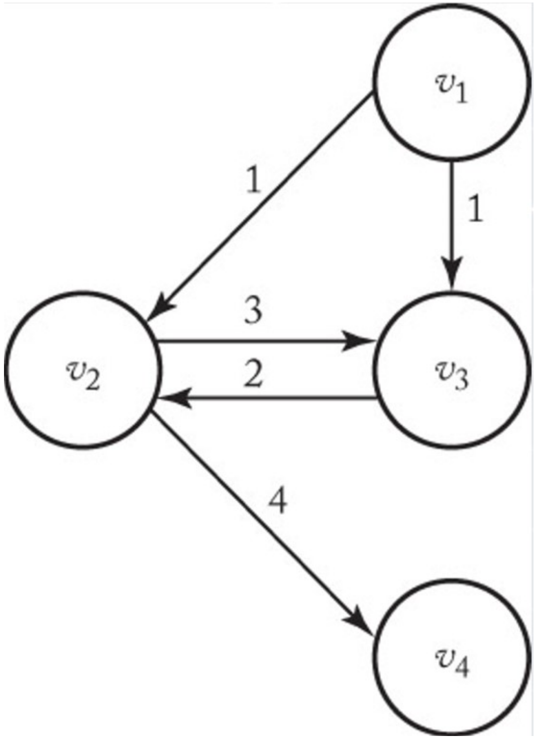


The Principle of Optimality

- If the principle of optimality applies in a given problem, we can develop a recursive property that gives an optimal solution to an instance by first finding optimal solutions to sub-instances.

- i.e. if we know the shortest path from v_1 to v_2 and the shortest path from v_2 to v_4, we know those paths combine to form the shortest path from v_1 to v_4.

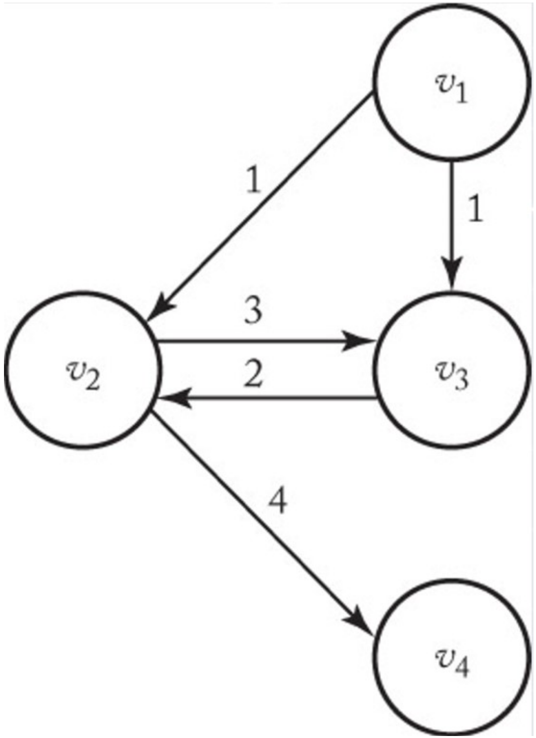
- We build our solution in a bottom-up fashion, solving lower-level sub-instances before calculating the solution to the instance.



The Principle of Optimality

Why the Principle of Optimality is important!

- This principle seems fairly obvious.
- However, it is necessary to show that it applies to a problem before assuming that an optimal solution can be obtained with dynamic programming.
- Consider the Longest Paths problem.
- What is the longest simple (no cycles) path from v_1 to v_4?

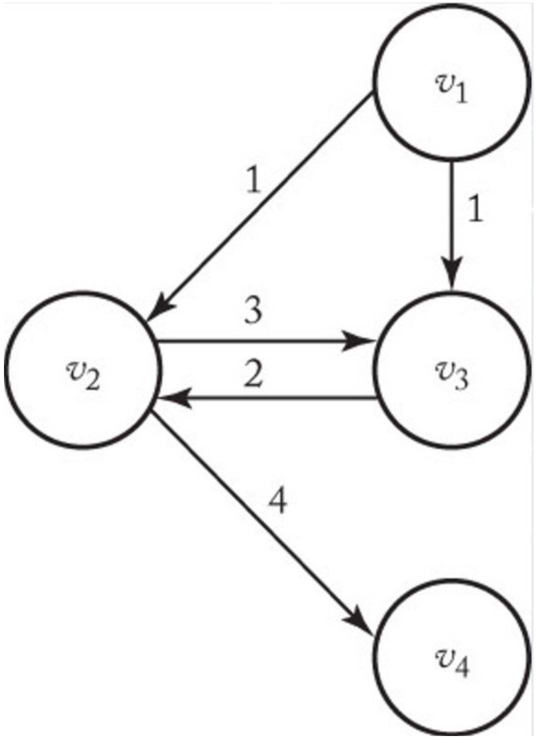


The Principle of Optimality

Why the Principle of Optimality is important!

- This principle seems fairly obvious.
- However, it is necessary to show that it applies to a problem before assuming that an optimal solution can be obtained with dynamic programming.
- Consider the Longest Paths problem.
- What is the longest simple (no cycles) path from v_1 to v_4? **[v_1, v_3, v_2, v_4]**

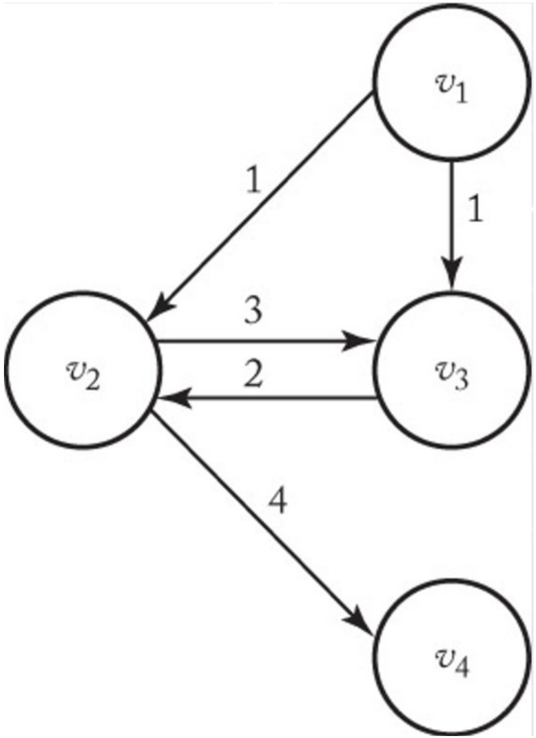
Does this path contain a solution to every sub-instance?



The Principle of Optimality

Why the Principle of Optimality is important!

- This principle seems fairly obvious.
- However, it is necessary to show that it applies to a problem before assuming that an optimal solution can be obtained with dynamic programming.
- Consider the Longest Paths problem.
- What is the longest simple (no cycles) path from v_1 to v_4? **[v_1, v_3, v_2, v_4]**



Does this path contain a solution to every sub-instance?
No! [v_1, v_3] is not an optimal solution to the subproblem of the longest path between v_1 and v_3. [v_1, v_2, v_3] is.

Traveling Salesperson Problem

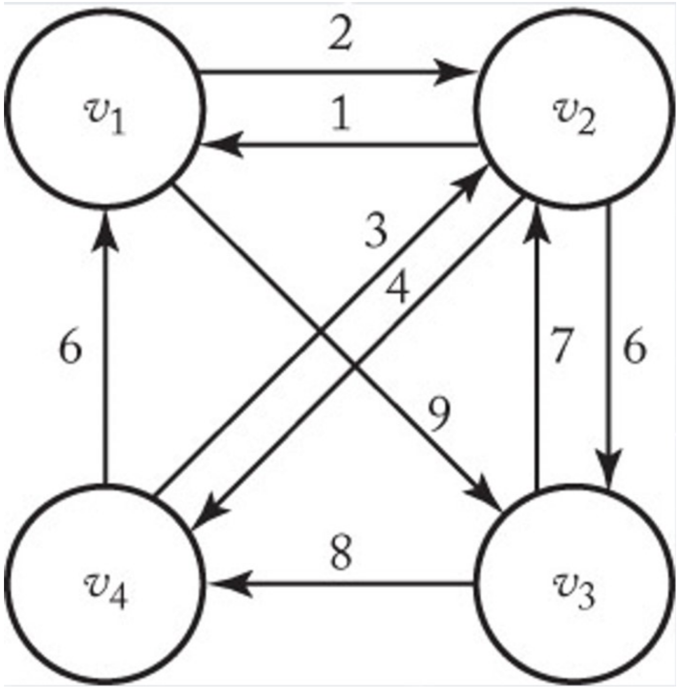
- Suppose a salesperson is planning a sales trip that includes 20 cities. Each city is connected to some of the other cities by a road. To minimize travel time, we want to determine a shortest route that starts at the salesperson's home city, visits each of the other cities exactly *once*, and ends back at the home city.

- A **tour** or **Hamiltonian Circuit** in a directed graph is a path from a vertex to itself that passes through each other vertex exactly once.

- An **optimal tour** in a weighted, directed graph is such a path of minimum length.

Traveling Salesperson Problem

Which vertex should we select as the start of our tour?



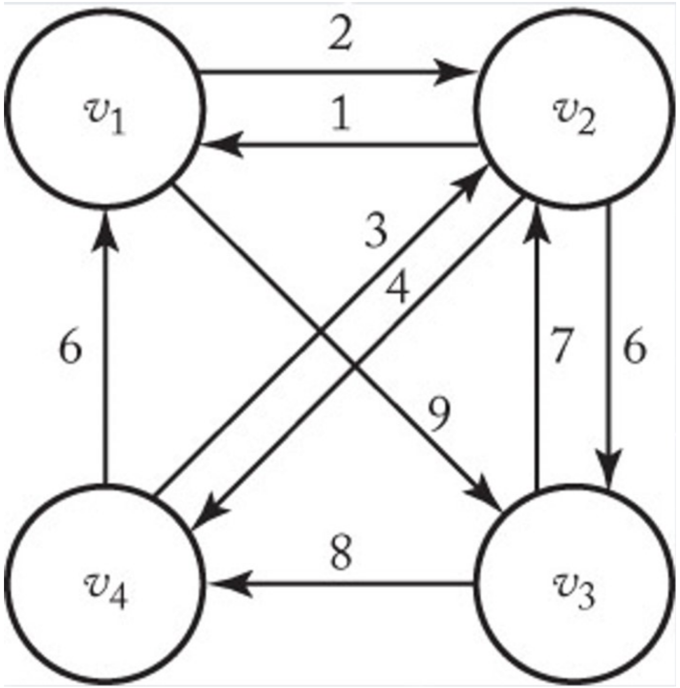
Traveling Salesperson Problem

Which vertex should we select as the start of our tour?

It doesn't matter!

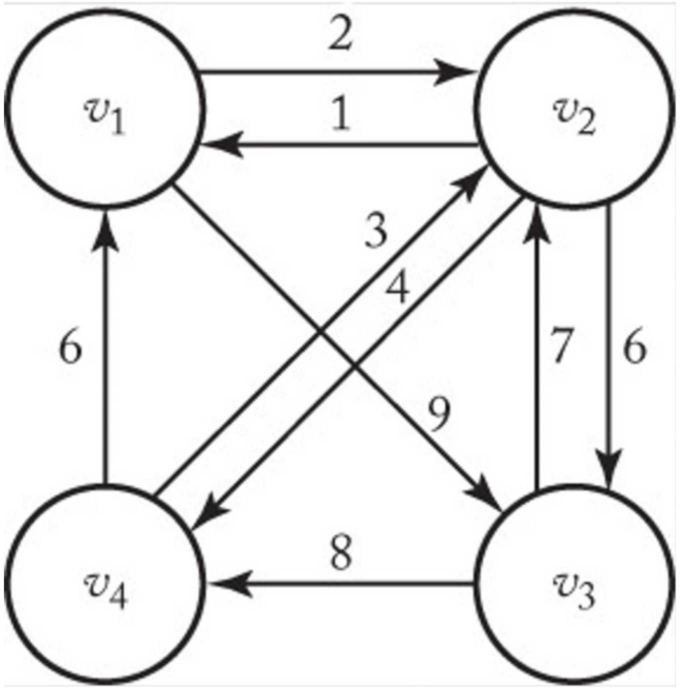
- ➢ $[v_1, v_2, v_3, v_4, v_1] = [v_2, v_3, v_4, v_1, v_2]$, etc.

- The starting vertex is irrelevant. For that reason, we will arbitrarily choose v_1 as our starting vertex.



Traveling Salesperson Problem

What are the possible tours of this graph?



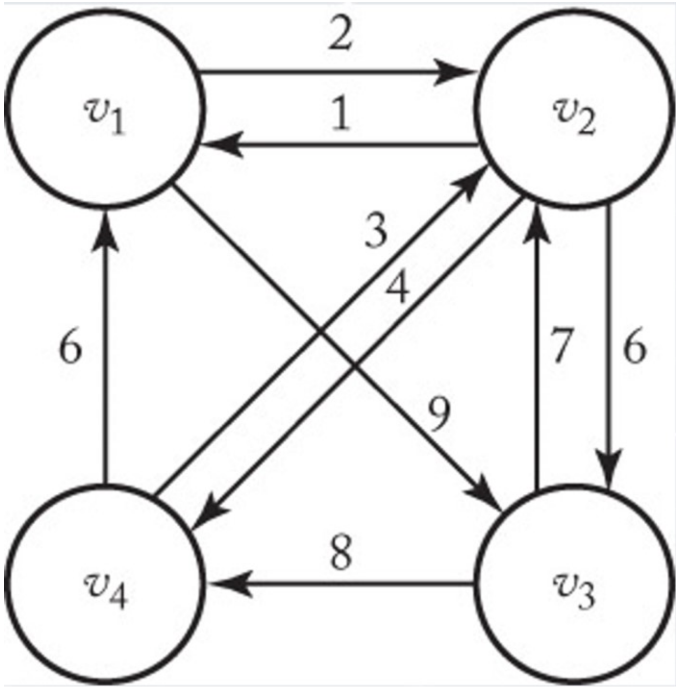
Traveling Salesperson Problem

What are the possible tours of this graph?

- $[v_1, v_2, v_3, v_4, v_1]$ with a length of 22
- $[v_1, v_3, v_2, v_4, v_1]$ with a length of 26
- $[v_1, v_3, v_4, v_2, v_1]$ with a length of 21

The third tour is optimal. In a brute force algorithm, we consider every existing tour.

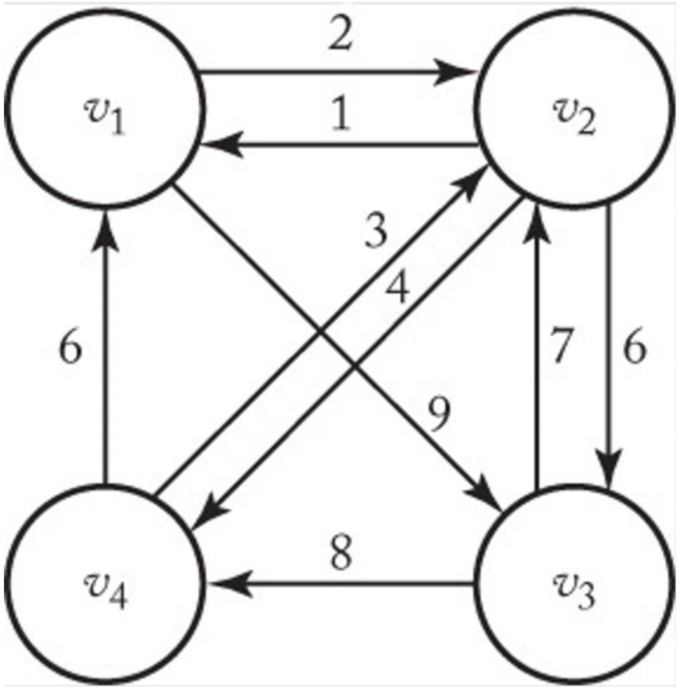
- What is the time complexity of this?



Traveling Salesperson Problem

What are the possible tours of this graph?

- $[v_1, v_2, v_3, v_4, v_1]$ with a length of 22
- $[v_1, v_3, v_2, v_4, v_1]$ with a length of 26
- $[v_1, v_3, v_4, v_2, v_1]$ with a length of 21



The third tour is optimal. In a brute force algorithm, we consider every existing tour.

- What is the time complexity of this?

The second vertex on a tour can be any of $(n - 1)$ vertices. The third can be any of $(n - 2)$, etc.

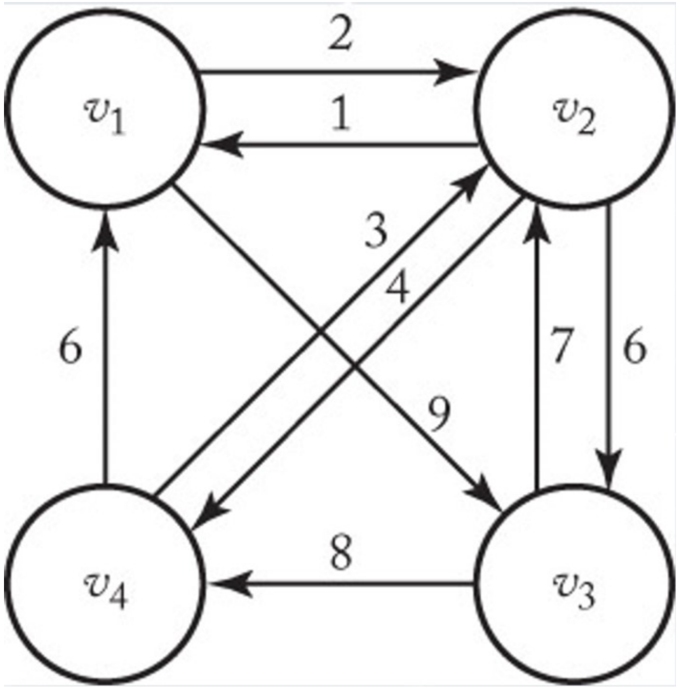
$(n - 1)(n - 2) \ldots 1 = \mathbf{(n - 1)!}$

Traveling Salesperson Problem

Suppose we have an optimal tour from v_1

- Let v_k be the first vertex after v_1 on that tour
- We can say that the path from v_k back to v_1 is a **subpath** of the optimal tour.
- This subpath <u>must</u> be the shortest path from v_k to v_1 that passes through each other vertex exactly once.

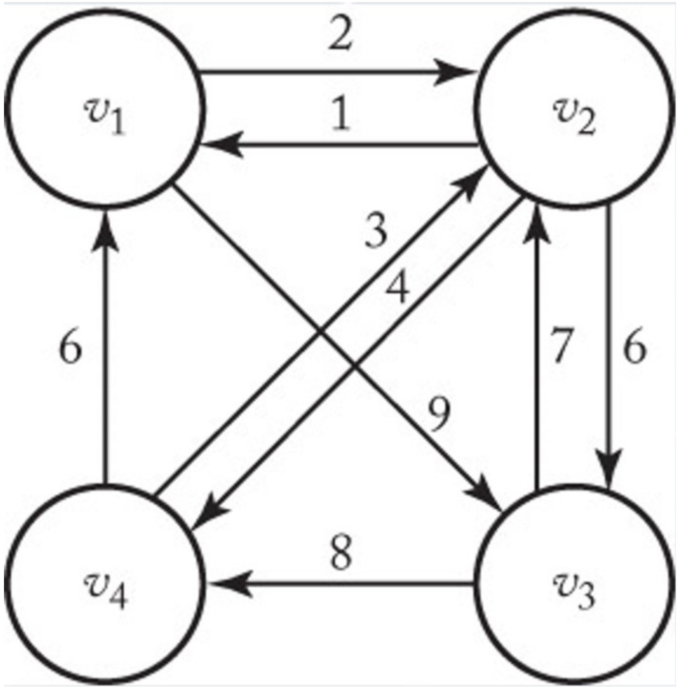
What does this tell us?



Traveling Salesperson Problem

Suppose we have an optimal tour from v_1

- Let v_k be the first vertex after v_1 on that tour
- We can say that the path from v_k back to v_1 is a **subpath** of the optimal tour.
- This subpath must be the shortest path from v_k to v_1 that passes through each other vertex exactly once.



What does this tell us?

➢ The **principle of optimality** applies and we can use dynamic programming!

Traveling Salesperson Problem

Let:

- V = a set of all vertices in the graph
- A = a subset of V
- $D[v_i][A]$ = the length of a shortest path from v_i to v_1 that passes through each vertex in A exactly once.

For example, $D[v_2][\{v_3, v_4\}]$ = length of a shortest path from v_2 to v_1 passing through both v_3 and v_4 exactly one time.

Traveling Salesperson Problem Example

Given the following V and A:

$V = \{v_1, v_2, v_3, v_4\}$
$A = \{v_3\}$

$D[v_2][A] = ?$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem Example

Given the following V and A:

$V = \{v_1, v_2, v_3, v_4\}$
$A = \{v_3\}$

$D[v_2][A] = \text{length}[v_2, v_3, v_1] = \infty$

However, given:
$A = \{v_3, v_4\}$

$D[v_2][A] = ?$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem Example

Given the following V and A:

$V = \{v_1, v_2, v_3, v_4\}$
$A = \{v_3\}$

$D[v_2][A] = \text{length}[v_2, v_3, v_1] = \infty$

However, given:
$A = \{v_3, v_4\}$

$D[v_2][A] = \mathit{min}\ (\text{length}[v_2, v_3, v_4, v_1], \text{length}[v_2, v_4, v_3, v_1])$
$= \mathit{min}\ (20, \infty) = 20$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Length of an optimal tour =

$$\underset{2 \le j \le n}{\textit{minimum}} \left(W[1][j] + D[v_j][V - \{v_1, v_j\}]\right)$$

- For each vertex j from 2 to n, calculate and add:
  - The weight on the edge from v_1 to v_j
  - The weight of the shortest path from v_j to v_1 that passes through every other vertex except itself and 1.
- Then, choose the minimum of these values.

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Length of an optimal tour =

$$\underset{2 \le j \le n}{minimum} \; (W[1][j] + D[v_j][V - \{v_1, v_j\}])$$

- More generally, for $j \neq 1$ and v_j not in A:

$D[v_i][A] =$

$$\underset{j:\; v_j \in A}{minimum} \; (W[i][j] + D[v_j][A - \{v_j\}]) \text{ if } A \neq \emptyset$$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step One

Since this is a dynamic programming algorithm, we build our solution from the bottom up.

Which subpaths of a potential optimal tour should we calculate first?

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step One

Since this is a dynamic programming algorithm, we build our solution from the bottom up.

Which subpaths of a potential optimal tour should we calculate first?

- The lengths of the paths from each vertex to v_1 that don't pass through any other vertices.

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step One

In this case, A = the empty set.

- $D[v_2][\emptyset] = ?$
- $D[v_3][\emptyset] = ?$
- $D[v_4][\emptyset] = ?$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step One

In this case, A = the empty set.

- $D[v_2][\emptyset] = 1$
  - i.e. the length of the shortest path from v_2 to v_1 that passes through no other vertices.
- $D[v_3][\emptyset] = \infty$
- $D[v_4][\emptyset] = 6$

- Each of these simply calculates the direct weight from each vertex to v_1.

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step One

D[v_2][Ø] = 1 D[v_3][Ø] = ∞ D[v_4][Ø] = 6

Step Two

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step One

$D[v_2][\emptyset] = 1$ $D[v_3][\emptyset] = \infty$ $D[v_4][\emptyset] = 6$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Step Two

- Determine the lengths of the shortest paths from each vertex to v_1 that pass through 1 other vertex

- $D[v_4][\{v_2\}]$
- $D[v_2][\{v_3\}]$
- $D[v_4][\{v_3\}]$
- $D[v_2][\{v_4\}]$
- $D[v_3][\{v_4\}]$

Traveling Salesperson Problem

Step One

$D[v_2][\emptyset] = 1$ $D[v_3][\emptyset] = \infty$ $D[v_4][\emptyset] = 6$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Step Two

- Determine the lengths of the shortest paths from each vertex to v_1 that pass through 1 other vertex
  - Let's calculate the length of the shortest path from v_3 to v_1 that passes through v_2

$D[v_3][\{v_2\}] =$

$$\underset{j:v_j \in \{v_2\}}{minimum} \left(W[3][j] + D[v_j][\{v_2\} - \{v_j\}]\right)$$

Traveling Salesperson Problem

Step One

$D[v_2][\emptyset] = 1$ $\quad$ $D[v_3][\emptyset] = \infty$ $\quad$ $D[v_4][\emptyset] = 6$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Step Two

- Determine the lengths of the shortest paths from each vertex to v_1 that pass through 1 other vertex
  - Let's calculate the length of the shortest path from v_3 to v_1 that passes through v_2

$$D[v_3][\{v_2\}] = \underset{j:v_j \in \{v_2\}}{minimum} \left(W[3][j] + D[v_j][\{v_2\} - \{v_j\}]\right)$$

$$= W[3][2] + D[v_2][\emptyset] = 7 + 1 = 8$$

Traveling Salesperson Problem

Step Two

Calculating the rest, we get:

- $D[v_4][\{v_2\}] = 3 + 1 = 4$
- $D[v_2][\{v_3\}] = 6 + \infty = \infty$
- $D[v_4][\{v_3\}] = \infty + \infty = \infty$
- $D[v_2][\{v_4\}] = 4 + 6 = 10$
- $D[v_3][\{v_4\}] = 8 + 6 = 14$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step Three

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step Three

- Determine the lengths of the shortest paths from each vertex v_i to v_1 that pass through 2 other vertices:

- D[v_4][{v_2, v_3}]
- D[v_3][{v_2, v_4}]
- D[v_2][{v_3, v_4}]

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step Three

- Determine the lengths of the shortest paths from each vertex v_i to v_1 that pass through 2 other vertices.
  - Let's calculate the length of the shortest path from v_4 to v_1 that passes through both v_2 and v_3

$$D[v_4][\{v_2, v_3\}] = \underset{j:v_j \in \{v_2, v_3\}}{minimum} (W[4][j] + D[v_j][\{v_2, v_3\} - \{v_j\}])$$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step Three

- Determine the lengths of the shortest paths from each vertex v_i to v_1 that pass through 2 other vertices.
  - Let's calculate the length of the shortest path from v_4 to v_1 that passes through both v_2 and v_3

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$D[v_4][\{v_2, v_3\}] =$

$$\underset{j: v_j \in \{v_2, v_3\}}{\textit{minimum}} (W[4][j] + D[v_j][\{v_2, v_3\} - \{v_j\}])$$

$= \mathit{min}\ (W[4][2] + D[v_2][\{v_3\}],\ W[4][3] + D[v_3][\{v_2\}])$

Traveling Salesperson Problem

Step Three

$D[v_4][\{v_2, v_3\}] =$

$$\underset{j:v_j \in \{v_2, v_3\}}{minimum} \; (W[4][j] + D[v_j][\{v_2, v_3\} - \{v_j\}])$$

$= min\ (W[4][2] + D[v_2][\{v_3\}],\ W[4][3] + D[v_3][\{v_2\}])$

$= min\ (3 + \infty, \infty + 8) = \infty$

Note: $D[v_2][\{v_3\}]$ and $D[v_3][\{v_2\}]$ were calculated in step two.

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step Three

Calculating the rest, we get:

- $D[v_3][\{v_2, v_4\}] = min\ (7 + 10, 8 + 4) = 12$
- $D[v_2][\{v_3, v_4\}] = min\ (6 + 14, 4 + \infty) = 20$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step Four

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step Four

- In this final step, we start from vertex 1
  - i.e. We determine the shortest path from v_1 to v_1 that passes through every other vertex

$$\underset{j:v_j \in \{v_2, v_3, v_4\}}{minimum} (W[1][j] + D[v_j][\{v_2, v_3, v_4\} - \{v_j\}])$$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

Step Four

$$\underset{j:v_j \in \{v_2, v_3, v_4\}}{\textit{minimum}} \; (W[1][j] + D[v_j][\{v_2, v_3, v_4\} - \{v_j\}])$$

$$= \textit{min}\ (W[1][2] + D[v_2][\{v_3, v_4\}],$$
$$W[1][3] + D[v_3][\{v_2, v_4\}],$$
$$W[1][4] + D[v_4][\{v_2, v_3\}])$$

$$= \textit{min}\ (2 + 20, 9 + 12, \infty + \infty) = 21$$

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

Traveling Salesperson Problem

- The dynamic programming solution to the traveling salesperson problem is $\Theta(n^2 2^n)$
  - While this is better than factorial, it's still extremely bad.
  - However, a bad algorithm is better than a terrible one!

Suppose two employees are competing for the same sales position. Their boss tells them that whoever covers a 20-city territory faster gets the position.

- One uses a brute-force to determine a route, the other uses dynamic programming. If their computers perform the basic operation in 1 microsecond:
  - Brute-force algorithm: 19! microseconds = 3857 years
  - Dynamic Programming Algorithm: $(20 - 1)(20 - 2)2^{20-3}$ = 45 seconds

With a very small n, even exponential algorithms can sometimes be useful.

In-Class Exercise

Find an optimal circuit for the weighted, directed graph represented by the following matrix. Show the entries in the D array for each step.

	1	2	3	4
1	0	8	13	18
2	3	0	7	8
3	4	11	0	10
4	6	6	7	0