# The Traveling Salesman Problem – Brute Force Method

Lecture 30
Sections 6.1, 6.3

Robb T. Koether

Hampden-Sydney College

Fri, Nov 3, 2017

# Outline

# The Traveling Salesman Problem

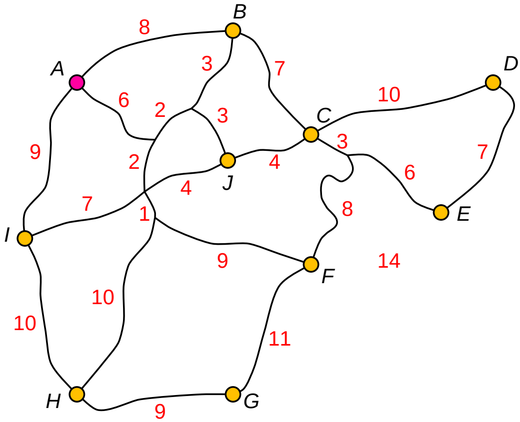
**Definition (Traveling Salesman Problem)**

The Traveling Salesman Problem is to find the *circuit* that visits *every* vertex (at least once) and *minimizes* the total weight of its edges.

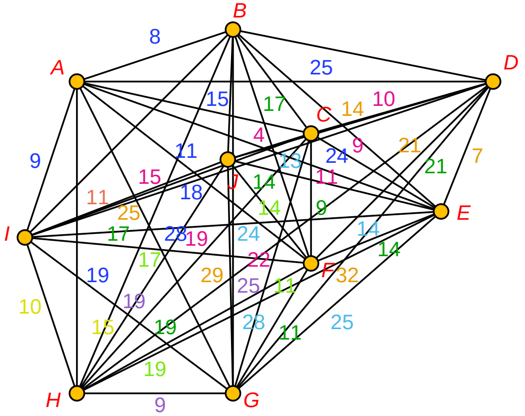
# The Traveling Salesman Problem

- The Traveling Salesman Problem could also be called the UPS Deliveryman Problem.
- There is a weight (or cost) to each edge of the graph.
- The weight could be expressed as
  - Distance – Find the shortest circuit.
  - Time – Find the fastest circuit.
  - Dollars (fuel, pay) – Find the least expensive circuit.

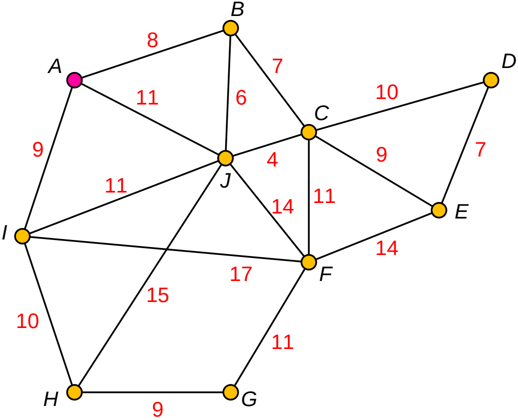
# The Traveling Salesman Problem



# The Traveling Salesman Problem



# The Traveling Salesman Problem



# The Traveling Salesman Problem

| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $G$ | $H$ | $I$ | $J$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $A$ | - | 8 | 15 | 25 | 24 | 25 | 28 | 19 | 9 | 11 |
| $B$ | 8 | - | 7 | 17 | 16 | 18 | 29 | 21 | 17 | 6 |
| $C$ | 15 | 7 | - | 10 | 9 | 11 | 22 | 19 | 15 | 4 |
| $D$ | 25 | 17 | 10 | - | 7 | 21 | 32 | 29 | 25 | 14 |
| $E$ | 24 | 16 | 9 | 7 | - | 14 | 25 | 28 | 24 | 13 |
| $F$ | 25 | 18 | 11 | 21 | 14 | - | 11 | 20 | 17 | 14 |
| $G$ | 28 | 29 | 22 | 32 | 25 | 11 | - | 9 | 19 | 25 |
| $H$ | 19 | 21 | 19 | 29 | 28 | 20 | 9 | - | 10 | 15 |
| $I$ | 9 | 17 | 15 | 25 | 24 | 17 | 19 | 10 | - | 11 |
| $J$ | 11 | 6 | 4 | 14 | 13 | 14 | 25 | 15 | 11 | - |

# Outline

# The Brute-Force Algorithm

**Definition (Brute-Force Algorithm)**

A brute-force algorithm is an algorithm that tries exhaustively every possibility, and then chooses the best one.

# The Brute-Force Algorithm

## Definition (Brute-Force Algorithm)

A brute-force algorithm is an algorithm that tries exhaustively every possibility, and then chooses the best one.

- If there are $n$ cities, then there are $(n-1)!$ possible circuits.

# The Brute-Force Algorithm

**Definition (Brute-Force Algorithm)**

A brute-force algorithm is an algorithm that tries exhaustively every possibility, and then chooses the best one.

- If there are $n$ cities, then there are $(n-1)!$ possible circuits.
- That is, $n-1$ choices for the first city.

# The Brute-Force Algorithm

## Definition (Brute-Force Algorithm)

A brute-force algorithm is an algorithm that tries exhaustively every possibility, and then chooses the best one.

- If there are $n$ cities, then there are $(n-1)!$ possible circuits.
- That is, $n-1$ choices for the first city.
- Followed by $n-2$ choices for the second city.

# The Brute-Force Algorithm

## Definition (Brute-Force Algorithm)

A brute-force algorithm is an algorithm that tries exhaustively every possibility, and then chooses the best one.

- If there are $n$ cities, then there are $(n-1)!$ possible circuits.
- That is, $n-1$ choices for the first city.
- Followed by $n-2$ choices for the second city.
- Followed by $n-3$ choices for the third city.

# The Brute-Force Algorithm

## Definition (Brute-Force Algorithm)

A brute-force algorithm is an algorithm that tries exhaustively every possibility, and then chooses the best one.

- If there are $n$ cities, then there are $(n-1)!$ possible circuits.
- That is, $n-1$ choices for the first city.
- Followed by $n-2$ choices for the second city.
- Followed by $n-3$ choices for the third city.
- And so on, until only 1 choice for the last city.

# The Brute-Force Algorithm

## Definition (Brute-Force Algorithm)

A brute-force algorithm is an algorithm that tries exhaustively every possibility, and then chooses the best one.

- If there are $n$ cities, then there are $(n-1)!$ possible circuits.
- That is, $n-1$ choices for the first city.
- Followed by $n-2$ choices for the second city.
- Followed by $n-3$ choices for the third city.
- And so on, until only 1 choice for the last city.
- Altogether

$$(n-1)(n-2)(n-3)\cdots 3\cdot 2\cdot 1=(n-1)!$$

choices.

# The Brute-Force Algorithm

- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were

# The Brute-Force Algorithm

- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities?

# The Brute-Force Algorithm

- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities?
  - 15 cities?

- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities?
  - 15 cities?
  - 20 cities?

# The Brute-Force Algorithm

- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities?
  - 15 cities?
  - 20 cities?
  - 25 cities?

# The Brute-Force Algorithm

- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities?
  - 15 cities?
  - 20 cities?
  - 25 cities?
  - 30 cities?

# The Brute-Force Algorithm

- Clearly, the brute-force algorithm is not adequate to solve the Traveling Salesman Problem.

# The Brute-Force Algorithm

- Clearly, the brute-force algorithm is not adequate to solve the Traveling Salesman Problem.
- What is the UPS driver to do?

# Outline

# Assignment

## Assignment

- Chapter 6: Exercises 27, 28, 29, 31, 33.