The Traveling Salesman Problem – Brute Force Method

Lecture 30
Sections 6.1, 6.3

Robb T. Koether

Hampden-Sydney College

Wed, Apr 4, 2018

Outline

The Traveling Salesman Problem

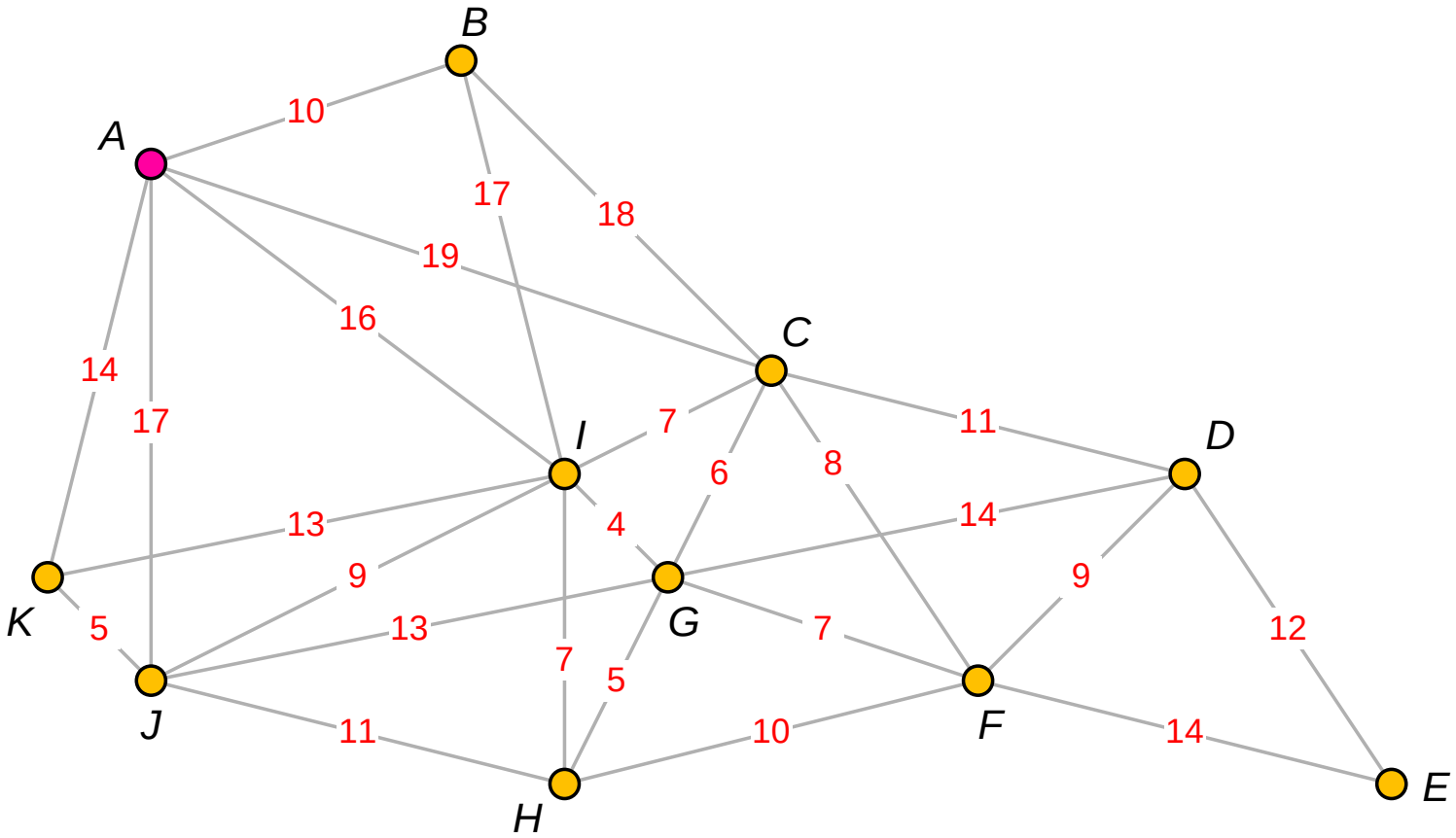
Definition (Traveling Salesman Problem)

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

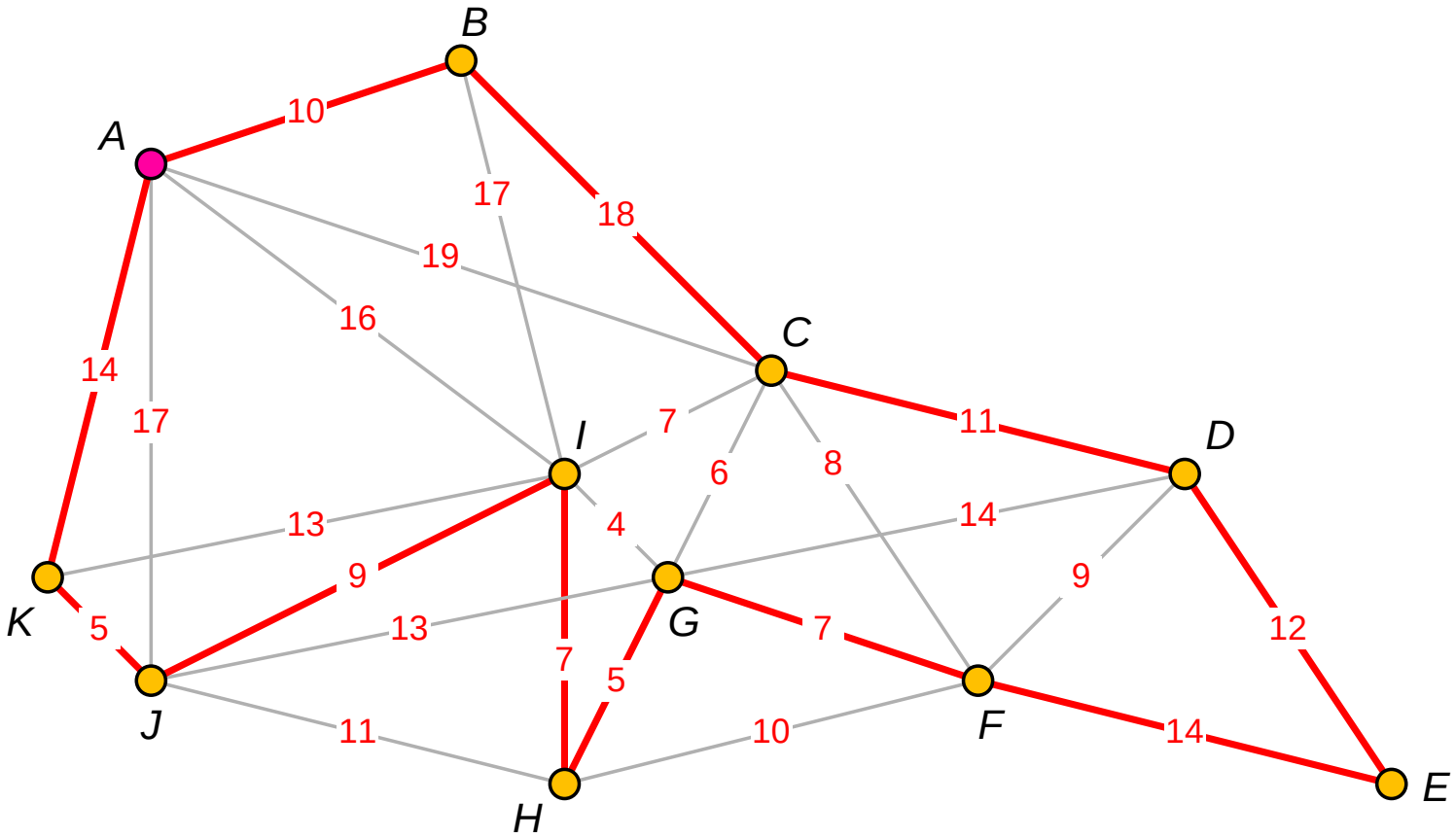
- The Traveling Salesman Problem could also be called the UPS Deliveryman Problem.
- There is a weight (or cost) to each edge of the graph.
- The weight could be expressed as
  - Distance – Find the shortest circuit.
  - Time – Find the fastest circuit.
  - Dollars (fuel, pay) – Find the least expensive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

The Traveling Salesman Problem



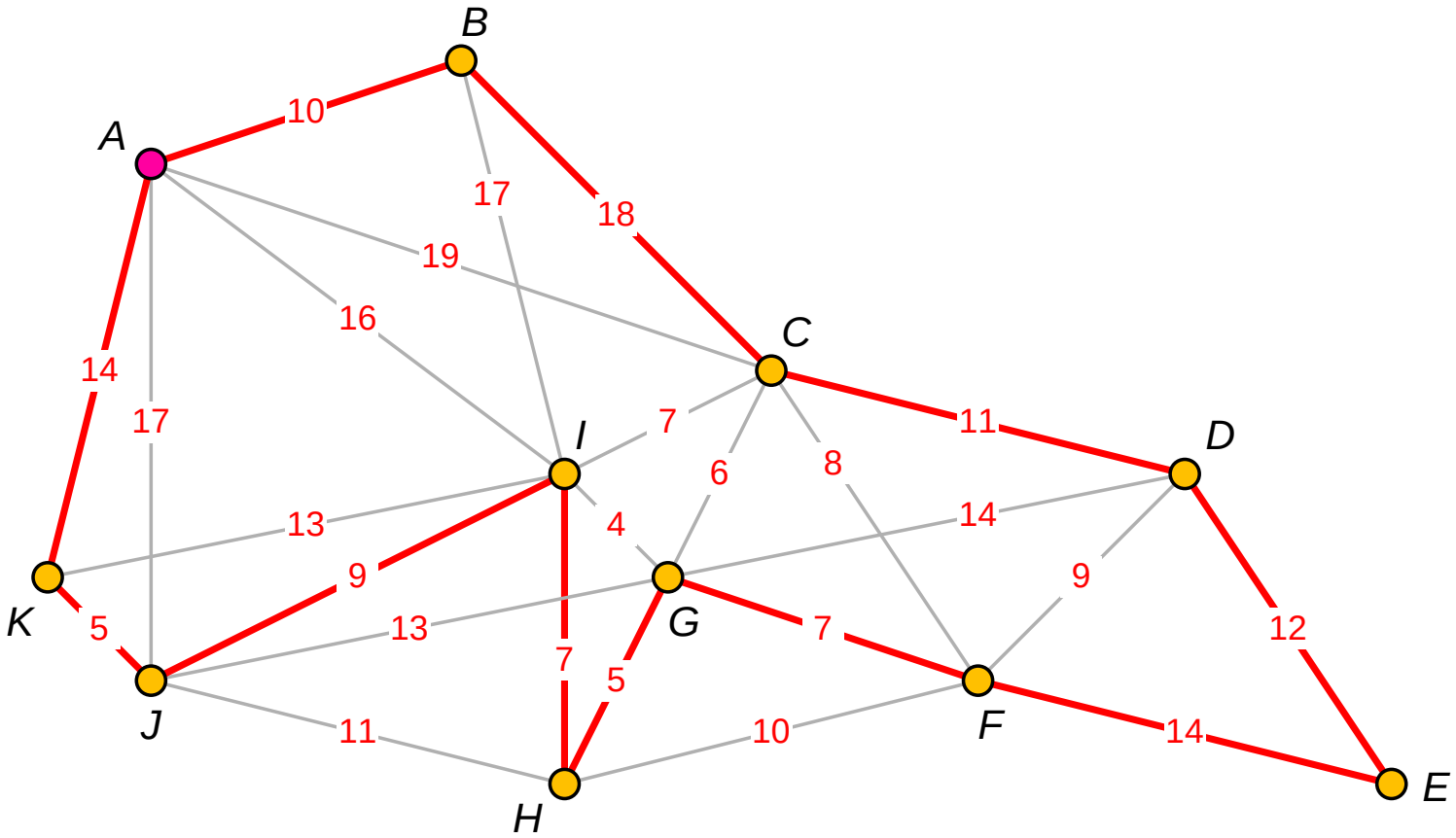
What is the shortest circuit through all cities?

The Traveling Salesman Problem



Shortest circuit is *ABCDEFGHIJKA*

The Traveling Salesman Problem



The length is 112

The Traveling Salesman Problem

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*
A	-	10	19	30	41	27	20	23	16	17	14
B	10	-	18	29	40	26	21	24	17	26	24
C	19	18	-	11	22	8	6	11	7	16	20
D	30	29	11	-	12	9	14	19	18	27	31
E	41	40	22	12	-	14	21	24	25	34	38
F	27	26	8	9	14	-	7	10	11	20	24
G	20	21	6	14	21	7	-	5	4	13	17
H	23	24	11	19	24	10	5	-	7	11	16
I	16	17	7	18	25	11	4	7	-	9	13
J	17	26	16	27	34	20	13	11	9	-	5
K	14	24	20	31	38	24	17	16	13	5	-

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? 0.00036 sec

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? 0.00036 sec
  - 15 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? 0.00036 sec
  - 15 cities? 1 1/2 mins

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? 0.00036 sec
  - 15 cities? 1 1/2 mins
  - 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? 0.00036 sec
  - 15 cities? 1 1/2 mins
  - 20 cities? 3.8 yrs

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? 0.00036 sec
  - 15 cities? 1 1/2 mins
  - 20 cities? 3.8 yrs
  - 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? 0.00036 sec
  - 15 cities? 1 1/2 mins
  - 20 cities? 3.8 yrs
  - 25 cities? 20 million yrs

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? 0.00036 sec
  - 15 cities? 1 1/2 mins
  - 20 cities? 3.8 yrs
  - 25 cities? 20 million yrs
  - 30 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? 0.00036 sec
  - 15 cities? 1 1/2 mins
  - 20 cities? 3.8 yrs
  - 25 cities? 20 million yrs
  - 30 cities? 280,000 billion yrs

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?

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- 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

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec
  - 20 cities?

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec
  - 20 cities? 0.42 sec

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec
  - 20 cities? 0.42 sec
  - 30 cities?

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec
  - 20 cities? 0.42 sec
  - 30 cities? 16 min

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec
  - 20 cities? 0.42 sec
  - 30 cities? 16 min
  - 40 cities?

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec
  - 20 cities? 0.42 sec
  - 30 cities? 16 min
  - 40 cities? 20 days

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec
  - 20 cities? 0.42 sec
  - 30 cities? 16 min
  - 40 cities? 20 days
  - 50 cities?

The Brute-Force Algorithm

- There are complex algorithms that can solve the problem in time proportional to $n^2 \cdot 2^n$, which is a lot faster than $n!$.
- If a computer could process 1,000,000,000 possibilities per second, how long would it take if there were
  - 10 cities? 0.0001 sec
  - 20 cities? 0.42 sec
  - 30 cities? 16 min
  - 40 cities? 20 days
  - 50 cities? 89 yrs

Outline

Assignment

Assignment

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