1 The Traveling Salesman Problem

2 The Brute-Force Algorithm

3 Assignment
1. The Traveling Salesman Problem

2. The Brute-Force Algorithm

3. Assignment
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 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

A
B
C
D
E
F
G
H
I
J

8
3
7
6
2
3
10
1
4
3
7
6
8
14
9
3
4
2
10
11
9

Robb T. Koether (Hampden-Sydney College) The Traveling Salesman Problem –Brute Force
The Traveling Salesman Problem

The Traveling Salesman Problem – Brute Force Method

Fri, Nov 3, 2017
The Traveling Salesman Problem

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

# The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a problem in operations research and computer science. It asks: given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? The problem is NP-hard, meaning that as the number of cities increases, the problem becomes exponentially more difficult to solve.

## Brute Force Method

The brute force method involves generating all possible routes and selecting the shortest one. For a small number of cities, this is feasible, but as the number of cities increases, the number of possible routes grows factorially, making this method impractical for large instances.

### Example Table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>8</td>
<td>15</td>
<td>25</td>
<td>24</td>
<td>25</td>
<td>28</td>
<td>19</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>B</td>
<td>8</td>
<td>-</td>
<td>7</td>
<td>17</td>
<td>16</td>
<td>18</td>
<td>29</td>
<td>21</td>
<td>17</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>15</td>
<td>7</td>
<td>-</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td>22</td>
<td>19</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>25</td>
<td>17</td>
<td>10</td>
<td>-</td>
<td>7</td>
<td>21</td>
<td>32</td>
<td>29</td>
<td>25</td>
<td>14</td>
</tr>
<tr>
<td>E</td>
<td>24</td>
<td>16</td>
<td>9</td>
<td>7</td>
<td>-</td>
<td>14</td>
<td>25</td>
<td>28</td>
<td>24</td>
<td>13</td>
</tr>
<tr>
<td>F</td>
<td>25</td>
<td>18</td>
<td>11</td>
<td>21</td>
<td>14</td>
<td>-</td>
<td>11</td>
<td>20</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>G</td>
<td>28</td>
<td>29</td>
<td>22</td>
<td>32</td>
<td>25</td>
<td>11</td>
<td>-</td>
<td>9</td>
<td>19</td>
<td>25</td>
</tr>
<tr>
<td>H</td>
<td>19</td>
<td>21</td>
<td>19</td>
<td>29</td>
<td>28</td>
<td>20</td>
<td>9</td>
<td>-</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>I</td>
<td>9</td>
<td>17</td>
<td>15</td>
<td>25</td>
<td>24</td>
<td>17</td>
<td>19</td>
<td>10</td>
<td>-</td>
<td>11</td>
</tr>
<tr>
<td>J</td>
<td>11</td>
<td>6</td>
<td>4</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>25</td>
<td>15</td>
<td>11</td>
<td>-</td>
</tr>
</tbody>
</table>
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 \((n-1)!(n-2)!(n-3)\cdot\ldots\cdot3\cdot2\cdot1 = (n-1)!\) choices.
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.
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?
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?
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?
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?
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?
1 The Traveling Salesman Problem

2 The Brute-Force Algorithm

3 Assignment
Assignment

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