ASSIGNMENT 9 TIPS AND TRICKS

- traveling salesperson problem
- nearest insertion heuristic
- smallest increase heuristic
- implementation
- beyond

http://princeton.edu/~cos126
Goals

- Implement a data structure using **linked lists**.
- Analyze the running time of a program.
- Explore a notoriously difficult problem (see Intractability lecture).
ASSIGNMENT 9 TIPS AND TRICKS

• traveling salesperson problem
  • nearest insertion heuristic
  • smallest increase heuristic
  • implementation
  • beyond
Euclidean TSP

Given \( n \) points in the plane, find a tour of minimum length that visits them all exactly once.
USA cities

13,509 cities in the United States
http://www.tsp.gatech.edu
General TSP

Given \( n \) points and pairwise “distances”, find a tour of minimum length that visits them all.

**Distances could represent:**
- Costs.
- Travel times.
- Fuel consumed.
- Some more abstract quantity.
- …
Applications

Traveling salespeople? Probably not.
Applications

**VLSI design.** Drill holes in a printed circuit board.
Applications

**VLSI design.** Drill holes in a printed circuit board.
Applications

**VLSI design.** Drill holes in a printed circuit board.

**Space telescope.** Re-position satellite to image celestial objects.

---

**Fuel-Saving Strategies for Dual Spacecraft Interferometry Missions**

Christopher A. Bailey, Timothy W. McLain, and Randal W. Beard

Abstract

Separated spacecraft interferometry missions will require that spacecraft move in a coordinated fashion to ensure minimal and balanced consumption of fuel. This paper develops strategies for determining interferometry mission plans that result in significant fuel savings over standard approaches. Simulation results demonstrate that valuable reductions in fuel consumption can be realized by combining the retargeting and imaging maneuvers required to image multiple stellar sources. Fuel-optimal imaging strategies have been developed for two-spacecraft interferometry missions similar to the proposed StarLight mission using chained local optimization methods. Based on these strategies, sampling-pattern guidelines for space-borne interferometry missions have been developed.

---

**Optimization and the Traveling Salesman Problem**

Charles A. Whitney  Harvard University
Applications

**VLSI design.** Drill holes in a printed circuit board.

**Space telescope.** Re-position satellite to image celestial objects.

**Computational biology.** Map the mouse genome.

---

**A radiation hybrid transcript map of the mouse genome**

Philip Arner¹,², Thomas Brule³, Isabelle Porras³, Lorraine Eley³, Shahnaz Gas³, Patricia Ruiz³, Michael V. Wiles³,⁴,⁵, Rita Sousa-Nunes⁶, Ross Kettleborough⁶, Amer Rana⁶, Jean Morissette⁶, Liz Bentley⁶, Michelle Goldsworthy⁶, Alison Haynes⁶, Efison Herbert⁶, Lorraine Southam⁶, Hans Lehrach⁶, Jean Weissenbach⁶, Giacomo Manenti⁷, Patricia Rodriguez-Tome⁷,⁸, Rosa Beddington*, Sally Dunwoodie⁷,⁸ & Roger D. Cox⁹

---
Applications

**VLSI design.** Drill holes in a printed circuit board.

**Space telescope.** Re-position satellite to image celestial objects.

**Computational biology.** Map the mouse genome.

**Combinatorial optimization.** Training ground for new techniques.

*seminal problem in operations research*
Brute-force algorithm

**Easy?** Try all possible tours; pick best one.

- ABCDE
- ABCED
- ABDCE
- ABDEC
- ABEDC
- ACBDE
- ACBED
- ACDBE
- ACEBD
- ADBCE
- ADCBE
**Brute-force algorithm**

Estimate how many possible tours among \( n = 1,000 \) cities?

A. \( 181,440 \)

B. \( 1,000,000 \)

C. \( 60,822,550,204,416,000 \)

D. \( 2.01 \times 10^{2,564} \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( # \text{ tours} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>181,440</td>
</tr>
<tr>
<td>20</td>
<td>60,822,550,204,416,000</td>
</tr>
<tr>
<td>50</td>
<td>( 3.04 \times 10^{62} )</td>
</tr>
<tr>
<td>100</td>
<td>( 4.66 \times 10^{155} )</td>
</tr>
<tr>
<td>1,000</td>
<td>( 2.01 \times 10^{2,564} )</td>
</tr>
</tbody>
</table>
Brute-force algorithm

Easy? Try all possible tours; pick best one.

Hard? Given $n$ points, number of possible tours is $n \times (n-1) \times (n-2) \times \cdots \times 1$
Assignment 9 Tips and Tricks

- traveling salesperson problem
- nearest insertion heuristic
- smallest increase heuristic
- implementation
- beyond
Nearest insertion heuristic

A

B

C

D

850.00

939.41

494.97
Nearest insertion heuristic
Nearest insertion heuristic
Nearest insertion heuristic
Nearest insertion heuristic

Q. Does nearest insertion heuristic guarantee to produce shortest tour?
A. No.

nearest insertion tour length = 2947.47
optimal tour length = 2512.09
ASSIGNMENT 9 TIPS AND TRICKS

- traveling salesman problem
- nearest insertion heuristic
- smallest increase heuristic
- implementation
- beyond
Smallest increase heuristic
Smallest increase heuristic

length = 2320.18
Smallest increase heuristic

2320.18
best

length = 2521.56
Smallest increase heuristic

2320.18
best

length = 2418.08
Smallest increase heuristic

length = 2595.19

2320.18

best
Smallest increase heuristic

2320.18

best
Smallest increase heuristic
Smallest increase heuristic

best
3005.58

length = 3005.58
Smallest increase heuristic

length = 2745.50

best
3005.58

2745.50
Smallest increase heuristic

length = 2691.32

best  2745.50
Smallest increase heuristic

length = 3459.86

best = 2691.32
Smallest increase heuristic

best

2691.32
Smallest increase heuristic
Q. Does smallest increase heuristic guarantee to produce shortest tour?
A. No.

smallest increase tour length = 2691.31

optimal tour length = 2254.11
ASSIGNMENT 9 TIPS AND TRICKS

- traveling salesperson problem
- nearest insertion heuristic
- smallest increase heuristic
- implementation
- beyond
**Point data type**

You will **not** write or submit this file.

```java
public class Point {
    public Point(double x, double y) {
        // creates the point (x, y)
    }

    public double distanceTo(Point that) {
        // Euclidean distance between the two points
    }

    public int drawTo(Point that) {
        // draws the line segment between the two points
    }

    public String toString() {
        // string representation of this point
    }
}
```
public class Tour

public Tour() creates an empty tour

public Tour(Point a, Point b, … ) creates a 4-point tour a→b→c→d→a

public int size() number of points in this tour

public double length() length of this tour

public void draw() draws this tour to standard drawing

public String toString() string representation of this tour

public void insertNearest(Point p) inserts the point p into tour using nearest insertion heuristic

public void insertSmallest(Point p) inserts the point p into the tour using the smallest increase heuristic
Circularly linked lists

Node data type.

```java
private class Node {
    private Point p;
    private Node next;
}
```

Visual representation.

```
(x0, y0)  (x1, y1)  (x2, y2)  (x4, y4)  (x3, y3)
```

First
Traversing a circularly linked list

Which of the following prints every node in a circularly linked list?

A. 
Node x = first;
while (x != first) {
    StdOut.println(x.p);
    first = first.next;
}

B. 
Node x = first;
while (x.next != first) {
    StdOut.println(x.p);
    x = x.next;
}

C. 
for (Node x = first; x != null; x = x.next) {
    StdOut.println(x.p);
}

D. 
Node x = first;
do {
    StdOut.println(x.p);
    x = x.next;
} while (x != first);
Traversing a circularly linked list

Which of the following prints every node in a circularly linked list?

A. Node x = first;
   while (x != first) {
      StdOut.println(x.p);
      first = first.next;
   }

B. Node x = first;
   while (x.next != first) {
      StdOut.println(x.p);
      x = x.next;
   }

C. for (Node x = first; x != null; x = x.next) {
   StdOut.println(x.p);
   }

D. Node x = first;
   do {
      StdOut.println(x.p);
      x = x.next;
   } while (x != first);
Traversing a circularly linked list

Node x = first;
    do {
        StdOut.println(x.p);
        x = x.next;
    } while (x != first);

standard output
(x0, y0)
(x1, y1)
(x2, y2)
(x4, y4)
(x3, y3)
Inserting a node into a circularly linked list

```plaintext
first

(x₀, y₀) -> (x₁, y₁) -> (x₂, y₂) -> (x₃, y₄) -> (x₅, y₅) -> first

newNode

(x₂, y₅) -> newNode
```

x

best

Smallest increase heuristic: performance trick

**Bottleneck.** Computing the tour length.

**Impact.** Requires \( n \) calls to `distanceTo()`.

\[
\text{length} = d_1 + d_2 + d_3 + d_4 + d_5 + d_6 + d_7
\]
Smallest increase heuristic: performance trick

Key optimization. Compute change in tour length; not tour length.

Impact. Only 3 calls to distanceTo() instead of \( n \).

\[ \Delta \text{length} = d_1 + d_2 - d_3 \]
Tips and tricks

Linked structures.
- Do not use a null-terminated linked list.
- You must use a circularly linked list.
- Use new Node() only to create new nodes (not new references to existing nodes).

Traversing the circularly linked list.
- Can use a for or while loop.
- Simpler with a do-while loop.  

Work incrementally. Constructors, size(), length(), toString(),

Test & Debug.  Print, print, print!

Dealing with ties. If tie, insert after first such point.

Corner cases.  (0- and 1-point tours).

String representation. Use StringBuilder in toString() for repeated string concatenation.
ASSIGNMENT 9 TIPS AND TRICKS

- traveling salesperson problem
- nearest insertion heuristic
- smallest increase heuristic
- implementation
- beyond
TSP art
TSP art
TSP books, apps, and movies