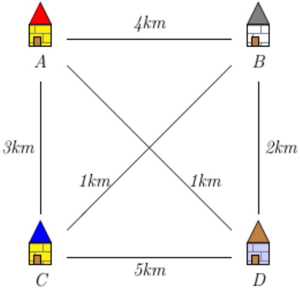
# Introduction — Problems!

Dr Kamal Bentahar

School of Computing, Electronics and Mathematics
Coventry University

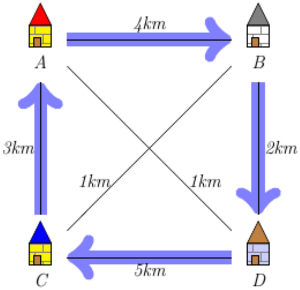
Week 1 – 22/01/2018

# Traveling Salesman Problem



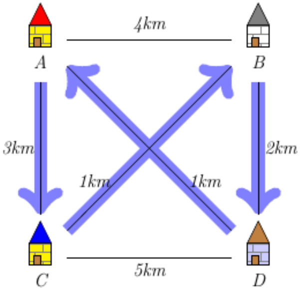

Shortest tour?

# Traveling Salesman Problem



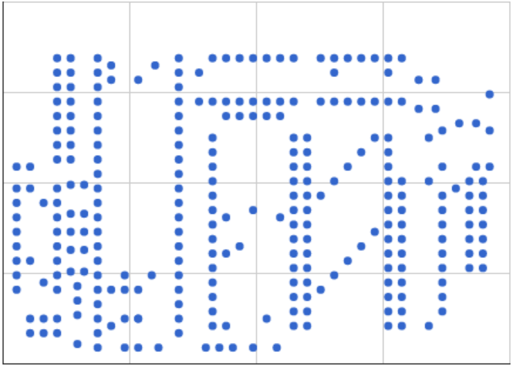

$$4 + 2 + 5 + 3 = 14$$

# Traveling Salesman Problem

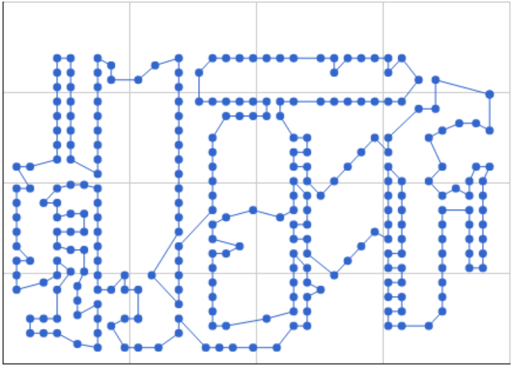



$3 + 1 + 2 + 1 = 7$

# Traveling Salesman Problem



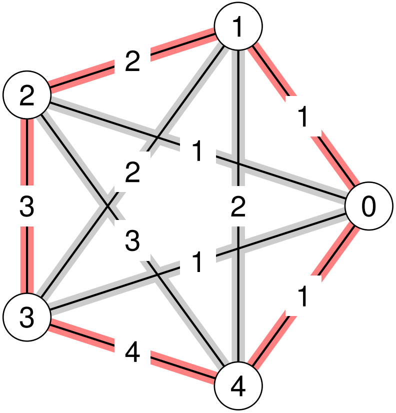
# Traveling Salesman Problem



# Traveling Salesman Problem

- One of the most famous problems in CS.
- Given a **list of cities** and the **distances between each pair of cities**, what is the shortest possible route that visits each city and returns to the origin city?
- **NP-hard** problem!

# Traveling Salesman Problem – Assignment example



# Traveling Salesman Problem – what is the issue?

| Number of cities $n$ | Number of paths (n-1)!/2 |
|---|---|
| 3 | 1 |
| 4 | 3 |
| 5 | 12 |
| 6 | 60 |
| 7 | 360 |
| 8 | $2,520$ |
| 9 | $20,160$ |
| 10 | $181,440$ |
| 15 | $43,589,145,600$ |
| 20 | $6.082 \times 10^{16}$ |
| 71 | $5.989 \times 10^{99}$ |

# Some more examples

## Problem (Cliques)

*Given a graph and an integer $n$, decide if it contains a clique with $k$ vertices.*

A clique in a graph is a set of vertices for which any two are connected.

## Problem (Subset-Sum Problem)

*Given a set $S = \{x_1, x_2, \ldots, x_n\}$ of integers, and an integer $t$ (called target) decide if there is a subset of $S$ whose sum is equal to $t$.*

## Problem (A Diophantine quadratic equation)

*Given three positive integers $a, b, c$, decide if the equation $ax^2 + by = c$ has a solution in positive integers.*

## Problem (Satisfiability)

*Given a Boolean expression, decide if there is a way of assigning the values true and false to the variables so that the expression is true.*

# History: What is computation?

Questions about this first arose in the context of pure Mathematics:

- Gottlob Frege (1848–1925)
- David Hilbert (1862–1943)
- George Cantor (1845–1918)
- Kurt Gödel (1906–1978)
- 1936:
  - Gödel and Stephen Kleene (1909-1994): **Partial Recursive Functions**
  - Gödel, Kleene and Jacques Herbrand (1908–1931)
  - Alonzo Church (1903–1995): **Lambda Calculus**
  - Alan Turing (1912–1954): **Turing Machine**
- 1943: Emil Post (1897–1954): **Post Systems**
- 1954: A.A. Markov: Theory of Algorithms – **Grammars**
- 1963: Shepherdson and Sturgis: **Universal Register Machines**

# A needle in a haystack — a search problem

**Problem:**

Given any (finite) haystack $H$, decide whether $H$ contains a needle.

Easy but tedious — **exhaustive search**: simply search every location within the haystack in some order and terminate answering **yes** if a needle is found. If search is completed with no needle found then terminate answering **no**.

This problem is a **decision problem**: given some data (the haystack) decide if the data has a certain property (needle containment). We may divide all possible instances of the problem into **yes-instances** and **no-instances** using our process.

# Types of problems

- Decision
- Search
- Computation/Construction
- Counting
- Optimization
- ...

**Important observation**

As far as “Can these problem be solved at all using computation?”, they can be converted to a **decision problem**.

# How hard is a problem?

Here are some example problems:

(1) What is $1 + 1$?

(2) What is the shortest route across the rail network from Coventry to London?

(3) What is the shortest tour around all the universities in the UK and back to your starting point (by car say)?

- How hard are they to solve?
- Why do we feel that (1) is “easier” than (2) and (3)?
- What about the last two? Is one much harder than the other, or are they both about the same?

# How can we measure the hardness of a problem?

- By the type/sophistication of the machine/process required to solve it?
  - Real physical machines
  - Theoretical (imaginary) machines
- By the amount of resources used by the machine?
  - Processor time
  - Memory space
- By the level of difficulty encountered by the (human) solver of the problem?

# Problems vs Problem Instances

**Problems:** Generalization of a problem instance.

- e.g. not interested in just $1 + 1$, but $x + y$ in general.

For a specific *problem instance*, we could measure exactly the amount of processor time and memory capacity required to solve it, using some suitable process.
However, when solving a general problem, we cannot always say exactly what resources will be used.

- We express resource usage as a function of the **instance's size.**
- When we ask questions about whether a problem is solvable by some machine, we allow the machine unlimited memory and time — all problems become unsolvable at some point if these are finite.
- It is for this reason, among others, that theoretical machines are used in classifying hardness.

## Problems vs Problem Instances

**Problems:** Generalization of a problem instance.

- e.g. not interested in just $1 + 1$, but $x + y$ in general.

For a specific *problem instance*, we could measure exactly the amount of processor time and memory capacity required to solve it, using some suitable process.
However, when solving a general problem, we cannot always say exactly what resources will be used.

- We express resource usage as a function of the **instance's size**.
- When we ask questions about whether a problem is solvable by some machine, we allow the machine unlimited memory and time — all problems become unsolvable at some point if these are finite.
- It is for this reason, among others, that theoretical machines are used in classifying hardness.

The examples on slide 7 are not *problems*, but **instances of problems**.

(1) What is $1 + 1$?
$\rightarrow$ instance of the problem called *addition*,

(2) What is the shortest route across the rail network from Coventry to London?
$\rightarrow$ instance of the *shortest path problem*,

(3) What is the shortest tour around all the universities in the UK and back to your starting point (by car say)?
$\rightarrow$ instance of the *travelling salesman problem*.

# Problems and Processes

**Difficulty:** type of machine, time, and space required may depend on our choice of process used to solve the problem.

## Problem (Addition of integers)

*Given two integers $a, b$, find $s = a + b$.*

## Suggested solution:

```
do
    Select a random number s
while s − b ≠ a
return s
```

How does this method work? Is it correct? What is wrong with it?
The hardness of the problem should be taken to be the hardness of the most efficient process capable of solving it.

# O-notation scale – reminder

In increasing order:

- Constant: $O(1)$
- Polynomial: $O(n^k)$ for $k \geq 1$. $[O(n), O(n^2), \ldots]$
- Exponential $O(c^n)$ for $c > 1$. $[O(2^n), O(3^n), \ldots]$
- Factorial: $O(n!)$
- Combinatorial: $O(n^n)$

"Tricks":

$$n^k \log n \sim n^{k+\varepsilon} \qquad (\varepsilon \text{ small})$$

$$n^n > n! \quad \text{because } n^n = \underbrace{n \times n \times n \times \cdots \times n \times n \times n}$$
$$> n(n-1)(n-2)\cdots 3 \times 2 \times 1 = n!$$

# Big-O notation cheat-sheet

1. $1$ — constant, does not depend on $n$
2. $\log n$ — think of this as $n^{\varepsilon}$ for a "small" $\varepsilon$
3. $n$
4. $n \log n$ — think $n \times n^{\varepsilon} = n^{1+\varepsilon}$
5. $n^2$
6. $n^2 \log n$ — think $n^{2+\varepsilon}$
7. $n^3$
8. $2^n$
9. $3^n$
10. $n!$
11. $n^n$

# Complexity Onion