

Complexity

Johan Montelius

KTH

VT26

big-O

An estimate of the change in execution time...
when the data set grows large.

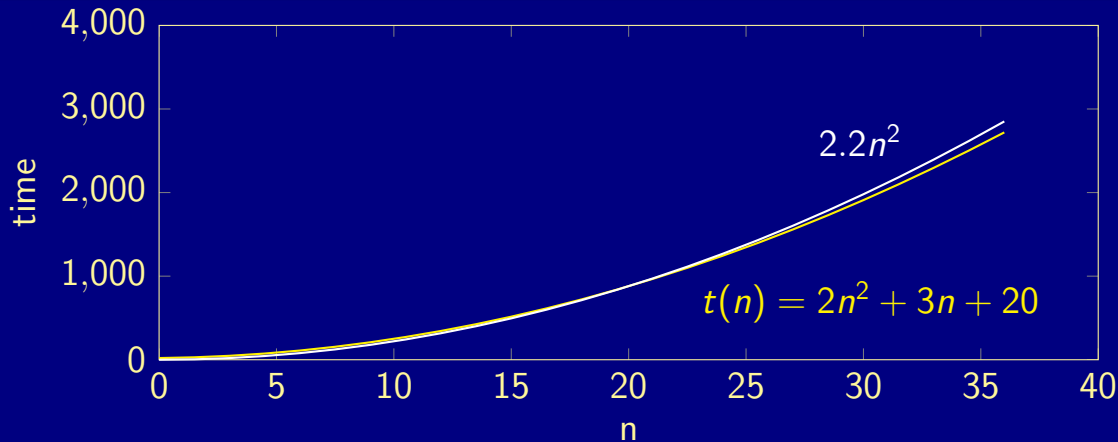
big-O

An estimate of the change in execution time...
when the data set grows large.

big-O

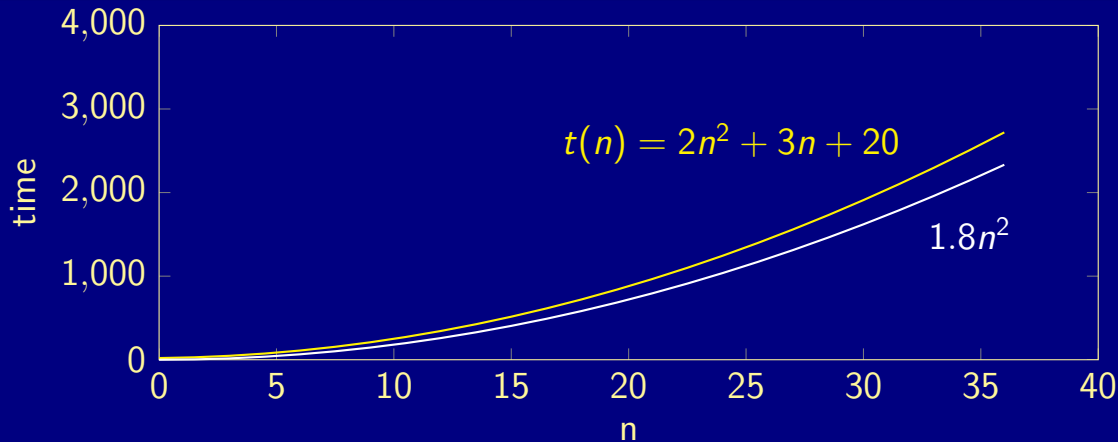
An estimate of the change in execution time...
when the data set grows large.

an upper bound



Ergo: $t(n)$ is in $O(n^2)$ since there is a k such that $k \times n^2 > t(n)$ above some n .

a lower bound



Ergo: $t(n)$ is in $\Omega(n^2)$ since there is a k such that $k \times n^2 < t(n)$ above some n .

Big-O, Ω and *Theta*

- A functions upper bound is limited by $O(g(n))$.
- Its lower bound is limited by $\Omega(g(n))$.
- If a function is limited by $O(g(n))$ and $\Omega(g(n))$ then it is limited by $\Theta(g(n))$.

Big-O, Ω and *Theta*

- A functions upper bound is limited by $O(g(n))$.
- Its lower bound is limited by $\Omega(g(n))$.
- If a function is limited by $O(g(n))$ and $\Omega(g(n))$ then it is limited by $\Theta(g(n))$.

Big-O, Ω and *Theta*

- A functions upper bound is limited by $O(g(n))$.
- Its lower bound is limited by $\Omega(g(n))$.
- If a function is limited by $O(g(n))$ and $\Omega(g(n))$ then it is limited by $\Theta(g(n))$.

Big-O, Ω and *Theta*

- A functions upper bound is limited by $O(g(n))$.
- Its lower bound is limited by $\Omega(g(n))$.
- If a function is limited by $O(g(n))$ and $\Omega(g(n))$ then it is limited by $\Theta(g(n))$.

order of ordo

Let's order the following functions:

order of ordo

Let's order the following functions:

$$0.1 \times n + \log_{10}(n)$$

$$5 \times \sqrt{n} + 34$$

$$\ln(n)^2 + 34$$

$$\log_2(n) + 30$$

$$n^4 + 20 \times n + 32$$

$$50 \times n + 100$$

$$20 \times n^2 + 100$$

$$n \times \log_{10}(n)$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$5 \times \sqrt{n} + 34$$

$$\ln(n)^2 + 34$$

$$\log_2(n) + 30$$

$$n^4 + 20 \times n + 32$$

$$50 \times n + 100$$

$$20 \times n^2 + 100$$

$$n \times \log_{10}(n)$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$\ln(n)^2 + 34$$

$$\log_2(n) + 30$$

$$n^4 + 20 \times n + 32$$

$$50 \times n + 100$$

$$20 \times n^2 + 100$$

$$n \times \log_{10}(n)$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$\log_2(n) + 30$$

$$n^4 + 20 \times n + 32$$

$$50 \times n + 100$$

$$20 \times n^2 + 100$$

$$n \times \log_{10}(n)$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$n^4 + 20 \times n + 32$$

$$50 \times n + 100$$

$$20 \times n^2 + 100$$

$$n \times \log_{10}(n)$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$O(n^4)$$

$$50 \times n + 100$$

$$20 \times n^2 + 100$$

$$n \times \log_{10}(n)$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$O(n^4)$$

$$O(n)$$

$$20 \times n^2 + 100$$

$$n \times \log_{10}(n)$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$O(n^4)$$

$$O(n)$$

$$O(n^2)$$

$$n \times \log_{10}(n)$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$O(n^4)$$

$$O(n)$$

$$O(n^2)$$

$$O(n \times \lg(n))$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$O(n^4)$$

$$O(n)$$

$$O(n^2)$$

$$O(n \times \lg(n))$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$O(n^4) \quad O(n^2)$$

$$O(n)$$

$$O(n \times \lg(n))$$

order of ordo

Let's order the following functions:

$$O(n)$$

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$O(n^4) \quad O(n^2) \quad O(n \times \lg(n))$$

$$O(n)$$

order of ordo

Let's order the following functions:

$$O(\sqrt{n})$$

$$O(\lg(n)^2)$$

$$O(\lg(n))$$

$$O(n^4) \quad O(n^2) \quad O(n \times \lg(n)) \quad O(n)$$

order of ordo

Let's order the following functions:

$$\begin{array}{ccccccc} & & O(\lg(n)^2) & & & & O(\lg(n)) \\ O(n^4) & O(n^2) & O(n \times \lg(n)) & O(n) & & & O(\sqrt{n}) \end{array}$$

order of ordo

Let's order the following functions:

$$O(\lg(n))$$

$$O(n^4) \quad O(n^2) \quad O(n \times \lg(n)) \quad O(n) \quad O(\sqrt{n}) \quad O(\lg(n)^2)$$

order of ordo

Let's order the following functions:

$$O(n^4) \quad O(n^2) \quad O(n \times \lg(n)) \quad O(n) \quad O(\sqrt{n}) \quad O(\lg(n)^2) \quad O(\lg(n))$$

order of ordo

Let's order the following functions:

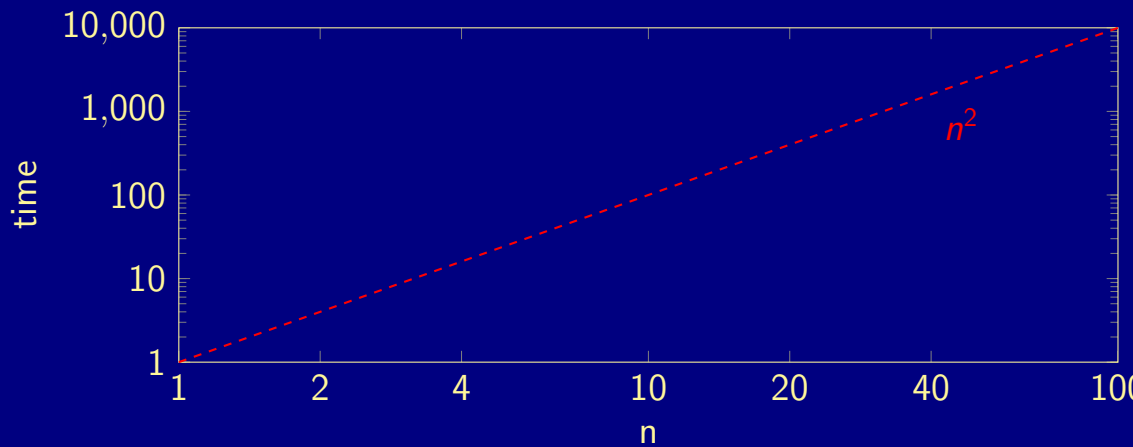
$$O(n^4) \quad O(n^2) \quad O(n \times \lg(n)) \quad O(n) \quad O(\sqrt{n}) \quad O(\lg(n)^2) \quad O(\lg(n))$$

orderd by complexity - not execution time given a specific n

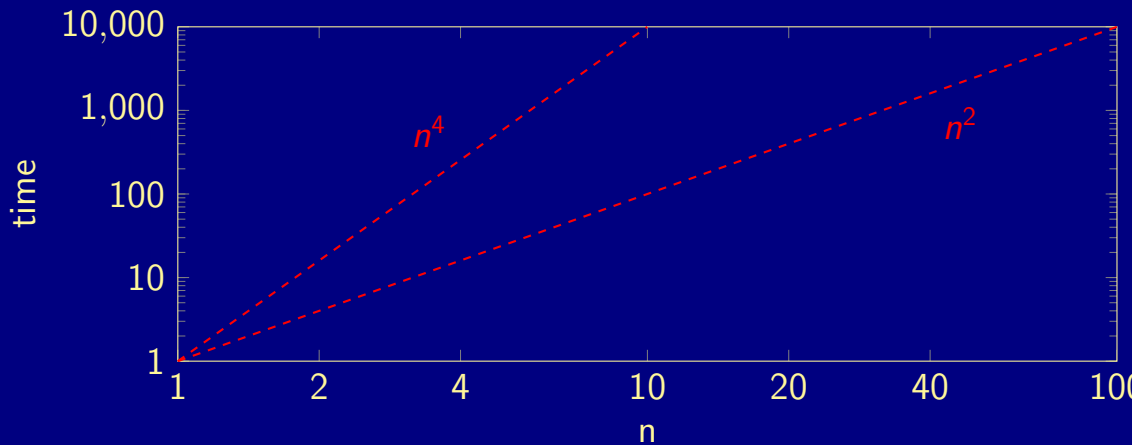
the cost of chess



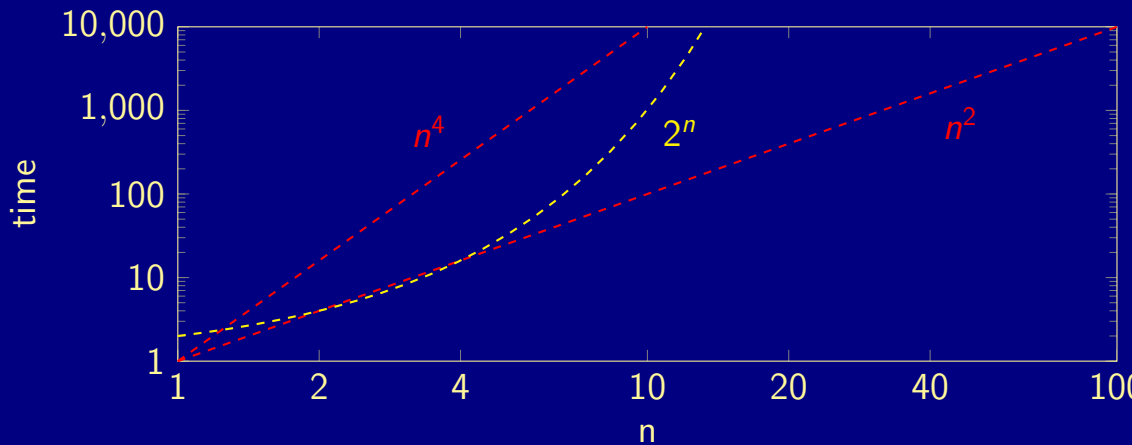
exponential time



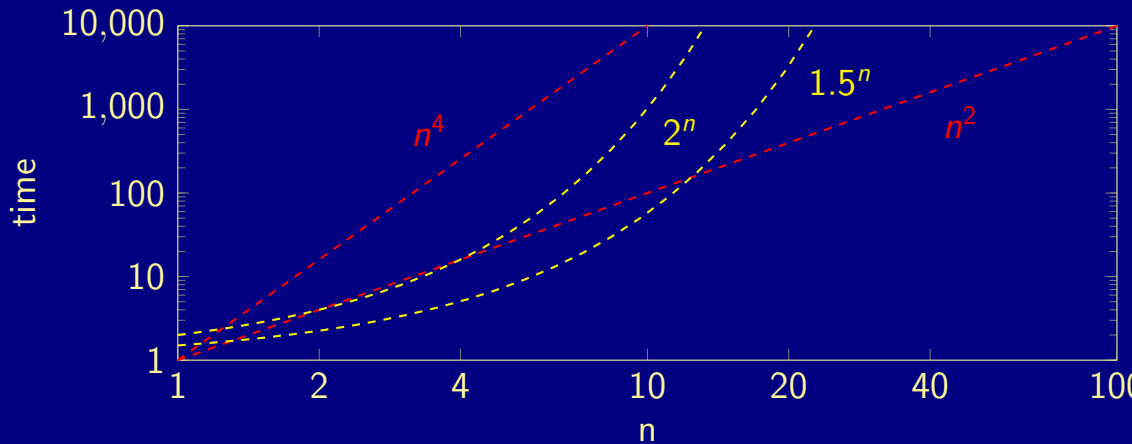
exponential time



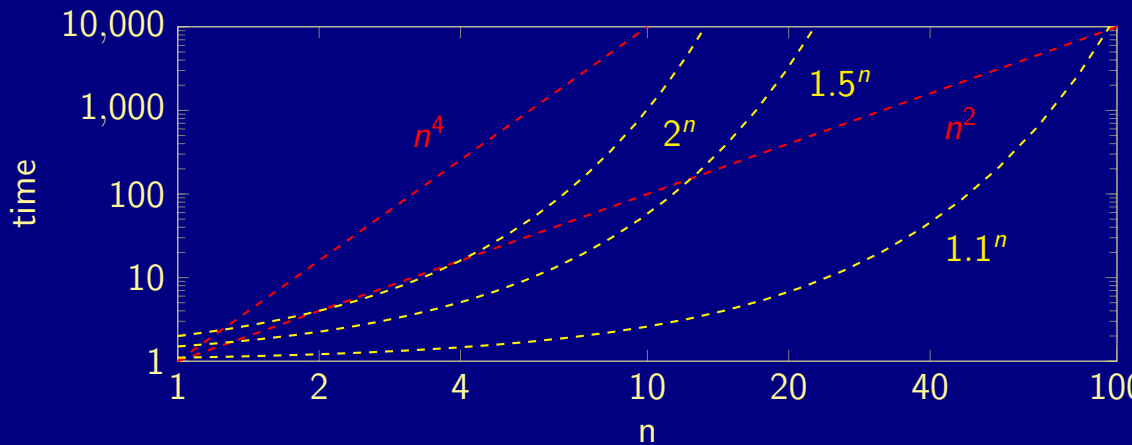
exponential time



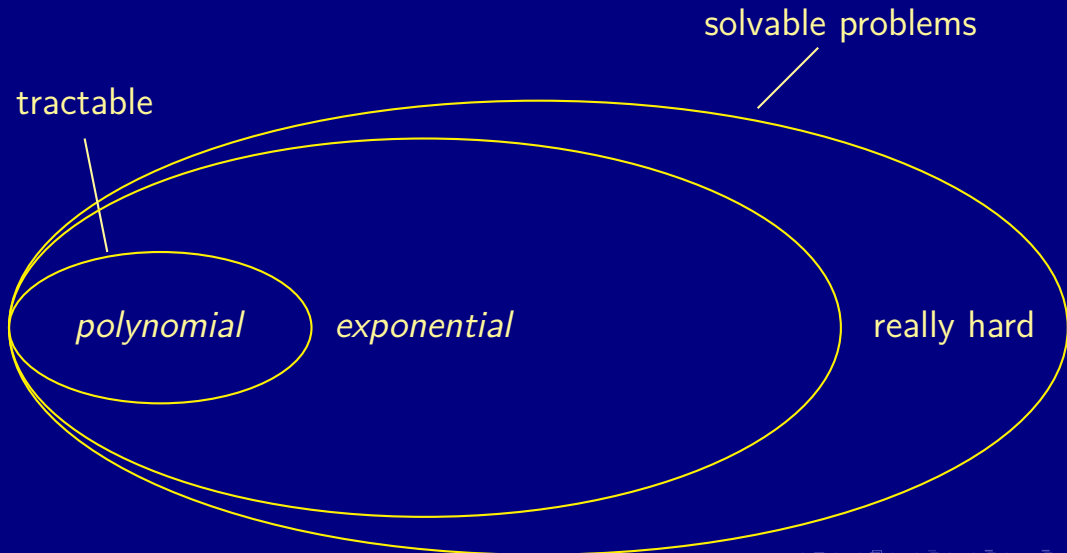
exponential time



exponential time



tractable, intractable and even worse



this sentence is false

this sentence is false



Kurt Gödel

*There are things
that can not be
decided.*

this sentence is false



Alonso Church
lambda calculus



Kurt Gödel
*There are things
that can not be
decided.*

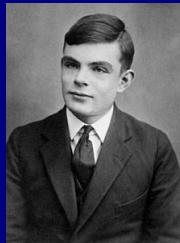
this sentence is false



Alonso Church
lambda calculus



Kurt Gödel
*There are things
that can not be
decided.*

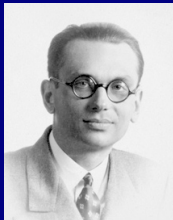


Alan Turing
the Turing machine

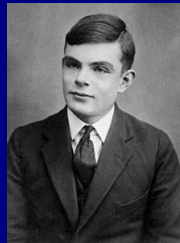
this sentence is false



Alonso Church
lambda calculus



Kurt Gödel
*There are things
that can not be
decided.*



Alan Turing
the Turing machine

Apart from things that can not be computed, we can compute everything :-)

The Imitation Game



the halting problem

Assume we have a program, `halt`, that can answer if a given *program* with its *input*, halts or ends up in an infinite loop i.e. given a program `p` and input `i`, `halt(p, i)` will terminate and return either *true* or *false*.

the halting problem

Assume we have a program, `halt`, that can answer if a given *program* with its *input*, halts or ends up in an infinite loop i.e. given a program `p` and input `i`, `halt(p, i)` will terminate and return either *true* or *false*.

Now create a program `wtf` that takes a program `p` as input:

```
bool wtf(prg p) {  
    if (halt(p,p)) {  
        loop();  
    } else {  
        return true;  
    }  
}
```

the halting problem

Assume we have a program, `halt`, that can answer if a given *program* with its *input*, halts or ends up in an infinite loop i.e. given a program `p` and input `i`, `halt(p, i)` will terminate and return either *true* or *false*.

Now create a program `wtf` that takes a program `p` as input:

```
bool wtf(prg p) {  
    if (halt(p,p)) {  
        loop();  
    } else {  
        return true;  
    }  
}
```

What will `wtf(wtf)` return?

problem complexity

problem complexity

It's not easy to determine the classification of a problem.

problem complexity

It's not easy to determine the classification of a problem.

If we have an algorithm at least we know the upper limit.

problem complexity

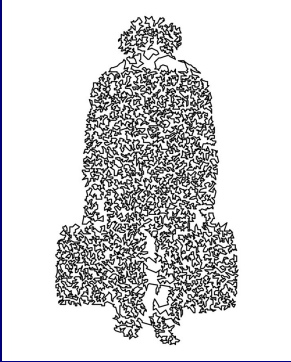
It's not easy to determine the classification of a problem.

If we have an algorithm at least we know the upper limit.

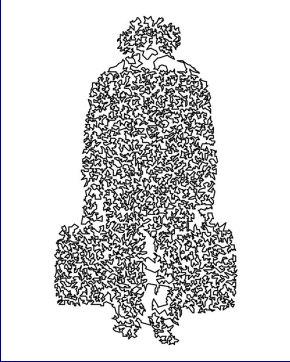
We have problems where it's easy to find an exponential algorithm, but no proof that there is no polynomial algorithm.

traveling salesman

traveling salesman

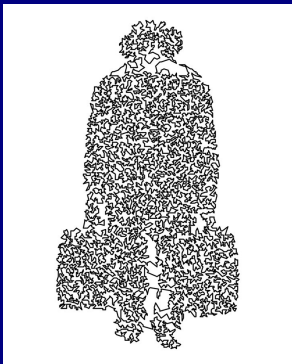


traveling salesman



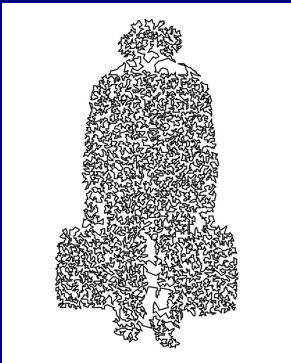
- Find the shortest road that visits all cities.

traveling salesman



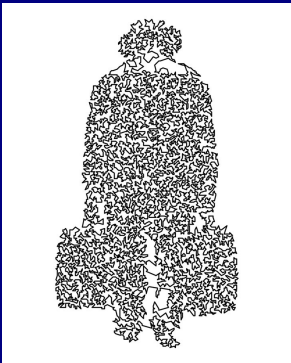
- Find the shortest road that visits all cities.
- Let's try them all ...

traveling salesman



- Find the shortest road that visits all cities.
- Let's try them all ...
- exponential solution :-)

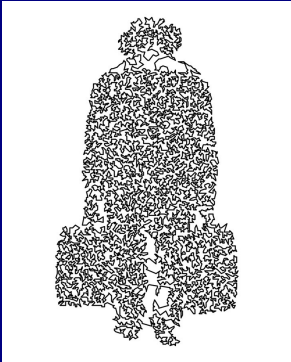
traveling salesman



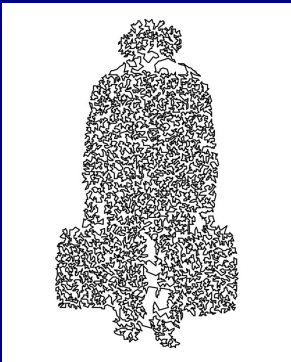
- Find the shortest road that visits all cities.
- Let's try them all ...
- exponential solution :-)
- If I give you the path - how can you verify that it is the shortest?

traveling salesman - almost the same

traveling salesman - almost the same

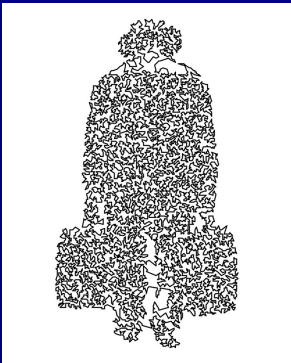


traveling salesman - almost the same



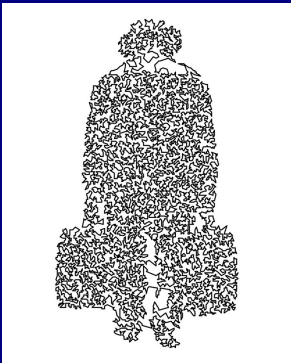
- Find a road visiting all cities but that is less than x km.

traveling salesman - almost the same



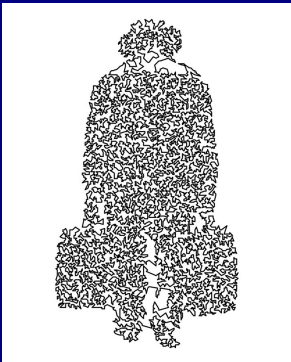
- Find a road visiting all cities but that is less than x km.
- Let's try them all ...

traveling salesman - almost the same



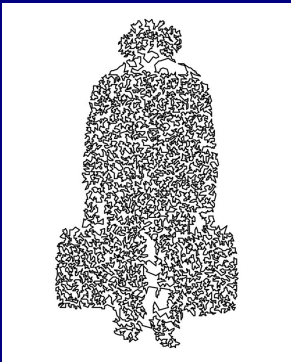
- Find a road visiting all cities but that is less than x km.
- Let's try them all ...
- exponential solution :-)

traveling salesman - almost the same



- Find a road visiting all cities but that is less than x km.
- Let's try them all ...
- exponential solution :-)
- A solution is easy to verify in polynomial time.

traveling salesman - almost the same

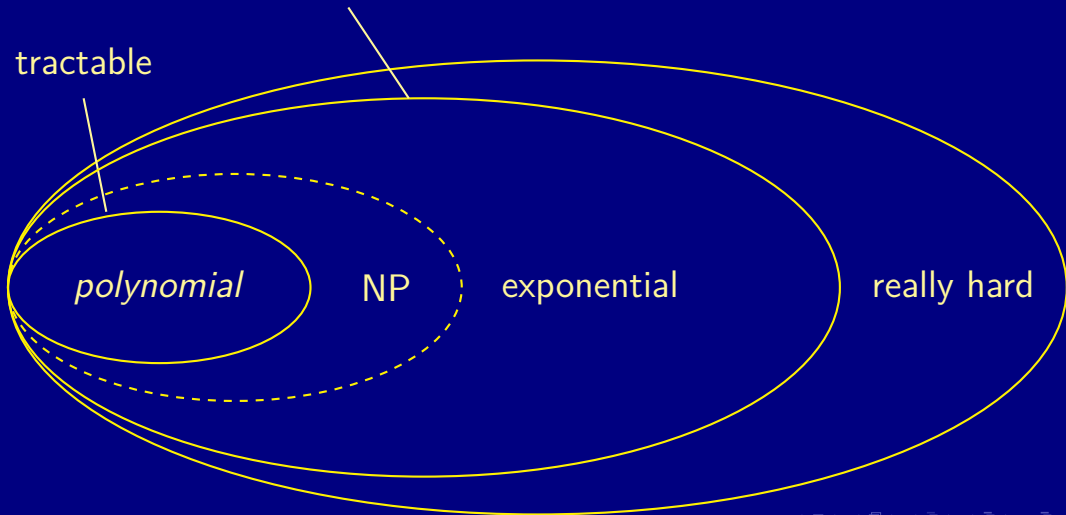


- Find a road visiting all cities but that is less than x km.
- Let's try them all ...
- exponential solution :-)
- A solution is easy to verify in polynomial time.
- A Non-deterministic Polynomial problem - NP.

non-deterministic polynomial

undecidable

tractable



extra income

$$P = NP$$

$$P = NP$$

If you can prove it, or prove that it does not holds,

$$P = NP$$

If you can prove it, or prove that it does not holds,
then you can claim a million dollar.

The Millennium Prize Problems

there is still hope

there is still hope

