Complexity

From computability to complexity

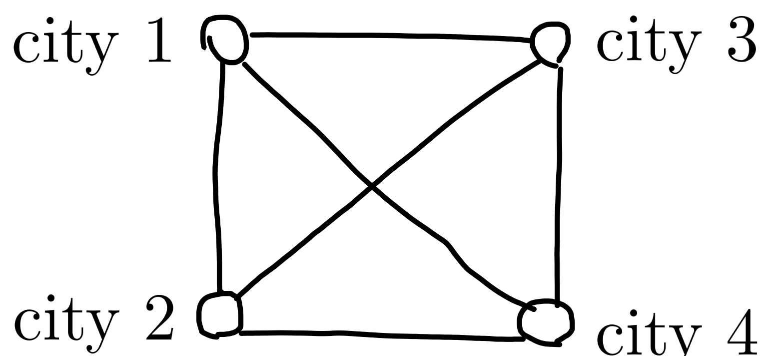
In computability: we identified some problems are decidable, and some others are undecidable

From computability to complexity

In computability: we identified some problems are decidable, and some others are undecidable

Even when a problem is decidable (i.e. algorithmic solvable), it may not be *practically* solvable as it may require too much time and/or space

Travelling Salesman Problem (TSP):



Find the shortest route that visits each city exactly once and returns to the starting city?

From computability to complexity

In computability: we identified some problems are decidable, and some others are undecidable

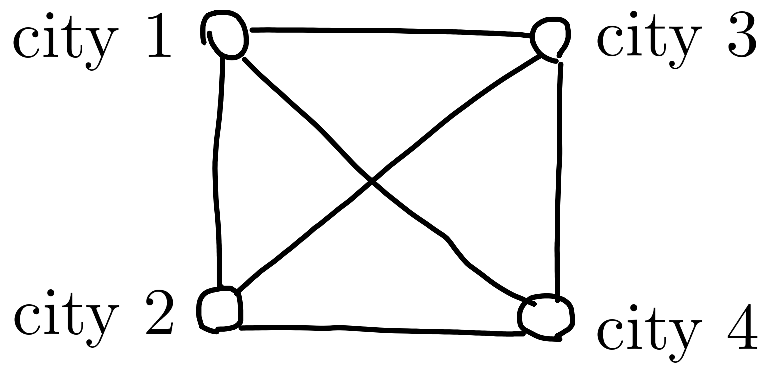
Even when a problem is decidable (i.e. algorithmic solvable), it may not be *practically* solvable as it may require too much time and/or space

From now on: let's focus on decidable problems, and identify problems that can be solved efficiently (as well as those that are intractable)

This is the (final) topic of complexity

Time complexity

Travelling Salesman Problem (TSP):



Find the shortest route that visits each city exactly once and returns to the starting city?

Suppose $\mathbf{M}_{TSP}$ is a decider (i.e. algorithm) that solves TSP.

Want: how many steps (i.e. head moves) $\mathbf{M}_{TSP}$ takes before it halts

Number of steps depends on length of input string (number of cities)

Time complexity

Given an algorithm (i.e. a TM), we define the
(worst-case) time complexity

Defn. Let $\mathbf{M}$ be a (deterministic) decider.

For an input string s, $\mathbf{M}$ accepts or rejects s after $t(s)$ steps.

The *time complexity* of $\mathbf{M}$ is

$$f(n) = \max_{\{s \in \Sigma^* | \ |s|=n\}} t(s)$$

Time complexity

Given an algorithm (i.e. a TM), we define the
(worst-case) time complexity

Defn. Let $\mathbf{M}$ be a (deterministic) decider.

For an input string s, $\mathbf{M}$ accepts or rejects s after $t(s)$ steps.

The *time complexity* of $\mathbf{M}$ is

$$f(n) = \max_{\{s \in \Sigma^* \mid \ |s| = n\}} t(s)$$

Exact number of steps of $\mathbf{M}$ is often complicated.
Consider *asymptotic* time complexity: when n is large.

Big-O

e.g. Let $f(n) = 6n^3$
Then $f(n) = O(\underline{n^3})$

$g(n) = n^3$

$c = 7$: $f(n) = 6n^3 \le 7n^3 = c \cdot g(n)$

$n_0 = 1$

e.g. Let $f(n) = 6n^3 + 10n^2$
Then $f(n) = O(n^3)$

$c = 16$: $f(n) = 6n^3 + 10n^2$
$\le 6n^3 + 10n^3$
$= 16 \cdot n^3 = c \cdot g(n)$

Write $f(n) = O(g(n))$ if
$(\exists c \ge 1)(\exists n_0 \ge 1)(\forall n \ge n_0) f(n) \le cg(n)$

Big-O

e.g. Let $f(n) = 6n^3 + 5n^2 + 4n + 100$
Then $f(n) = O(n^3)$

e.g. Let $f(n) = 3n^2 + 20n\log_2 n + 10^2$
Then $f(n) = O(n^2)$

e.g. Let $f(n) = 2\log_2 n + 7\log_2(\log_2 n) + 4$
Then $f(n) = O(\log_2 n) = \underline{O}(\log n)$

Big-O

e.g. Let $f(n) = 6n^3 + 5n^2 + 4n + 100$
Then $f(n) =$

e.g. Let $f(n) = 3n^2 + 20n\log_2 n + 10^2$
Then $f(n) =$

e.g. Let $f(n) = 2\log_2 n + 7\log_2\log_2 n + 4$
Then $f(n) =$

e.g. Let $f(n) = 99$
Then $f(n) = O(1)$

e.g. Let $f(n) = 88n^{99} + 2^n$
Then $f(n) = \cancel{O(n^{99})}$
$O(2^n)$

polynomial complexity
exponential complexity

Big-O

e.g. $f(n) = O(n^2) + O(n) =$ $O(n^2)$

e.g. $f(n) = O(n\log n) + \frac{n}{2}O(n) =$ $O(n^2)$

e.g. $f(n) = 2^{O(n)} + n^{O(1)} =$ $2^{O(n)}$

$2^{c \cdot n}$ n^c

expo. poly.

Example

Consider the following TM (decider) **M1** that decides

$L = \{\cancel{a^n \beta^n} \mid n = 0, 1, \ldots\}$: $\qquad \downarrow \boxed{X\,0\,0\,X\,1\,1\,B\,B\,B\cdots}$

$0^n 1^n$ $\qquad \underbrace{X00X11}_{n}$

M1 = "On input string s:

1) Scan the tape; if a 0 is to the right of a 1, reject

2) Scan the tape and cross off one 0 and one 1. $\rightarrow O(n)$
Repeat until no 0 or no 1 on the tape. $\frac{n}{2}$

3) If only 0 remains or only 1 remains on the tape, reject;
if neither 0 nor 1 remains on the tape, accept."

$|s| = n$

Time complexity $f(n) = O(n) + \frac{n}{2} \cdot O(n) + O(n)$
$= O(n^2)$

$n \to \frac{n}{2} \to \frac{n}{4} \to \cdots$

Example

$\underbrace{x\ 0\ x\ 0\ x\ 1\ x\ 1}_{n}\ B \cdots B$

Consider the following TM (decider) **M2** that decides $L = \{\cancel{a^n b^n}\ 0^n 1^n \mid n = 0, 1, \ldots\}$:

M2 = "On input string s:

1) Scan the tape; if a 0 is to the right of a 1, reject

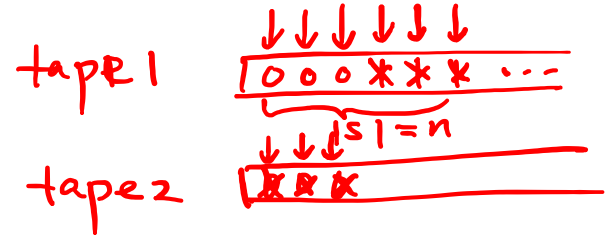
2) Scan the tape and cross off *every other* 0 starting from the first 0, then *every other* 1 starting from the first 1. Repeat until no 0 or no 1 on the tape.

3) If only 0 remains or only 1 remains on the tape, reject; if neither 0 nor 1 remains on the tape, accept."

Time complexity $f(n) = O(n) + \log_2 n\, O(n) + O(n)$

$= O(n \log n)$

Example



Consider the following TM (decider) **M3** with 2 tapes that decides $L = \{\underset{0^n 1^n}{\cancel{a^n b^n}} \mid n = 0, 1, \ldots\}$:

M3 = "On input string s:

1) Scan tape1; if a 0 is to the right of a 1, reject $O(n)$

2) Scan 0s on tape1 until the first 1, and copy the 0s onto tape2. $O(n)$

3) Scan 1s on tape1 until B. For each 1 read on tape 1 cross off a 0 on tape2. If all 0s are crossed off on tape2 before 1s on tape1 are read, reject $O(n)$

4) If 0 remains on tape2, reject; if all 0s are crossed off on tape2, accept." $O(n)$

Time complexity $f(n) =$ $O(n)$

Recap

For language $L = \{\alpha^n \beta^n \mid n = 0, 1, \ldots\}$:

	M1 (1-tape)	**M2** (1-tape)	**M3** (2-tape)
Time complexity	$O(n^2)$	$O(n \log n)$	$O(n)$

$O(n \log n)$ is the *best* a single-tape TM can do
if the language it decides is non-regular

Time complexity is dependent on models
(Decidability was not)

Polynomial difference

	Multi-tape TM	Single-tape TM
Time complexity $(t(n) \geq n)$	$f(n) = O(t(n))$	$f(n) = O(t^2(n))$

Time complexity of nondeterministic TM

Defn. Let **N** be a (nondeterministic) decider.

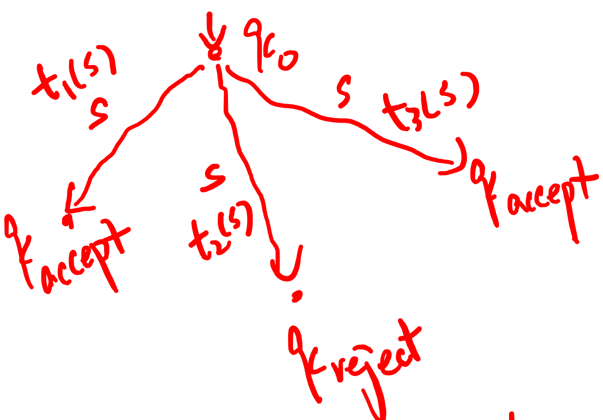
For an input string s, **N** accepts/rejects s on *multiple branches* and let $t(s)$ be the maximum steps among these branches.

The *time complexity* of **N** is

Non-det.

$$f(n) = \max_{\{s \in \Sigma^* \mid \ |s|=n\}} t(s)$$

Det.



$t(s) = \max\{t_1(s), t_2(s), t_3(s)\}$

Exponential difference

	Nondeterministic single-tape TM	Deterministic single-tape TM
Time complexity $(t(n) \geq n)$	$f(n) = O(t(n))$	$f(n) = 2^{O(t(n))}$

Class P and NP

Difference between multi-tape and single-tape: polynomial

Difference between nondeterministic and deterministic: exponential

e.g. n^3 and 2^n

Class P and NP

From now on: draw a line for decidable languages:

Class P: languages that can be decided by a polynomial time $O(n^c)$ algorithm (tractable, practically computable)

Class NP: languages that can be decided by an exponential time $O(2^{cn})$ algorithm, but a polynomial time algorithm has not been found (intractable, practically unsolvable)

Σ^*

decidable

undecidable

This classification allows us to ignore *polynomial difference*

P

NP

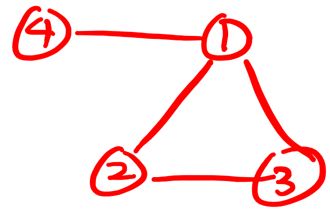
In practice *polynomial difference* is still important

Example

Language $L = \{\alpha^n \beta^n \mid n = 0, 1, \ldots\}$ can be decided by a TM with time complexity $O(n)$ ($O(n \log n)$, $O(n^2)$) so it is in Class P

Every context-free language is in Class P

G



$n =$ "number of nodes"

Example

Consider $L := \{<\mathcal{G}>|\ \mathcal{G}$ is a undirected connected graph$\}$

We designed the following TM that decides L

M = "On input string $<\mathcal{G}>$:

1) Select the first node and mark it. $O(1)$

2) From a marked node v, select an unmarked node v' that is connected to v and mark v'. $O(n) \cdot O(n)$
Repeat this step until no unmarked nodes can be selected.

3) Scan all nodes to determine if they are all marked. If so, accept. Otherwise, reject." $O(n)$

Let n be the number of nodes in $\mathcal{G}$

Time complexity $f(n) = O(n^2)$

Class P

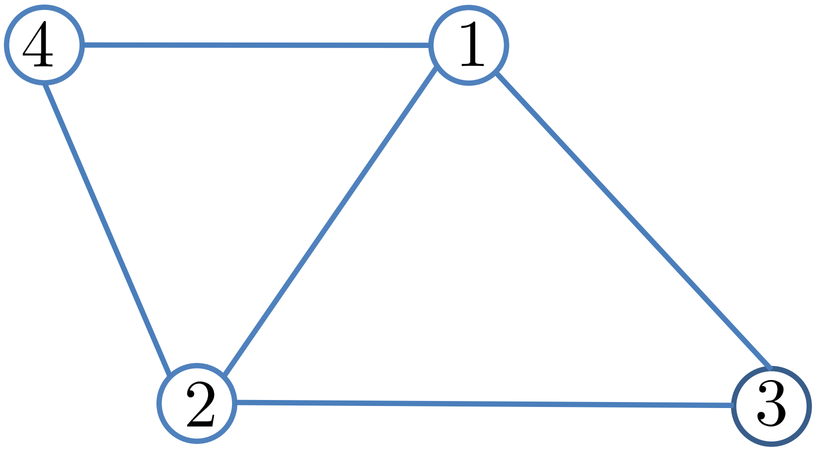
Class NP

These are languages for which a polynomial time algorithm has not been found.
Existence is unknown.

Instead, a polynomial time algorithm using nondeterministic TM has been found, which corresponds to an exponential time algorithm using deterministic TM

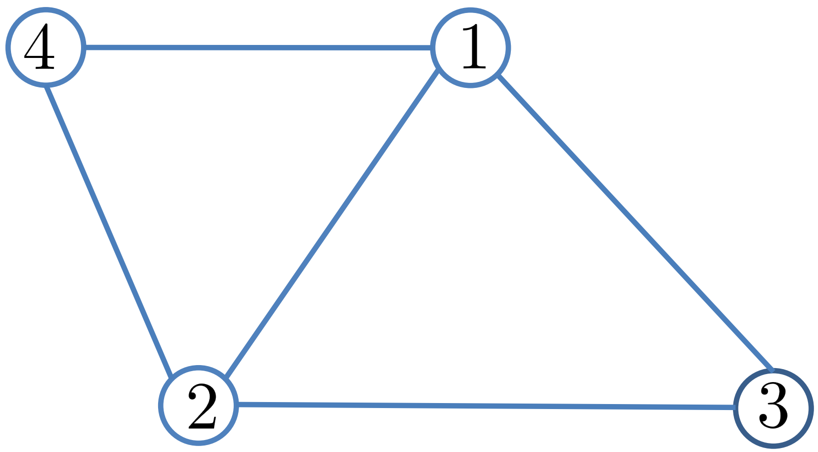
Example

Let $\mathcal{G}$ be an undirected graph. A *clique* is a subgraph of $\mathcal{G}$, where every two nodes are connected by an edge.



Example

Let $\mathcal{G}$ be an undirected graph. A *clique* is a subgraph of $\mathcal{G}$, where every two nodes are connected by an edge.



A k-clique is a clique that contains k nodes.

Problem: given a graph $\mathcal{G}$ and a number k, determine if $\mathcal{G}$ contains a k-clique.

Example

Consider $L := \{< \mathcal{G}, k >|\ \mathcal{G}$ is an undirected graph that contains a k-clique$\}$.

Design a TM that decides L

Example

Consider $L := \{< \mathcal{G}, k >|\ \mathcal{G}$ is an undirected graph that contains a k-clique$\}$.

Design a TM that decides L

$\mathbf{N}$ = "On input string $< \mathcal{G}, k >$:

1) Nondeterministically select k nodes.

2) Scan the edge set $\mathcal{E}$ to check if an edge exists between every two of the selected k nodes

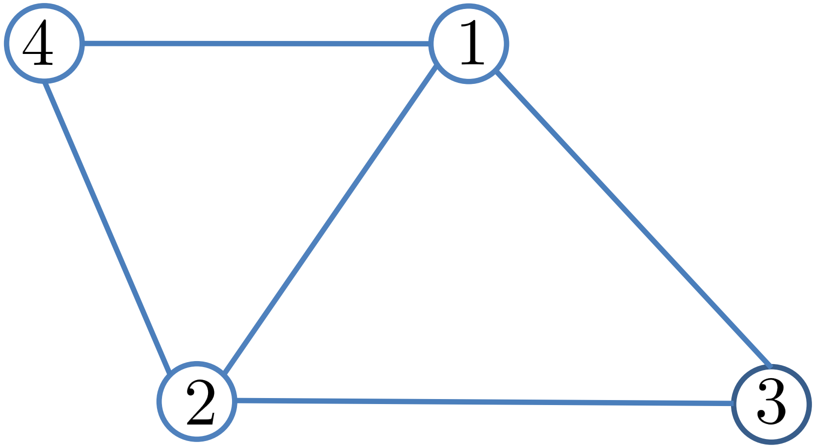
3) If there is a branch of $\mathbf{N}$ checks positively, accept. Otherwise, reject."

Let n be the number of nodes in $\mathcal{G}$

Time complexity $f(n) =$

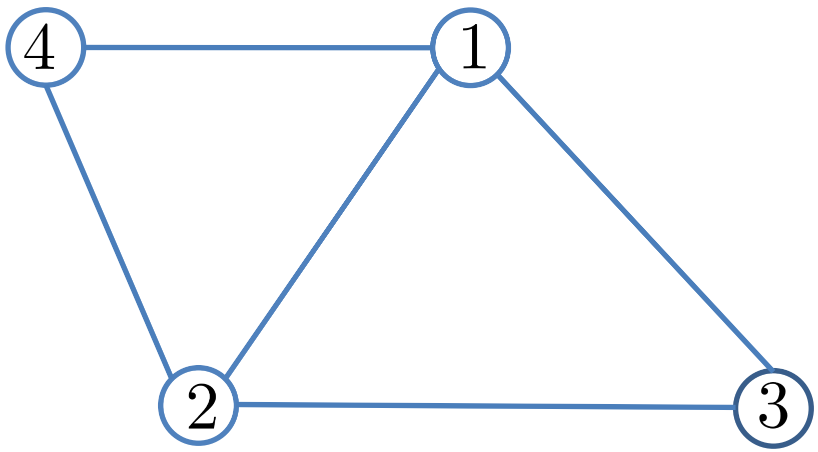
Example

Let $\mathcal{G}$ be an undirected graph. A *Hamiltonian cycle* is a cycle that visits each node exactly once.



Example

Let $\mathcal{G}$ be an undirected graph. A *Hamiltonian cycle* is a cycle that visits each node exactly once.



Problem: given a graph $\mathcal{G}$,
determine if $\mathcal{G}$ contains a Hamiltonian cycle.

Example

Consider $L := \{< \mathcal{G} >| \ \mathcal{G}$ is an undirected graph that contains a Hamiltonian cycle$\}$.

Design a TM that decides L

Example

Consider $L := \{< \mathcal{G} >|\ \mathcal{G}$ is an undirected graph that contains a Hamiltonian cycle$\}$.

Design a TM that decides L

$\mathbf{N}$ = "On input string $< \mathcal{G} >$:

1) Nondeterministically select n edges.

2) Scan the edge set $\mathcal{E}$ to check if the edges form a cycle that includes all nodes.

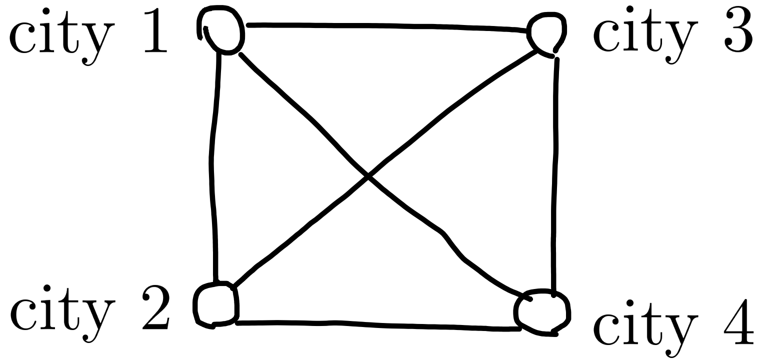
3) If there is a branch of $\mathbf{N}$ checks positively, accept. Otherwise, reject."

Let n be the number of nodes in $\mathcal{G}$

Time complexity $f(n) =$

Example

Travelling Salesman Problem (TSP):



Find the shortest route that visits each city exactly once and returns to the starting city?

Such a route is a Hamiltonian cycle

Problem: given a (weighted) undirected graph $\mathcal{G}$ and a number d, determine if there is a Hamiltonian cycle with distance $\leq d$

Example

Consider $L := \{< \mathcal{G}, d >| \ \mathcal{G}$ is a (weighted) undirected graph that contains a Hamiltonian cycle with distance $\leq d\}$.

Design a TM that decides L

Example

Consider $L := \{< \mathcal{G}, d >\mid \mathcal{G}$ is a (weighted) undirected graph that contains a Hamiltonian cycle with distance $\leq d\}$.

Design a TM that decides L

$\mathbf{N} =$ "On input string $< \mathcal{G}, d >$:

1) Nondeterministically select n edges.

2) Scan the edge set $\mathcal{E}$ to check
if the edges form a cycle that includes all nodes
and the sum of distances is $\leq d$.
3) If there is a branch of $\mathbf{N}$ checks positively, accept.
Otherwise, reject."

Let n be the number of nodes in $\mathcal{G}$
Time complexity $f(n) =$

Class P and NP

There are two possibilities:

$\mathrm{P} \not\subseteq \mathrm{NP}$

$\mathrm{P} = \mathrm{NP}$

One of the greatest open questions, worth of 1M USD