The Classes P and NP
Polynomial Time
Polynomial time

**Definition.** $P$ is the class of languages that are decidable in polynomial time on a deterministic Turing machine. In other words,

$$P = \bigcup_{k} \text{TIME}(n^k)$$
Graph Problems

Definition. \( PATH = \{ <G, s, t> \mid \exists \) a directed path from \( s \) to \( t \) in \( G \} \)

**Question.** Can you give an efficient algorithm (polynomial time) to decide PATH?
$PATH \in P$

$M =$ “On input $<G, s, t>$:

1. Place a mark on node $s$.
2. Repeat until no additional nodes are marked.
3. Scan all edges of $G$. If $(a, b)$ found from marked node to unmarked node, mark $b$.
4. If $t$ is marked, accept. Otherwise, reject.”
Proving that a Language is in P

- Construct a deterministic TM and show that it takes at most polynomial time, that is, $O(n^k)$ time on an input of size $n$, where $k$ is some constant.

- It does not matter how good the running time of your algorithm is as long as it is polynomially bounded.

- Assume reasonable encoding of input. For example, a graph might be encoded as adjacency matrix.
Importance of the Class $\textbf{P}$

- Invariant for all models of computation that are polynomially equivalent to the deterministic single-tape Turing machine

- Roughly corresponds to the class of all problems that we know are realistically solvable, that is, problems that have efficient algorithms for them
# Tractable Problems

<table>
<thead>
<tr>
<th>Time complexity function</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>.00001 second</td>
<td>.00002 second</td>
<td>.00003 second</td>
<td>.00004 second</td>
<td>.00005 second</td>
<td>.00006 second</td>
</tr>
<tr>
<td>$n^2$</td>
<td>.0001 second</td>
<td>.0004 second</td>
<td>.0009 second</td>
<td>.0016 second</td>
<td>.0025 second</td>
<td>.0036 second</td>
</tr>
<tr>
<td>$n^3$</td>
<td>.001 second</td>
<td>.008 second</td>
<td>.027 second</td>
<td>.064 second</td>
<td>.125 second</td>
<td>.216 second</td>
</tr>
<tr>
<td>$n^5$</td>
<td>.1 second</td>
<td>3.2 seconds</td>
<td>24.3 seconds</td>
<td>1.7 minutes</td>
<td>5.2 minutes</td>
<td>13.0 minutes</td>
</tr>
<tr>
<td>$2^n$</td>
<td>.001 second</td>
<td>1.0 second</td>
<td>17.9 minutes</td>
<td>12.7 days</td>
<td>35.7 years</td>
<td>366 centuries</td>
</tr>
<tr>
<td>$3^n$</td>
<td>0.59 second</td>
<td>58 minutes</td>
<td>6.5 years</td>
<td>3855 centuries</td>
<td>$2 \times 10^8$ centuries</td>
<td>1.3×10$^{13}$ centuries</td>
</tr>
</tbody>
</table>
Exercise

Let $CONNECTED = \{ <G> \mid G$ is a connected undirected graph$\}$. 

Show that $CONNECTED$ is in P.
What About Nondeterminism?

**Definition.** The running time of a nondeterministic Turing Machine $N$ is the function $f : \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the maximum number of steps that $N$ uses on any branch of its computation on any input of length $n$. 
How Costly Is Nondeterminism?

**Theorem.** Let $f(n)$ be a function, where $f(n) \geq n$. Every $f(n)$ time nondeterministic Turing machine has an equivalent $2^{O(f(n))}$ time deterministic Turing Machine.
How Costly Is Nondeterminism?

**Theorem.** Let \( f(n) \) be a function, where \( f(n) \geq n \). Every \( f(n) \) time nondeterministic Turing machine has an equivalent \( 2^{O(f(n))} \) time deterministic Turing Machine.

**Proof.** Compare the time complexity of the given nondeterministic machine with the equivalent deterministic machine. Let \( b \) be the branching factor of the NTM.

The number of nodes in the tree is \( 1 + b + b^2 + b^3 + \ldots + b^{f(n)} = O(b^{f(n)}) \).

The cost to explore from the root configuration down to each node is at most \( f(n) \).

Thus, the runtime is \( O(f(n) \times b^{f(n)}) = 2^{O(f(n))} \).
A Hamiltonian path is a path between two vertices of a graph that visits each vertex exactly once.

**Definition.** $HAMPATH = \{<G, s, t> | \exists$ a Hamiltonian path from $s$ to $t \}$
Guessing a Solution

$N =$ “On input $<G, s, t>$:

1. Guess an ordering, $p_1, p_2, \ldots, p_n$, of the nodes in $G$.
2. Check whether $s = p_1$ and $t = p_n$.
3. For each $i = 1$ to $n-1$, check whether $(p_i, p_{i+1})$ is an edge in $G$. 
   If any are not, reject. Otherwise, accept.
Checking for Hamiltonian Paths

$E =$ “On input <$G, s, t>$:

1. Generate all orderings, $p_1, p_2, \ldots, p_n$, of the nodes in $G$.
2. Check whether $s = p_1$ and $t = p_n$.
3. For each $i = 1$ to $n-1$, check whether $(p_i, p_{i+1})$ is an edge in $G$.
   
   If any are not, $reject$. Otherwise, $accept$. 
The Class NP

**Definition.** The class NP is the class of languages that are decidable in polynomial time on a nondeterministic Turing machine. In other words,

\[
NP = \bigcup_k \text{NTIME}(n^k)
\]

where NTIME\((t(n))\) = \{ \(L\) | \(L\) is decided in \(O(t(n))\) time by an NTM\}. 
The Class NP

- Roughly speaking the class NP corresponds to problems that are *easily verifiable*, that is, if someone claimed to have a solution, you *can verify it in polynomial time*.

- For example, \( \text{COMPOSITE} = \{ x \mid x = pq \text{ for integers } p,q > 1 \} \) is in NP. Why?
Certificates

NP is the class of languages that have polynomial time verifiers. That is, for a language \( L \in \text{NP} \), given input \( w \) and a certificate (string) \( c \), we can decide in polynomial-time (using \( c \)) whether \( w \in L \).

What is a certificate for \( PATH \)?

What is a certificate for \( HAMPATH \)?

What is a certificate for \( COMPOSITE \)?

What is a certificate for \( HAMPATH \)?
Exercise

A **clique** in an undirected graph is a subgraph wherein every two nodes are connected by an edge. A $k$-clique is a clique that contains $k$ nodes.

$$CLIQUE = \{ <G, k> \mid G \text{ is an undirected graph with a } k\text{-clique} \}.$$  
Show that $CLIQUE$ is in $NP$.  

The Classes P and NP

- Are there problems that given a candidate solution we can verify in polynomial time but we cannot find the solutions in polynomial time?

- How do we know some problems are not solvable in polynomial-time?
The Classes P and NP
"I can't find an efficient algorithm, I guess I'm just too dumb."
“I can’t find an efficient algorithm, because no such algorithm is possible!”
"I can't find an efficient algorithm, but neither can all these famous people."
Refining Our Picture

All Languages

Turing-Recognizable Languages

Decidable Languages

EXPTIME

P

CFLs

$\{a^n b^n c^n\}$

Regular Languages

$\{a^* b^*\}$
Refining Our Picture

- All Languages
- Turing-Recognizable Languages
- Decidable Languages
- EXPTIME
  - NP
  - P
  - CFLs $a^n b^n$
  - Reg $a^* b^*$ Languages

Whole bunch of problems