For some problems it is difficult to find an algorithm in P

We first discuss an example of finding a Hamiltonian Path

Definition: for a given path find a path going through all nodes once

Fig 7.17
Hamiltonian Path II

- HAMPATH = \{⟨G, s, t⟩ | G : directed, a Hamiltonian path from s to t\}
- A brute-force way: checking all possible paths
  But the number is exponential
- Polynomial verification
  for a path, in P time ⇒ a Hamiltonian path or not
This is an example where verification is easier than determination
We discuss another example where verification is easier than determination.

An integer is composite if

\[ x = pq, \ p > 1, \ q > 1 \]

- Given \( x \), difficult to find \( p, q \)
- Given \( x, p, q \) easily verify \( x = pq \) or not
Some problems are difficult so even a polynomial verifier cannot be easily obtained.

**HAMPATH**: given \( \langle G, s, t \rangle \) no Hamiltonian path from \( s \) to \( t \)

Verification may still be difficult.

Given \( s \) and \( t \) it seems we still need to check all paths.
Verifier I

- **Definition:** An algorithm $V$ is a verifier of a language $A$ if

\[ A = \{ w \mid V \text{ accepts } \langle w, c \rangle \text{ for some strings } c \} \]

- **Example:** Compositness. $V$ accepts $\langle w, c \rangle = \langle x, p \rangle$, where $p$ is a divisor

- **Example:** Hamiltonian path. $V$ accepts

\[ \langle w, c \rangle = \langle \langle G, s, t \rangle, \text{ a path from } s \text{ to } t \rangle \]

- $c$ is called a “certificate”
Verifier II

- Definition: a polynomial verifier if it takes polynomial time of $|w|$
- A: polynomially verifiable if $\exists$ a polynomial verifier
- Note that we measure time on $|w|$ without considering $|c|$
- For a polynomial verifier, $|c|$ should be in polynomial of $|w|$
- Otherwise, reading $|c|$ already non-polynomial
NP is a class of languages

Definition: a language $\in$ NP if it has a polynomial verifier

We will prove that this definition is equivalent to that the language is decided by nondeterministic polynomial TM

This is where the name comes from

Some use this as the definition

Note that for nondeterministic TM time is by checking the longest branch
Definition:

$\text{NTIME}(t(n)) = \{ L \mid L \text{ decided by } O(t(n)) \text{ nondeterministic TM} \}$

$\text{NP} = \bigcup_k \text{NTIME}(n^k)$
NTM for HAMPATH I

A list \( p_1 \cdots p_m \) is nondeterministically determined

For each list:

1. Check repetitions
2. Check \( s = p_1; \ t = p_m \)
3. Check that for \( 1 \cdots m - 1 \), \( (p_i, p_{i+1}) \) is an edge of \( G \)

Cost on each list is polynomial:

repetitions: \( O(m^2) \)

\( s = p_1, \ t = p_m : O(m) \)

edge check: \( O(m^2) \)
\textbf{NP} \equiv \text{Polynomial NTM I}

- Polynomial verifier \iff polynomial NTM
- Idea:
  - \(\Rightarrow\) NTM by guessing certificate
  - \(\Leftarrow\) using NTM’s accepting branch as certificate
- Proof:
  - \(\Rightarrow\) : now we have a verifier \(V\) in time \(O(n^k)\)

Recall the definition below

\[ A = \{ w \mid V \text{ accepts } \langle w, c \rangle \text{ for some strings } c \} \]

We have

\[ |c| \leq n^k \]
Use an NTM to

1. nondeterministically select $c$
2. run $V$ on $\langle w, c \rangle$

That is, run $c$ in parallel and each is polynomial

We have that for any $w \in A$, the NTM accepts it in polynomial time

"$\iff$": now $w$ is accepted by a polynomial NTM

Let $c$ be the accepting branch

Note that for polynomial NTM, each branch is polynomial
Then we have a verifier $V$ that handles input $\langle w, c \rangle$ in polynomial time.

Note: the definition of $V$ requires only “some $c$.”

So finding one is sufficient.
SUBSET-SUM I

Given \( x_1, \ldots, x_k \) and \( t \), is sum of a subset \( = t \) ?

Formally

\[
\{\langle s, t \rangle \mid s = \{x_1, \ldots, x_k\} \text{ and } \exists \{y_1, \ldots, y_l\} \subset \{x_1, \ldots, x_k\} \text{ such that } \sum y_i = t\}
\]

Example

\[\langle \{4, 11, 16, 21, 27\}, 25 \rangle \text{ OK as } 4 + 21 = 25\]
Note: allow repetition here

\[ \langle \{4, 11, 11, 16, 21, 27\}, 25 \rangle \]

We prove that this problem is NP
Consider any input

\[ \langle \langle s, t \rangle, c \rangle \]

we

1. check if \( \sum c_i = t \)
2. check if all \( c_i \in S \)
Here

length of $c < \text{length of } s$

The verification can be done in polynomial time
Roughly

P: problems decided quickly
NP: problems verified quickly

Question: is $P = NP$?
   This is one of the greatest unsolved problems

Most believe $P \neq NP$
It has been shown that some problems in NP are related.

For certain NP problems:
- If there exists a polynomial algorithm of one NP problem, then $P = NP$.
- These problems are called NP-complete problems.
- They are useful to study the issue of $P$ versus $NP$.

To prove $P \neq NP$, only need to focus on NP-complete problems.

To prove $P = NP$, need only polynomial algorithms for an NP-complete problem.