- Reduction:

A function f can be shown uncomputable by reducing the Halting problem to it.

I.e design a computable fn. f' s.t. $\forall x$, $HALT(x) = f \circ f'(x)$.

We then write $HALT \leq_T f$.

Complexity classes P & NP

- We now study computable functions.

- For a TM M, we denote the maximum number of steps, on an input of size n, by $time_M(n)$.

- We collect all the problems solvable in time $\approx T(n)$ in a class:

$Dtime(T(n)) := \{L \subseteq \{0,1\}^* \mid TM\ M \text{ decides } L,\ time_M(n) = O(T(n))\}$.

- As discussed before, we consider poly-time "fast". So,

<u>Defn</u>: · The poly-time solvable problems are
$\underline{P} := \bigcup_{c \in \mathbb{N}} \mathrm{Dtime}(n^c)$.

· Formally, P is called the <u>deterministic polynomial time</u> class.

- For practical people poly-time may not always be "fast".
 For eg. $g(n) = n^{\log\log n}$ is "faster" than $f(n) = n^{100}$ time, up to input size $n < 2^{2^{100}}$!!

- Historically, P has served as a rather good abstraction of <u>fast</u>.

- Problems in P? Many interesting examples.

- Let us now consider some practically "hard" problems.

(1) Travelling salesperson:

Let G be a graph on n vertices with edges labelled with "distances". Is there a closed tour, visiting all the nodes exactly once, of length $\leq k$?

(2) Subset sum:

Let S be a set $\{a_1, \ldots, a_n\} \subseteq \mathbb{N}$ & $t \in \mathbb{N}$. Is there a subset of S summing to t?

(3) Integer programming:

Given m linear inequalities in m variables. Is there a boolean feasible point?

Exercise: (1), (2), (3) are easily solved in time $O(n!)$, $O(2^n)$, $O(2^m)$ respectively.

- So, these three problems have algos. that run in <u>exponential-time</u>.

<u>Defn:</u> $\underline{\underline{EXP}} := \bigcup_{c \in \mathbb{N}} \text{Dtime}(2^{n^c})$.

- Can these problems be solved any faster? Not known!

- They have a common feature: Given a closed tour, a subset, or a boolean point; it can be "verified" in poly-time whether it is the right one!

- This feature motivated Cook (1971) & Levin (1973) to study all the problems that have <u>poly-time verifiers</u>.

Defn: A language $L \subseteq \{0,1\}^*$ is in <u>NP</u> if $\exists c > 0$ & a poly-time TM M s.t.

$$\forall x \in \{0,1\}^*, \quad x \in L \text{ iff}$$
$$\exists u \in \{0,1\}^{|x|^c}, \quad M(x,u) = 1.$$

M is called the <u>verifier</u> for L

u is called the <u>certificate</u> for x

- Eg. in (2) the corresponding language is $L = \{(S,t) \mid S \text{ has a subset of sum } t\}$

On input (S,t):

- The <u>certificate u</u> is the subset of S of sum t.
- The <u>verifier M</u> is TM that given (S,t,u) checks whether $u \subseteq S$ & sum of u is t.

- Clearly, $(S,t) \in L$ iff $\exists u, M(S,t,u) = 1$.
- Also, $|u| \le |(S,t)|$ & M is poly-time.

- Thus, $L \in NP$. (Same with (1) & (3)).

- Let us "place" NP :

▷ $P \subseteq NP$.

Pf: • Say, $L \in P$ is decided by a poly-time TM M. Then, $\forall x \in \{0,1\}^*$,

$$x \in L \text{ iff } M(x) = 1 .$$

• So, with the empty u, we can deduce $L \in NP$. $\square$

▷ $NP \subseteq EXP$.

Pf: • Say, $L \in NP$ with the setting:
$x \in L$ iff $\exists u \in \{0,1\}^{|x|^c}$, $M(x,u) = 1$.

• Say, $\text{time}_M(n) \leq n^d$.

• On input x, we can test $x \overset{?}{\in} L$ by running M on all possible certificates! Output 1 iff one of these time M outputs 1.

• The time-complexity is: $2^{|x|^c} \times (|x| + |x|^c)^d$

- which is, clearly, $O(2^{|x|^{c+1}})$ for $|x|$ large enough.

- $\Rightarrow L \in \mathrm{Dtime}(2^{n^{c+1}}) \subseteq \mathrm{EXP}$. $\square$

▷ $P \subseteq NP \subseteq EXP$.

- It is not known whether NP is different from the other two.

- Though, later, we will prove that P & EXP are different!

- The 'N' in NP refers to <u>nondeterminism</u>.

- To study NP we generalize TMs to <u>nondeterministic TMs</u> (NDTM).