Reading: Sipser §7.3.
“Nondeterministic Time”

We say that a nondeterministic TM $M$ decides a language $L$ iff for every $w \in \Sigma^*$,

1. Every computation by $M$ on input $w$ halts (in state $q_{\text{accept}}$ or state $q_{\text{reject}}$);

2. $w \in L$ iff there exists at least one accepting computation by $M$ on $w$.

3. $w \notin L$ iff every computation by $M$ on $w$ rejects (or dies, with no applicable transitions).

$M$ decides $L$ in nondeterministic time $t(\cdot)$ iff for every $w$, every computation by $M$ on $w$ takes at most $t(|w|)$ steps.
More on Nondeterministic Time

1. Linear speedup holds.

2. “Polynomial equivalence” holds among nondeterministic models
   
   \[ L \text{ decided in time } T \text{ by a nondeterministic multitape TM} \]
   
   \[ \Rightarrow L \text{ decided in time } O(T^2) \text{ by a nondeterministic 1-tape TM} \]

Definition:

\[ \text{NTIME}(t(n)) = \{ L : L \text{ is decided in time } t(n) \text{ by some nondet. multitape TM} \} \]

\[ \text{NP} = \bigcup_{\text{polynomial } p} \text{NTIME}(p) = \bigcup_{k \geq 0} \text{NTIME}(n^k). \]
P vs. NP

• Clearly $P \subseteq NP$. But there are problems in NP that are not obviously in P ($\neq$ “obviously not”)

• TSP = TRAVELLING SALESMAN PROBLEM.
  • Let $m > 0$ be the number of cities,
  • $D : \{1, \ldots, m\}^2 \to \mathbb{N}$ give the distance $D(i, j)$ between city $i$ and city $j$, and
  • $B$ be a distance bound

Then $TSP =$

$$\{\langle m, D, B \rangle : \exists \text{ tour of all cities of length } \leq B \}. $$
Traveling Salesman Problem: Example

There are many variants of TSP, eg require visiting every city exactly once, triangle inequality on distances...

“tour” = visits every city and returns to starting point
TSP $\in$ NP

• Why is TSP $\in$ NP?

  Because if $\langle m, D, B \rangle \in$ TSP, the following nondeterministic strategy will accept in time $O(n^3)$, where $n =$ length of representation of $\langle m, D, B \rangle$.

  – nondeterministically write down a sequence of cities $c_1, \ldots, c_t$, for $t \leq m^2$. (“guess”)

  – trace through that tour and verify that all cities are visited and the length is $\leq B$. If so, halt in $q_{\text{accept}}$. If not, halt in $q_{\text{reject}}$. (and “check”)

If $\langle m, D, B \rangle \notin$ TSP, above has no accepting computations.

But any obvious deterministic version of this algorithm takes exponential time.
A useful characterization of NP

• A **verifier** for a language \( L \) is an algorithm \( V \) such that

\[
L = \{ x : V \text{ accepts } \langle x, y \rangle \text{ for some string } y \}.
\]

• A **polynomial-time verifier** is one that runs in time polynomial in \(|x|\) on input \( \langle x, y \rangle \).

• A string \( y \) that makes \( V(\langle x, y \rangle) \) accept is a “proof” or “certificate” that \( x \in L \).

• **Example:** TSP

  certificate \( y = ? \)

  \[
  V(\langle x, y \rangle) = ?
  \]

• Without loss of generality, \(|y|\) is at most polynomial in \(|x|\).
NP is the class of easily verified languages

- **Theorem**: NP equals the class of languages with polynomial-time verifiers.

**Proof**:

⇒

⇐

- “$L$ is in NP iff members of $L$ have short, efficiently verifiable certificates”
More problems in NP

- Hamiltonian Circuit

\[ HC = \{ G : G \text{ is an undirected graph with a circuit that touches each node exactly once} \}. \]

Really just a special case of TSP. (why?)

- We are not fussy about the precise method of representing a graph as a string, because all reasonable methods are within a polynomial of each other in length.
A “similar” problem that is in P

- **Eulerian Circuit**

  \[ EC = \{ G : G \text{ is an undirected graph with a circuit that passes through each edge exactly once} \} \]

It is easy to check if \( G \) is Eulerian…

So \( EC \in P \).
Composite Numbers

- **COMPOSITES** = \{ w : w a composite number in binary \}.

\[ \text{COMPOSITES} \in \text{NP} \]

Not obviously in P, since an exhaustive search for factors can take time proportional to the value of \( w \), which grows as \( 2^n = \text{exponential in the size of } w \).

Only recently (2002), it was shown that **COMPOSITES** \( \in \) P (equivalently, **PRIMES** \( \in \) P).
Boolean logic

Boolean formulas

**Def**: A **Boolean formula** (B.F.) is either:

- a “Boolean variable” $x, y, z, \ldots$
- $(\alpha \lor \beta)$ where $\alpha, \beta$ are B.F.’s.
- $(\alpha \land \beta)$ where $\alpha, \beta$ are B.F.’s.
- $\neg \alpha$ where $\alpha$ is a B.F.

**e.g.** $(x \lor y \lor z) \land (\neg x \lor \neg y \lor \neg z)$

[Omitting redundant parentheses]
Boolean satisfiability

**Def:** A truth assignment is a mapping $a : \text{Boolean variables} \rightarrow \{0, 1\}$. \([0 = \text{false, } 1 = \text{true}]\)

The \([0, 1]\) value of a B.F. $\gamma$ on a truth assignment $a$ is given by the usual rules of logic:

- If $\gamma$ is a variable $x$, then $\gamma(a) = a(x)$.
- If $\gamma = (\alpha \lor \beta)$, then $\gamma(a) = 1$ iff $\alpha(a) = 1$ or $\beta(a) = 1$.
- If $\gamma = (\alpha \land \beta)$, then $\gamma(a) = 1$ iff $\alpha(a) = 1$ and $\beta(a) = 1$.
- If $\gamma = \neg \alpha$, then $\gamma(a) = 1$ iff $\alpha(a) = 0$.

$a$ satisfies $\gamma$ (sometimes written $a \models \gamma$) iff $\gamma(a) = 1$.

In this case, $\gamma$ is satisfiable. If no $a$ satisfies $\gamma$, then $\gamma$ is unsatisfiable.
Boolean Satisfiability

\[ \text{SAT} = \{ \alpha : \alpha \text{ is a satisfiable Boolean formula} \} \]

**Prop:** \( \text{SAT} \in \text{NP} \)
A “similar” problem in P: 2-SAT

A 2-CNF formula is one that looks like

\[(x \lor y) \land (\neg y \lor z) \land (\neg y \lor \neg x)\]

i.e., a conjunction of clauses, each of which is the disjunction of 2 literals (or 1 literal, since \((x) \equiv (x \lor x)\))

2-SAT = the set of satisfiable 2-CNF formulas.

   e.g. \((x \lor y) \land (\neg x \lor \neg y) \land (\neg x \lor y) \land (x \lor \neg y) \notin \text{SAT}\)
2-SAT $\in \mathbf{P}$

Method (resolution):

1. If $x$ and $\neg x$ are both clauses, then not satisfiable
   
   e.g. $(x) \land (z \lor y) \land (\neg x)$

2. If $(x \lor y) \land (\neg y \lor z)$ are both clauses, add clause $(x \lor z)$ (which is implied).

3. Repeat. If no contradiction emerges $\Rightarrow$ satisfiable.

$O(n^2)$ repetitions of step 2 since only 2 literals/clause.

Proof of correctness: omitted