$P \subseteq NP$

**Def** $H$ is NP-hard (under polynomial time reductions, $\leq_P$) if for every $Y \in NP$

$Y \leq_P H$

**Def** $X$ is NP-complete if $X$ is NP-hard and $X \in NP$.

**Easy observation**; if $X$ is NP-complete then

$$X \in P \iff P = NP$$

(if $X \in P$, then $\forall A \in NP$ $A \leq_X X$, so $A \in P$; If $X \notin P$ then it witnesses $P \neq NP$)

**Thm 0.** There are NP-complete problems.

**Thm 1.** There are interesting NP-complete problems. [proof later]

In particular, CIRCUIT-SAT (circuit satisfiability)

**List of interesting NP-complete problems:**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Input</th>
<th>Yes-Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIRCUIT-SAT</td>
<td>$T$: Boolean circuit with $n$ inputs, $t$-bit output</td>
<td>There is an assignment to inputs that yields output 1</td>
</tr>
<tr>
<td>(CNF-)SAT</td>
<td>literal: variable or negated variable clause: OR ($v$) of literals CNF: AND ($v$) of clauses</td>
<td>as above, but each clause has $\leq 3$ literals</td>
</tr>
<tr>
<td>3-(CNF-)SAT</td>
<td>$G = (V, E)$, integer $k$</td>
<td>$G$ has a vertex cover of size $\leq k$</td>
</tr>
<tr>
<td>Vertex Cover</td>
<td>$G = (V, E)$, integer $k$</td>
<td>$G$ is an independent set of size $\geq k$</td>
</tr>
<tr>
<td>Independent Set</td>
<td>$U = {1, \ldots, n}$, $S \subseteq U$ $i \neq j$, integer $k$</td>
<td>There are $m \leq k$, $S_j$ whose union is $U$</td>
</tr>
<tr>
<td>Set Cover</td>
<td>$G = (V, E)$</td>
<td>$G$ has a cycle that visits every vertex exactly once.</td>
</tr>
<tr>
<td>Hamiltonian Cycle</td>
<td>$G = (V, E)$ $E = {(u, v)}$ distance $d: (u_i, v_i) \rightarrow d(u_i, v_i)$</td>
<td>There is a path of length $\leq D$ that starts at $u_i$, ends at $v_i$, and visits every vertex.</td>
</tr>
<tr>
<td>Traveling Salesman</td>
<td>$G = (V, E)$</td>
<td>There are $n$ triples in $T$ such that every element of $xuv$ is in one triple in $T$</td>
</tr>
<tr>
<td>3-D Matching</td>
<td>$G = (V, E)$</td>
<td>$G$ has a valid $k$-coloring (even for $k=3$)</td>
</tr>
<tr>
<td>Graph $k$-coloring</td>
<td>$G = (V, E)$</td>
<td>$G$ has a valid $k$-coloring (even for $k=3$)</td>
</tr>
</tbody>
</table>
**SUBSET SUM:** \( w_1, \ldots, w_n \) positive integers
\( W \) positive integer

There is \( J \subseteq \{1, \ldots, n\} \)
\[ \sum_{j \in J} w_j = W \]

**0-1 INTEGER (LINEAR) PROGRAMMING**

**SCHEDULING w/RELEASE TIMES AND DEADLINES**

Objective function:
\[ f = \sum_{i} c_i x_i \quad (c_i \text{ given}) \]

Matrix \( A = (a_{ij}) \quad i = 1, n \]
\( j = 1, m \)

Vector \( b = (b_i) \quad i = 1, n \)

Value \( v \)

n jobs

Job \( i \) has release time \( r_i \)
Deadline \( d_i \)
Processing duration \( t_i \)

A job once scheduled must run its duration

There are choices for \( x_i \)
in \( \{0, 1\} \)
such that
\[ \sum_{i} c_i x_i \geq v \]
\[ A x \leq b \]

There is a schedule that makes all jobs run within the constraints.