Remember the definition of NP: languages decidable in polynomial time. I.e.:

$$NP = \bigcup_{k=1}^{\infty} NTIME(n^k)$$

**Language Funtime**

Here are some fun languages:

1. **SAT** = \{ \langle \Phi \rangle \mid \Phi(x_1...x_k) \text{ is a Boolean formula, and } \exists a_1...a_k \text{ such that } \Phi(a_1...a_k) = T \}\n
2. **3SAT** = \{ \langle \Phi \rangle \mid \Phi(x_1...x_k) \text{ is a 3-CNF, and } \exists a_1...a_k \text{ such that } \Phi(a_1...a_k) = T \}\n
3. **Hampath** = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph, } s \text{ and } t \text{ are vertices } \in V(G), \text{ and there is a path from } s \text{ to } t \text{ that visits all } v \in V(G) \text{ exactly once } \}\n
4. **Composites** = \{ x \mid x \text{ is a composite integer } \}\n
5. **GraphIso** = \{ \langle G, H \rangle \mid G, H \text{ are graphs and } \exists \pi \text{, a permutation } V(G) \to V(G) \text{ such that } H = \pi(G) \}\n
6. **Clique** = \{ \langle G, k \rangle \mid G \text{ is a graph with a clique}^1 \text{ of size } k \}\n
7. **VertexCover** = \{ \langle G, k \rangle \mid G \text{ is a graph with a vertex cover}^2 \text{ of size } k \}\n
8. **IndependentSet** =\{ \langle G, k \rangle \mid G \text{ is a graph with an independent set}^3 \text{ of size } k \}\n
**Exempli Gratia**

**SAT**

We show the NTM decider for SAT:

On input \langle \Phi \rangle:
- Nondeterministically select an assignment \( (a_1, a_2...a_k) \).
- Evaluate \( \Phi(a_1, a_2...a_k) \).
- Accept if \( \Phi = T \), reject otherwise

Runtime: The runtime is clearly polynomial.

Correctness: If \( \Phi \in SAT \), then there will be an accepting choice for \( (a_1, a_2...a_k) \), and we will accept. If \( \Phi \notin SAT \), then no selection of \( (a_1, a_2...a_k) \) will result in True, and there will be no accepting branch of our NTM.

---

1. a clique of \( G \) is a subset \( C \), of \( V(G) \) such that \( \forall (u, v) \in C, (u, v) \in E(G) \)
2. a vertex cover of \( G \) is a subset \( C \), of \( V(G) \) such that \( \forall (u, v) \in E(G), u \in C \text{ or } v \in C \)
3. an independent set of \( G \) is a subset \( C \), of \( V(G) \) such that \( \forall (u, v) \in C, (u, v) \notin E(G) \)
HamPath

We show the NTM decider for HAMPath:
On input \(<G,s,t>\):
  Non-deterministically select an ordering of all the vertices in \(G\), \((v_1,v_2...v_k)\), such that \(v_1 = s\) and \(v_k = t\)
  Check that each \((v_i,v_{i+1})\) is an edge in \(G\).
  Accept if so, reject otherwise

Runtime: The runtime is clearly polynomial in the number of vertices in the graph, and we can represent a graph on the tape as a list of vertices and edges, so the runtime is also polynomial in input.
Correctness: If \(<G,s,t>\in\text{HAMPath}\), then there will be an accepting choice for \((v_1,v_2...v_k)\), and we will accept. If \(<G,s,t>\not\in\text{HAMPath}\), then no selection of \((v_1,v_2...v_k)\) will be accepted, and we reject.

Composites

We show the NTM decider for COMPOSITES:
On input integer \(<x>\):
  Non-deterministically select some integer \(i\) \(1 < i \leq \sqrt{x}\)
  Check that \(i\) divides \(x\).
  Accept if so, reject otherwise

Runtime: The runtime is clearly polynomial in the representation of \(x\), which takes \(\log(x)\) bit. Note that we can’t select all values from 1 to \(x\) because that would be exponential in \(\log(x)\).
Correctness: P. Obv.

Graph Isomorphism

We show the NTM decider for GraphIso:
On input \(<G,H>\):
  Non-deterministically select a permutation \(\pi\) of all the vertices in \(G\)
  Check that each \(\pi(G) = H\)
  Accept if so, reject otherwise

Runtime: Poly Correctness: Correct.

Another definition of NP

We can consider a deterministic definition of NP.
Def: A Verifier \(V\) for a language \(A\) is a Turing Machine such that \(A = \{w | \exists c \text{ such that } V(w,c) \text{ accepts or rejects in polynomial time}\}\)
V’s runtime is measured in the size of \( w \), not in \( w \) and \( c \). The language \( A \) is poly-time-verifiable if \( \exists V \), a poly-time verifier for \( A \).

We propose a definition, \( NP_2 = \{ A \mid A \text{ is poly-time-verifiable} \} \)

Now consider a Verifier for CLIQUE:

On input \( w = \langle G, k \rangle \), and \( c \), a \( k \)-sized subset of \( V(G) \), Verify that for all \( (u, v) \in c \), \( (u, v) \in E(G) \).

Runtime is clearly polynomial, as we simply check each pair of vertices in the subset against the edge-set.

**Thm:** \( NP = NP_2 \).

**Proof:** Forward Direction: Suppose \( L \in NP \) (meaning that it has a poly-time NTM, \( N \), that decides it), then \( \exists \text{Verifier} V \) for it:

On input \( \langle w, c \rangle \): Simulate \( N \) on input \( w \), using \( c \) to dictate \( N \)'s choices. If \( N \) accepts, accept, else reject.

Other Direction: Suppose \( L \in NP_2 \) (meaning that it has a poly-time Verifier), then \( \exists \text{N} \), a poly-time NTM that decides it:

On input \( w \): Nondeterministically select \( c \) Run \( V(w, c) \). Accept if it accepts, reject otherwise.

So we can say that \( P = \) languages decidable in polynomial time and \( NP = \) languages verifiable in polynomial time.

**Poly-Time Reducibility**

\( f \) is a poly-time reducible function if it is computable by a poly-time TM.

\( A \preceq_p B : \exists \) a poly-time computable function \( f \) such that \( w \in A \) iff \( f(w) \in B \).

Example: Let us poly-time reduce CLIQUE to INDEPENDENTSET:

On \( \langle G, k \rangle \):

Let \( G' \) have the same vertices as \( G \), and let the edges of \( G', E' = (u, v) \) such that \( (u, v) \notin E(G) \).

Output \( \langle G', k \rangle \).

Runtime: It poly.

Correctness: If \( G' \) has a independent set of size \( k \), then inverting the edges means that that same set would be totally connected (because it was totally unconnected before). If \( G \) has a clique of size \( k \), then inverting the edges means all the vertices in the clique are totally disconnected if we invert the edges.

\footnote{That \( c \) represents the runtime of \( V \), and thus is polynomial in the input, \( w \). Therefore you only need to consider polynomial lengths for \( c \).}
Example: Let us poly-time reduce \textsc{IndependentSet} to \textsc{VertexCover}:

On $< G, k >$:
- Output $< G, |V(G)| - k >$.

Runtime: You just did a subtraction....
Correctness: If there is an independent set of size $k$, then all the vertices not in the independent set form a vertex cover by definition (so we have a vertex cover of size $V(G) - k$). If there is not an independent set of size $k$, then there are not $V(G) - k$ vertices that form a vertex cover.

\textbf{IMPORTANT THEOREM STUFF}

\textbf{Thm} If $A \leq_p B$ and $B \in P$, then $A \in P$

\textbf{Proof:} Let $M$ be a poly-time decider for $B$
Then the poly-time decider for $A$ first computes $w' = f(w)$ and runs $M$ on $w'$.
This is clearly polynomial time.

\textbf{NP-Complete:} $L$ is NP-Complete if
1. $L \in \text{NP}$
2. $\forall A \in \text{NP}, A \leq_p L$