With P, we worked with TIME, which required there to be a deterministic decider for languages in the class. We can define a similar class of languages for nondeterministic deciders.

Let $t: \mathbb{N} \rightarrow \mathbb{N}$ be a function. The nondeterministic time complexity class $\text{NTIME}(t(n)) = \{L \mid L$ is a language decided by an $O(t(n))$ time nondeterministic TM$\}$. That is, a decider.

Note that the running time of a NTM $N$ is a function $f: \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the maximum number of steps $N$ takes on any branch of its computation on any input of length $n$.

![Diagram](image)

We define now the nondeterministic version of P.

$$NP = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$$

NP is a less wieldy class than P as it stands. NTMs are somewhat difficult to work with. Fortunately, there's an equivalent class that is way easier to use.
A verifier for a language $A$ is a deterministic decider $V$ where

$$A = \{ \omega \in \Sigma^* \mid \exists c \in \Sigma^*: \langle \omega, c \rangle \in L(V) \}.$$  

In this definition, $c$ is a certificate. Another way to phrase this definition is that, given a solution, a certificate, a witness, etc, $V$ can verify if $\omega$ is an instance of $A$.

**Ex**  
$\text{HAMPATH} = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t \}$.

A Hamiltonian path is a path that visits each vertex exactly once.

$\text{5BEADCE}$ is a Hamiltonian path.

Determining if there is a Hamiltonian path in a graph is hard. However, if we're given a path, it's easy to determine if it's a Hamiltonian path of the graph. Just check that it visits every vertex once and the edges are all valid.

The fact that verification is easy is actually a defining feature of NP. We'll define the class $\text{PV}$ to be

$$\text{PV} = \{ A \mid \text{there is a poly time verifier for } A \}.$$  

Note that we measure the runtime of verifiers only in terms of the length of $\omega$. So if $c$ has length $|c|^2$ and the verifier scans all of $c$, it runs in time $\Omega(|c|^3)$.

**Ex**  
Let $V$ be a verifier such that

- On input $\langle G, (V, E), s, t, c \rangle$,
  1. Check that $c$ visits each $v \in V$
  2. Check that $c$ starts at $s$ and ends at $t$
  3. Check that each $(v, c)$ is in $E$
  4. Reject if either is not true
  5. Accept

This verifier runs in polytime, clearly, so $\text{HAMPATH} \in \text{PV}$.
Alternatively, we can write down an NTM for HAMPATH.

**Ex: N** = "On input \((G=(V,E), s, t)\),

1) Write |V| numbers between 1 and |V| out, each number determined nondeterministically.
2) Check that \(p_1, \ldots, p_{|V|}\) has no repeats and reject if so.
3) Check that \(p_1 = s\) and \(p_{|V|} = t\) and reject if not.
4) For \(i = 1\) to \(|V|-1\)
   a) Check that \((p_i, p_{i+1}) \in E\) and reject if not.
5) Accept."

This shows that HAMPATH \(\in NP\).

From these examples, it should not surprise that \(PV = NP\).

\[\text{Hm} \quad PV = NP\]

**Pf:** It suffices to show \(PV \leq NP\) and \(NP \leq PV\).

- Suppose \(A \in NP\). On input \(w\), there exists some branch of computation that accepts \(w\) if \(w \in A\). In either case, it must finish in time \(p(w)\) for some polynomial \(p\) on all inputs \(w\). If we record a branch of computation as a sequence of states \(c\), then \(|c| \leq p(w)\). Then we can construct a poly time verifier that simulates \(M\) along \(c\) in poly time and accepts iff \(M\) accepts.

So \(A \in PV\).

In the opposite direction, assume \(A \in PV\). Then there is a poly time verifier \(V\) for \(A\). That runs in time \(O(nk)\).

We can construct a NTM \(N\) for \(A\) as follows.
N = "On input \( w \),

1) Nondeterministically select a string \( c \) of length \( O(w^k) \)
2) Run \( V(w, c) \) and accept if it does
3) Reject"

If \( \text{wEA} \), then there is some certificate \( c \) of length \( O(w^k) \)
such that \( V(<w, c>) \) accepts. Moreover, if \( w \notin A \), then
there is no such \( c \). Together, we get \( L(N) = A \) and
\( N \) runs in poly time.

\[ \square \]