User contains an NDTM M_L, for any given NP language, in such a way that $\forall x \in L$, M_L accepts x on minimum single branch in maximum $p_L(|x|)$ steps. Here, $p_L(\ )$ denotes a fixed polynomial depending on the machine.

• It is also known that any $x \notin L$ will not be accepted by M_L. Then, user can use a polynomial time to create $y = \left\langle M_L, x, \#^{p_L(|x|)} \right\rangle$ for the given x.

• Consider the previous argument, according to this argument, $x \in L$ if and only if $y \in U$. Therefore, ***U* is NP-hard**.

To show that *U* is also in NP, users have to create an NDTM M_U, which given an input $y = \left\langle M_L, x, \#^t \right\rangle$, simulates *M* on *x* for *t* steps.

• Every branches of *M* are guessed by M_U non-deterministically and accepts *u* if and only if *u* is accepted by *M*.

• Since the minimum length of input is *t* and user simulate *M* for maximum *t* steps, the running time is polynomial in the length of the input.

• Now it can be easily seen that the language *U* is exactly accepted by M_U, therefore, $U \in NP$.

Hence**, *U* is NP-complete.**