获得的答案 **返回**

The prefix free of regular expression is defined as,

$$PREFIX-FREE_{REX}=\{\langle R\rangle \mid R \text{ is a regular expression and } L(R) \text{ is prefix free}\}.$$

The string *x* is said to be a prefix of a string *y* if there exists a string *z* such that *xz=y*. If $x \neq y$ then *x* is said to be the proper prefix of *y*. A language is said to be prefix free if none of the strings have proper prefix of its members (i.e., strings).

If there exists a TM for any language, then it said to be decidable. Construct the Turing machine *M* to check the decidability of the language.

M = "On input R where R is a regular expression

1. If R is not a valid regular expression then reject.
2. Construct a DFA (Deterministic Finite Automata) for the language accepted by R i.e., L(R).
3. Run the DFS (Depth First Search) from the start state q_0 and remove all the unreachable states from it.
4. For each final state q_f, run the DFS to check if there exists path to any other final state from it or any self loop to itself.
5. If there exists a path to another final state or self loop from q_f then reject. Otherwise, accept."

Thus, there exists a TM for $PREFIX-FREE_{REX}$. Therefore, it is decidable.

The DFA accepts the prefix free language if and only if there is no accepting string goes through more than one final state. It can be checked by simple graph traversal technique like DFS. In this case, the DFA cannot have two final states for an accepting run. It is decidable.

In the case of context free grammar (CFG), it fails because every context free language cannot have deterministic PDA. In case of regular expressions, every regular expression has a DFA. If the PDA cannot be drawn deterministically for a language, then it cannot be restricted to have a single final state. Therefore, the similar approach fails to prove $PREFIX-FREE_{CFG}$ is decidable.