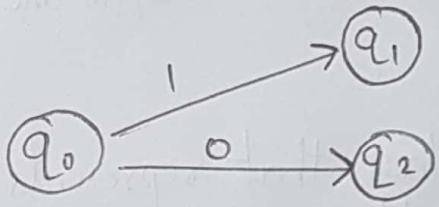
Deterministic & Non-Deterministic Algo

- **Deterministic Algo: —**

for a given particular input, the computer will always produce the same output going through the same states

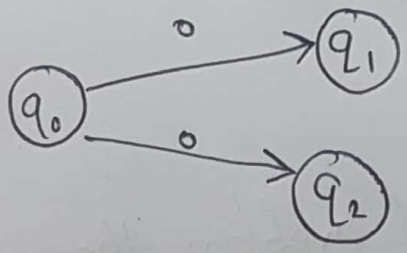
e.g.:



- **Non-Deterministic Algo: —**

for the same input, the compiler may produce different output in different runs. They can't be solved in polynomial time and also cannot determine what is the next step.

ex:



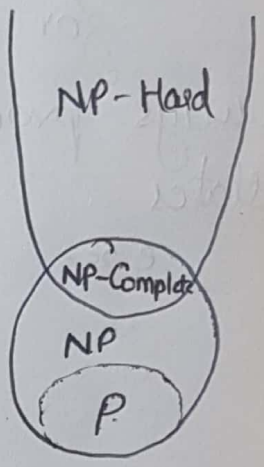
↳ Some terms with Non-Deterministic Algo:

① choice(X): chooses any value randomly from the set X
② failure(): denotes the unsuccessful soln.
③ success(): soln is successfull & current thread terminates.

NP-COMPLETENESS

- NP is a complexity class used to classify decision problems

  decision problem: a prob with YES/NO ans.



↳ P :—

it is a complexity class that represents the set of all decision problems that can be solved in polynomial time.

ex: Selection Sort ($O(n^2)$).

↳ NP :—

it is a complexity class that represents the set of all decision problems for which the instances where the answer is YES have proof that can be verified in polynomial time

ex:- Integer factorization

it is a decision based problem the given integers 'n' & 'm' there exists an int 'f' where $1 < f < m$ and is a factor of 'n'.

Validatn: checking in polynomial time by performing the division n/f.

↳ NP Complete:—

a complexity class which represents the set of all problems X in NP for which it is possible to reduce any other NP problem Y to X in polynomial time.

ex: Vertex Cover problem

↳ NP Hard:—

a problem X is NP-hard, if there is an NP-complete problem Y, such that Y is reducible to X in polynomial time.

ex: TSP

NOTE:

- out of all the other types NP Complete is the hardest of all to solve
- based on Non-Deterministic Turing Machine