(1) Divide and Conquer

(2) Greedy

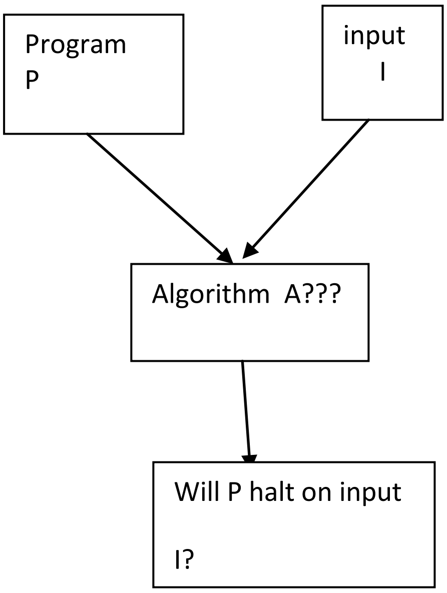
(3) dynamic programming useful for solving some optimization problems

But there are many real world problems which cannot be solved in polynomial time using any of the above methods or not yet known how the above methods can be applied to solve those problems in polynomial time.

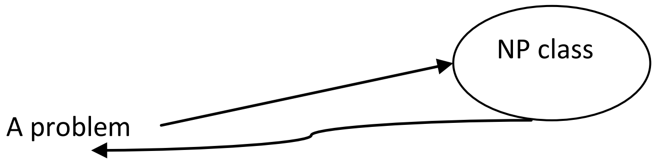
search, sorting, finding shortest path on a graph, finding spanning tree given a graph...... for which we know polynomial time algorithm.

NP-completeness

Towers of Hanoi: O(2^n)



Whether A will be able to detect that P will fall in infinite loop or not with the input I.



Is there any problem which is NP-complete?

Is there any problem x, which belongs to NP class and all problem in NP class reducible to X?

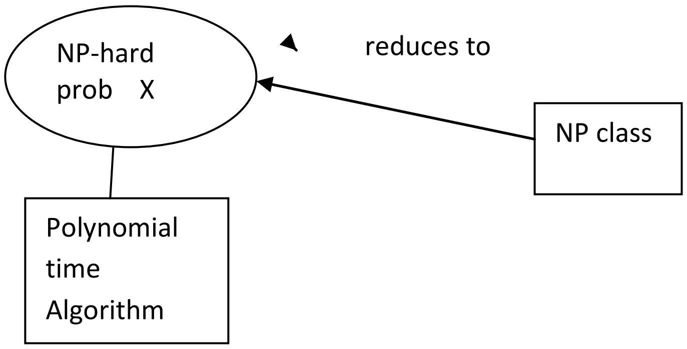
COOK's theorem

Satisfiablity problem is problem which belongs to NP class and all problems in NP class are reducible to Satisfiablity problem.

SAT problem is a NP-hard problem

SAT problem is NP-complete problem

If we know the polynomial time algorithm for NP hard problem, then there will have the polynomial time algorithm for every problem in NP , then P=NP



is P=NP?

P is proper sub set of NP?

For any problem X belonging to class P , solutions can be also verified in polynomial time and solutions can be computed in polynomial time. So, X belongs to NP

There exists some problems Y belonging to NP class for which solutions can be also verified in polynomial time, but solutions cannot be computed in polynomial time.

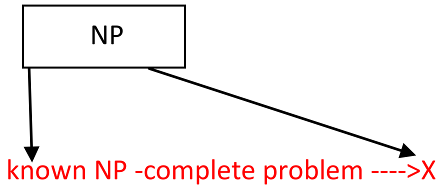
Y does not belong the class P

Given a problem, how to check a problem is NP complete

(1) Check whether the problem is NP or not. To prove it, check whether a given solution is verifiable in polynomial time

(2) Whether it is NP-hard (not easy).

We can choose any one of the known NP-complete problem or NP-hard problem X and use reductions to reduce X to your problem



NP

known NP -complete problem ---->X