**NATIONAL INSTITUTE OF TECHNOLOGY, DELHI**

**ASSIGNMENT**

**DESIGN AND ANALYSIS OF ALGORITHMS**

**P and NP PROBLEMS**



Submitted By : VARTIKA CHATURVEDI

Roll No. : - 181210056

BRANCH : CSE 2nd year(group-II)

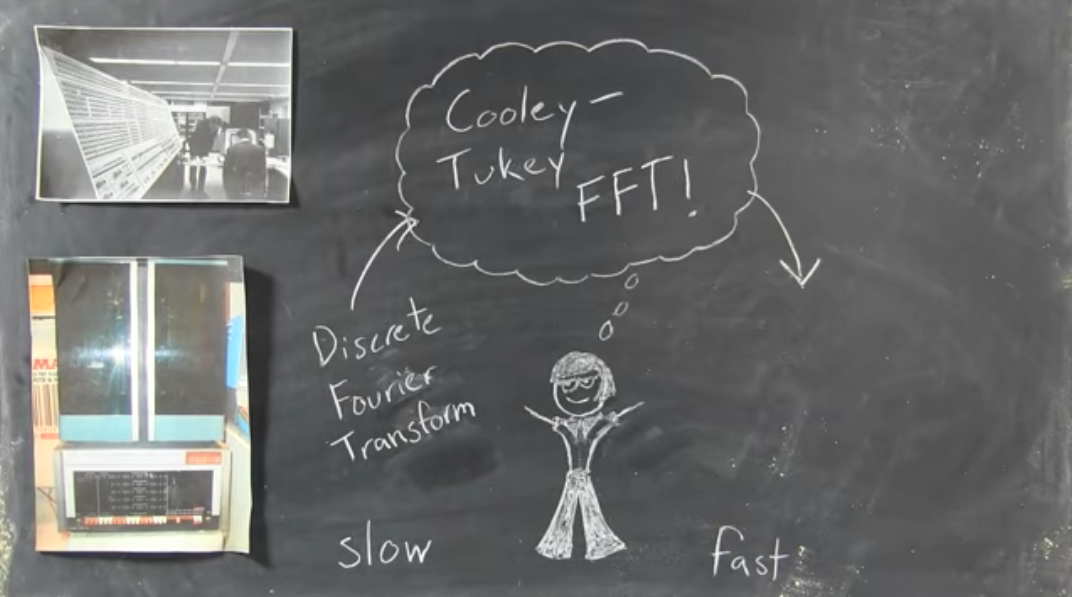
In 2000, the Clay Institute offered 1 Million Dollar each for the solutions to 7 key Math problems:

The Millennium Prize Problems:

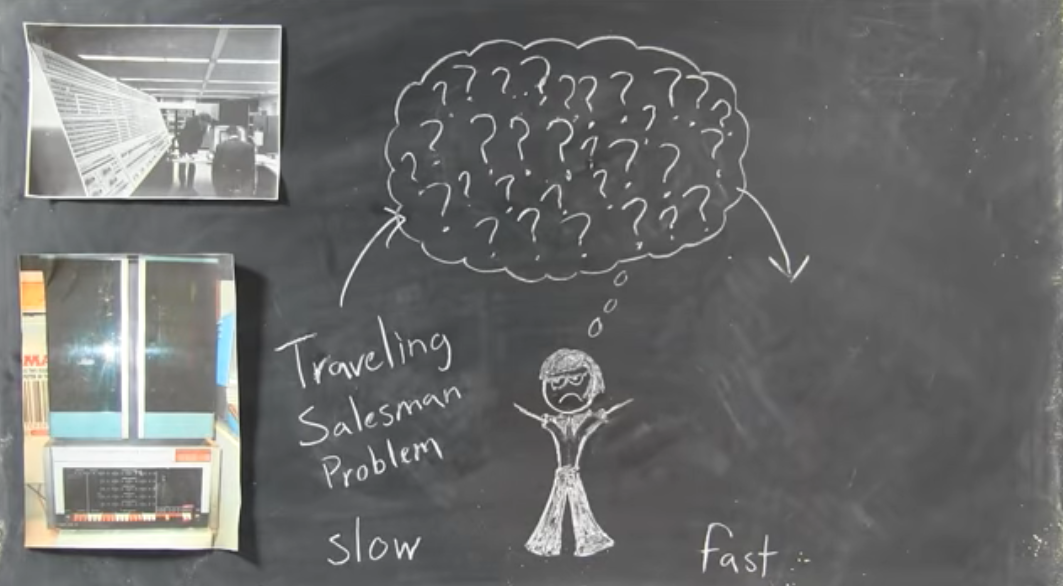
1. Birch and Swinnerton- Dyer Conjecture
2. Hodge Conjecture
3. Navier- Stokes Equations
4. P vs NP
5. Poincare Conjecture
6. Reimann Hypothesis
7. Yang- Mills Theory

P vs NP problem was the most recent problem (1971) and was the easiest to explain.

Sometimes, the first program anyone could think for a particular problem would be unworkably slow, but then overtime people would come up with clever ways to make it faster.



For other problems, people were still struggling for better solutions.



**P vs NP**

* P is a class that basically includes all the problems that can be solved by a reasonably fast program.

1. Minimum spanning tree problems using prim’s and Kruskal algorithms.

1. Fractional knapsack problem.

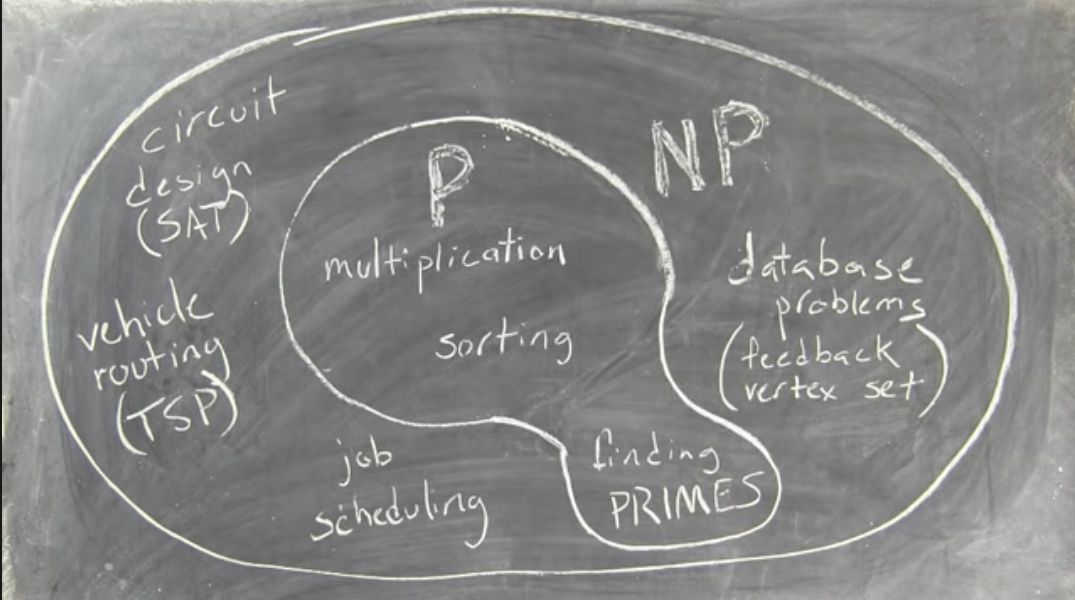
* Around and including P, there is a class called NP.
* That is all the problem where, if you’re given a correct solution you can at least check it in a reasonable amount of time.

1. If we are able to solve a problem in polynomial time, we will

surely be able to verify in polynomial time, so every P problem will also be a NP problem.

1. Travelling salesman problem :- we are not able to find polynomial time solution for this problem but we can verify this in polynomial time so this a NP problem but not a P problem.

* Often, we find that an NP problem was actually a part of P and we would have a fast program.



**Polynomial time reduction algorithm :-**

To prove P = NP we have to prove that every problem which lies in NP can be solved in polynomial time

There are millions of NP problems we can not solve each problem to prove this, here comes reduction.

A problem ‘A’ is said to be polynomial time reducible to a problem ‘B’ if :-

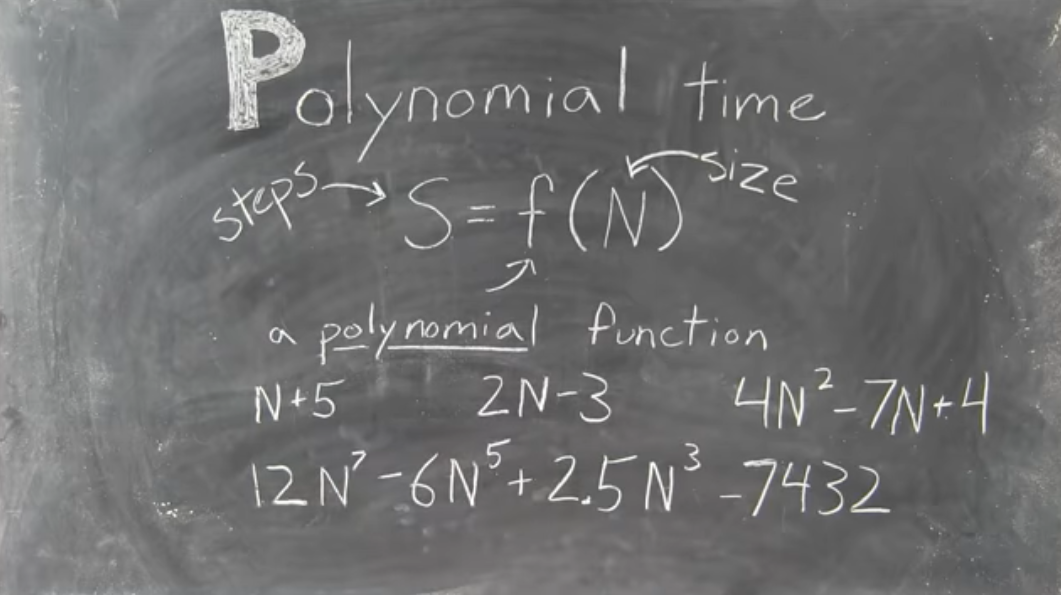
1. Every instance ‘a’ of ‘A’ can be transformed to some instance ‘b’ of ‘B’ in polynomial time.
2. Answer of ‘a’ is ‘YES’ if and only if answer of ‘b’ is ‘YES’.

So

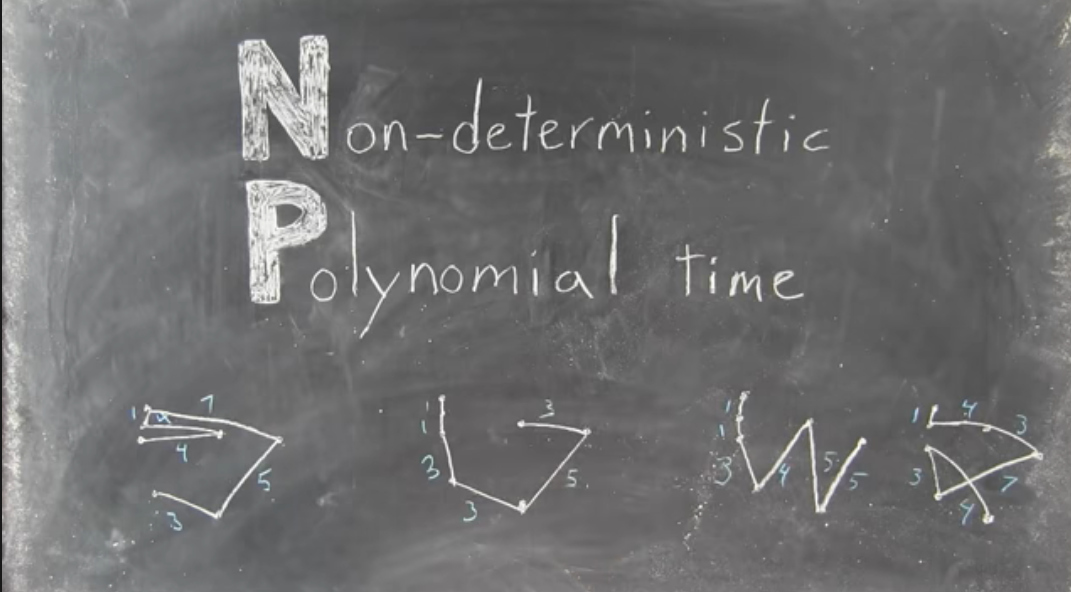
If A is reduced to B in polynomial time then :-

* If B is easy then A is also easy.
* If B is in P then A is also in P.
* If this is proven that A cannot be solved in polynomial time then B is also cannot be solved in polynomial time.
* If A is not in P then B is also not in P

**P Type:**



**NP Type:**



**NP Complete Problems:**

* After a long time researchers realized that dozens of NP problems that they were struggling with were essentially all the same problem with some easy polynomial time complications thrown here and there.

Eg. Sudoku, protein folding, puzzles etc.

* It consists of really hard parts of NP Problem.

**EXP Class:**

* Beyond NP there are even harder classes of problems- EXP the class of problems including figuring out the best move in CHESS, that takes exponential time to even check.

**NP Hard:**

* The problems that are at least as hard as NP complete is also known as NP Hard type of problems.

**Co-NP:**

* The class of problems where instead of being easy to check the right answers, it is easy to exclude wrong answers, which may or may not be the same of NP.

