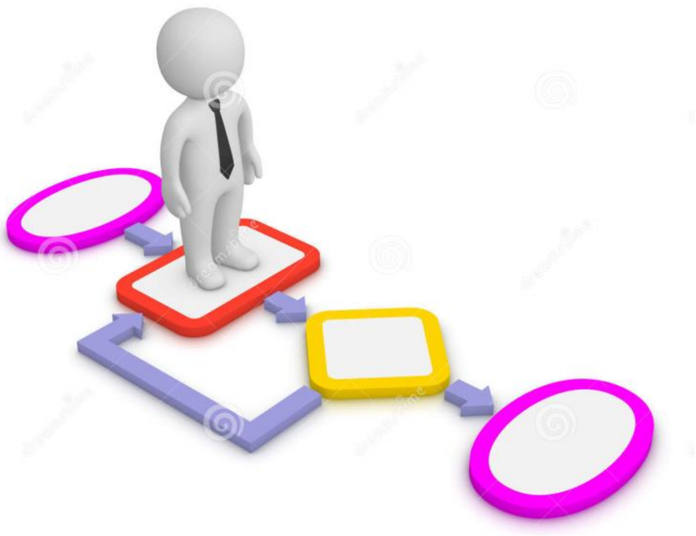
Design & Analysis of Algorithms
CS 4103

DR. SAMAN RIAZ

P, NP, NP-Complete Problems



Saman Riaz (PhD)

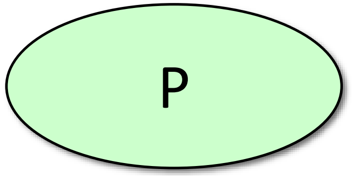
Polynomial Problems (P Family)

The set of problems that can be *solved* in polynomial time

n^c = polynomial
C^n = Exponential

These problems form the P family

All problems we covered so far are in P



Examples: Polynomial time

Linear Search----n
Binary Search----logn
Insertion Sort-----n^2
Merge Sort--------nlogn

Examples: Exponetial time

0/1 Knapsack----2^n
Traveling SP-----2^n
Sum of Subsets-----2^n

Nondeterministic Polynomial (NP Family)

The set of *decision* problems that can be *verified* in polynomial time

Not necessarily *solvable* in polynomial time

NP

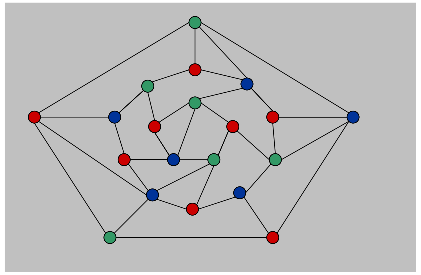
What does it mean:

"decision problem"
"verifiable"

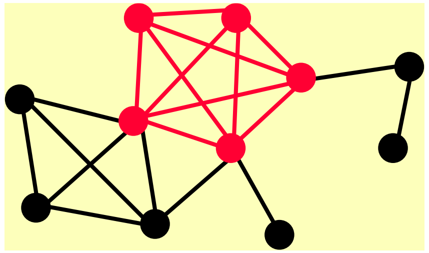
Nondeterministic Polynomial (NP Family) (Cont’d)

Decision Problem

- Problem where its outcome is either Yes or No



Is there a way to color the graph 3-way such that no two adjacent nodes have the same color?



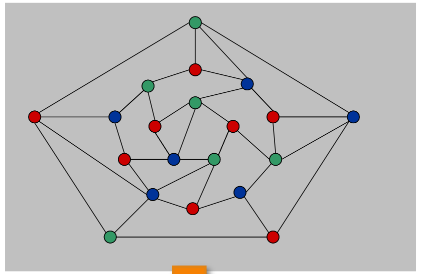
Is there a clique of size 5?
Clique: Set of vertices where each pair of vertices is connected.

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

Is there assignment of 0’s and 1’s to these Xi variables that make the expression = true?

- **Verifiable in Polynomial Time**
  - **If I give you a candidate answer, you can verify whether it is correct or wrong in polynomial time**
  - **That is different from finding the solution in polynomial time**

Verifiable in Polynomial Time



If I give you color assignment:

>> Check the number of colors is 3

>> Each that no two vertices are the same O(E)

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

If I give you assignment for each Xi:

>> if the expression is True

- But the find a solution from scratch, it can be hard

P vs. NP

P is definitely subset of NP

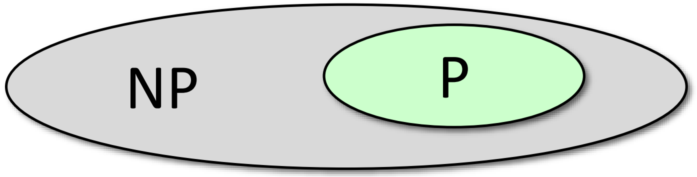
- Every problem with poly-time solution is verifiable in poly-time

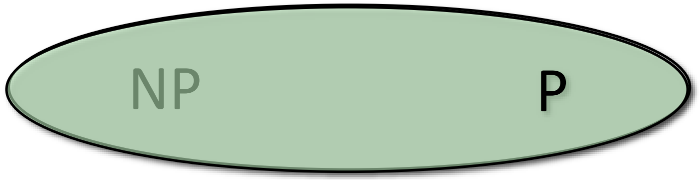
Is it proper subset or equal?

- No one knows the answer

- P=NP most famous problem in CS.
- [Clay Institute]() is offering one million dollar

Most guesses are
leaning towards P ≠ NP



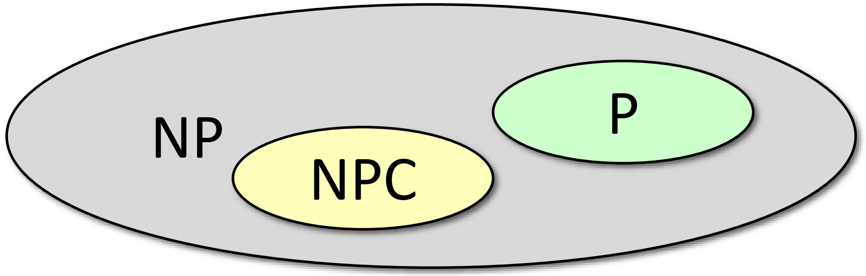


- **NP family has set of problems known as "NP-Complete"**
  - Hardest problems in NP
  - No poly-time solution for NP-Complete problems yet

NP-Complete (NPC)

A set of problems in NP

- So, they are decision problems
- Can be verified quickly (poly-time)



They are hardest to solve

- The existing solutions are all exponential
- Known for 30 or 40 years, and no one managed to find poly-time solution for them
- Still, no one proved that no poly-time solution exist for NPC problems

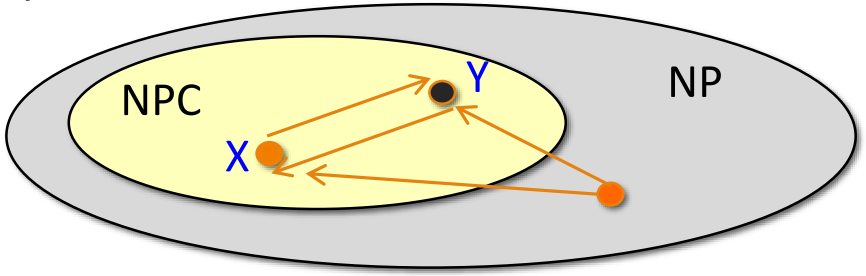
Property in NPC problem

- Problem X is NPC if any other problem in NP can be mapped (transformed) to X in polynomial time

NP-Complete (NPC) Cont'd

Property in NPC problems

- Problem X is NPC if any other problem in NP can be mapped (transformed) to X in polynomial time
  - So, Any two problems in NPC must transform to each other in poly-time
  - X ----PolyTime-------> Y
  - Y -----PolyTime ------> X



- **This means if any problem in NPC is solved in poly-time ➔ Then all NPC problems are solved in Poly-Time**
  - **This will lead to P = NP**

How to prove a new problem “Y” is NPC?

Show it is in NP

- It is a decision problem
- Verifiable in poly-time

Select any problem from NPC family (say X)

- Show that X transforms to Y in poly-time

NP-Hard Family

It is a family of problems as hard as NPC problems

But they are not decision problems

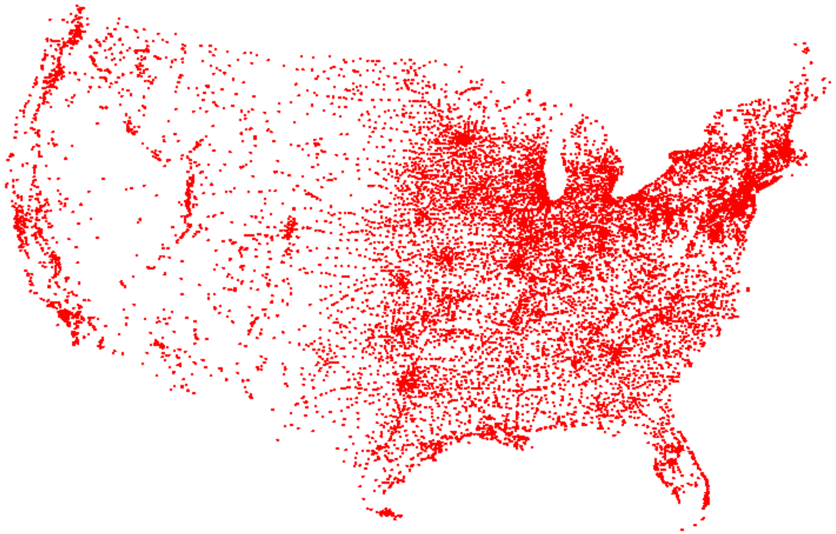
- Can be any type

NP-Hard problems have exponential time solutions

NP-Hard Example: Travelling Salesman Problem

- **Given a set of n cities and a pairwise distance function d(u, v)**

- **What is the shortest possible route that visits each city once and go back to the starting point**

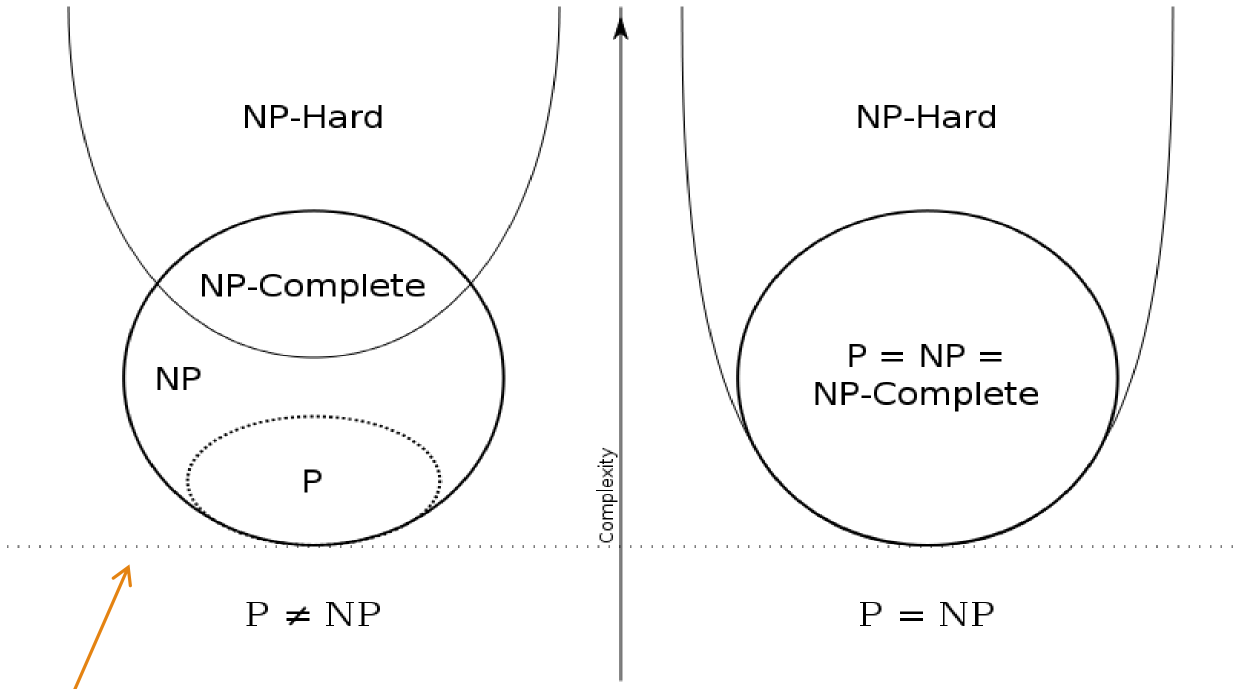


• Imagine you need to visit 5 cities on your sales tour. You know all the distances. Which is the shortest round-trip to follow?

An obvious solution is to check all possibilities. But this only works for small problems. If you add a new city it needs to be tried out in every previous combination.

So this method takes "factorial time": t = n! (Actually t = (n-1)! but it is still factorial.)

Eular Diagram



Most probable