# UCSC Silicon Valley Extension
## Advanced C Programming

## NP Complete

Radhika Grover

# Overview

- P vs NP
- Example of class NP
- NP, NP-hard, NP-complete

# P vs NP problem

- Do questions exist
  - Whose answer can be **verified** quickly
  - But which require an impossibly long time to solve ?
- Proposed by Stephen Cook and Leonid Levin in 1971
- Unsolved to date ( a Millennium prize problem ) http://www.claymath.org/millennium-problems

# Example of class NP

- Finding the prime factors of a large number
    - Given a solution, can **verify** it quickly (using multiplication)
    - Can we **solve** this problem quickly ?
- Used in cryptography

# Decision problem

- Check if something (such as a proposed solution) is true
- Yes/No answers

# Knapsack problem

- Decision form
- Optimization form
- Special case : subset-sum

# Knapsack problem - decision

Example : Knapsack can hold a weight W, each item i has weight $w_i$ and value $v_i$

Can you find the items with a value of at least V that don't exceed the knapsack's capacity of W?

Can check easily if this a solution : $\{v_1, v_3, v_5\}$

# Knapsack problem - optimization

Example : Knapsack can hold a weight W, each item i has weight $w_i$ and value $v_i$

Can you find the maximum value of the items that don't exceed the knapsack's capacity of W

Cannot check a given solution without solving problem

# Subset - sum

Given a set of integer and an integer s, is there any non-empty subset with a sum of s?

Example : Given the set {-3, -5, 1, 10, 7} and s = 3

Answer = {7, -5, 1}

# Set cover problem

Example: Universe U = {1, 2, 3, 4, …, 10}

Sets S = { {1, 2, 4, 9}, {3, 5}, {3, 5,7}, {6, 8, 10} }

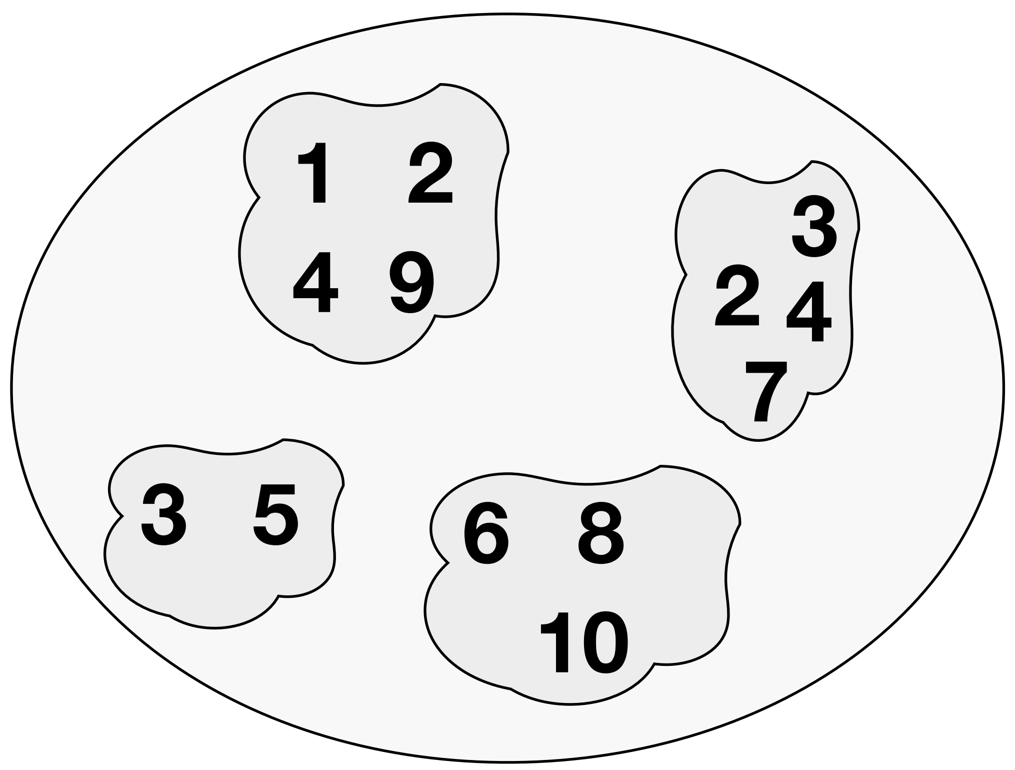
S is a **cover** for U because the union of the sets contains all the elements of u

Decision form: Is there a set cover of at **least size k**?

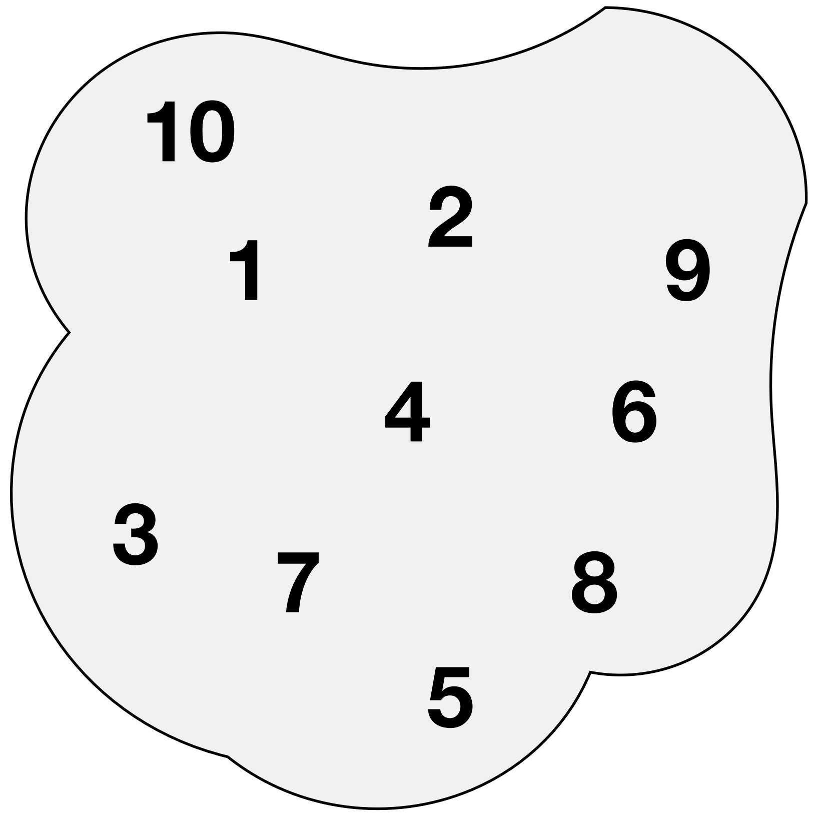
k = 3, yes {{1, 2, 4, 9},{3, 5, 7},{6, 8,10}}

# Set cover problem

- Optimization form: what is the set cover with the **fewest** number of sets ?




set S




universe

# Boolean satisfiability problem (SAT)

Find an assignment of TRUE/FALSE values to the variable of a boolean expression that makes the expression TRUE

Example : $f = (a + b + c)(a' + b)(a + b' + c')$

Solution : f is true with $a = b = c = $ true

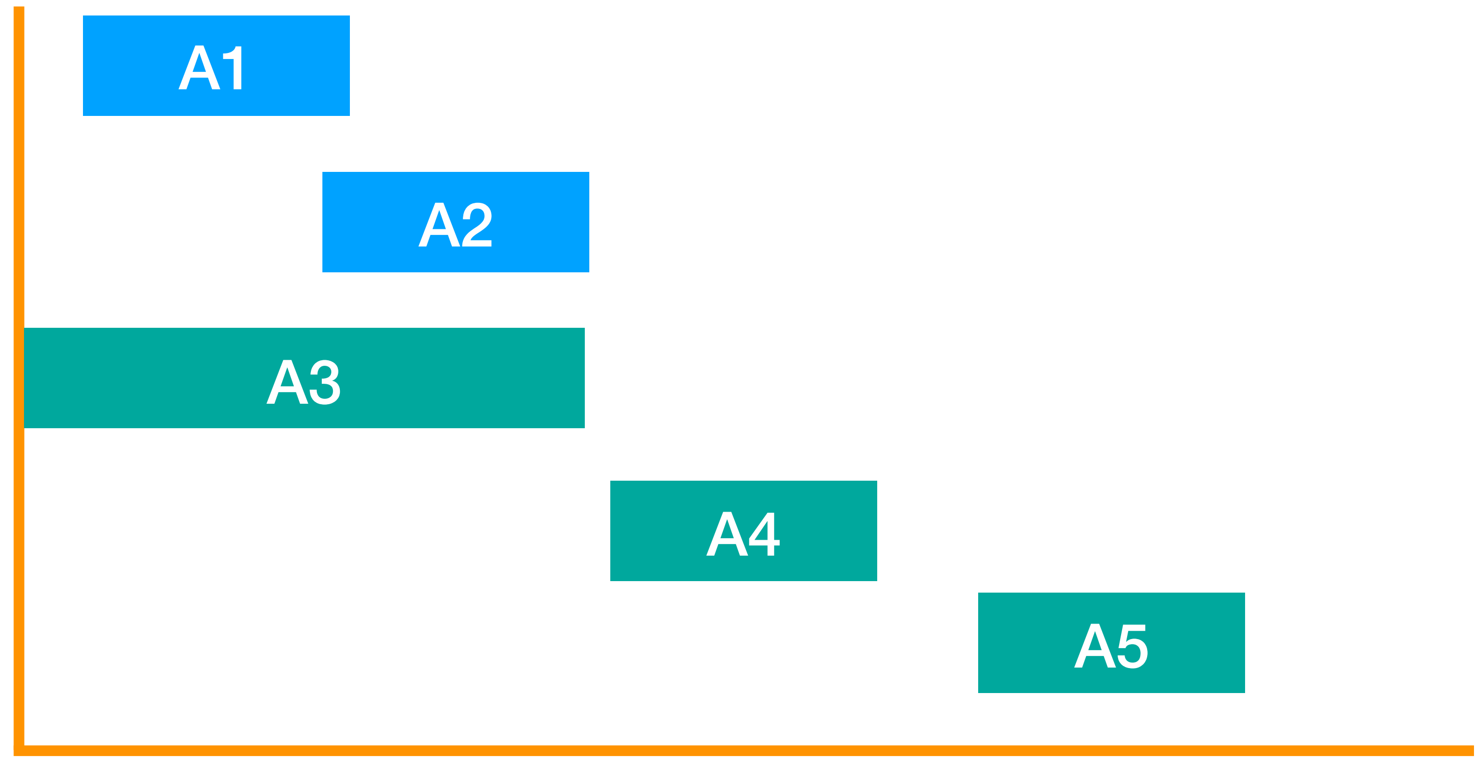
# Traveling salesman problem

- A salesman has to tour n cities and there is an integer cost to travel between cities
- Find the tour that meets these condition
  - Has lowest cost and
  - Visits each city exactly once and
  - Ends at the city it starts from

# Activity selection

- Selection of non-conflicting activities to perform within a given time frame
- Select the maximum number of activities that can be performed by a single person or machine

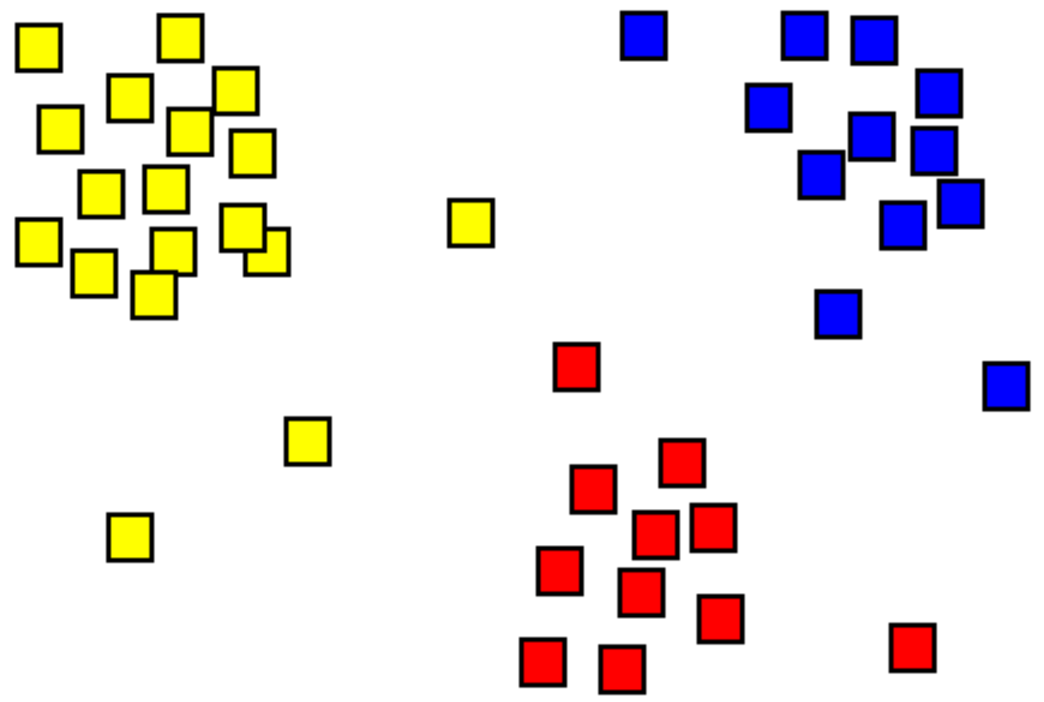
Note : person can only work on a single activity at a time.




Greedy polynomial-time algorithm exists :sort activities by finish time, pick one with earlier finish time

# K-means clustering



- NP-hard problem
- Partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster.

# Class P

Problem is in class P when :

- An algorithm for solving the problem, runs in polynomial time (i.e not exponential time)

Examples :

- Minimum weight spanning tree
- Edmonds - Karp for maximum flow
- Some sorting algorithms

# Class NP

Problem is in class NP (non-deterministic polynomial time)

- An algorithm can **verify** a proposed solution to the problem in polynomial time

- Not known if problem is in P

Examples :

- Scheduling jobs on machines

# Class NP - complete

- Problem A is NP-complete when :
    - A is in class NP and
    - Every problem in class NP is reducible in polynomial time to A
- An NP - complete problem is as hard or harder than all problem in NP

# Class NP - complete example

- 3 - SAT (every NP problem can be reduced to it)
- Sudoku
- 3 colorability
- Knapsack problem

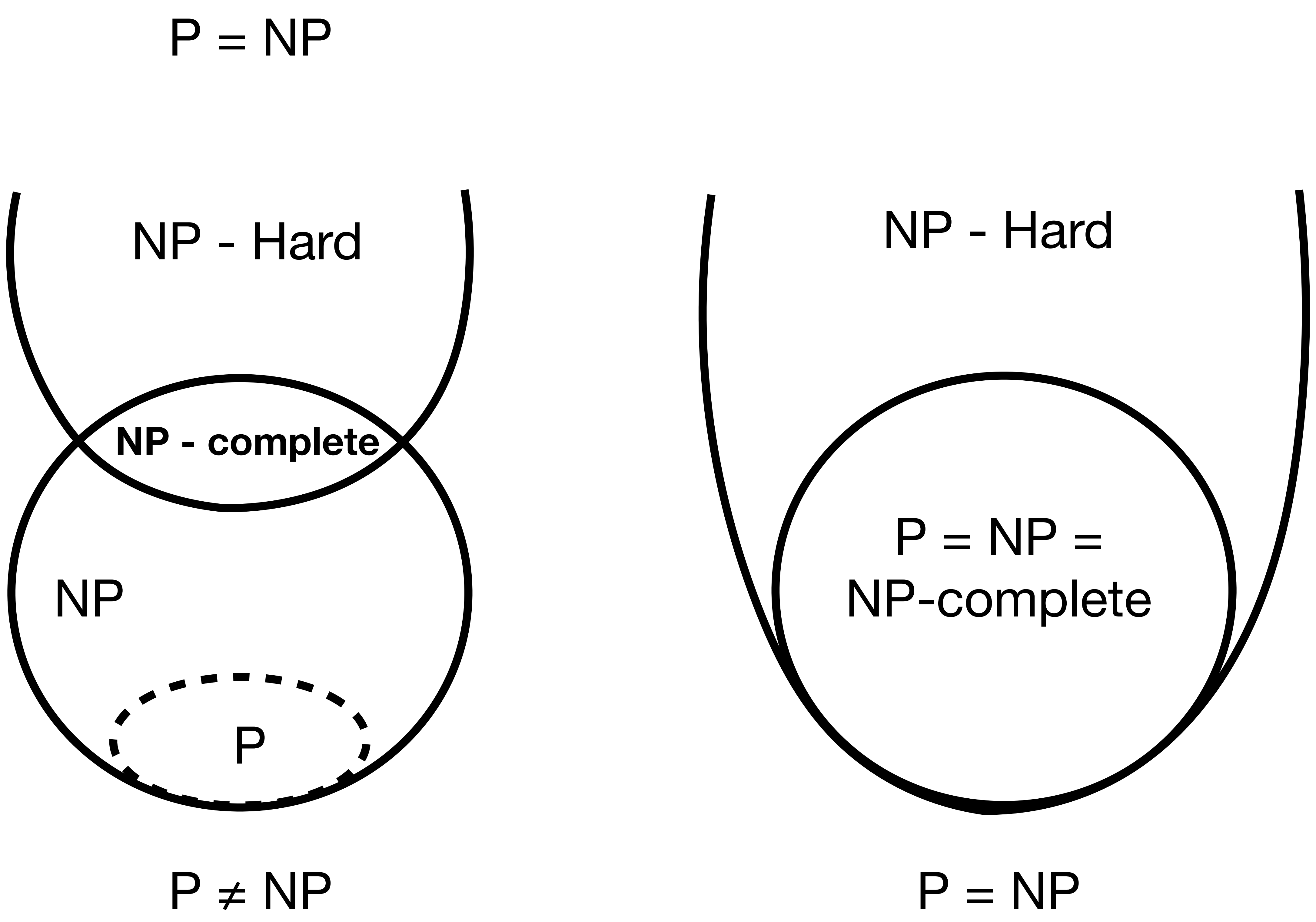
# NP - hard

- NP - complete problems can be **verified** by an algorithm in polynomial time
- NP - hard problems **cannot be verified** in polynomial time

Example :

- Independent set
- Vertex cover

# Diagram




**P = NP Traveling Salesman - the movie**

# NP - complete reduction

To show new problem is NP complete :

- Prove it is in NP
- Reduce to another NP-complete problem to it

# NP - completeness

NP - complete problems that appear deceptively similar to class P

| NP-complete | Polynomial |
|---|---|
| Hamiltonian path | Euler path |
| Longest path between two vertices in a graph | Shortest path between two vertices in a graph |
| 3-SAT | 2-SAT |
| Steiner tree, k-minimum MST | minimum spanning tree |

# Some problems in class P

- Shortest path problem (BFS)
- Sorting problem (Various algorithms)
- Solving linear equations (Gaussian elimination )
- Linear programming problem
- Minimum spanning tree problem
- Maximum flow problems
- Route inspection problem (using T-join)
- 2-color graph coloring

# NP-complete problems

- Boolean satisfiability problem
- Decision from of set cover problem (related to vertex cover, edge cover, set packing , etc
- Hamiltonian cycle problems
- Integer linear programming
- Knapsack problem (decision form)
- Decision form of graph coloring

# NP-hard problem

- Traveling salesman problem
- Knapsack problem (optimizing form)
- Set cover (optimizing form)

# References


P versus NP problem, Stephen cook

www.claymath.org/sites/default/files/pvsnp.pdf

http://www.claymath.org/millennium-problems/p-vs-np-problem

en.wikipedia.org/wiki/integer-factorization

en.wikipedia.org/wiki/NP-completeness

“Computer scientists find new shortcuts for infamous traveling salesman problem” https://www.wired.com/2013/01/traveling-salesman-problem/

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Greedy/actSelectionGreedy.htm