

NPTEL MOOC, JAN-FEB 2015
Week 8, Module 6

DESIGN AND ANALYSIS OF ALGORITHMS

Intractability: Checking algorithms

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
<http://www.cmi.ac.in/~madhavan>

Efficient algorithms

- * Shortest path, minimum cost spanning tree, maximum flow, ... have polynomial time algorithms
- * Search space for solutions is exponential
 - * All possible paths, all possible spanning trees, all possible subsets of edges, ...
 - * Brute force: scan exponential possibilities and choose the best

Efficient algorithms ...

- * Do all problems admit such efficient solutions?
- * Unfortunately not
- * For a large class of “natural” problems, no shortcut is known to exist

NPTTEL

Generating vs checking

- * A teacher assigns homework:
 - * Factorize a large number that is the product of two primes
- * **Student:** Given N , find p, q such that $pq = N$
 - * Generate a solution
- * **Teacher:** Given a student's solution p, q , verify that $pq = N$
 - * Check a solution

Checking algorithms

- * Checking algorithm C for problem P
- * Takes in an input instance I for P and a solution “certificate” S for I
- * C outputs yes if S represents a valid solution for I , no otherwise
- * For factorization, I is N , S is $\{p,q\}$ and C involves verifying that $pq = N$

Boolean satisfiability

- * Boolean variables x, y, z, \dots
- * $\neg x$ — negation of x , $x \vee y$ — x or y , $x \wedge y$ — x and y
- * Clause — formula C of the form
$$(x \vee \neg y \vee z \vee \dots \vee w)$$
 - * Disjunction of literals (variables, negated variables)
- * Formula — conjunction of clauses
$$C \wedge D \wedge \dots \wedge E$$

Boolean satisfiability

- * Assign suitable values {True,False} to x,y,z,\dots so that the formula evaluates to true

$$(x \parallel y \parallel z) \& (x \parallel !y) \& (y \parallel !z) \& (!x \parallel !y \parallel !z)$$

- * $x = \text{True}, y = \text{True}, z = \text{False}$ makes this true

$$(x \parallel y \parallel z) \& (x \parallel !y) \& (y \parallel !z) \& (z \parallel !x) \& (!x \parallel !y \parallel !z)$$

- * Now there is no satisfying assignment

Boolean satisfiability

- * Generating a solution
 - * Try each possible assignment to x, y, z, \dots
 - * N variables — 2^N possible assignments
 - * Is there a better algorithm? Not known
- * Checking a solution
 - * Given formula F and valuation $V(x)$ for each x , substitute into formula and evaluate

Boolean satisfiability

- * Input format is important
- * Suppose a clause is a conjunction of literals ...

$$(x \ \& \ !y \ \& \ z \ \& \ \dots \ \& \ w)$$

- * ... and a formula is a disjunction of clauses

$$C \ || \ D \ || \ \dots \ || \ E$$

- * Each clause forces a unique valuation
- * Try each clause in sequence

Travelling salesman

- * A network of cities with distances between each pair
- * A **complete** graph $G = (V, E)$ with edge weights
- * Find the shortest tour that visits each city exactly once
- * Simple cycle x, y, z, \dots, x visiting all vertices, of minimum cost

Travelling salesman

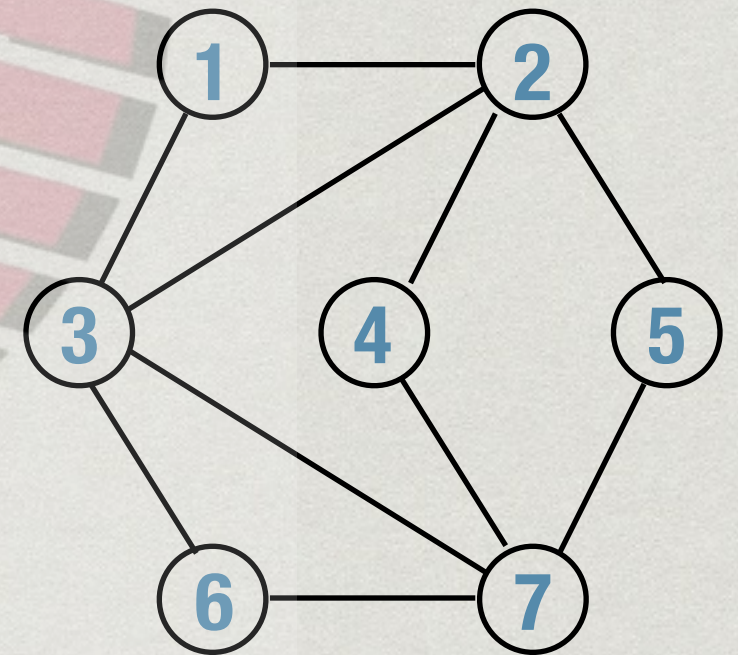
- * Designing a checking algorithm
- * Checking algorithm must give a yes/no answer
- * Given a graph G and a proposed solution S we can
 - * Verify that S is a cycle
 - * Compute its cost
 - * How to check that S is the least cost cycle?

Travelling salesman

- * Transform the problem
- * Is there a tour with cost at most K ?
- * Now, given a solution S , we can check it
- * For the original problem, cost is at most the sum of all the edge weights in the graph
- * Find optimum K — test different values using binary search

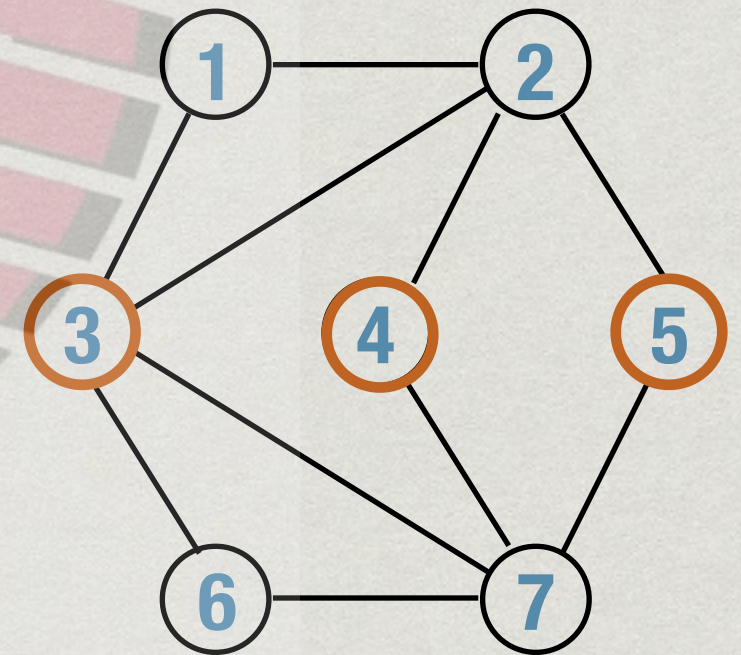
Independent set

- * u, v are independent if there is no edge (u, v)
- * U is an independent set if each pair $\{u, v\}$ in U is independent
 - * Constitute a neutral committee where none of the members know each other
- * Find the largest independent set in a given graph
- * Checking version: Is there an independent set of size K ?



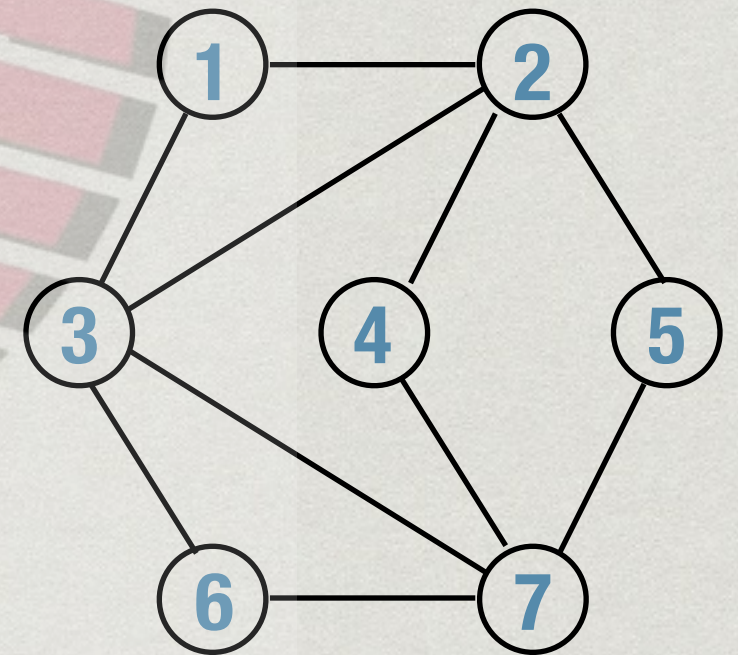
Independent set

- * u, v are independent if there is no edge (u, v)
- * U is an independent set if each pair $\{u, v\}$ in U is independent
 - * Constitute a neutral committee where none of the members know each other
- * Find the largest independent set in a given graph
- * Checking version: Is there an independent set of size K ?



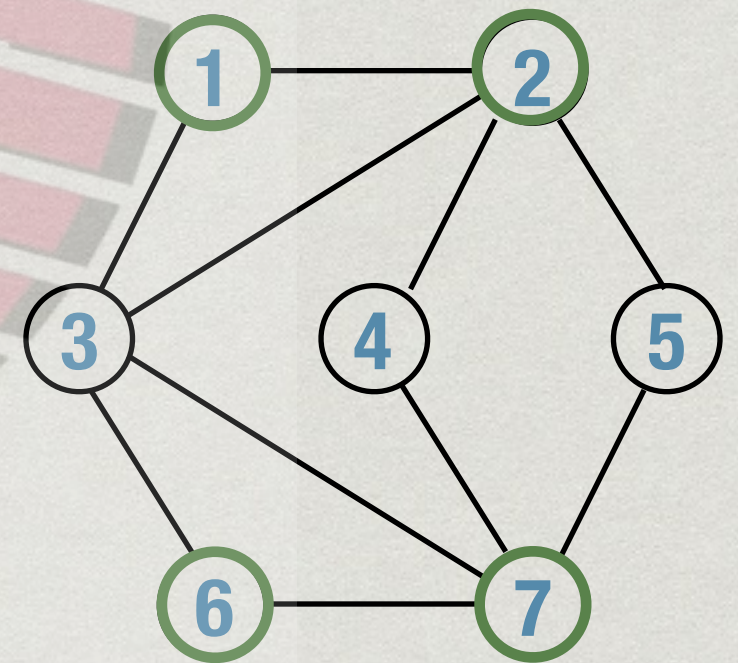
Vertex cover

- * Node u covers every edge (u,v) incident on u
- * U is a vertex cover if each edge in the graph is covered by some vertex in U
 - * Position surveillance cameras at intersections to watch all roads
- * Find the smallest vertex cover in a given graph
- * Checking version: Is there an vertex cover of size K ?



Vertex cover

- * Node u covers every edge (u,v) incident on u
- * U is a vertex cover if each edge in the graph is covered by some vertex in U
 - * Position surveillance cameras at intersections to watch all roads
- * Find the smallest vertex cover in a given graph
- * Checking version: Is there an vertex cover of size K ?



Connecting independent set and vertex cover

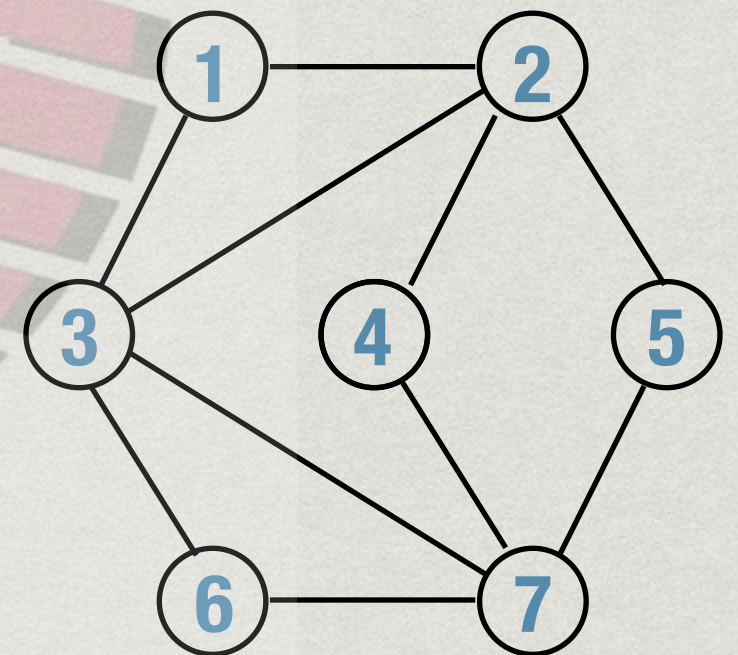
- * U is an independent set of size K iff $V-U$ is a vertex cover of size $N-K$

- * (\Rightarrow)

Every edge (u,v) has at most one end point in U , so at least one end point in $V-U$

- * (\Leftarrow)

For any edge (u,v) , at least one endpoint in $V-U$, so no edges (u,v) within U



Connecting independent set and vertex cover

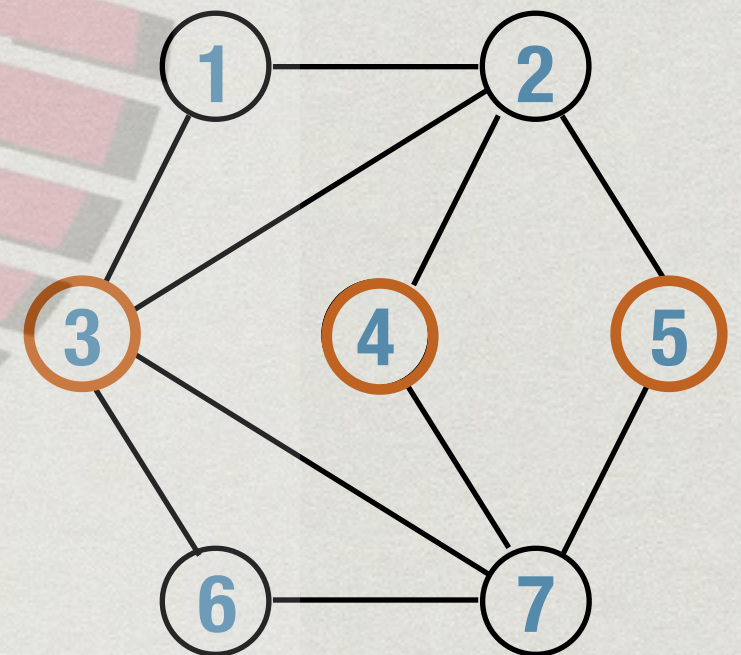
- * U is an independent set of size K iff $V-U$ is a vertex cover of size $N-K$

- * (\Rightarrow)

Every edge (u,v) has at most one end point in U , so at least one end point in $V-U$

- * (\Leftarrow)

For any edge (u,v) , at least one endpoint in $V-U$, so no edges (u,v) within U



Connecting independent set and vertex cover

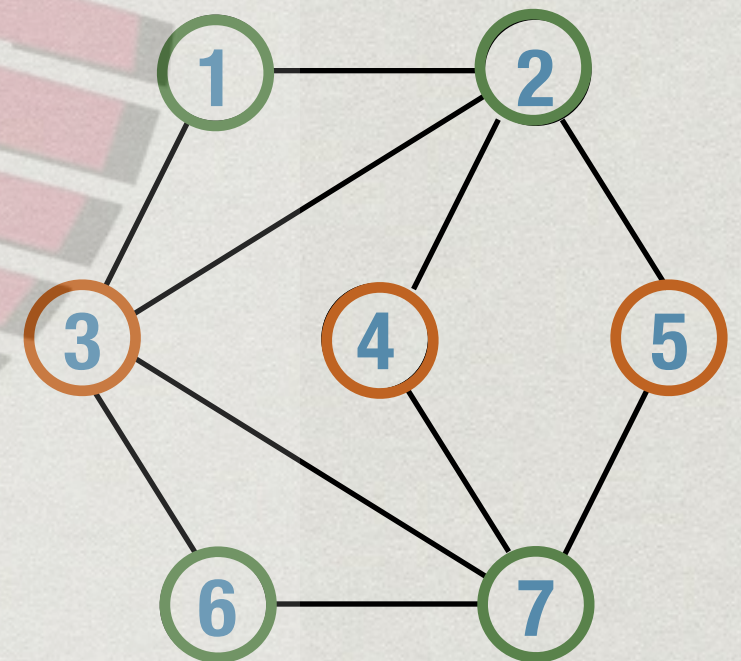
- * U is an independent set of size K iff $V-U$ is a vertex cover of size $N-K$

- * (\Rightarrow)

Every edge (u,v) has at most one end point in U , so at least one end point in $V-U$

- * (\Leftarrow)

For any edge (u,v) , at least one endpoint in $V-U$, so no edges (u,v) within U



Reductions

- * Independent set and vertex cover reduce to each other
- * Recall: if A reduces to B and A is intractable, so is B
- * Many pairs of checkable problems are inter-reducible
 - * All “equally” hard

Algorithm for A

