CS 5003: Parameterized Algorithms

Lectures 1 and 2

Krithika Ramaswamy

IIT Palakkad

* Text Books
    * Parameterized Algorithms by Cygan et al.
    * Kernelization by Fomin et al.

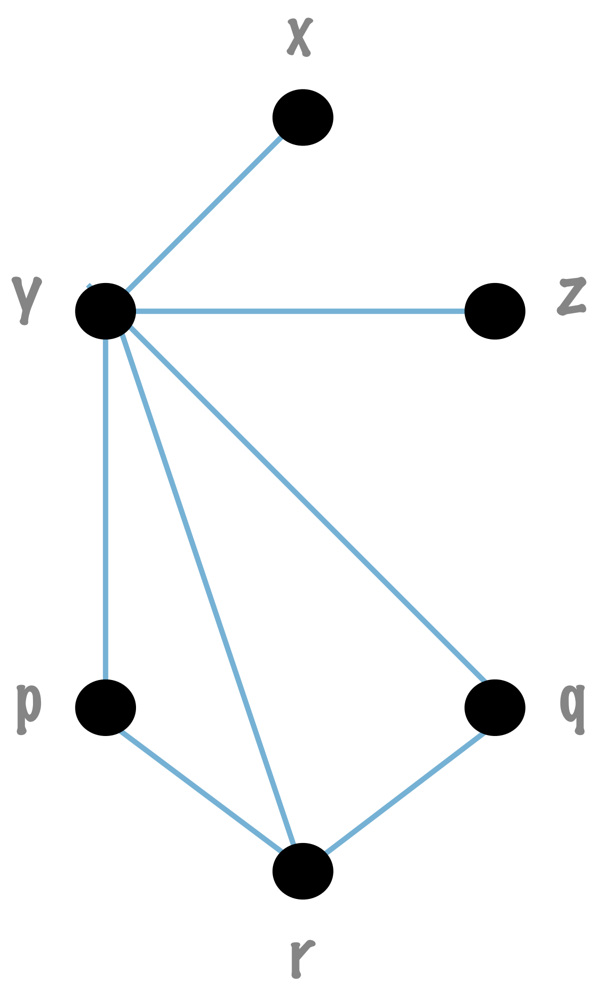
* Evaluation Scheme
    * Quiz 1 - 20%
    * Quiz 2 - 20%
    * Assignments - 15%
    * Endsem - 45%
    * Presentation (optional)

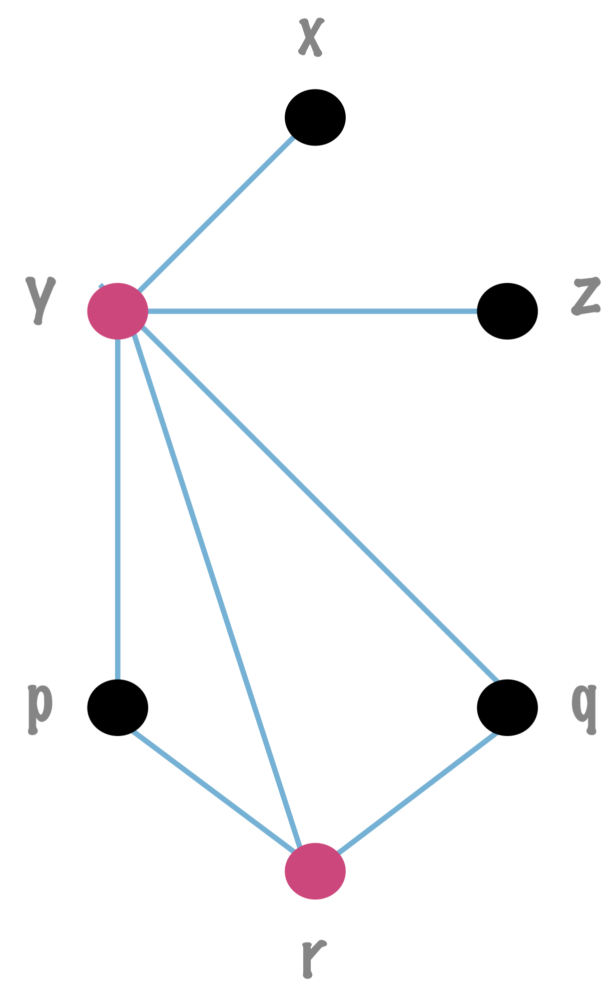
Review of Basic Concepts: Algorithms and Problems

- Algorithm
  - A finite sequence of steps to solve a problem
- Computational Problem
  - Input encoded in binary
  - Specification of the output desired for each input
- Instance of a problem
  - Specific input to the problem

Review of Basic Concepts: Vertex Cover

Vertex cover - set of vertices that has at least one endpoint of each edge

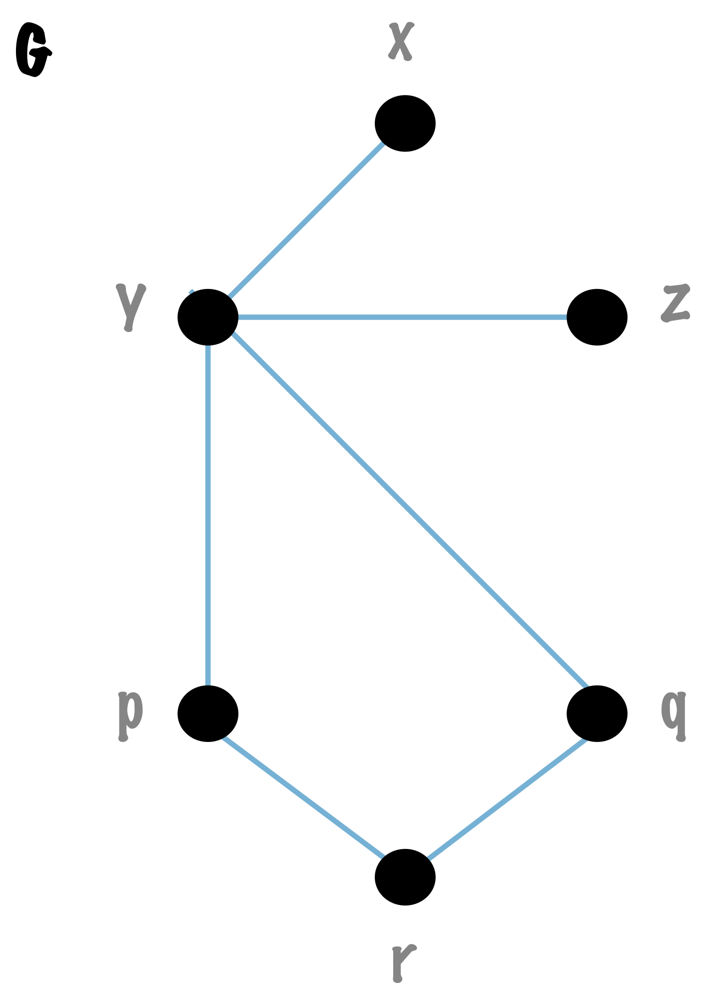


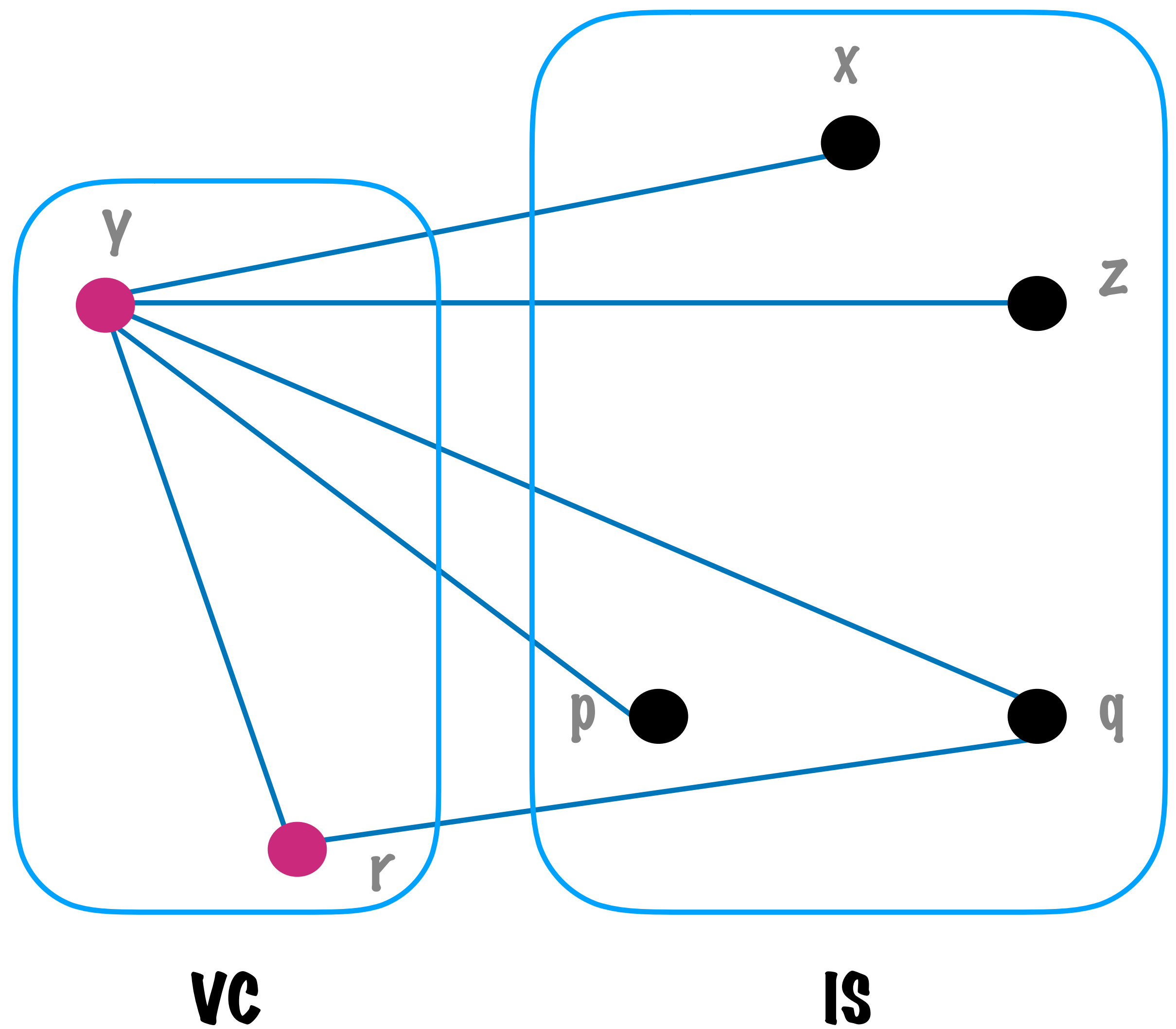


Review of Basic Concepts: Independent Set

Independent set - set of vertices that are pairwise non-adjacent

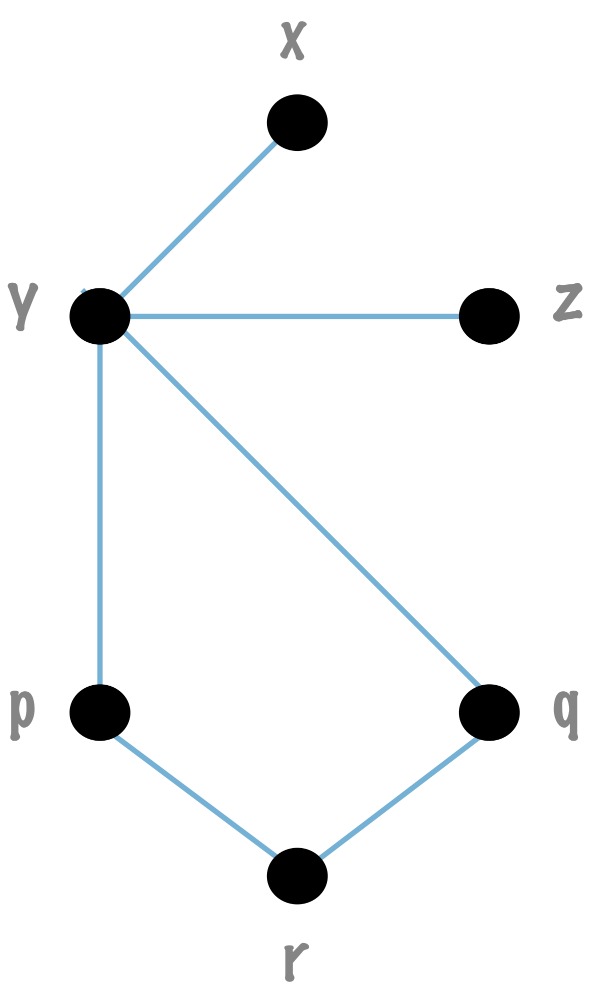
S is a vertex cover => V(G)-S is an independent set

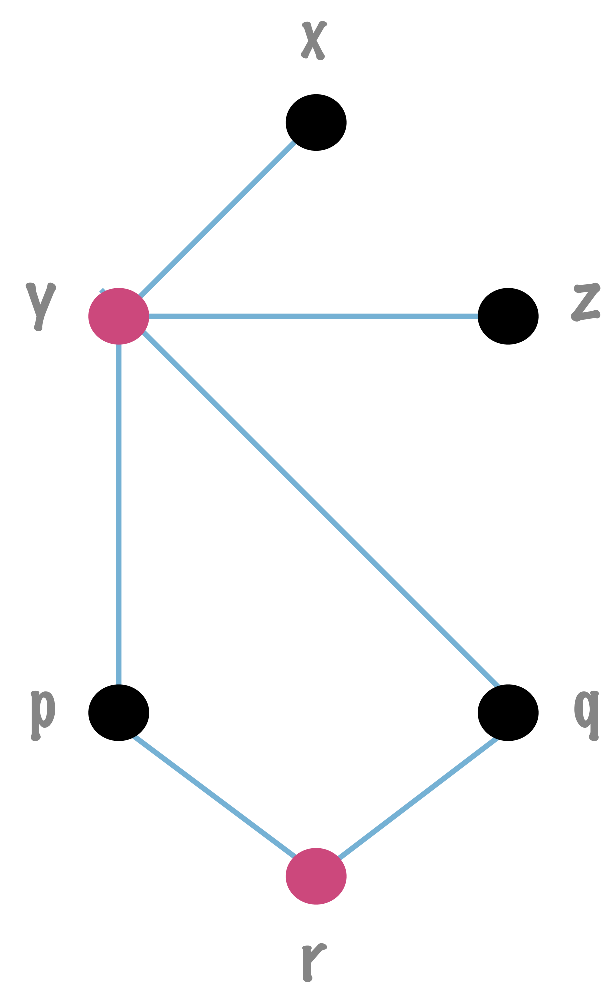




Review of Basic Concepts: Example of a Problem

Minimum Vertex Cover
Instance: An undirected graph G
Output: A minimum-size vertex cover of G

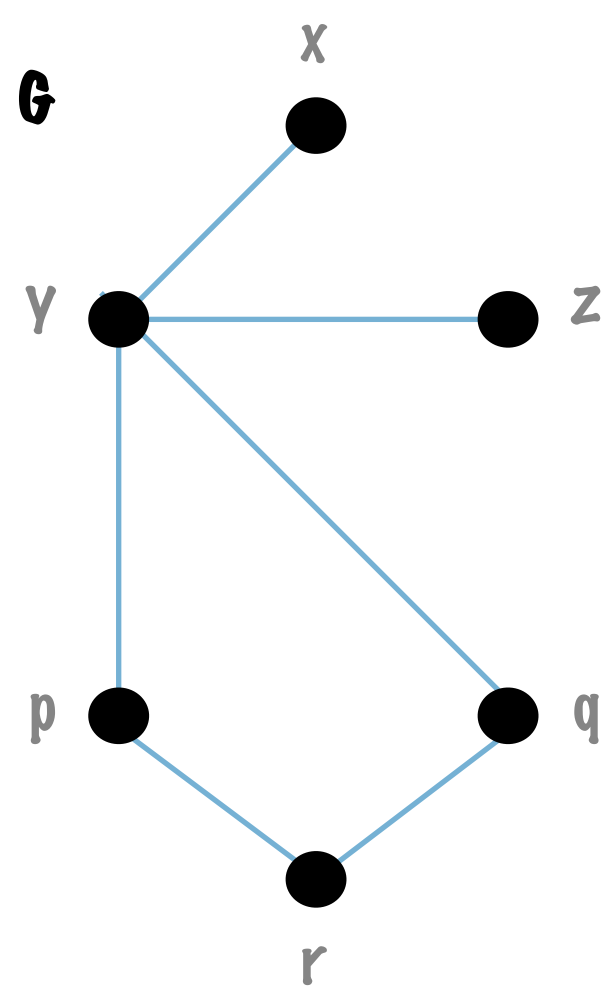




Review of Basic Concepts: Decision Problems

- Decision Problem: answer is yes or no

Vertex Cover
Instance: An undirected graph G and an integer k
Question: Does there exist a vertex cover of G of size at most k?



- (G,1) is a no-instance of Vertex Cover
- (G,2) is an yes-instance of Vertex Cover
- (G,3) is an yes-instance of Vertex Cover

Review of Basic Concepts: Classes P and EXP

* Complexity Class P
    * P is the set of all decision problems solvable in polynomial time
    * $O(n^c)$ time for some constant c where n is the input size
    * Examples: Shortest Path, Matching, Longest Path in directed acyclic graphs

* Complexity Class EXP
    * EXP is the set of all decision problems solvable in exponential time
    * $O(2^{n^c})$ time for some constant c where n is the input size
    * Examples: Vertex Cover, Travelling Salesperson, Longest Path

Review of Basic Concepts: Class NP

* Complexity Class NP
  * A problem is in NP if any yes-instance can be verified that it is indeed an yes-instance in polynomial time
    * polynomial-size certificate

* Vertex Cover
  * Certificate is a vertex cover of size at most k
    * Certificate is polynomial-size
    * Can verify if a set of vertices is a vertex cover or not in polynomial time

$$NP \subseteq EXP$$

Review of Basic Concepts: Reductions

* Polynomial-time Reductions
  * Problem A reduces to problem B if there is a polynomial time algorithm h such that for every instance x of A
    * h(x) is an instance of B
    * x is an yes-instance of A if and only if h(x) is an yes-instance of B
* Complexity Classes NP-hard and NP-complete
  * Problem A is NP-hard if every problem in NP reduces to it in polynomial time
    * If an NP-hard problem has a polynomial-time algorithm then P=NP
  * Problem A is NP-complete if it is in NP and NP-hard

Review of Basic Concepts: Approaches to NP-hardness

- Consequence of a problem being NP-hard
  - No polynomial time (in the worst case) algorithm that solves all instances optimally is likely to exist
    - Near-optimum solution: approximation algorithms
    - Average case: Randomized algorithms
    - Restrict the input
    - Exponential time

Parameterized algorithms (or) Fixed-parameter tractable algorithms

Parameterized Problem

* Each instance is associated with a non-negative integer called parameter

Parameterized Graph Problem Template
<u>Instance:</u> A graph G and integer k
<u>Question:</u> Does G have a solution of size k?
<u>Parameter:</u> k

$2^{O(k^2)}$ poly(n)

$2^{O(k \log k)}$ poly(n)

size of an instance (G,k) is |G| + k

Goal: Design f(k) poly(n) algorithm

n - input size

k - parameter

fixed-parameter tractable algorithm
or
parameterized algorithm

Complexity Class FPT
is the set of all parameterized problems that are fixed-parameter tractable

Parameterized Algorithms

- Find solution in exponential time
  - Exponential factor in running time is restricted to only the parameter

Vertex Cover (parameterized by solution size)
Instance: A graph G on n vertices m edges and integer k
Question: Does G have a vertex cover of size at most k?
Parameter: k

Goal: f(k) poly(n,m) algorithm

Multiple Parameterizations

Vertex Cover (parameterized by max degree)
Instance: A graph G with max degree r and integer k
Question: Does G have a vertex cover of size at most k?
Parameter: r

Goal: $f(r)$ poly(n,m) algorithm

Vertex Cover (parameterized by min degree)
Instance: A graph G with min degree q and integer k
Question: Does G have a vertex cover of size at most k?
Parameter: q

Goal: $f(q)$ poly(n,m) algorithm

Not all parameterized problems are FPT

Vertex Cover Parameterized by Solution Size

<u>Instance:</u> A graph G on n vertices m edges and integer k
<u>Question:</u> Does G have a vertex cover of size at most k?
<u>Parameter:</u> k

Instance: (G,k)

Preprocessing Rule 1: Delete isolated vertices ● u

Resulting Instance: (G-u,k)

(G,k) is an yes-instance => (G-u,k) is an yes-instance

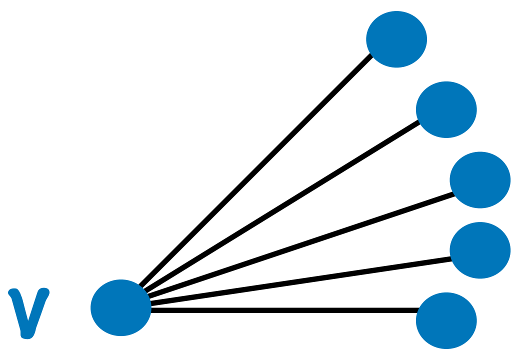
(G-u,k) is an yes-instance => (G,k) is an yes-instance

Time: O(n)

Vertex Cover Parameterized by Solution Size

Instance: (G′,k′)

Preprocessing Rule 2: Delete high degree vertices



>= k′+1 neighbours

Add v into the solution

Resulting Instance: (G′-v,k′-1)

(G′,k′) is an yes-instance => (G′-v,k′-1) is an yes-instance

(G′-v,k′-1) is an yes-instance => (G′,k′) is an yes-instance

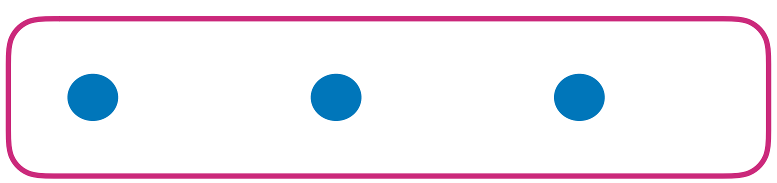
Time: $O(nk^2)$

Vertex Cover Parameterized by Solution Size

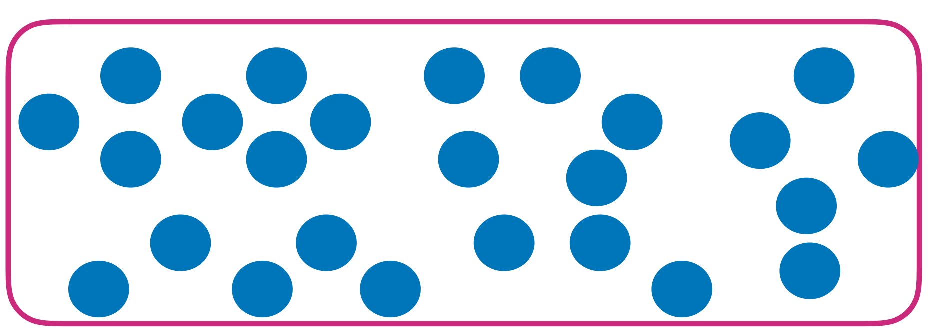
Instance: (H,r)

(G,k) is an yes-instance iff (H,r) is an yes-instance

Suppose (H,r) is an yes-instance



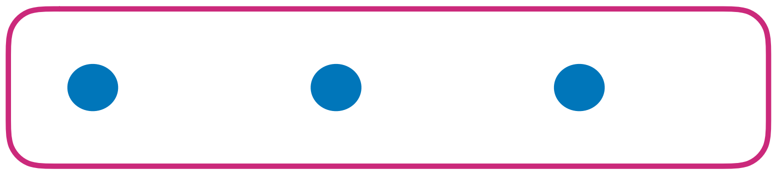
S a vertex cover of H with |S| <= r <=k



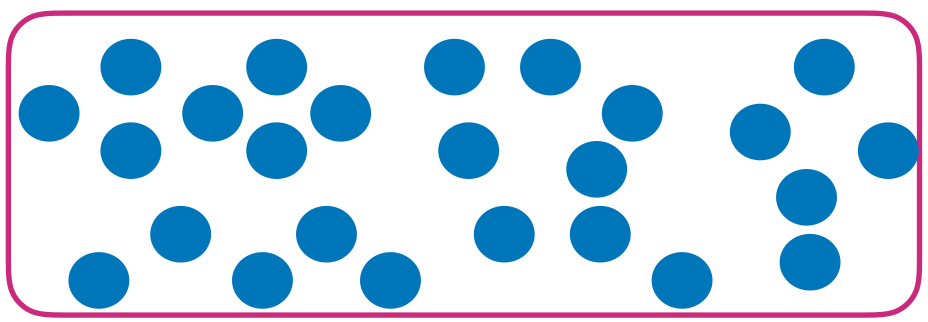
Independent Set

Vertex Cover Parameterized by Solution Size

Suppose (H,r) is an yes-instance



S a vertex cover of H with |S| <= r <=k



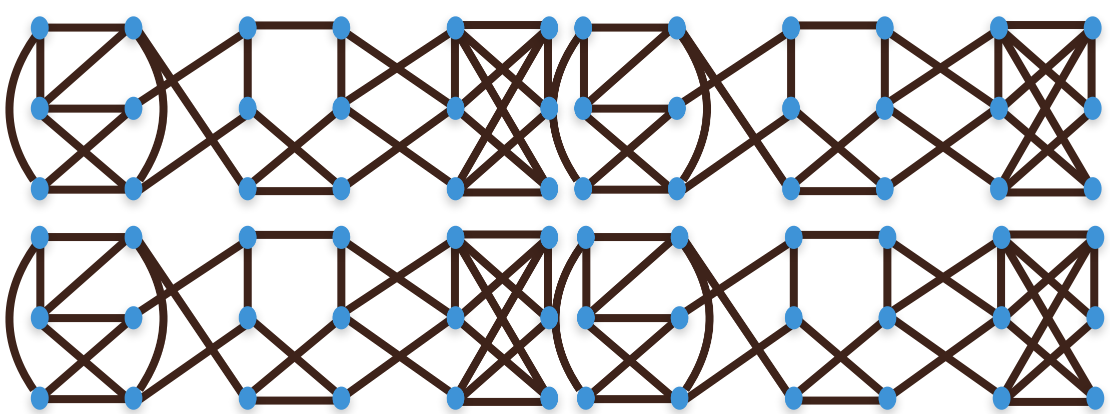
Independent Set

Can H have more than r^2 edges?

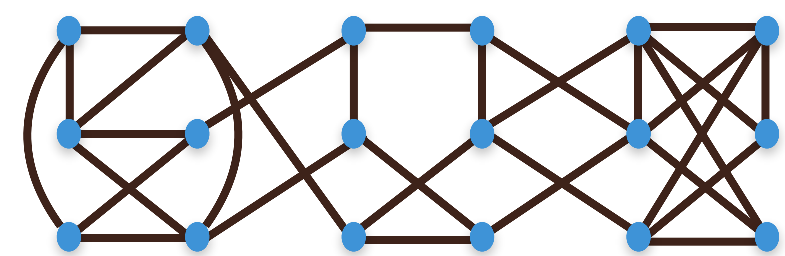
Vertex Cover Parameterized by Solution Size

If H has more than r^2 edges then (H,r) is no yes-instance

Otherwise, H has at most r^2 edges and at most $r+r^2$ vertices



$O(n^3)$ time



G on n vertices and m edges
Does G have a VC of size <= k?
Parameter: k

H on $r+r^2$ vertices and r^2 edges
Does H have a VC of size <= r?
Parameter: r

$|(H,r)| = f(k)$ Kernel

(G,k) is an yes-instance iff (H,r) is an yes-instance

Kernel => FPT