

Assignment Project Exam Help

Add WeChat powcoder

CSC373

Assignment Project Exam Help

<https://powcoder.com>

Weeks 7 & 8:  
Add WeChat powcoder  
Complexity

# Recap

Add WeChat powcoder

- Linear Programming

- Standard formulation
- Slack formulation
- Simplex
- Duality
- Formulating given problems as LPs

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help This & Next Week

- Applications of linear programming
  - Shortest path
  - Network flow
- A note about integer programming
- Complexity <https://powcoder.com>
  - Turing machines, computability, efficient computation
  - P, NP, and NP-completeness
  - Reductions
  - Idea behind NP-completeness of SAT and 3SAT
  - NP vs co-NP
  - Other complexity classes

# Assignment Project Exam Help

## Network Flow via LP

Add WeChat powcoder

- Problem

- **Input:** directed graph  $G = (V, E)$ , edge capacities  $c: E \rightarrow \mathbb{R}_{\geq 0}$

- **Output:** Value  $v(f^*)$  of a maximum flow  $f^*$

- Flow  $f$  is valid if:

- **Capacity constraints:**  $\forall (u, v) \in E: 0 \leq f(u, v) \leq c(u, v)$

- **Flow conservation:**  $\forall u: \sum_{(u,v) \in E} f(u, v) = \sum_{(v,u) \in E} f(v, u)$

- Maximize  $v(f) = \sum_{(s,v) \in E} f(s, v)$

Linear constraints

Linear objective!

# Assignment Project Exam Help

## Network Flow via LP

Add WeChat powcoder

$$\text{maximize } \sum_{v \in V} f_{sv}$$

Assignment Project Exam Help

$$0 \leq f_{uv} \leq c(u, v) \text{ for all } (u, v) \in E$$

<https://powcoder.com>

$$\sum_{(u,v) \in E} f_{uv} = \sum_{(v,w) \in E} f_{v,w} \text{ for all } v \in V \setminus \{s, t\}$$

Add WeChat powcoder

Exercise: Write the dual of this LP.  
What is the dual trying to find?

# Assignment Project Exam Help

## Shortest Path via LP

Add WeChat powcoder

- Problem

- **Input:** directed graph  $G = (V, E)$ , edge weights  $w: E \rightarrow \mathbb{R}_{\geq 0}$ , source vertex  $s$ , target vertex  $t$
- **Output:** weight of the shortest weight path from  $s$  to  $t$

- **Variables:** for each vertex  $v$ , we have variable  $d_v$

Why max?

maximize  $d_t$   
subject to

$$\begin{aligned} d_v &\leq d_u + w(u, v) \quad \text{for each edge } (u, v) \in E, \\ d_s &= 0. \end{aligned}$$

Exercise: prove formally that this works!

If objective was min., then we could set all variables  $d_v$  to 0.

## Assignment Project Exam Help

But...but...  
Add WeChat powcoder

- For these problems, we have different combinatorial algorithms that are much faster and run in strongly polynomial time

## Assignment Project Exam Help

- Why would we use LP?  
<https://powcoder.com>

Add WeChat powcoder

- For some problems, we don't have faster algorithms than solving them via LP

# Assignment Project Exam Help

# Multicommodity-Flow

Add WeChat powcoder

## • Problem:

- **Input:** directed graph  $G = (V, E)$ , edge capacities  $c: E \rightarrow \mathbb{R}_{\geq 0}$ ,  $k$  commodities  $(s_i, t_i, d_i)$ , where  $s_i$  is source of commodity  $i$ ,  $t_i$  is sink, and  $d_i$  is demand.
- **Output:** valid multicommodity flow  $(f_1, f_2, \dots, f_k)$ , where  $f_i$  has value  $d_i$  and all  $f_i$  jointly satisfy the constraints

Add WeChat powcoder

The only known polynomial time algorithm for this problem is based on solving LP!

$$\sum_{i=1}^k f_{iuv} \leq c(u, v) \quad \text{for each } u, v \in V,$$

$$\sum_{v \in V} f_{iuv} - \sum_{v \in V} f_{ivu} = 0 \quad \text{for each } i = 1, 2, \dots, k \text{ and for each } u \in V - \{s_i, t_i\},$$

$$\sum_{v \in V} f_{i, s_i, v} - \sum_{v \in V} f_{i, v, s_i} = d_i \quad \text{for each } i = 1, 2, \dots, k,$$

$$f_{iuv} \geq 0 \quad \text{for each } u, v \in V \text{ and for each } i = 1, 2, \dots, k.$$



# Integer Linear Programming

- Variable values are restricted to be integers

- Example:

➤ Input:  $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$

➤ Goal:

Maximize  $c^T x$

Subject to  $Ax \leq b$   
 $x \in \{0, 1\}^n$

- Does this make the problem easier or harder?
  - Harder. We'll prove that this is "NP-complete".

# Assignment Project Exam Help

## LPs are everywhere...

Add WeChat powcoder

- Microeconomics
- Manufacturing
- VLSI (very large scale integration) design
- Logistics/transportation
- Portfolio optimization
- Bioengineering (flux balance analysis)
- Operations research more broadly: maximize profits or minimize costs, use linear models for simplicity
- Design of approximation algorithms
- Proving theorems, as a proof technique
- ...

# Assignment Project Exam Help

# Introduction to Complexity

Add WeChat powcoder

- You have a problem at hand
- You try every technique known to humankind for finding a polynomial time algorithm, but fail.
- You try every technique known to humankind for proving that there cannot exist a polynomial time algorithm for your problem, but fail.
- **What do you do?**
  - Prove that it is NP-complete, of course!

# Assignment Project Exam Help

# Turing Machines

Add WeChat powcoder

- “Which problems can a computer (not) solve in a certain amount of time?”

- How do we mathematically define what a computer is?

Assignment Project Exam Help

- Alan Turing (“Father of Computer Science”), 1936

- Introduced a mathematical model

- “Turing machine”

https://powcoder.com

Add WeChat powcoder

- All present-day computers can be simulated by a Turing machine

- **Fun fact:** So can all the quantum computers

- But TM might take longer to solve the same problem

# Assignment Project Exam Help

# Turing Machines

Add WeChat powcoder

- We won't formally introduce...but at a high level...
- Turing machine
  - Tape
    - Input is given on tape
    - Intermediate computations can be written there
    - Output must to be written there
  - Head pointer
    - Initially pointing at start of input on tape
  - Maintains an internal “state”
  - A transition function describes how to change state, move head pointer, and read/write symbols on tape

# Assignment Project Exam Help

# Computability

Add WeChat powcoder

- Church-Turing Thesis

- “Everything that is computable can be computed by a Turing machine”
- Widely accepted, cannot be “proven”
- There are problems which a Turing machine cannot solve, regardless of the amount of time available
  - E.g., the halting problem

Add WeChat powcoder

- What about the problems we *can* solve? How do we define the time required?
  - Need to define an encoding of the input and output

# Encoding

Add WeChat powcoder

- What can we write on the tape?

- $S$  = a set of finite symbols
- $S^* = \bigcup_{n \geq 0} S^n$  = set of all finite strings using symbols from  $S$

Assignment Project Exam Help

- Input:  $w \in S^*$

- Length of input =  $|w|$  = length of  $w$  on tape

<https://powcoder.com>

- Output:  $f(w) \in S^*$

- Length of output =  $|f(w)|$

Add WeChat powcoder

- **Decision problems:** output = “YES” or “NO”

- E.g. “does there exist a flow of value at least 7 in this network?”

# Encoding

Add WeChat powcoder

- Example:

- “Given  $a_1, a_2, \dots, a_n$ , compute  $\sum_{i=1}^n a_i$ ”

- Suppose we are told that  $a_i \leq C$  for all  $i$

- What  $|S|$  should we use?

- $S = \{0,1\}$  ( $|S| = 2$ , binary representation)

- Length of input =  $O(\log_2 a_1 + \dots + \log_2 a_n) = O(n \log_2 C)$

- What about 3-ary ( $|S| = 3$ ) or 18-ary ( $|S| = 18$ )?

- Only changes the length by a constant factor, still  $O(n \log C)$

- What about unary (conceptually,  $|S| = 1$ )?

- Length blows up exponentially to  $O(nC)$

- Binary is already good enough, but unary isn't

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# Assignment Project Exam Help

# Efficient Computability

Add WeChat powcoder

- Polynomial-time computability

- A TM solves a problem in polynomial time if there is a polynomial  $p$  such that on every instance of  $n$ -bit input and  $m$ -bit output, the TM halts in at most  $p(n, m)$  steps
- Polynomial:  $n, n^2, 5n^{100} + 1000n^3, n \log^{100} n = o(n^{1.001})$
- Non-polynomial:  $2^n, 2^{\sqrt{n}}, 2^{\log^2 n}$

- Extended Church-Turing Thesis

- “Everything that is computable in polynomial time”
- Much less widely accepted, especially today
- But in this course, **efficient = polynomial-time**

If you ask the Turing machine to write a  $2^n$ -bit output, it's only reasonable to let it take  $2^n$  time...but usually, we'll look at problems where output is  $O(\text{length of input})$ , so we can ignore this  $m$

# Assignment Project Exam Help

P

Add WeChat powcoder

- P (polynomial time)

- The class of all decision problems computable by a TM in polynomial time

Assignment Project Exam Help

- Examples

- Addition, multiplication, square root
- Shortest paths
- Network flow
- Fast Fourier transform
- Checking if a given number is a prime [Agrawal-Kayal-Saxena 2002]
- ...

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

## NP

Add WeChat powcoder

- NP (nondeterministic polynomial time) intuition
  - Subset sum problem:  
Given an array  $\{-7, -3, -2, 5, 8\}$ , is there a zero-sum subset?  
<https://powcoder.com>
  - Enumerating all subsets is exponential
  - But...given  $\{-3, -2, 5\}$ , we can verify in polynomial time that it is indeed a valid subset and has zero sum
  - A nondeterministic Turing machine could “guess” the solution and then test if it has zero sum in polynomial time

# Assignment Project Exam Help

## NP

Add WeChat powcoder

- NP (nondeterministic polynomial time)

- The class of all decision problems for which a YES answer can be verified by a TM in polynomial time given polynomial length “advice” or “witness”.

Assignment Project Exam Help

- There is a polynomial-time verifier TM  $V$  and another polynomial  $p$  such that

<https://powcoder.com>

- For all YES inputs  $x$ , there exists advice  $y$  with  $|y| = p(|x|)$  on which  $V(x, y)$  returns YES
- For all NO inputs  $x$ ,  $V(x, y)$  returns NO for every possible  $y$

Add WeChat powcoder

- Informally: “Whenever the answer is YES, there’s a short proof of it.”
  - When the answer is NO, there may not be any short proof for it.

# Assignment Project Exam Help

## co-NP

Add WeChat powcoder

- co-NP

- Same as NP, except whenever the answer is NO, there is a short proof of it

# Assignment Project Exam Help

- Open questions

- NP = co-NP?

<https://powcoder.com>

- $P = NP \cap \text{co-NP}$ ?

- And...drum roll please...

Add WeChat powcoder

$$P = NP?$$

# P versus NP

Add WeChat powcoder

- Lance Fortnow in his article on P and NP in Communications of the ACM, Sept 2009

Assignment Project Exam Help

*“The P versus NP problem has gone from an interesting problem related to logic to perhaps the most fundamental and important mathematical question of our time, whose importance only grows as computers become more powerful and widespread.”*

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

# Millenium Problems

Add WeChat powcoder

- Award of \$1M for each problem by the Clay Math institute

1. Birch and Swinnerton-Dyer Conjecture

2. Hodge Conjecture

3. Navier-Stokes Equations

4.  $P = NP?$

<https://powcoder.com>

Claim: Worth >> \$1M

5. Poincare Conjecture (Solved)<sup>1</sup>

Add WeChat powcoder

6. Riemann Hypothesis

7. Yang-Mills Theory

<sup>1</sup>Solved by Grigori Perelman (2003): Prize unclaimed

# Cook's Conjecture

Assignment Project Exam Help  
Add WeChat powcoder

- Cook's conjecture

- (And every sane person's belief...)
- $P$  is likely not equal to  $NP$

Assignment Project Exam Help

- Why do we believe this?

- There is a large class of problems (NP-complete)
- By now, contains thousands and thousands of problems
- Each problem is the "hardest problem in NP"
- If you can efficiently solve *any one of them*, you can efficiently solve *every problem in NP*
  - Despite decades of effort, no polynomial time solution has been found for *any of them*

<https://powcoder.com>

Add WeChat powcoder



# Reductions

Add WeChat powcoder

- Problem  $A$  is **p-reducible** to problem  $B$  (denoted  $A \leq_p B$ ) if an “oracle” (subroutine) for  $B$  can be used to efficiently solve  $A$

➤ You can solve  $A$  by making polynomially many calls to the oracle and doing additional polynomial-time computation

Assignment Project Exam Help

<https://powcoder.com>

- **Question:** If  $A$  is p-reducible to  $B$ , then which of the following is true?  
a) If  $A$  cannot be solved efficiently, then neither can  $B$ .  
b) If  $B$  cannot be solved efficiently, then neither can  $A$ .  
c) Both.  
d) None.

Add WeChat powcoder

# Reductions

Add WeChat powcoder

- Problem  $A$  is **p-reducible** to problem  $B$  (denoted  $A \leq_p B$ ) if an “oracle” (subroutine) for  $B$  can be used to efficiently solve  $A$

➤ You can solve  $A$  by making polynomially many calls to the oracle and doing additional polynomial computation

Assignment Project Exam Help

<https://powcoder.com>

- **Question:** If I want to prove that my problem  $X$  is “hard”, I should:

Add WeChat powcoder

- a) Reduce my problem  $X$  to a known hard problem.
- b) Reduce a known hard problem to my problem  $X$ .
- c) Both.
- d) None.

# NP-completeness

Add WeChat powcoder

- NP-completeness

- A problem  $B$  is NP-complete if it is in NP and **every** problem  $A$  in NP is p-reducible to  $B$
- Hardest problems in NP
- If one of them can be solved efficiently, every problem in NP can be solved efficiently, implying  $P=NP$

Assignment Project Exam Help

<https://powcoder.com>

- Observation: Add WeChat powcoder

- If  $A$  is in NP, and some NP-complete problem  $B$  is p-reducible to  $A$ , then  $A$  is NP-complete too
  - “If I could solve  $A$ , then I could solve  $B$ , and then I could solve any problem in NP”

# NP-completeness

Add WeChat powcoder

- But this uses an already known NP-complete problem to prove another problem is NP-complete

Assignment Project Exam Help

- How do we find the *first* NP-complete problem?

- How do we know there are any NP-complete problems at all?
- Key result by Cook
- First NP-complete problem: SAT
  - By a reduction from an arbitrary problem in NP to SAT
  - “From first principles”

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

# CNF Formulas

Add WeChat powcoder

- Conjunctive normal form (CNF)

- Boolean variables  $x_1, x_2, \dots, x_n$
- Their negations  $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$
- Literal  $\ell$ : a variable or its negation
- Clause  $C = \ell_1 \vee \ell_2 \vee \dots \vee \ell_r$  is a disjunction of literals
- CNF formula  $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_m$  is a conjunction of clauses
  - $k$ CNF: Each clause has at most  $k$  literals
    - We'll abuse notation a little and assume there are *exactly*  $k$
    - Example of 3CNF

$$\varphi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee x_1)$$

# Assignment Project Exam Help

## SAT and 3SAT

Add WeChat powcoder

- Example of 3CNF

$$\varphi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee x_1)$$

- “SAT” (Satisfiability) Problem:

- A CNF formula  $\varphi$  is satisfiable if there is an assignment of truth values (TRUE/FALSE) to variables under which the formula evaluates to TRUE
  - That means, in each clause, at least one literal is TRUE
- SAT: “Given a CNF formula  $\varphi$ , is it satisfiable?”
- 3SAT: “Given a 3CNF formula  $\varphi$ , is it satisfiable?”

# Assignment Project Exam Help

## SAT and 3SAT

Add WeChat powcoder

- Cook-Levin Theorem

- SAT (and even 3SAT) is NP-complete

- Doesn't use any known NP-complete problem

- Directly reduces any given NP problem to SAT
- Reduction is a bit complex, so we'll defer it until later
- But for now, let's assume SAT and 3SAT are NP-complete and reduce them to other problems (and then those problems to other problems...)

# Assignment Project Exam Help

# NP-Complete Examples

Add WeChat powcoder

- NP-complete problems

- SAT = first NP complete problem
- Decision TSP: Is there a route visiting all  $n$  cities with total distance at most  $k$ ? Assignment Project Exam Help
- 3-Colorability: Can the vertices of a graph be colored with at most 3 colors such that no two adjacent vertices have the same color? <https://powcoder.com>
- Karp's 21 NP-complete problems

Add WeChat powcoder

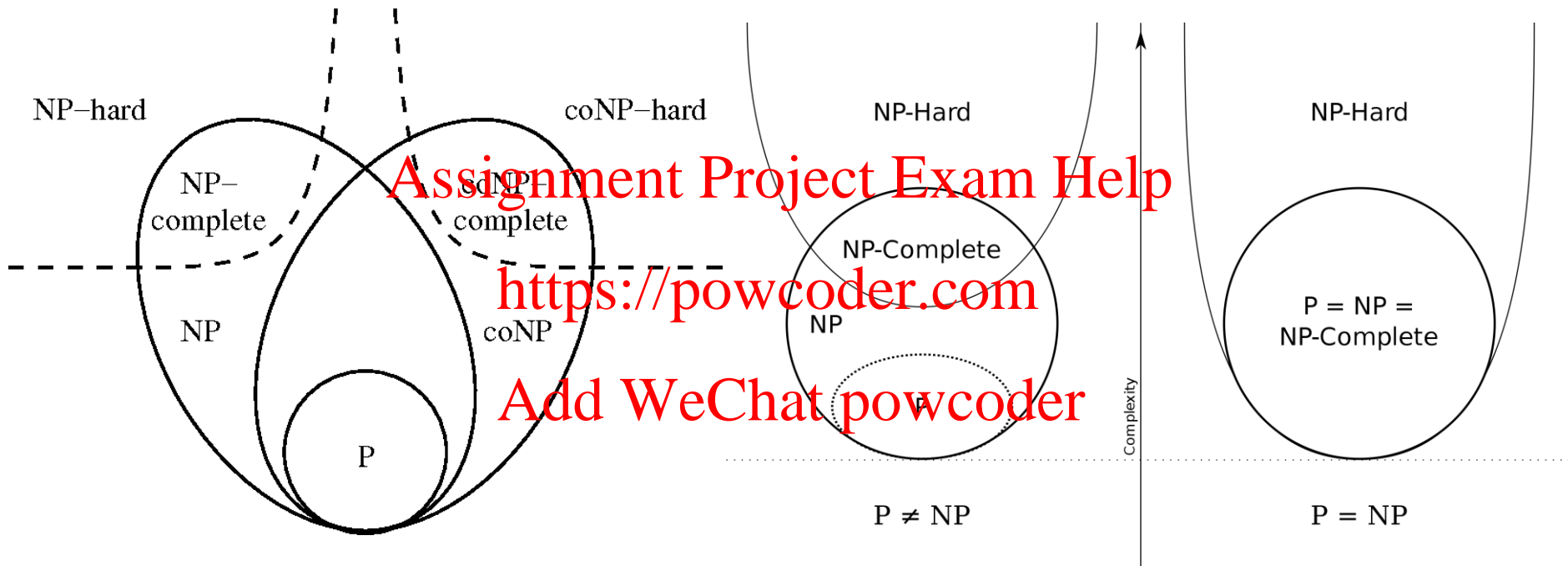
- co-NP-complete

- Tautology problem (“negation” of SAT):
  - “Given a CNF formula  $\varphi$ , does it always evaluate to TRUE regardless of variable assignments?”



# Complexity

Assignment Project Exam Help  
Add WeChat powcoder

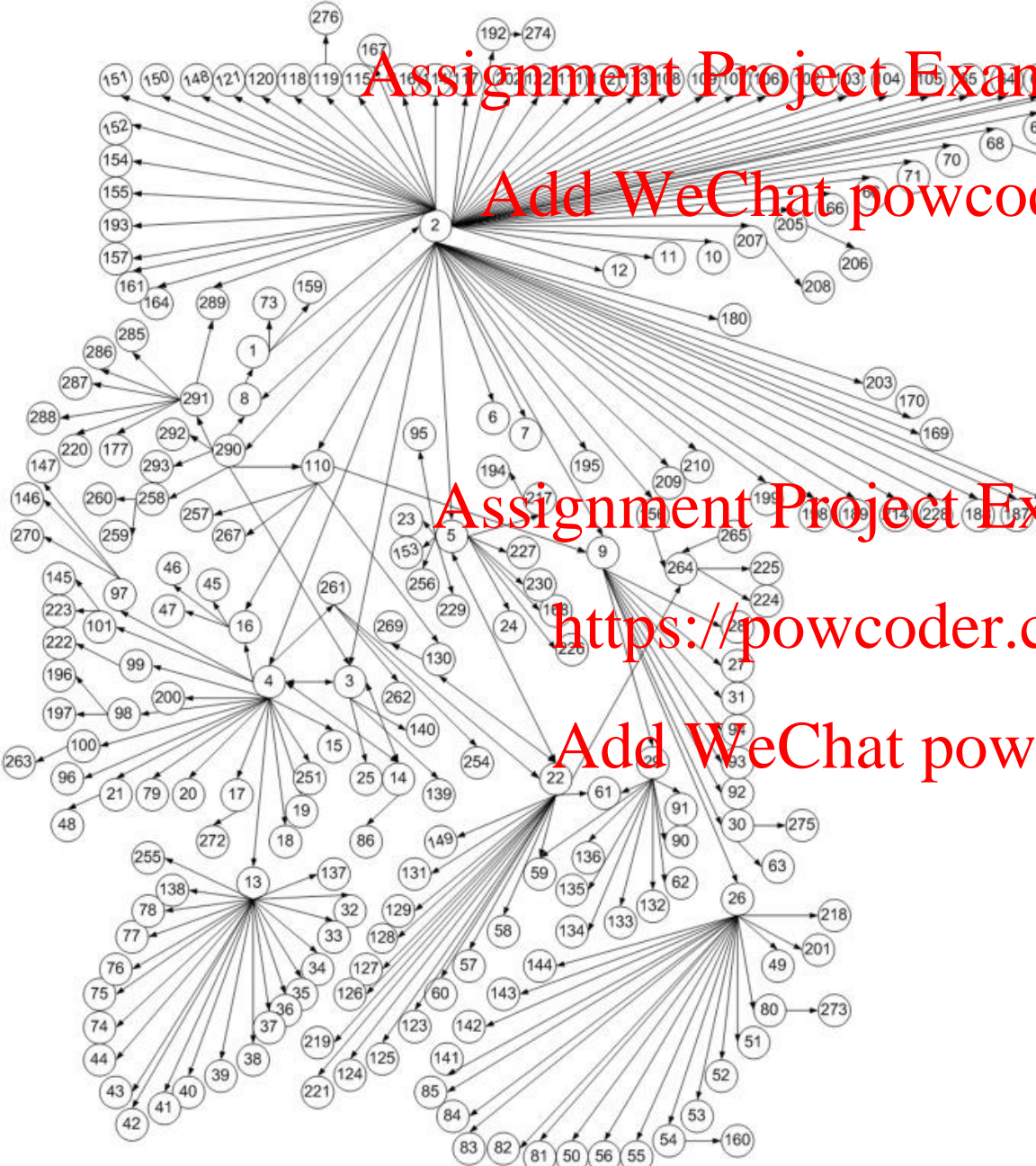


By Behnam Esfahbod, CC BY-SA 3.0,  
<https://commons.wikimedia.org/w/index.php?curid=3532181>



### Survey of polynomial transformations between NP-complete problems

Jorge A. Ruiz-Vanoye <sup>a</sup>, Joaquín Pérez-Ortega <sup>b</sup>, Rodolfo A. Pazos R. <sup>c</sup>, Ocotlán Díaz-Parra <sup>d</sup>, Juan Frausto-Solís <sup>e</sup>, Hector J. Fraire Huacuja <sup>c</sup>, Laura Cruz-Reyes <sup>c</sup>, José A. Martínez F. <sup>c</sup>



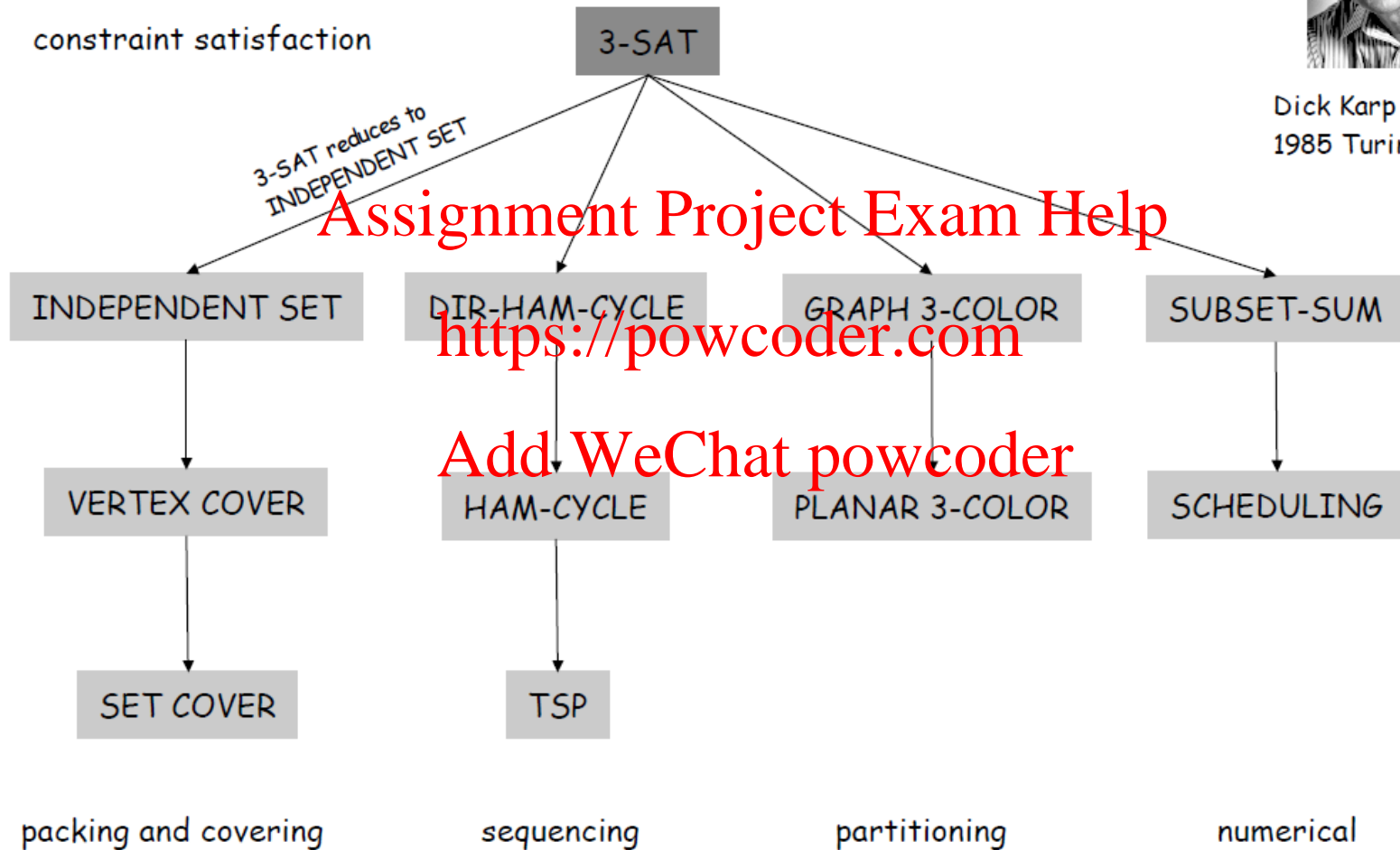
Number	Name of problem	Number	Name of problem
1	Satisfiability (SAT)	2	3-Satisfiability (3SAT)
3	Clique (clique cover)	4	Vertex cover
5	Subset sum	6	Hitting string
7	Chinese postman for mixed graphs	8	Graph colorability
9	Three-Dimensional matching (3DM)	10	Rectilinear picture compression
11	Tableau equivalence	12	Consistency of databases frequency tables
13	Hamiltonian Circuit (Directed Hamiltonian path, Undirected Hamiltonian path)	14	Independent set
15	Setbasis	16	Hitting set
17	Generalized containment	18	Multiple copy file allocation
19	Shortest common supersequence	20	Longest common subsequence
21	Minimum cardinality key	22	Partition
23	K'th largest subset	24	Capacity assignment
25	Conjunctive Boolean query	26	Exact cover by 3-sets (X3C)
27	Minimum test set	28	3-Matroid intersection
29	3-Partition	30	Numerical three-dimensional matching

# Assignment Project Exam Help

Add WeChat powcoder



Dick Karp (1972)  
1985 Turing Award



# Assignment Project Exam Help

## Just A Tad Bit of History

Add WeChat powcoder

- [Cook 1971]
  - Proved 3SAT is NP-complete in seminal paper
- [Karp 1972] Assignment Project Exam Help
  - Showed that 20 other problems are also NP-complete
  - “Karp's 21 NP-complete problems”
  - Renewed interest in this idea

<https://powcoder.com>

Add WeChat powcoder
- 1982: Cook won the Turing award

# Assignment Project Exam Help

## Independent Set

Add WeChat powcoder

- Problem

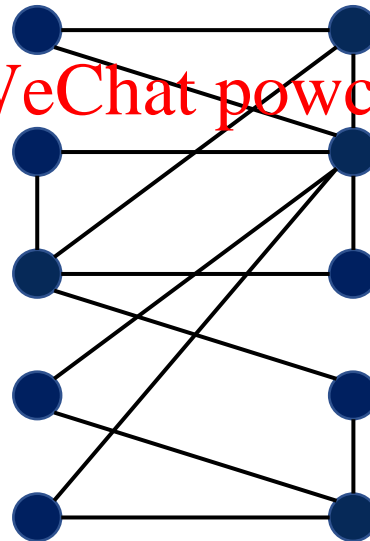
- **Input:** Undirected graph  $G = (V, E)$ , integer  $k$
- **Question:** Does there exist a subset of vertices  $S \subseteq V$  with  $|S| = k$  such that for each edge, at most one of its endpoints is in  $S$ ?

Assignment Project Exam Help

<https://powcoder.com>

**Example:**

- Does this graph have an independent set of size 6?
  - Yes!
- Does this graph have an independent set of size 7?
  - No!



● = independent set

Add WeChat powcoder

# Assignment Project Exam Help

# Independent Set

Add WeChat powcoder

- Claim: Independent Set is in NP

- Recall: We need to show that there is a polynomial-time algorithm which
  - Can accept every YES instance with the right polynomial-size advice
  - Will not accept a NO instance with any advice
- **Advice:** the actual independent set  $S$
- **Algorithm:** check if  $S$  is an independent set and if  $|S| = k$
- Simple!

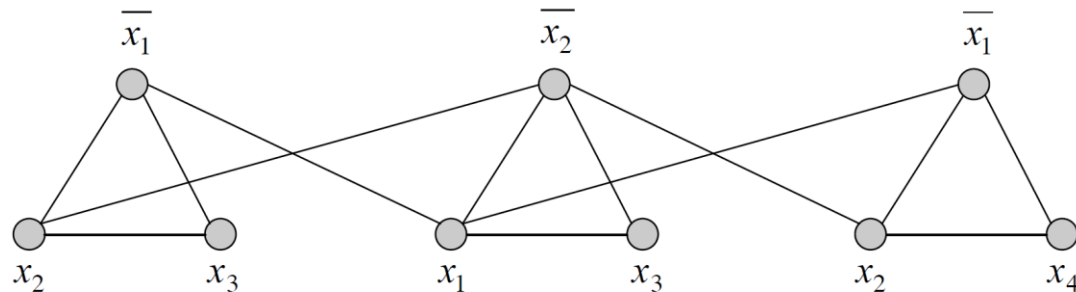
# Assignment Project Exam Help

## Independent Set

Add WeChat powcoder

- Claim:  $3SAT \leq_p$  Independent Set

- Given a formula  $\varphi$  of 3SAT with  $k$  clauses, construct an instance  $(G, k)$  of Independent Set as follows
  - Create 3 vertices for each clause (one for each literal)
  - Connect them in a triangle
  - Connect the vertex of each literal to each of its negations



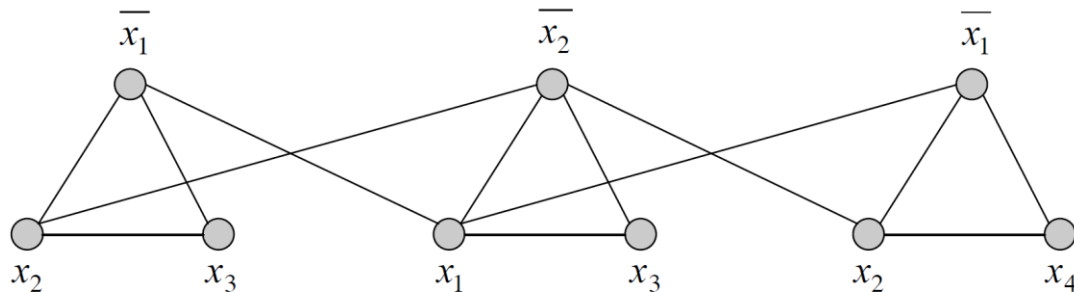
$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

# Assignment Project Exam Help

## Independent Set

➤ Why does this work?

- $3SAT = YES \Rightarrow Independent\ Set = YES$ 
  - From each clause, take any literal that is TRUE in the assignment
- $Independent\ Set = YES \Rightarrow 3SAT = YES$ 
  - Independent set  $S$  must contain one vertex from each triangle
  - No literal and its negation are both in  $S$
  - Set literals in  $S$  to TRUE, their negations to FALSE, and the rest to arbitrary values



$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$



# Assignment Project Exam Help

# Different Types of Reductions

Add WeChat powcoder

- $A \leq B$ 
  - Karp reductions
    - Take an arbitrary instance of  $A$ , and in polynomial time, **construct** a single instance of  $B$  with the same answer
    - Very restricted type of reduction
    - The reduction we just constructed was a Karp reduction
  - Turing/Cook reductions
    - Take an arbitrary instance of  $A$ , and solve it by **making polynomially many calls to an oracle for solving  $B$  and some polynomial-time extra computation**
    - Very general reduction
    - In this course, we'll allow Turing/Cook reductions, but whenever possible, see if you can construct a Karp reduction

# Assignment Project Exam Help

## Subset Sum

Add WeChat powcoder

- Problem

- **Input:** Set of integers  $S = \{w_1, \dots, w_n\}$ , integer  $W$
- **Question:** Is there  $S' \subseteq S$  that adds up to exactly  $W$ ?

Assignment Project Exam Help

- Example

<https://powcoder.com>

- $S = \{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}$ ,  $W = 3754$ ?
- Yes!
  - $1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754$

Add WeChat powcoder

# Assignment Project Exam Help

## Subset Sum

Add WeChat powcoder

- Claim: Subset Sum is in NP

- Recall: We need to show that there is a polynomial-time algorithm which
  - Can accept every YES instance with the right polynomial-size advice
  - Will not accept a NO instance with any advice
- **Advice:** the actual subset  $S$
- **Algorithm:** check that  $S'$  is indeed a subset of  $S$  and sums to  $W$
- Simple!

# Assignment Project Exam Help

## Subset Sum

Add WeChat powcoder

- Claim:  $3SAT \leq_p \text{Subset Sum}$

- Given a formula  $\varphi$  of 3SAT, we want to construct  $(S, W)$  of Subset Sum with the same answer
- In the table in the following slide:
  - Columns are for variables and clauses
  - Each row is a number in  $S$ , represented in decimal
  - Number for literal  $\ell$  : has 1 in its variable column and in the column of every clause where that literal appears
    - Number selected = literal set to TRUE
  - “Dummy” rows: can help make the sum in a clause column 4 if and only if at least one literal is set to TRUE

# Assignment Project Exam Help

## Subset Sum

Add WeChat powcoder

Decimal representation

- Claim:  $3SAT \leq_p$  Subset Sum

Assignment Project Exam Help

$$C_1 = x \vee y \vee z$$

$$C_2 = x \vee \bar{y} \vee z$$

$$C_3 = \bar{x} \vee \bar{y} \vee \bar{z}$$

<https://www.powcoder.com>

Add WeChat powcoder

dummies to get  
clause columns  
to sum to 4

	x	y	z	$C_1$	$C_2$	$C_3$
x	1	0	0	0	1	0
$\neg x$	1	0	0	1	0	1
y	0	1	0	1	0	0
$\neg y$	0	1	0	0	1	1
z	0	0	1	1	1	0
$\neg z$	0	0	1	0	0	1
	0	0	0	1	0	0
	0	0	0	2	0	0
	0	0	0	0	1	0
	0	0	0	0	2	0
	0	0	0	0	0	1
	0	0	0	0	0	2
W	1	1	1	4	4	4

# Assignment Project Exam Help

## Subset Sum

Add WeChat powcoder

- Note

- The Subset Sum instance we constructed has “large” numbers
  - Their values are exponentially large ( $\sim 10^{\#variables + \#clauses}$ )
  - But the number of bits required to write them is polynomial
- Can we hope to construct Subset Sum instance with numbers whose values are only  $\text{poly}(\#variables, \#clauses)$  large?
  - Unlikely, as that would prove  $P = NP$ !
  - Like Knapsack, Subset Sum can be solved in pseudo-polynomial time
    - That is, in polynomial time if the numbers are only polynomially large in value

# Assignment Project Exam Help

## 3-Coloring

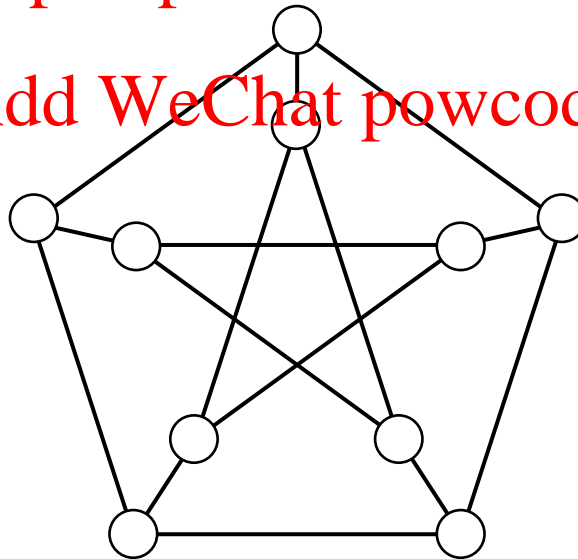
Add WeChat powcoder

- Problem

- **Input:** Undirected graph  $G = (V, E)$
- **Question:** Can we color each vertex of  $G$  using at most three colors such that no two adjacent vertices have the same color?

<https://powcoder.com>

Add WeChat powcoder



# Assignment Project Exam Help

## 3-Coloring

Add WeChat powcoder

- Claim: 3-coloring is in NP

- Recall: We need to show that there is a polynomial-time algorithm which
  - Can accept every YES instance with the right polynomial-size advice
  - Will not accept a NO instance with any advice
- **Advice:** colors of the nodes in a valid 3-coloring
- **Algorithm:** check that this is a valid 3-coloring
- Simple!



# Assignment Project Exam Help

## 3-Coloring

Add WeChat powcoder

• Claim:  $3SAT \leq_p 3\text{-Coloring}$

- Given a 3SAT formula  $\varphi$ , we want to construct a graph  $G$  such that  $G$  is 3-colorable if and only if  $\varphi$  has a satisfying assignment
- $G$  will have the following nodes:
  - Type 1: true, false, base, one for each  $x_i$ , one for each  $\bar{x}_i$
  - Type 2: additional nodes for each clause  $C_j$
- 1-1 correspondence between valid 3-colorings of type 1 nodes and valid truth assignments:
  - All literals with the same color as “true” node are set to true
  - All literals with the same color as “false” node are set to false
- **Claim:** Fix any colors of type 1 nodes. There exists a valid 3-coloring of  $G$  giving these colors to type 1 nodes if and only if the corresponding truth assignment is satisfying for  $\varphi$ .

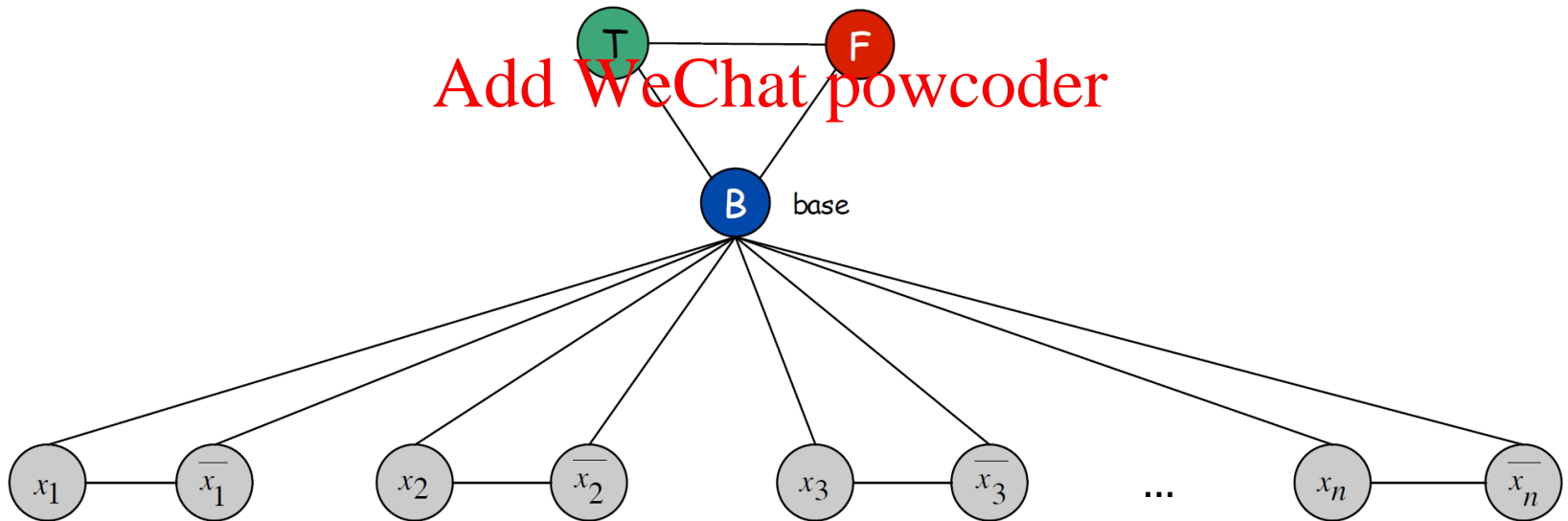
# Assignment Project Exam Help

## 3-Coloring

Add WeChat powcoder

- Create 3 new nodes T, F, and B, and connect them in a triangle
- Create a node for each literal, connect it to its negation and to B
- T-F-B must have different colors, and so must  $B-x_i-\bar{x}_i$ 
  - Each literal has the color of T or F, its negation has the other color
  - Valid 3-coloring  $\Leftrightarrow$  valid truth assignment (set all with color T to true)

<https://powcoder.com>



# Assignment Project Exam Help

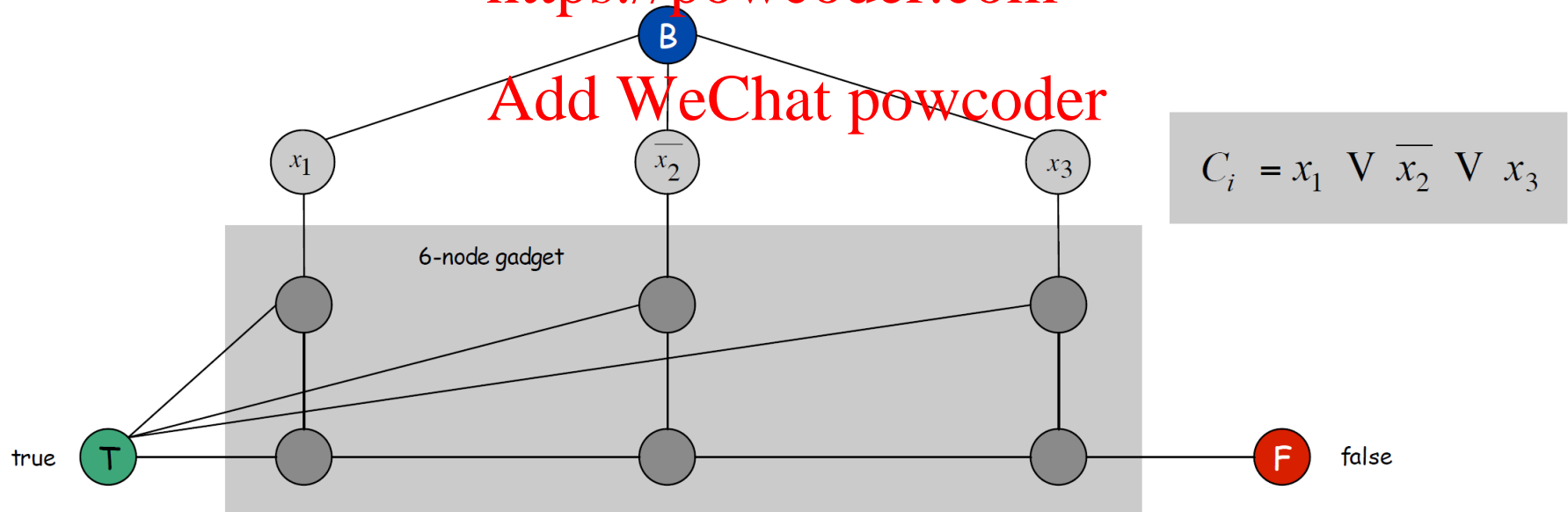
## 3-Coloring

Add WeChat powcoder

- We also need valid 3-coloring  $\Leftrightarrow$  *satisfying* truth assignment
  - For each clause, add the following gadget with 6 nodes and 13 edges
  - **Claim:** Clause gadget is 3-colorable  $\Leftrightarrow$  at least one of the nodes corresponding to the literals in the clause is assigned color of T

<https://powcoder.com>

Add WeChat powcoder



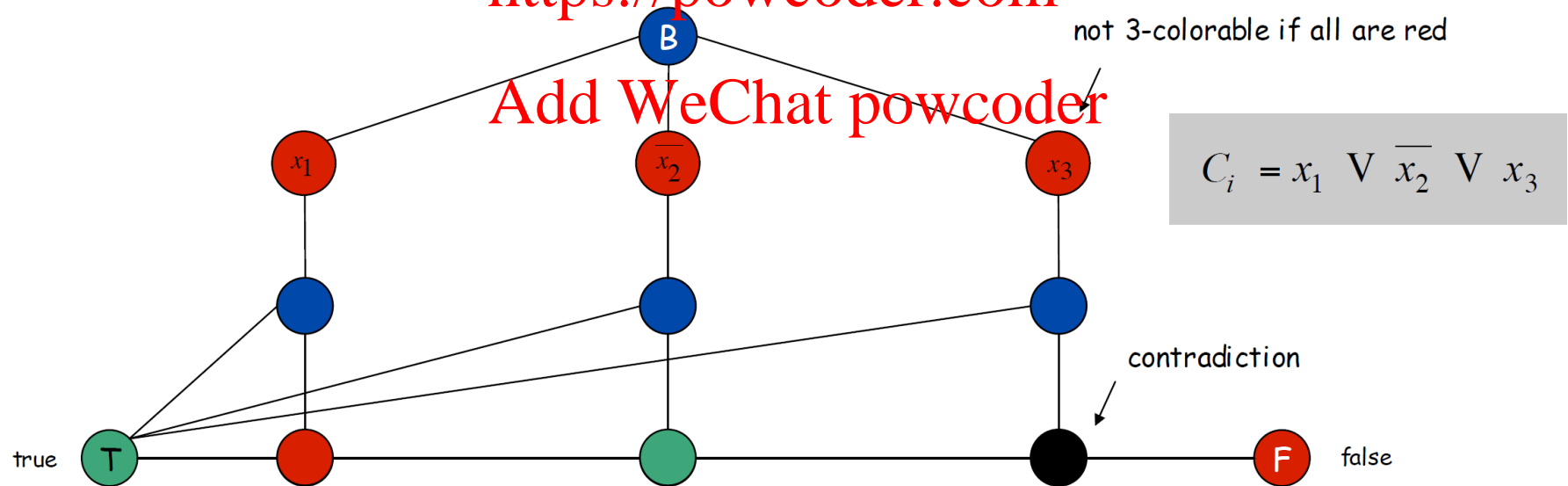
# Assignment Project Exam Help

## 3-Coloring

Add WeChat powcoder

- **Claim:** Valid 3-coloring  $\Rightarrow$  truth assignment satisfies  $\varphi$
- Suppose a clause  $C_i$  is not satisfied, so all its three literals must be F
  - Then the 3 nodes in top layer must be B
  - Then the first two nodes in bottom layer must be F and T
  - No color left for the remaining node  $\Rightarrow$  contradiction!

<https://powcoder.com>



# Assignment Project Exam Help

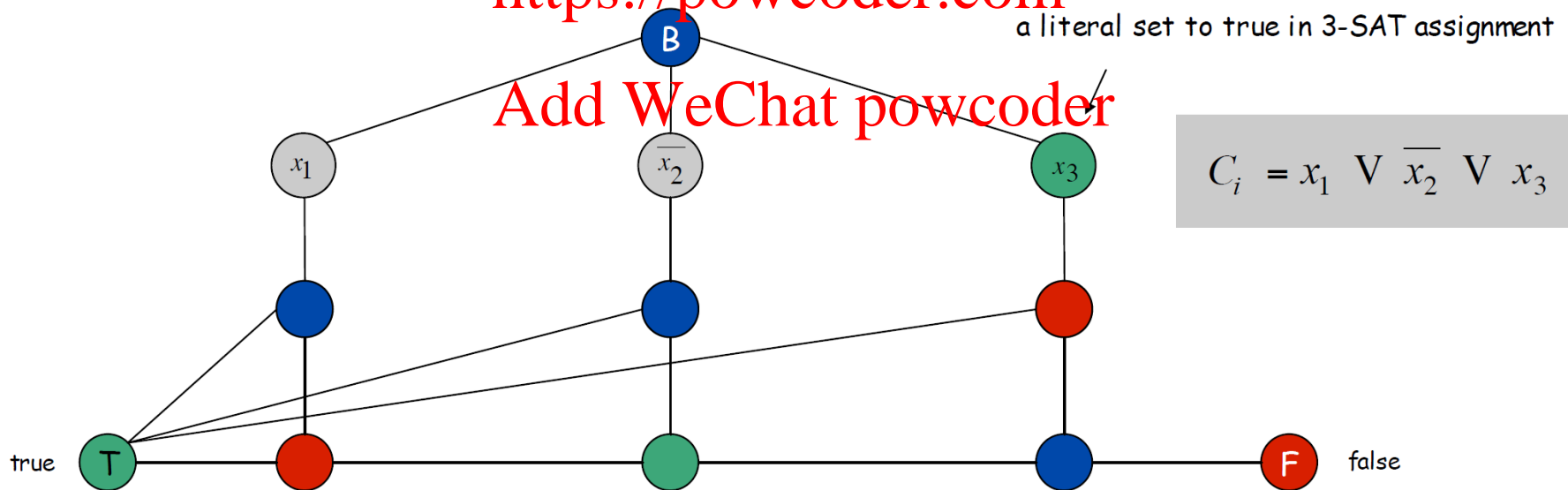
## 3-Coloring

Add WeChat powcoder

- We just proved: valid 3-coloring  $\Rightarrow$  satisfying assignment
- **Claim:** satisfying assignment  $\Rightarrow$  valid 3-coloring
  - Each clause has at least one literal with color T
  - **Exercise:** Regardless of which literal has color T and which color (T/F) the other literals have, the clause widget can always be 3-colored

<https://powcoder.com>

Add WeChat powcoder ✓



# Assignment Project Exam Help

# Review of Reductions

Add WeChat powcoder

- If you want to show that problem B is NP-complete
- **Step 1: Show that B is in NP**
  - Some polynomial-size advice should be sufficient to verify a YES instance in polynomial time
  - No advice should work for a NO instance
  - Usually, the solution of the “search version” of the problem works
    - But sometimes, the advice can be non-trivial
    - For example, to **check LP optimality**, one possible advice is the **values of both primal and dual variables**, as we saw in the last lecture

# Assignment Project Exam Help

## Review of Reductions

Add WeChat powcoder

- If you want to show that problem B is NP-complete
- Step 2: Find a known NP-complete problem A and reduce it to B (i.e. show  $A \leq_p B$ )
  - This means taking an arbitrary instance of A, and solving it in polynomial time using an oracle for B
    - Caution 1: Remember the direction. You are “reducing known NP-complete problem to your current problem”.
    - Caution 2: The size of the B-instances you construct should be polynomial in the size of the original A-instance
  - This would show that if B can be solved in polynomial time, then A can be as well
  - Some reductions are trivial, some are notoriously tricky...

# Binary Integer Linear Programming (BILP)

- Problem

- Input:  $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}, k \in \mathbb{R}$
- Question: Does there exist  $x \in \{0,1\}^n$  such that  $c^T x \geq k$  and  $Ax \leq b$ ?

- Decision variant of “maximize  $c^T x$  subject to  $Ax \leq b$ ” but instead of any  $x \in \mathbb{R}^n$  with  $x \geq 0$ , we are restricting  $x$  to binary.
- Does restricting search space make the problem easier or harder?
  - This is actually NP-complete!



# Assignment Project Exam Help

## BILP Feasibility

Add WeChat powcoder

- An even simpler problem
  - Special case where  $c = k = 0$ , so  $c^T x \geq k$  is always true

Assignment Project Exam Help

- Problem
  - Input:  $b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$
  - Question: Does there exist  $x \in \{0,1\}^n$  such that  $Ax \leq b$ ?

Add WeChat powcoder

- Just need to find a feasible solution
- This is still NP-complete!

# Assignment Project Exam Help

# BILP Feasibility

Add WeChat powcoder

- Claim: BILP Feasibility is in NP

- Recall: We need to show that there is a polynomial-time algorithm which
  - Can accept every YES instance with the right polynomial-size advice
  - Will not accept a NO instance with any advice
- **Advice:** simply a vector  $x$  satisfying  $Ax \leq b$
- **Algorithm:** Check if  $Ax \leq b$
- Simple!

# Assignment Project Exam Help

# BILP Feasibility

Add WeChat powcoder

- Claim:  $3SAT \leq_p$  BILP Feasibility

- Take any formula  $\varphi$  of 3SAT
- Create a binary variable  $x_i$  for each variable  $x_i$  in  $\varphi$ 
  - We'll represent its negation  $\bar{x}_i$  with  $1 - x_i$
- For each clause  $C$ , we want at least one of its three literals to be TRUE
  - Just make sure their sum is at least 1
  - E.g.  $C = x_1 \vee \bar{x}_2 \vee \bar{x}_3 \Rightarrow x_1 + (1 - x_2) + (1 - x_3) \geq 1$
- Easy to check that
  - this is a polynomial reduction
  - Resulting system has a feasible solution iff  $\varphi$  is satisfiable

# Assignment Project Exam Help

So far... Add WeChat powcoder

- To establish NP-completeness of problem B, we always reduced 3SAT to B
    - But we can reduce any other problem A that we have already established to be NP-complete
    - Sometimes this might lead to a simpler reduction because A might already be “similar” to B
- <https://powcoder.com>

- Let's see an example! Add WeChat powcoder

# Vertex Cover

Add WeChat powcoder

- Problem

- **Input:** Undirected graph  $G = (V, E)$ , integer  $k$

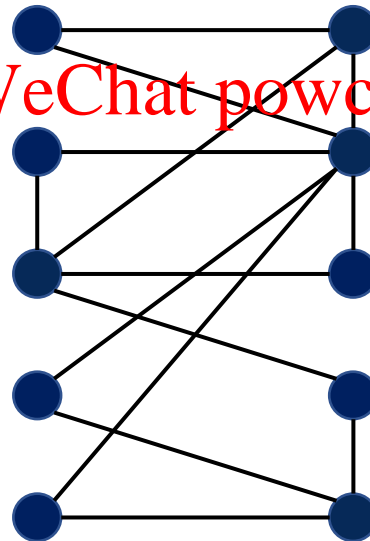
- **Question:** Does there exist a vertex cover of size  $k$ ?

- That is, does there exist  $S \subseteq V$  with  $|S| = k$  such that every edge is incident to at least one vertex in  $S$ ?

<https://powcoder.com>

**Example:**

- Does this graph have a vertex cover of size 4?
  - Yes!
- Does this graph have a vertex cover of size 3?
  - No!



● = vertex cover

Add WeChat powcoder

# Assignment Project Exam Help

## Vertex Cover

Add WeChat powcoder

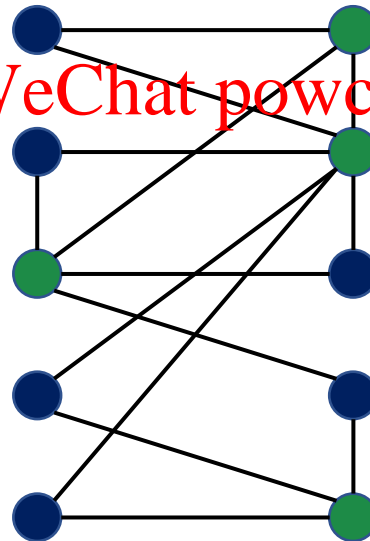
- Problem

- **Input:** Undirected graph  $G = (V, E)$ , integer  $k$
- **Question:** Does there exist a vertex cover of size  $k$ ?
  - That is, does there exist  $S \subseteq V$  with  $|S| = k$  such that every edge is incident to at least one vertex in  $S$ ?

<https://powcoder.com>

**Question:**

- Did we see this graph in the last lecture?
  - Yes!
  - For independent set of size 6



● = vertex cover  
● = independent set

# Vertex Cover

Add WeChat powcoder

- Vertex cover and independent set are intimately connected!
- **Claim:**  $G$  has a vertex cover of size  $k$  if and only if  $G$  has an independent set of size  $n - k$

Assignment Project Exam Help

<https://powcoder.com>

- **Stronger claim:**  $S$  is a vertex cover if and only if  $V \setminus S$  is an independent set

Add WeChat powcoder

# Vertex Cover

Add WeChat powcoder

- **Claim:**  $S$  is a vertex cover if and only if  $V \setminus S$  is an independent set

- **Proof:**

- $S$  is a vertex cover

- IFF: For every  $(u, v) \in E$ , at least one of  $\{u, v\}$  is in  $S$

- IFF: For every  $(u, v) \in E$ , at most one of  $\{u, v\}$  is in  $V \setminus S$

- IFF: No two vertices of  $V \setminus S$  are connected by an edge

- IFF:  $V \setminus S$  is an independent set ■

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# Vertex Cover

Add WeChat powcoder

- Claim: Independent Set  $\leq_p$  Vertex Cover

- Take an arbitrary instance  $(G, k)$  of Independent Set
- We want to check if there is an independent set of size  $k$
- Just convert it to the instance  $(G, n - k)$  of Vertex Cover
- Simple!
  - A reduction from 3SAT would have basically repeated the reduction we already did for  $3\text{SAT} \leq_p \text{Independent Set}$
- **Note:** I didn't argue that Vertex Cover is in NP
  - This is simple as usual. Just give the actual vertex cover as the advice.

# Assignment Project Exam Help

## Set Cover

Add WeChat powcoder

- Problem

- **Input:** A universe of elements  $U$ , a family of subsets  $S$ , and an integer  $k$
- **Question:** Do there exist  $k$  sets from  $S$  whose union is  $U$ ?

Assignment Project Exam Help

- Example

- $U = \{1, 2, 3, 4, 5, 6, 7\}$
- $S = \{\{1, 3, 7\}, \{2, 4, 6\}, \{4, 5\}, \{1\}, \{1, 2, 6\}\}$
- $k = 3$ ? Yes!  $\{\{1, 3, 7\}, \{4, 5\}, \{1, 2, 6\}\}$
- $k = 2$ ? No!

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

## Set Cover

Add WeChat powcoder

- Claim: Set Cover is in NP

➤ Easy. Let the advice be the actual  $k$  sets whose union is  $U$ .

## Assignment Project Exam Help

- Claim: Vertex Cover  $\leq_p$  Set Cover

<https://powcoder.com>

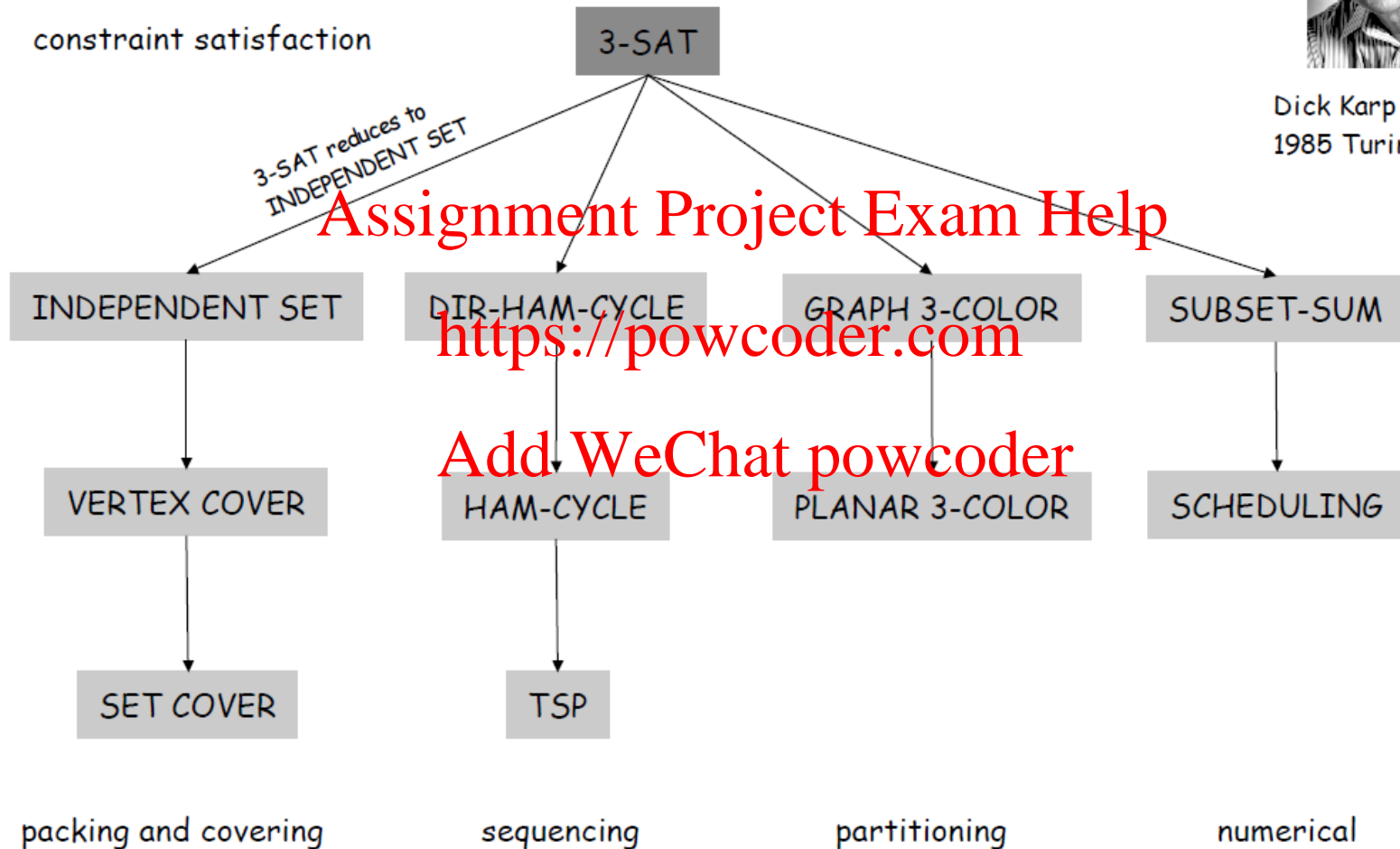
- Given an instance of vertex cover with graph  $G = (V, E)$  and integer  $k$ , create the following set cover instance
- Set  $U = E$
  - For each  $v \in V$ ,  $S$  contains a set  $S_v$  of all edges incident on  $v$
  - Selecting  $k$  set whose union is  $U$  = selecting  $k$  vertices such that union of their incident edges covers all edges
  - Hence, the two problems obviously have the same answer
- Add WeChat powcoder

# Assignment Project Exam Help

Add WeChat powcoder



Dick Karp (1972)  
1985 Turing Award



# Cook-Levin Theorem

- We did not prove “the first NP-completeness” result
- Theorem: 3SAT is NP-complete
  - We need to prove this without using any other “known NP-complete” problem
  - We want to directly show that *every problem in NP* can be reduced to 3SAT
- We will first reduce any NP problem to SAT, and then reduce SAT to 3SAT

# Cook-Levin Theorem

- We're not going to prove it in this class, but the key idea is as follows
  - If a problem is in NP, then  $\exists$  Turing machine  $T(x, y)$  which
    - takes as input a problem instance  $x$  and an advice  $y$  of size  $p(|x|)$
    - verifies in  $q(|x|)$  time whether  $x$  is a YES instance
    - both  $p$  and  $q$  are polynomials
  - $x$  is a YES instance iff  $\exists y T(x, y) = ACCEPT$

# Cook-Levin Theorem

Add WeChat powcoder

NOT IN SYLLABUS

- $x$  is a YES instance iff  $\exists y T(x, y) = ACCEPT$ 
  - We need to convert  $\exists y T(x, y) = ACCEPT$  into whether a SAT formula  $\varphi$  is satisfiable

Assignment Project Exam Help

- Recall that a Turing machine  $T$  consists of a memory tape, a head pointer, a state, and a transition function
- What describes  $T$  at a given step of its computation?
  - What is written in each cell of its memory tape?
  - Which cell of the tape is the read/write head currently pointing to?
  - What state is the Turing machine in?

<https://powcoder.com>

Add WeChat powcoder

## Cook-Levin Theorem

Add WeChat powcoder

NOT IN SYLLABUS

- $x$  is a YES instance iff  $\exists y T(x, y) = ACCEPT$ 
  - We need to convert  $\exists y T(x, y) = ACCEPT$  into  $\exists z \varphi(z) = TRUE$ , where  $z$  consists of Boolean variables and  $\varphi$  is a SAT formula

## Assignment Project Exam Help

## • Variables:

- $T_{i,j,k}$  = True if machine's tape cell  $i$  contains symbol  $j$  at step  $k$  of the computation
- $H_{i,k}$  = True if the machine's read/write head is at tape cell  $i$  at step  $k$  of the computation
- $Q_{q,k}$  = True if machine is in state  $q$  at step  $k$  of the computation
- Cell index  $i$  and computation step  $k$  only need to be polynomially large as  $T$  works in polynomial time



## Cook-Levin Theorem

NOT IN SYLLABUS

Add WeChat powcoder

- $x$  is a YES instance iff  $\exists y T(x, y) = ACCEPT$ 
  - We need to convert  $\exists y T(x, y) = ACCEPT$  into  $\exists z \varphi(z) = TRUE$ , where  $z$  consists of Boolean variables and  $\varphi$  is a SAT formula

Assignment Project Exam Help

- **Clauses:**
  - Express how the variables must be related using the transition function
  - Express that the Turing machine must reach the state *ACCEPT* at some step of the computation
- This establishes that SAT is NP-complete.
- Next:  $SAT \leq_p 3SAT$ .

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

## Cook-Levin Theorem

Add WeChat powcoder

- Claim:  $SAT \leq_p 3SAT$

- Take an instance  $\varphi = C_1 \wedge C_2 \wedge \dots$  of SAT
- Replace each clause with multiple clauses with exactly 3 literals each

### Assignment Project Exam Help

- For a clause with one literal,  $C = \ell_1$ :

- Add two variables  $z_1, z_2$ , and replace  $C$  with four clauses

$$(\ell_1 \vee z_1 \vee z_2) \wedge (\ell_1 \vee \bar{z}_1 \vee z_2) \wedge (\ell_1 \vee z_1 \vee \bar{z}_2) \wedge (\ell_1 \vee \bar{z}_1 \vee \bar{z}_2)$$

- Verify that this is logically equivalent to  $\ell_1$

- For a clause with two literals,  $C = (\ell_1 \vee \ell_2)$ :

- Add variable  $z_1$  and replace it with the following:

$$(\ell_1 \vee \ell_2 \vee z_1) \wedge (\ell_1 \vee \ell_2 \vee \bar{z}_1)$$

- Verify that this is logically equal to  $(\ell_1 \vee \ell_2)$

# Assignment Project Exam Help

# Cook-Levin Theorem

Add WeChat powcoder

- Claim:  $SAT \leq_p 3SAT$

➤ For a clause with three literals,  $C = \ell_1 \vee \ell_2 \vee \ell_3$ :

- Perfect. No need to do anything!

➤ For a clause with 4 or more literals,  $C = (\ell_1 \vee \ell_2 \vee \dots \vee \ell_k)$ :

- Add variables  $z_1, z_2, \dots, z_{k-3}$  and replace it with:

$$(\ell_1 \vee \ell_2 \vee z_1) \wedge (\ell_3 \vee \bar{z}_1 \vee z_2) \wedge (\ell_4 \vee \bar{z}_2 \vee z_3) \wedge \dots \wedge (\ell_{k-2} \vee \bar{z}_{k-4} \vee z_{k-3}) \wedge (\ell_{k-1} \vee \ell_k \vee \bar{z}_{k-3})$$

- Check:

- If any  $\ell_i$  is TRUE, then there exists an assignment of  $z$  variables to make this TRUE
- If all  $\ell_i$  are FALSE, then no assignment of  $z$  variables will make this TRUE

# NP vs co-NP

Add WeChat powcoder

- Complements of each other

- NP = short proof for YES, co-NP = short proof for NO
- If a problem “Does there exist...” is in NP, then its complement “Does there not exist...” is in co-NP, and vice-versa
- The same goes for NP-complete and co-NP-complete

Assignment Project Exam Help

<https://powcoder.com>

- Example

- SAT is NP-complete (“Does there exist  $x$  satisfying  $\varphi$ ?”)
  - So “Does there exist no  $x$  satisfying  $\varphi$ ?”, i.e., “Is  $\varphi$  always FALSE?” is coNP-complete
- Then, Tautology (“Is  $\varphi$  always TRUE?”) is also coNP-complete

Add WeChat powcoder

# Assignment Project Exam Help

## NP $\cap$ co-NP

Add WeChat powcoder

- Clearly,  $P \subseteq NP \cap \text{co-NP}$ 
  - No advice needed; can just solve the problem in polytime
  - Major open question: Is  $P = NP \cap \text{co-NP}$ ?

# Assignment Project Exam Help

- NP  $\cap$  co-NP: Short proof of both YES and NO  
<https://powcoder.com>
  - Hunt for problems not known in P but still in NP  $\cap$  co-NP

Add WeChat powcoder

# Assignment Project Exam Help

## NP $\cap$ co-NP

Add WeChat powcoder

- Linear programming

- [Gale–Kuhn–Tucker 1948]: LP is in NP  $\cap$  co-NP

- **Question:** max objective value  $\geq$  threshold?

- **Proof of YES:** Provide a feasible solution with objective  $\geq$  threshold

- **Proof of NO:** Provide optimal primal and dual solutions

<https://powcoder.com>

### CHAPTER XIX

#### LINEAR PROGRAMMING AND THE THEORY OF GAMES <sup>1</sup>

BY DAVID GALE, HAROLD W. KUHN, AND ALBERT W. TUCKER <sup>2</sup>

The basic “scalar” problem of *linear programming* is to maximize (or minimize) a linear function of several variables constrained by a system of linear inequalities [Dantzig, II]. A more general “vector” problem calls for maximizing (in a sense of partial order) a system of linear functions of several variables subject to a system of linear inequalities and, perhaps, linear equations [Koopmans, III]. The purpose of this chapter is to establish theorems of duality and existence for general “matrix” problems of linear programming which contain the “scalar” and “vector” problems as special cases, and to relate these general problems to the theory of zero-sum two-person games.

# Assignment Project Exam Help

## NP $\cap$ co-NP

Add WeChat powcoder

- Linear programming

- But later, Khachiyan [1979] proved that LP is in P

# Assignment Project Exam Help

ЖУРНАЛ  
ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ  
Том 20 Январь 1989 - Февраль № 1

<https://powcoder.com>

Add WeChat powcoder

УДК 519.852

## ПОЛИНОМИАЛЬНЫЕ АЛГОРИТМЫ В ЛИНЕЙНОМ ПРОГРАММИРОВАНИИ

Л. Г. ХАЧИЯН

(Москва)

Построены точные алгоритмы линейного программирования, трудоемкость которых ограничена полиномом от длины двоичной записи задачи.

# Assignment Project Exam Help

## NP $\cap$ co-NP

Add WeChat powcoder

- Primality testing (“Is  $n$  a prime?”)

- [Pratt 1975]: PRIMES is in NP  $\cap$  co-NP

- Proof of NO: Easy, provide a non-trivial factor

- Proof of YES: relies on interesting math

SIAM J. COMPUT.  
Vol. 4, No. 3, September 1975

<https://powcoder.com>

### EVERY PRIME HAS A SUCCINCT CERTIFICATE\*

Add WeChat powcoder

**Abstract.** To prove that a number  $n$  is composite, it suffices to exhibit the working for the multiplication of a pair of factors. This working, represented as a string, is of length bounded by a polynomial in  $\log_2 n$ . We show that the same property holds for the primes. It is noteworthy that almost no other set is known to have the property that short proofs for membership or nonmembership exist for all candidates without being known to have the property that such proofs are easy to come by. It remains an open problem whether a prime  $n$  can be recognized in only  $\log_2^\alpha n$  operations of a Turing machine for any fixed  $\alpha$ .

The proof system used for certifying primes is as follows.

AXIOM.  $(x, y, 1)$ .

INFERENCE RULES.

$R_1$ :  $(p, x, a), q \vdash (p, x, qa)$  provided  $x^{(p-1)/q} \not\equiv 1 \pmod{p}$  and  $q|(p-1)$ .

$R_2$ :  $(p, x, p-1) \vdash p$  provided  $x^{p-1} \equiv 1 \pmod{p}$ .

THEOREM 1.  $p$  is a theorem  $\equiv p$  is a prime.

THEOREM 2.  $p$  is a theorem  $\supset p$  has a proof of  $[4 \log_2 p]$  lines.



# Assignment Project Exam Help

## NP $\cap$ co-NP

Add WeChat powcoder

- Primality testing (“Is  $n$  a prime?”)
  - Later, Agrawal, Kayal, and Saxena [2004] proved that PRIMES is in P
    - Milestone result!

# Assignment Project Exam Help

<https://powcoder.com>

Annals of Mathematics, 160 (2004), 781–793

Add WeChat powcoder

## PRIMES is in P

By MANINDRA AGRAWAL, NEERAJ KAYAL, and NITIN SAXENA\*

### Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

# NP $\cap$ co-NP

Add WeChat powcoder

- Factoring (“Does  $n$  have a factor  $\leq k$ ?”)

- FACTOR is in NP  $\cap$  co-NP

- Proof of YES: Just present such a factor

- Proof of NO:

<https://powcoder.com>

- Present the entire prime factorization of  $n$  along with a short proof that each presented factor is a prime
    - Verifier TM can check that each factor is indeed a prime, their product is indeed  $n$ , and none of the factors is  $\leq k$
    - Actually, proofs of primality are not required anymore since we know the TM can just run the AKS algorithm to check if the factors are prime

# Assignment Project Exam Help

## NP $\cap$ co-NP

Add WeChat powcoder

- Factoring (“Does  $n$  have a factor  $\leq k$ ?”)

- Major open question: Is FACTOR in P?

- Basis of several cryptographic procedures

## Assignment Project Exam Help

- Challenge: Factor the following number.

<https://powcoder.com>

Add WeChat powcoder

7403756347956171128280467960974295731142593188889231289  
08493623263897276503402826627689199641962511784399589  
43305021275853701189680982867331732731089309005525051  
16877063299072396380786710086096962537934650563796359

RSA-704

(\$30,000 prize if you can factor it)

# NP $\cap$ co-NP

Add WeChat powcoder

- Factoring (“Does  $n$  have a factor  $\leq k$ ?”)
  - [Shor 1994]: We can factor an  $n$ -bit integer in  $O(n^3)$  steps on a quantum computer.
  - \*Scalable\* quantum computers can help
    - 2001: Factored  $15 = 3 \times 5$
    - 2012: Factored  $21 = 3 \times 7$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help Other Complexity Classes

Add WeChat powcoder

- Based on the exact time complexity
  - $\text{DTIME}(n)$ ,  $\text{NTIME}(n^2)$ , ...
    - Deterministic / nondeterministic time complexity
- Based on space complexity
  - $\text{DSPACE}(n)$ ,  $\text{NSPACE}(\log n)$
- Using randomization
  - ZPP (expected polynomial time, no errors)
    - Is  $P = ZPP$ ?
- Allowing probabilistic errors
  - RP (polynomial time, one-sided error)
  - BPP (polynomial time, two-sided errors)