

Exercise walk-through

10.6

Reductions and NP-completeness

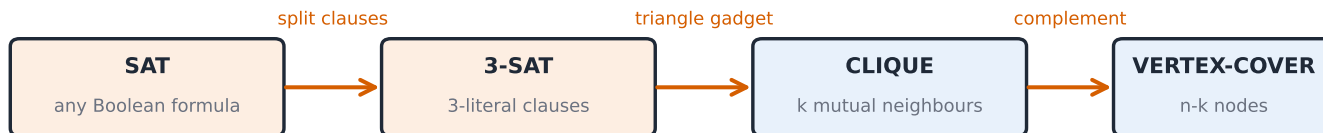
University Computer Science

Reductions: the arrow carries the difficulty

- 1 **Chain** SAT to 3-SAT (clause splitting) to CLIQUE (the triangle gadget) to VERTEX-COVER (complement the graph) - solve one, solve the chain.
- 2 **Course** Triangle ABC plus isolated D: complement edges are A-D, B-D, C-D; cover $\{D\}$ of size $n - k = 1$ covers them all.
- 3 **Direction** Reducing YOUR problem to SAT proves nothing - the arrow must carry difficulty INTO your problem, from a known hard one.

Reduction map

difficulty flows along the arrows



each arrow is a polynomial translator: a solver at the right end solves the left end

course case: G = triangle ABC + isolated D (3-clique); complement edges A-D, B-D, C-D

cover $\{D\}$ of size $n - k = 1$ covers them all

NP-complete = in NP + every NP problem reduces to it

the arrow carries the difficulty: reduce the KNOWN hard problem TO yours

P sits inside NP; nobody knows if they are equal

SAT to 3-SAT to CLIQUE to VERTEX-COVER - solve one, solve the chain.

Check yourself

- One level of $2T(n/2) + n$ costs...? (n .)
- $4T(n/2) + n$ solves to...? (n^2 .)
- Naive fib(30) makes... calls? (2,692,537.)
- To prove B NP-hard, reduce...? (A known hard problem TO B.)

Exercises 2.1-2.3

2.1 Which direction carries hardness: $A \rightarrow B$ means – is hard? [3]

2.2 What two conditions define NP-completeness? [3]

2.3 Translate: $G = \text{triangle ABC} + \text{isolated D}$, $k = 3$. What VERTEX-COVER instance results? [3]

Exercises 2.1-2.3

2.1 Which direction carries hardness: $A \rightarrow B$ means – is hard? [3]

2.1

- B (the target): if A is hard, B must be at least as hard ✓.

2.2 What two conditions define NP-completeness? [3]

2.2

- The problem is in NP (polynomial verification); every NP problem reduces to it ✓.

2.3 Translate: $G = \text{triangle ABC} + \text{isolated D}, k = 3$. What VERTEX-COVER instance results? [3]

2.3

- $(\bar{G}, n - k) = (\bar{G}, 1)$: complement edges A–D, B–D, C–D; cover $\{D\}$ ✓.

Exercise 3.1

3.1 Reduce CLIQUE to VERTEX-COVER in full.

State the translator (graph, parameter) and its running time. Prove both directions: a k -clique exists in G iff a cover of size $n - k$ exists in $\bar{G}$. Run it on $G = \text{triangle } ABC + \text{isolated } D$ with $k = 3$: list $\bar{G}$'s edges and the cover. State what a polynomial vertex-cover solver would then imply. [8]

Exercise 3.1

3.1 Reduce CLIQUE to VERTEX-COVER in full.

State the translator (graph, parameter) and its running time. Prove both directions: a k -clique exists in G iff a cover of size $n - k$ exists in $\bar{G}$. Run it on $G = \text{triangle ABC} + \text{isolated D}$ with $k = 3$: list $\bar{G}$'s edges and the cover. State what a polynomial vertex-cover solver would then imply. [8]

3.1

- Translator: $(G, k) \mapsto (\bar{G}, n - k)$ – build the complement in $O(V^2)$ ✓.
- Forward: a k -clique's nodes are mutually non-adjacent in $\bar{G}$, so every $\bar{G}$ -edge has an endpoint outside it – the other $n - k$ nodes form a cover ✓.
- Backward: a cover S of size $n - k$ leaves $n - |S| = k$ nodes touching no $\bar{G}$ -edge – an independent set in $\bar{G}$, i.e. a clique in G ✓.
- Run: $\bar{G}$'s edges are exactly A–D, B–D, C–D; $\{D\}$ covers them all at size $1 = n - k$ ✓.
- Implication: a polynomial vertex-cover solver would solve CLIQUE, hence SAT, hence every NP problem – $P = NP$ ✓.

Exercise 3.2

3.2 A colleague proves their problem WARS hard by writing a SAT encoding of WARS and running a SAT solver.

Name the direction they actually proved ($\text{WARS} \leq_p \text{SAT}$) and what it implies about a SAT solver. State what they must do instead to prove WARS NP-hard. State the two checks a full NP-completeness claim needs, and what remains unproven if the membership check is skipped.

[7]

Exercise 3.2

3.2 A colleague proves their problem WARS hard by writing a SAT encoding of WARS and running a SAT solver.

Name the direction they actually proved ($\text{WARS} \leq_p \text{SAT}$) and what it implies about a SAT solver. State what they must do instead to prove WARS NP-hard. State the two checks a full NP-completeness claim needs, and what remains unproven if the membership check is skipped. [7]

3.2

- They proved $\text{WARS} \leq_p \text{SAT}$: a SAT solver solves WARS instances – it says nothing about WARS being hard ✓.
- Instead: reduce a known NP-complete problem (SAT, 3-SAT, CLIQUE, ...) TO WARS with a polynomial translator ✓.
- NP-completeness needs: membership in NP (a verifiable certificate in polynomial time) plus NP-hardness by such a reduction ✓.
- Skipping membership: WARS might be harder than NP (e.g. outside NP) – then calling it NP-complete is simply unproven ✓.