$\mathcal{NP}$-Completeness

We consider the hardest problems in $\mathcal{NP}$.

Reductions Revisited

A function f (mapping strings to strings) is ***polynomial-time computable*** if there is constant k and TM that computes f in $O(n^k)$ time.

A language A is ***polynomial-time reducible*** to language B, if A is reducible to B via a polynomial-time computable function. Written $A \leq_p B$.

Reductions Preserve Hardness

The key result is as before:

Fact. *a) If* $A \le_p B$ *and* B *in* $\mathcal{P}$*, then* A *in* $\mathcal{P}$*.*
b) If $A \le_p B$ *and* A *not in* $\mathcal{P}$*, then* B *not in* $\mathcal{P}$*.*

Proof (of a). Say reduction from A to B given by f computable in $O(n^k)$ time, and one can decide membership in B in $O(n^\ell)$ time.

Then build the obvious decider for A: it takes input w, computes $f(w)$ and sees whether $f(w) \in B$. This runs in $O(n^{k\ell})$ time.

$\mathcal{NP}$-*Complete*

Definition. *Language S is **$\mathcal{NP}$-complete** if*
a) $S \in \mathcal{NP}$; and
b) for all A in $\mathcal{NP}$ it holds that $A \leq_P S$.

Note that this means that:

If S is $\mathcal{NP}$-complete and S in $\mathcal{P}$, then $\mathcal{P} = \mathcal{NP}$.

The First Theorem

There are many $\mathcal{NP}$-complete problems. What started the whole process was the great idea:

Cook's Theorem. `SAT` *is* $\mathcal{NP}$*-complete.*

We omit the proof.

Examples

- HAMPATH is $\mathcal{NP}$-complete.
- SUBSET_SUM is $\mathcal{NP}$-complete.

(Proof of latter later.) We saw earlier that both are in $\mathcal{NP}$.

More Graph Terminology

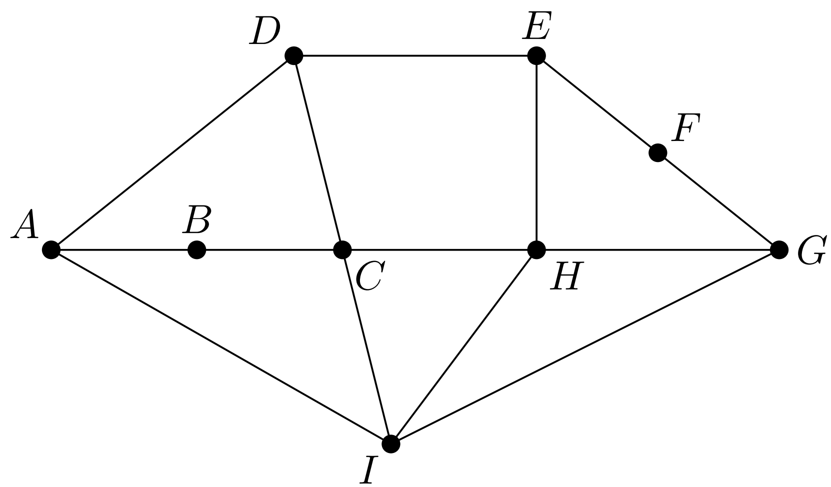
A set of nodes C is a ***clique*** if every two nodes in C are joined by an edge.

A set of nodes D is a ***dominating set*** if every other node is adjacent to at least one node in D.

A set of nodes V is a ***vertex cover*** if the removal of V destroys every edge.

Example Graph

Here $\{C, H, I\}$ is clique, $\{A, C, F\}$ is dominating set, and $\{A, C, E, G, I\}$ is vertex cover.



Example Graph Problems

We write our examples as decision problems. The following are $\mathcal{NP}$-complete:

The `CLIQUE` problem:
Input: graph G and integer k
Question: is there clique of at least k nodes?

The `DOMINATION` problem:
Input: graph G and integer k
Question: is there dominating set of at most k nodes?

The `VERTEX_COVER` problem:
Input: graph G and integer k
Question: is there vertex cover of at most k nodes?

3SAT *is* $\mathcal{NP}$*-complete*

The 3SAT problem is $\mathcal{NP}$-complete:

Input: ϕ a boolean formula in conjunctive normal form with 3 literals per clause (***3CNF***).
Question: is there a satisfying assignment?

Summary

The $\mathcal{NP}$-complete languages are the hardest languages in $\mathcal{NP}$ and every language in $\mathcal{NP}$ polynomially reduces to these. Examples of $\mathcal{NP}$-complete languages include `SAT` and `HAMPATH`.