THE BIG QUESTION

It follows from the definition that $P \subseteq NP$.

~~IS N=NP?~~ $P = NP$?

Since 1971 this is the most famous open problem in computer science.

Most people believe that the answer is no. Then there must be problems in NP - P. SAT would be a plausible candidate.

It seems as if hard NP-Problems can be reduced to each other. This observation leads us to the following definition.

NP-Completeness: A problem Q is NP-Complete if

1. Q is in NP.
2. For each A in NP, there is a reduction from A to Q, i.e. all NP problems can be reduced to Q.

Are there any NP-Complete problems? Well, there are:

Cook's Theorem: SAT is NP-Complete

Other NP-Complete problems

It is ease to see that reductions are transitive, i.e.

$A \lesssim B$ and $B \lesssim C \Rightarrow A \lesssim C$

We know that SAT $\lesssim$ INDEPENDENT SET. We also know that for each A in NP we have $A \leq$ SAT. But this means that for all A in NP we have $A \leq$ INDEPENDENT SET

So INDEPENDENT SET is an NP-Complete problem.

We realize that the NP-Complete problems must be the hardest problems in NP. If any NP-Complete problem can be solved efficiently then all can!

So we wouldn't expect to be able to find efficient solutions to NP-Complete problems.

The best way to "show" that a problem is impossible to solve efficiently is to show that it is NP-Complete.

This is the core of applied Complexity Theory.

But how do we show that a problem is NP-Complete?

Proving NP-Completeness

In order to show that A is NP-Complete it is enough to show that $A \in NP$ and $SAT \leq_P A$. **Why:** If $X \in$ we know that $X \leq SAT$. If we also have $SAT \leq A$ we know that $X \leq A$! This shows that A is NP-Complete.

Another approach: We can form i directed graph such that $A \rightarrow B$ means $A \leq B$. $SAT \rightarrow A \rightarrow B \rightarrow C \rightarrow$ ... tells us that $A, B, C, ...$ are NP-Complete.

To show that A is NP-Complete we can try to find a known NP-Complete problem B such that $B \leq A$.

NP-problems continued

Since SAT and INDEPENDENT SET can be reduced to each other we might think that there would be some similarities between the two problems. In fact, there is one such similarity.

In SAT we want to know if something exists. We are looking for aset of values for to coordinate such that the formula is true. It is hard to find such a set of values but if we have found it, it is easy to check if it makes the formula true.

In INDEPENDENT SET we are looking for a set of nodes of size K such that the set forms an independent set. I is hard to find the set but if we have found it, it is easy to check if it really is an independent set.

Both the problems have a so called yes-certificate, something that tells us that the answer to the problem is yes. For SAT, the certificate is the values for the variables. For INDEPENDENT SET, the certificate is the K-set.

Informally, the class NP is the set of decision problems such that if the answer to the problem with input x is yes, then is a certificate y, at most polynomial in the size of x such that it can be checked in polynomial time (in the size of x) that y is a yes- certificate.

We will give a more formal definition of this. The definition identify problems with something we will call languages. Then we will describe the property of being an NP-problems as a property for languages.

A formal definition of NP

A *verifies* the instance x of the problem L if there is a certificate y such that $|y| \in O(|x|^s)$ and

$$A(x, y) = \text{Yes} \quad \Leftrightarrow \quad x \in L$$

This means that A decides the language
$$L = \{x \in \{0,1\}^* : \exists y \in \{0,1\}^* : A(x,y) = \text{Ja}\}$$

$NP = \{L : \exists A$ that verifies L in polynomial time$\}$

$P \subseteq$ NP since all problem that can be decided in polynomial time also can be verified in polynomial time.

A second definition of NP:

A non-deterministic algorithm is an algorithm that makes random choices. The output is stochastic. We say that A decides a language L if:

$$\begin{aligned} x \in L \Rightarrow A(x) = \text{Yes} &\quad \text{with probabilty} > 0 \\ x \notin L \Rightarrow A(x) = \text{No} &\quad \text{with probability } 1 \end{aligned}$$

$NP = \{L$: $\exists$polynomial time non-deterministic algorithm that decides $L\}$

Reductions: Case studies

We have already seen that INDEPENDENT SET is NP-Complete by reducing 3-CNF-SAT to IS.

It can be shown that HAMILTONIAN CYCLE is NP-C by reducing 3-CNF-SAT to HC. This reduction is rather complicated and we don't give it here.

If we now know that HC is NP-C, we can show that some other problems X are NP-C by explicitly describing reductions HC $\leq$ X. We will do this for the problems TSP and SUBGRAPH ISOMORPHISM.

When you prove that a problem is NP-C you must remember that it is not enough to give a reduction. You also have to show that the problem is in NP. This essentially means that you have to show that solutions (certificates) can be verified in polynomial times. In most cases this is quite simple.

HAMILTONIAN CYCLE $\leq$ TSP

TSP

Input: A weighted complete graph G and a number K.
Goal: Is there a Hamiltonian cycle of length at most $\leq K$ in G?

HAMILTONIAN CYCLE

Input: A graph G.
Goal : Is there a Hamiltonian cycle in G?

Let $x = G$ be input to HC. We construct a complete graph G' with $w(e) = 0$ if $e \in G$ and $w(e) = 1$ if $e \notin G$. Then set $K = 0$. This will be the input to the TSP.

Subgraph isomorphism is NP-Complete

Given two graphs G_1 and G_2, Is G_1 a subgraph of G_2?

The problem obviously belongs to NP.

We reduce from Hamilton Cycle.

A graph $G = (V, E)$ contains a Hamiltonian cycle if and only if it contains a subgraph that is a cycle C with $|V|$ nodes. So we can set $G_1 = C$ and $G_2 = G$. som G.

Other NP-Complete problems

Exact Cover
Given a set of subsets of a set M, is it possible to find a selection of the subsets such that each element in M is in exactly one of the subsets?

Subset Sum
Given a set P of positive integers and an integer K, is there a subset of the numbers in P with sum K?

Integer Programming
Given an $m \times n$-matrix A, an m-vektor b, an n-vektor c and a number K, is there an n-vektor x with integer coefficients such that $Ax \leq b$ and $c \cdot x \geq K$?

If we relax the condition that the coefficients x should be integers we get a special case of **Linear Programming**.

Some more advanced reductions

We will look at some reductions that are more complicated. We will show that 3-COLORABILITY, EXACT COVER and SUBSET SUM are NP-C.

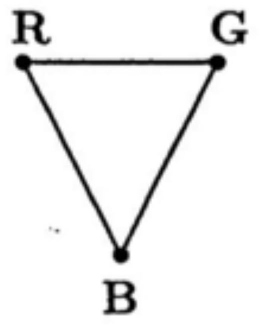
3-COLORABILITY

This is the problem of deciding if a given graph G can be colored with 3 colors or not. We will reduce 3-CNF-SAT to this problem. This means that given a general instance of 3-CNF-SAT, we will construct a very special instance of 3-COLORABILITY such that the formula is satisfiable if and only if the graph is 3-colorable.

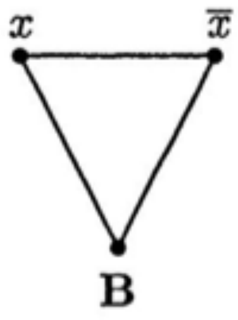
We present one possible solution, take from the literature.

Let $\mathcal{B}$ be a Boolean formula in CNF. We will construct a graph G that is 3-colorable iff $\mathcal{B}$ is satisfiable.

There will be three special vertices called **R**, **B**, and **G**, which will be connected in a triangle. In any 3-coloring, they will have to be colored with different colors, so we assume without loss of generality that they are colored red, blue, and green, respectively.

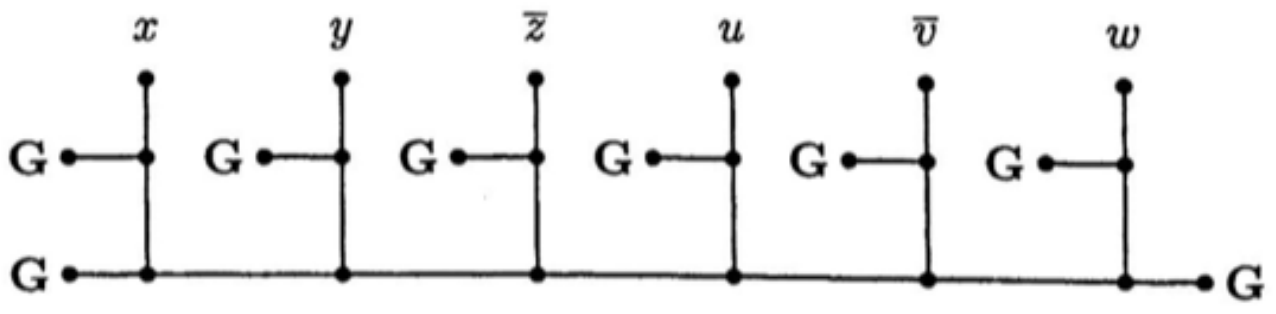


We include a vertex for each literal, and connect each literal to its complement and to the vertex **B** as shown.



In any 3-coloring, the vertices corresponding to the literals x and $\overline{x}$ will have to be colored either red or green, and not both red or both green. Intuitively, a legal 3-coloring will represent a satisfying truth assignment in which the green literals are true and the red literals are false.

To complete the graph, we add a subgraph like the one shown below for each clause in $\mathcal{B}$. The one shown below would be added for the clause $(x \vee y \vee \overline{z} \vee u \vee \overline{v} \vee w)$. The vertices in the picture labeled **G** are all the same vertex, namely the vertex **G**.



This subgraph has the property that a coloring of the vertices on the top row with either red or green can be extended to a 3-coloring of the whole subgraph iff at least one of them is colored green. If all vertices on the top row are colored red, then all the vertices on the middle row adjacent to vertices on the top row must be colored blue. Starting from the left, the vertices along the bottom row must be colored alternately red and green. This will lead to a conflict with the last vertex in the bottom row. (If the number of literals in the clause is odd instead of even as pictured, then the rightmost vertex in the bottom row is **R** instead of **G**.)

Conversely, suppose one of the vertices on the top row is colored green. Pick one such vertex. Color the vertex directly below it in the middle row red and the vertex directly below that on the bottom row blue. Color all other vertices on the middle row blue. Starting from the left and right ends, color the vertices along the bottom row as forced, either red or green. The coloring can always be completed.

Thus if there is a legal 3-coloring, then the subgraph corresponding to each clause must have at least one green literal, and truth values can be assigned so that the green literals are true. This gives a satisfying assignment. Conversely, if there is a satisfying assignment, color the true variables green and the false ones red. Then there is a green literal in each clause, so the coloring can be extended to a 3-coloring of the whole graph.

From this it follows that $\mathcal{B}$ is satisfiable iff G is 3-colorable, and the graph G can be constructed in polynomial time. Therefore CNFSat $\leq_{\mathrm{m}}^{\mathrm{p}}$ 3-colorability.

One can trivially reduce 3-colorability to k-colorability for $k > 3$ by appending a $k-3$ clique and edges from every vertex of the $k-3$ clique to every other vertex.

Proof. Suppose we are given an undirected graph $G = (V, E)$. We show how to produce an instance (X, S) of the exact cover problem for which an exact cover exists iff G has a 3-coloring.

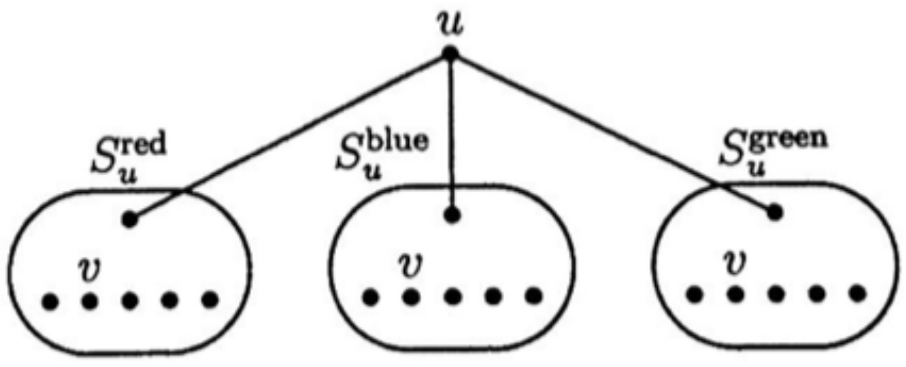
Let $C = \{\text{red, blue, green}\}$. For each $u \in V$, let $N(u)$ be the set of neighbors of u in G. Since G is undirected, $u \in N(v)$ iff $v \in N(u)$.

For each $u \in V$, we include u in X along with $3(|N(u)| + 1)$ additional elements of X. These $3(|N(u)| + 1)$ additional elements are arranged in three disjoint sets of $|N(u)| + 1$ elements each, one set corresponding to each color. Call these three sets S_u^{red}, S_u^{blue}, S_u^{green}. For each color $c \in C$, pick a special element p_u^c from S_u^c and associate the remaining $|N(u)|$ elements of S_u^c with the elements of $N(u)$ in a one-to-one fashion. Let q_{uv}^c denote the element of S_u^c associated with $v \in N(u)$.

The set S will contain all two element sets of the form

$$\{u, p_u^c\} \tag{31}$$

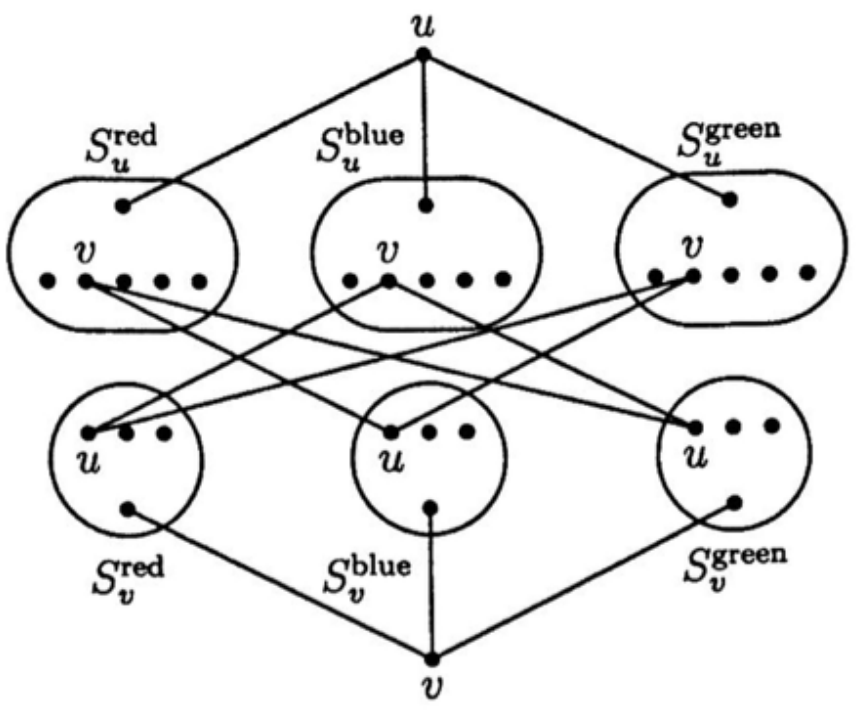
for $u \in V$ and $c \in C$, as well as all the sets S_u^c for $u \in V$ and $c \in C$. Here is a picture of what we have so far for a vertex u of degree 5 with $v \in N(u)$. The ovals represent the three sets S_u^c and the lines represent the three two-element sets (31).



To complete S, we include all two element sets of the form

$$\{q_{uv}^c, q_{vu}^{c'}\} \tag{32}$$

for all $(u, v) \in E$ and $c, c' \in C$ with $c \neq c'$. Here is a picture showing a part of the construction for two vertices u and v of degrees 5 and 3 respectively, where (u, v) in E. The six lines in the center represent the two-element sets (32).



We now argue that the instance (X, S) of Exact Cover just constructed is a "yes" instance, *i.e.* an exact cover $S' \subseteq S$ of X exists, iff the graph G has a 3-coloring. Suppose first that G has a 3-coloring $\chi : V \to C$. We construct an exact cover $S' \subseteq S$ as follows. For each vertex u, let S' contain the sets $\{u, p_u^{\chi(u)}\}$ and S_u^c for $c \neq \chi(u)$. This covers everything except points of the form $q_{uv}^{\chi(u)}$, where $(u, v) \in E$. For each edge (u, v), let S' also contain the set $\{q_{uv}^{\chi(u)}, q_{vu}^{\chi(v)}\}$. This set is in S since $\chi(u) \neq \chi(v)$. This covers all the remaining points, and each point is covered by exactly one set in S'.

Conversely, suppose S' is an exact cover. Each u is covered by exactly one set in S', and it must be of the form $\{u, p_u^c\}$ for some c. Let $\chi(u)$ be that c; we claim that χ is a valid coloring, *i.e.* that if $(u, v) \in E$ then $\chi(u) \neq \chi(v)$. For each u, since $\{u, p_u^{\chi(u)}\} \in S'$, we cannot cover p_u^c for $c \neq \chi(u)$ by any set of the form (31), since u is already covered; therefore they must be covered by the sets S_u^c, which are the only other sets containing the points p_u^c. The sets $\{u, p_u^{\chi(u)}\}$ and S_u^c, $c \neq \chi(u)$ cover all points except those of the form $q_{uv}^{\chi(u)}$, $(u, v) \in E$. The only way S' can cover these remaining points is by the sets (32). By construction of S, these sets are of the form $\{q_{uv}^{\chi(u)}, q_{vu}^{\chi(v)}\}$ for $(u, v) \in E$ and $\chi(u) \neq \chi(v)$. □

Both Subset Sum and Partition reduce to Knapsack. To reduce Partition to Knapsack, take $b = w$ and $W = B = \frac{1}{2}\Sigma$.

We show that these three problems are as hard as Exact Cover by reducing Exact Cover to Subset Sum. Assume that $X = \{0, 1, \ldots, m-1\}$ in the given instance (X, S) of Exact Cover. For $x \in X$, define

$$\#x \quad = \quad |\{A \in S \mid x \in A\}| \ ,$$

the number of elements of S containing x. Let p be a number exceeding all $\#x$, $0 \leq x \leq m-1$. Encode $A \in S$ as the number

$$w(A) \quad = \quad \sum_{x \in A} p^x$$

and take

$$B \quad = \quad \sum_{x=0}^{m-1} p^x \quad = \quad \frac{p^m - 1}{p - 1} \ .$$

In p-ary notation, $w(A)$ looks like a string of 0's and 1's with a 1 in position x for each $x \in A$ and 0 elsewhere. The number B in p-ary notation looks like a string of 1's of length m. Adding the numbers $w(A)$ simulates the union of the sets A. The number p was chosen big enough so that we do not get into trouble with carries. Asking whether there is a subset sum that gives B is the same as asking for an exact cover of X.

PARTITIONING $\leqslant$ KNAPSACK

Let $\{a_1, a_2, ..., a_n\}$ be an instance of PARTITIONING. In KNAPSACK we have pairs of numbers $\{(u_i, w_i)\}, U, W$ and we want to know if there is a selection of pairs such that $\sum u_i \geqslant U$ and $\sum w_i \leqslant W$. Given our instance of PARTITIONING we can set $A = \sum a_i$ and give $\{(a_i, a_i)\}, A/2, A/2$ as an instance to KNAPSACK. This KNAPSACK-problem has a solution if and only if there is a partitioning.

We already know that SUBSET SUM $\leqslant$ PARTITIONING. This shows that SUBSET SUM $\leqslant$ KNAPSACK by transitivity. It is easy to give a direct proof this, similar to the proof above. We have shown that all three problems are NP-Complete.

0/1-programming is NP-Complete

Given an $m \times n$-matris A and an m-vektor b. **Is there an n-vektor x with coefficients $\in \{0, 1\}$ such that $Ax \leq b$?**

The problem is in NP since, given x, we can check in time $O(n^2)$ if $Ax \leq b$.

We reduce from 3-CNF-SAT:
Let Φ be an instance of 3-CNF-SAT With n variables. To each x_i in Φ we define a corresponding variable $y_i \in \{0, 1\}$ and let 1 Mean True and 0 mean False.

FOr each clause $c_j = l_1 \vee l_2 \vee l_3$ we define an inequality

$$T(l_1) + T(l_2) + T(l_3) \geq 1$$

where $T(x_i) = y_i$ and $T(\neg x_i) = (1 - y_i)$.

And that's it!

We have now showed that a set of NP problems are NP-Complete by chains of reductions.

$$\begin{tikzcd}
 & & \text{TSP} & \\
 & & \text{HAMILTONIAN CYCLE} \arrow[u] & \\
\text{CNF-SAT} \arrow[rr] & & \text{3-CNF-SAT} \arrow[u] \arrow[r] \arrow[dl] \arrow[d] & \text{3-COLORABILITY} \arrow[d] \\
 & \text{0/1-PROGRAMMING} & \text{INDEPENDENT SET} \arrow[d] & \text{EXACT COVER} \arrow[d] \\
 & & \text{VERTEX COVER} \arrow[d] & \text{SUBSET SUM} \arrow[d] \\
 & & \text{SUBGRAPH ISOMORPHISM} & \text{PARTITIONING} \arrow[d] \\
 & & & \text{KNAPSACK}
\end{tikzcd}$$