 

We present reductions from 3SAT to several provlems.

# 1 Independent Set

Given an undirected non-weighted graph $G = (V, E)$, an *independent set* is a subset $I \subseteq V$ of the vertices such that no two vertices of $I$ are adjacent.

We will be interested in the following optimization problem: given a graph, find a largest independent set. We have seen that this problem is solvable in polynomial time in trees using dynamic programming. In the general case, unfortunately, it is much harder.

The problem models the execution of conflicting tasks, it is related to the construction of error-correcting codes, and it is a special case of more interesting problems. We are going to prove that it is not solvable in polynomial time unless P = NP.

First of all, we need to formulate it as a search problem:

- Given a graph $G$ and an integer $k$, find an independent set in $G$ with at least $k$ vertices, if it exists.

It is easy to see that the problem is in NP. We have to see that it is NP-hard. We will reduce 3SAT to Maximum Independent Set.

Starting from a formula $\phi$ with $n$ variables $x_1, \ldots, x_n$ and $m$ clauses, we generate a graph $G_\phi$ with $3m$ vertices, and we show that the graph has an independent set with at least $m$ vertices if and only if the formula is satisfiable, and that it is possible to map satisfying assignments back to independent sets of size at least $m$. (In fact we show that the size of the largest independent set in $G_\phi$ is equal to the maximum number of clauses of $\phi$ that can be simultaneously satisfied. — This is more than what is required to prove the NP-completeness of the problem)

The graph $G_\phi$ has a triangle for every clause in $\phi$. The vertices in the triangle correspond to the three literals of the clause.

Vertices in different triangles are joined by an edge iff they correspond to two literals that are one the complement of the other. In Figure 1 we see the graph resulting by applying the reduction to the following formula:

$$(x_1 \vee \bar{x}_5 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_3 \vee x_2 \vee x_4)$$

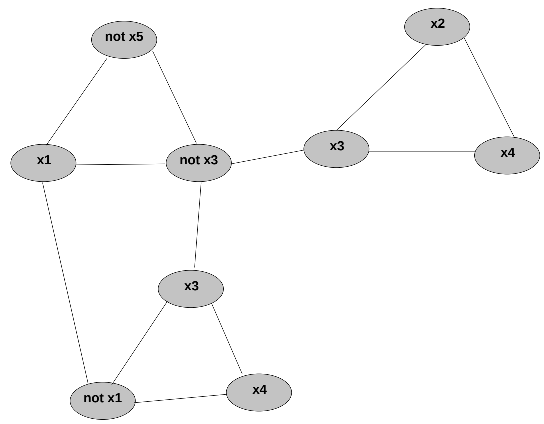


Figure 1: The reduction from 3SAT to Independent Set.

To prove the correctness of the reduction, we need to show that:

- If $\phi$ is satisfiable, then there is an independent set in $G_\phi$ with at least $m$ vertices.
- If there is an independent set in $G$ with at least $m$ vertices, then $\phi$ is satisfiable.

**From Satisfying Assignment to Independent Set.** Suppose we have an assignment of Boolean values to the variables $x_1, \ldots, x_n$ of $\phi$ such that all the clauses of $\phi$ are satisfied. This means that for every clause, at least on of its literals is satisfied by the assignment. We construct an independent set as follows: for every triangle we pick a node that corresponds to a satisfied literal (we break ties arbitrarily). It is impossible that two such nodes are adjacent, since only nodes that corresponds to a literal and its negation are adjacent; and they cannot be both satisfied by the assignment.

**From Independent Set to Satisfying Assignment.** Suppose we have an independent set $I$ with $m$ vertices. We better have exactly one vertex in $I$ for every triangle. (Two vertices in the same triangle are always adjacent.) Let us fix an assignment that satisfies all the literals that correspond to vertices of $I$. (Assign values to the other variables arbitrarily.) This is a consistent rule to generate an assignment, because we cannot have a literal and its negation in the independent set). Finally, we note how every clause is satisfied by this assignment.

Wrapping up:

- We showed a reduction $\phi \to (G_\phi, m)$ that given an instance of 3SAT produces an instance of the decision version of Maximum Independent Set.

- We have the property that $\phi$ is satisfiable (answer YES for the 3SAT problem) if and only if $G_\phi$ has an independent set of size at least $m$, and how to translate back a satisfying assignment to an independent set of size at least $m$.

- We knew 3SAT is NP-hard.

- Then also Max Independent Set is NP-hard; and so also NP-complete.

## 2 Set Cover

In the Set Cover problem, we are given a collection $S_1, \ldots, S_m$ of subsets of a set $X$, such that $\bigcup_{i=1}^m S_i = X$, and we want to find a minimal subcollection of sets whose union is still $X$. To formulate it as an NP search problem, we say that given sets $S_1, \ldots, S_m$ and an integer $k$ we ask whether there is a subset $I \subseteq \{1, \ldots, m\}$ such that $|I| = k$ and $\bigcup_{i \in I} S_i = X$.

We reduce Vertex Cover to Set Cover. Given a graph $G = (V, E)$ with $n$ vertices and $m$ edges, we create a set system where $X = E$, and there is a set $S_v$ for every vertex $v \in V$ of the graph, defined as

$$S_v := \{(u, v) : u \in V mbox and (u, v) \in E\}$$

that is, as the set of edges that touch the vertex $v$. It is easy to see that there is a vertex cover with $k$ vertices in $G$ if and only if there is a set cover with $k$ sets in the instance of set cover described above.

## 3 Subset Sum

The **Subset Sum** problem is defined as follows:

- Given a sequence of integers $a_1, \ldots, a_n$ and a parameter $k$,

- Find a subset of the integers whose sum is exactly $k$. Formally, find a subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} a_i = k$.

Subset Sum is a true *search problem*, not an optimization problem forced to become a search problem. It is easy to see that Subset Sum is in NP.

We prove that Subset Sum is NP-complete by reduction from Independent Set. We have to proceed as follows:

- Start from a graph $G$ and a parameter $k$.

- Create a sequence of integers and a parameter $k'$.
- Prove that the graph has vertex cover with $k$ vertices iff there is a subset of the integers that sum to $k'$.

Let then $G = (V, E)$ be our input graph with $n$ vertices, and let us assume for simplicity that $V = \{1, \ldots, n\}$, and let $k$ be the parameter of the independent set problem.

We define integers $a_1, \ldots, a_n$, one for every vertex; and also integers $b_{(i,j)}$, one for every edge $(i, j) \in E$; and finally a parameter $k'$. We will define the integers $a_i$ and $b_{(i,j)}$ so that if we have a subset of the $a_i$ and the $b_{(i,j)}$ that sums to $k'$, then: the subset of the $a_i$ corresponds to an independent set $I$ in the graph; and the subset of the $b_{(i,j)}$ corresponds to the edges in the graph that are not touched by $I$. Furthermore the construction will force $C$ to be of size $k$.

How do we define the integers in the subset sum instance so that the above properties hold? We represent the integers in a matrix. Each integer is a row, and the row should be seen as the base-10 (but it would also work in base-4) representation of the integer, with $|E| + 1$ digits.

The first column of the matrix (the "most significant digit" of each integer) is a special one. It contains 1 for the $a_i$s and 0 for the $b_{(i,j)}$s.

Then there is a column (or digit) for every edge. The column $(i, j)$ has a 1 in $a_i$, $a_j$ and $b_{(i,j)}$, and all 0s elsewhere.

The parameter $k'$ is defined as

$$k' := k \cdot 4^{|E|} + \sum_{j=0}^{|E|-1} 1 \cdot 4^i$$

This completes the description of the reduction. Let us now proceed to analyze it.

**From Independent Sets to Subsets** Suppose there is an independent set $I$ of size $k$ in $G$. Then we choose all the integers $a_i$ such that $i \in I$ and all the integers $b_{(i,j)}$ such that neither of $i$ and $j$ are in $I$. Then, when we sum these integers, we have a 1 in all digits except for the most significant one. In the most significant digit, we are summing the digit 1 a number of times equal to $|I| = k$. The sum of the integers is thus $k'$.

**From Subsets to Covers** Suppose we find a subset $I \subseteq V$ and $E' \subseteq E$ such that

$$\sum_{i \in I} a_i + \sum_{(i,j) \in E'} b_{(i,j)} = k'$$

First note that we never have a carry in the $|E|$ less significant digits: operations are in base 10 and there are at most 3 ones in every column. This means that for every edge $(i, j)$ $I$ must not contain both $i$ and $j$. So $I$ is a cover. The "most significant digit" (technically, the quotient of $k'$ divided by $4^{|E|}$) is $k$, so $I$ contains $k$ elements.

# 4 Partition

The **Partition** problem is defined as follows:

- Given a sequence of integers $a_1, \ldots, a_n$.
- Determine whether there is a partition of the integers into two subsets such the sum of the elements in one subset is equal to the sum of the elements in the other.

  Formally, determine whether there exists $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} a_i = (\sum_{i=1}^{n} a_i)/2$.

Clearly, Partition is a special case of Subset Sum. We will prove that Partition is is NP-hard by reduction from Subset Sum.[1]

Given an instance of Subset Sum we have to construct an instance of Partition. Let the instance of Subset Sum have items of size $a_1, \ldots, a_n$ and a parameter $k$, and let $A = \sum_{i=1}^{n} a_i$.

Consider the instance of Partition $a_1, \ldots, a_n, b, c$ where $b = 2A - k$ and $c = A + k$.

Then the total size of the items of the Partition instance is $4A$ and we are looking for the existence of a subset of $a_1, \ldots, a_n, b, c$ that sums to $2A$.

It is easy to prove that the partition exists if and only if there exists $I \subseteq \{1, \ldots, n\}$ such that $\sum_i a_i = k$.

[1]The reduction goes in the non-trivial direction!