36.5-2 – Given an integer $m \times n$ matrix A, and in integer m-vector b, the 0-1 integer programming problem asks whether there is an integer n-vector x with elements in the set $(0, 1)$ such that $A\times \leq b$. Prove that 0-1 integer programming is NP-hard (hint: reduce from 3-SAT).

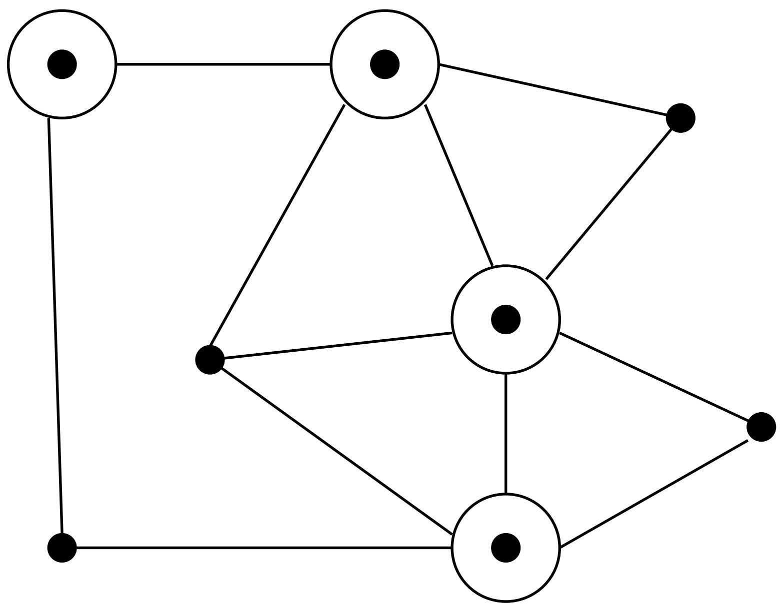
This is really the exact same problem as the previous integer programming problem, slightly concealed by:

- The linear algebra notation – each row is one constraint.
- All inequalities are $\leq$ – multiply both sides by -1 to reverse the constraint from $\geq$ to $\leq$ if necessary.

Vertex Cover

Instance: A graph $G = (V, E)$, and integer $k \leq V$

Question: Is there a subset of at most k vertices such that every $e \in E$ has at least one vertex in the subset?



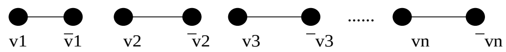
Here, four of the eight vertices are enough to cover. It is trivial to find *a* vertex cover of a graph – just take all the vertices. The tricky part is to cover with as small a set as possible.

Theorem: Vertex cover is NP-complete.

Proof: VC in in NP – guess a subset of vertices, count them, and show that each edge is covered.

To prove completeness, we show 3-SAT and VC. From a 3-SAT instance with n variables and C clauses, we construct a graph with $2N + 3C$ vertices.

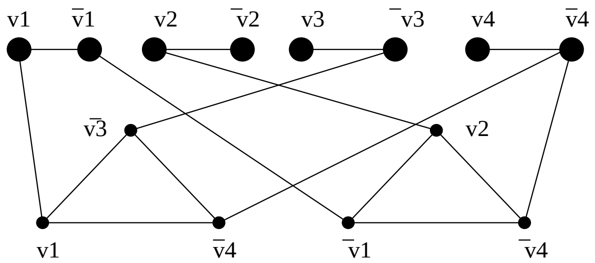
For each variable, we create two vertices connected by an edge:



To cover each of these edges, at least n vertices must be in the cover, one for each pair. For each clause, we create three new vertices, one for each literal in each clause. Connect these in a triangle.

At least two vertices per triangle must be in the cover to take care of edges in the triangle, for a total of at least $2C$ vertices.

Finally, we will connect each literal in the flat structure to the corresponding vertices in the triangles which share the same literal.



Claim: This graph will have a vertex cover of size $N+2C$ if and only if the expression is satisfiable.

By the earlier analysis, any cover must have at least $N+2C$ vertices. To show that our reduction is correct, we must show that:

1. *Every satisfying truth assignment gives a cover.*

   Select the N vertices cooresponding to the TRUE literals to be in the cover. Since it is a satisfying truth assignment, at least one of the three cross edges associated with each clause must already be covered - pick the other two vertices to complete the cover.

2. *Every vertex cover gives a satisfying truth assignment.*

   Every vertex cover must contain n first stage vertices and $2C$ second stage vertices. Let the first stage vertices define the truth assignment.

   To give the cover, at least one cross-edge must be covered, so the truth assignment satisfies.

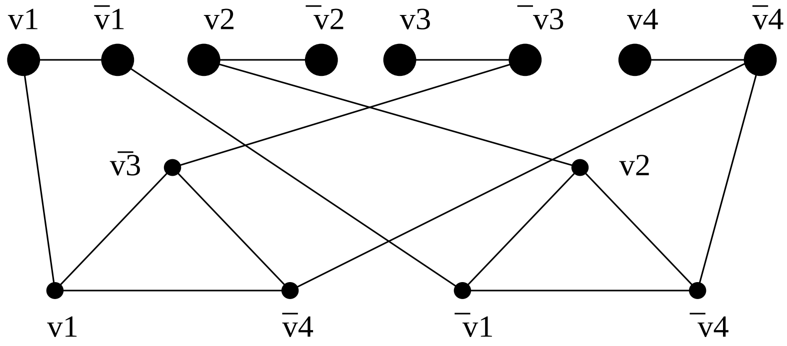
For a cover to have $N+2C$ vertices, all the cross edges must be incident on a selected vertex.

Let the N selected vertices from the first stage coorespond to TRUE literals. If there is a satisfying truth assignment, that means at least one of the three cross edges from each triangle is incident on a TRUE vertex.

By adding the other two vertices to the cover, we cover all edges associated with the clause.

Every SAT defines a cover and Every Cover Truth values for the SAT!

Example: $V_1 = V_2 = True$, $V_3 = V_4 = False$.



Starting from the Right Problem

As you can see, the reductions can be very clever and very complicated. While theoretically any NP-complete problem can be reduced to any other one, choosing the correct one makes finding a reduction much easier.

$$3 - Sat \propto VC$$

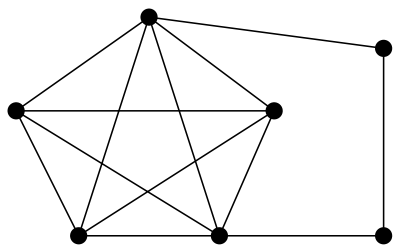
As you can see, the reductions can be very clever and complicated. While theoretically any NP-complete problem will do, choosing the correct one can make it much easier.

Maximum Clique

Instance: A graph $G = (V, E)$ and integer $j \leq v$.

Question: Does the graph contain a clique of j vertices, ie. is there a subset of v of size j such that every pair of vertices in the subset defines an edge of G?

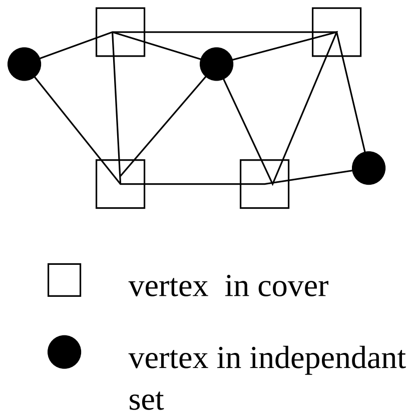
Example: this graph contains a clique of size 5.



When talking about graph problems, it is most natural to work from a graph problem - the only *NP*-complete one we have is vertex cover!

Theorem: Clique is *NP*-complete

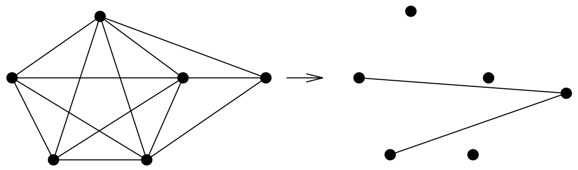
Proof: If you take a graph and find its vertex cover, the remaining vertices form an indepentent set, meaning there are no edges between any two vertices in the independent set, for if there were such an edge the rest of the vertices could not be a vertex cover.



Clearly the smallest vertex cover gives the biggest independent set, and so the problems are equivallent – Delete the subset of vertices in one from the total set of vertices to get the order!

Thus finding the maximum independent set must be NP-complete!

In an independent set, there are no edges between two vertices. In a clique, there are always between two vertices. Thus if we complement a graph (have an edge iff there was no edge in the original graph), a clique becomes an independent set and an independent set becomes a Clique!



Max Clique = 5
Max IS = 2

Max Clique = 2
Max IS = 5

Thus finding the largest clique is NP-complete:

If VC is a vertex cover in G, then $V - VC$ is a clique in G'. If C is a clique in G, $V - C$ is a vertex cover in G'.

36.5-1 Prove that subgraph isomorphism is NP-complete.

---

1. Guessing a subgraph of G and proving it is isomorphism to h takes $O(n^2)$ time, so it is in NP.

2. Clique and subgraph isomorphism. We must transform all instances of clique into some instances of subgraph isomorphism. Clique is a special case of subgraph isomorphism!

   Thus the following reduction suffices. Let $G = G'$ and $H = K_k$, the complete subgraph on k nodes.

Integer Partition (Subset Sum)

Instance: A set of integers S and a target integer t.

Problem: Is there a subset of S which adds up exactly to t?

Example: $S = \{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}$ and $T = 3754$

Answer: $1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = T$

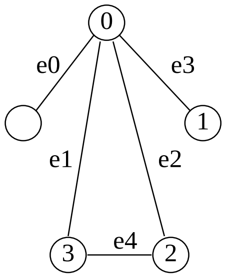
Observe that integer partition is a number problem, as opposed to the graph and logic problems we have seen to date.

Theorem: Integer Partition is *NP*-complete.

Proof: First, we note that integer partition is in *NP*. Guess a subset of the input number and simply add them up.

To prove completeness, we show that vertex cover $\propto$ integer partition. We use a data structure called an incidence matrix to represent the graph G.

$$\begin{array}{c|ccccc|} & e4 & e3 & e2 & e1 & e0 \\ \hline v0 & 0 & 1 & 1 & 1 & 1 \\ v1 & 0 & 1 & 0 & 0 & 0 \\ v2 & 1 & 0 & 1 & 0 & 0 \\ v3 & 1 & 0 & 0 & 1 & 0 \\ v4 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$



How many $1'$s are there in each column? Exactly two.

How many $1'$s in a particular row? Depends on the vertex degree.

The reduction from vertex cover will create $n+m$ numbers from G.

The numbers from the vertices will be a base-4 realization of rows from the incidence matrix, plus a high order digit:

$x_i = 4^{|E|} + \sum_{j=0}^{|E|-1} b[i,j] \times 4^j$

ie. $V_2 = 10100$ becomes $4^5 + (4^4 + 4^2)$.

The numbers from the edges will be $y_i = 4^j$.

The target integer will be

$$t = k \times 4^{|E|} + \sum_{j=0}^{|E|-1} 2 \times 4^j$$

Why? Each column (digit) represents an edge. We want a subset of vertices which covers each edge. We can only use k x vertex/numbers, because of the high order digit of the target.

$x_0 = 100101 = 1041$ $x_2 = 111000 = 1344$ $y_1 = 000010 = 4$

We might get only one instance of each edge in a cover - but we are free to take extra edge/numbers to grab an extra 1 per column.

VC in $G \rightarrow$ Integer Partition in S

Given k vertices covering G, pick the k cooresponding vertex/numbers. Each edge in G is incident on one or two cover vertices. If it is one, includes the cooresponding edge/number to give two per column.

Integer Partition in $S \rightarrow VC$ in G

Any solution to S must contain *exactly* k vertex/numbers. Why? It cannot be more because the target in that digit is k and it cannot be less because, with at most 3 $1'$s per edge/digit-column, no sum of these can carry over into the next column. (This is why base-4 number were chosen).

This subset of k vertex/numbers must contain at least one edge-list per column, since if not there is no way to account for the two in each column of the target integer, given that we can pick up at most one edge-list using the edge number. (Again, the prevention of carrys across digits prevents any other possibilites).

Neat, sweet, and *NP*-complete!

Notice that this reduction could not be performed in polynomial time if the number were written in unary $5 = 11111$. Big numbers is what makes integer partition hard!