1. Subset-Sum Problem (SSP): Given n positive integers $a_1, a_2, \cdots, a_n$, and a target sum t (again a positive integer), decide whether there exists a subcollection of the given integers, whose sum is t.

   (a) Prove that SSP is NP-Complete.

A reduction from the VERTEX-COVER problem can prove the NP-Hardness of SUBSET-SUM. The construction is illustrated in Figure 136. Let $G = (V, E)$ be an undirected graph with $m = |E|$ edges. We want to decide whether G has a vertex cover of size k. For the sake of understanding, we represent to resulting integers in the subset-sum problem in base four. We number the edges by $0, 1, 2, \ldots, m-1$. In particular, we assume that $E = \{0, 1, 2, \ldots, m-1\}$.

For $u \in V$, let $I_u \subseteq E$ denote the set of edges incident on u (that is, the set of edges, of which u is an endpoint). For each $u \in V$, we generate the integer

$$a_u = 4^m + \sum_{i \in I_u} 4^i.$$

Moreover, for each $i \in E$, we generate the integer

$$b_i = 4^i.$$

The set in the converted instance of SUBSET-SUM is

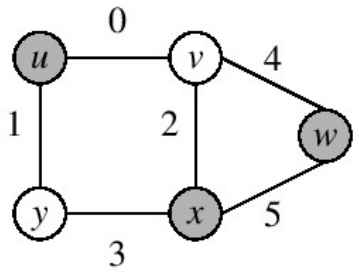
$$S = \{a_u \mid u \in V\} \cup \{b_i \mid i \in E\}.$$

Finally, the target sum is

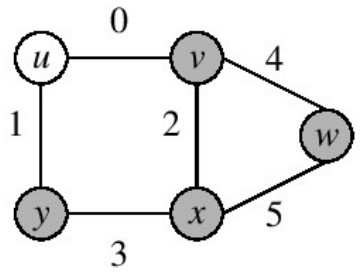
$$t = k \times 4^m + \sum_{i=0}^{m-1} 2 \times 4^i.$$

This completes the construction. In order to see that this construction works, consider two cases.

Figure 136: Reduction from VERTEX-COVER to SUBSET-SUM



	4^6	4^5	4^4	4^3	4^2	4^1	4^0
a_u	1	0	0	0	0	1	1
a_v	1	0	1	0	1	0	1
a_w	1	1	1	0	0	0	0
a_x	1	1	0	1	1	0	0
a_y	1	0	0	1	0	1	0
b_0	0	0	0	0	0	0	1
b_1	0	0	0	0	0	1	0
b_2	0	0	0	0	1	0	0
b_3	0	0	0	1	0	0	0
b_4	0	0	1	0	0	0	0
b_5	0	1	0	0	0	0	0
Target	3	2	2	2	2	2	2



	4^6	4^5	4^4	4^3	4^2	4^1	4^0
a_u	1	0	0	0	0	1	1
a_v	1	0	1	0	1	0	1
a_w	1	1	1	0	0	0	0
a_x	1	1	0	1	1	0	0
a_y	1	0	0	1	0	1	0
b_0	0	0	0	0	0	0	1
b_1	0	0	0	0	0	1	0
b_2	0	0	0	0	1	0	0
b_3	0	0	0	1	0	0	0
b_4	0	0	1	0	0	0	0
b_5	0	1	0	0	0	0	0
Target	4	2	2	2	2	2	2

Case 1: G contains a vertex cover $C = \{u_1, u_2, \ldots, u_k\}$ of size k.

Choose the integers $a_{u_1}, a_{u_2}, \ldots, a_{u_k} \in S$, and the integers b_i if and only of the i-th edge has exactly one endpoint in C. Since C is a vertex cover of G, every edge is covered by one or two vertices of C. We leave out those b_i in case both the endpoints of the i-th edge are present in C. It is straightforward to verify that the chosen integers add up to the target sum t.

Case 2: G does not contain a vertex cover of size k.

The above costruction ensure that at all positions 4^i, $i = 0, 1, 2, \ldots, m-1$, there are exactly three one digits (the rest are all zero). Therefore adding the integers in any subset of S never results in a carry in the m less significant digit positions. At the most significant digit, there may be a carry (or overflow), but we keep the total sum there without forwarding the excess amount to higher positions.

In order to achieve the target t, it therefore follows that we have to choose exactly k integers of the form a_u. Let U denote a choice of k vertices. Since G does not have a vertex cover of size k, the subset U in particular fails to cover all the edges of G. Let $i = (v, w)$ be an uncovered edge. This means that $v, w \notin U$, that is, a_v and a_w are not chosen. That is, in the column for 4^i, two one-digits are not chosen. Even if we choose b_i, the sum of the chosen integers will have one in the i-th digit position, since there is no carry from lower positions. We can thus conclude that the target sum t cannot be achieved.

(b) What if we allow the integers to be positive, negative, and zero?

SSP as we defined earlier is a special case of this generalized version of SSP. In fact, the reduction SSP $\leqslant$ GenSSP taking $I \longmapsto I$ works.

2. MAX-CUT problem: Let G = (V,E) be an undirected graph. We want to find a cut X,Y of V such that the number of edges connecting X and Y is as large as possible.

(a) Propose a decision version of the MAX-CUT problem. Prove that the maximization problem can be solved in polynomial time if and only if the decision problem can be solved in polynomial time.

Decision version: Given G and a positive integer k, decide whether G has a cut of size $\geqslant$ k.

If the maximization problem can be solved in polynomial time, the decision version can evidently be solved in poynomial time.

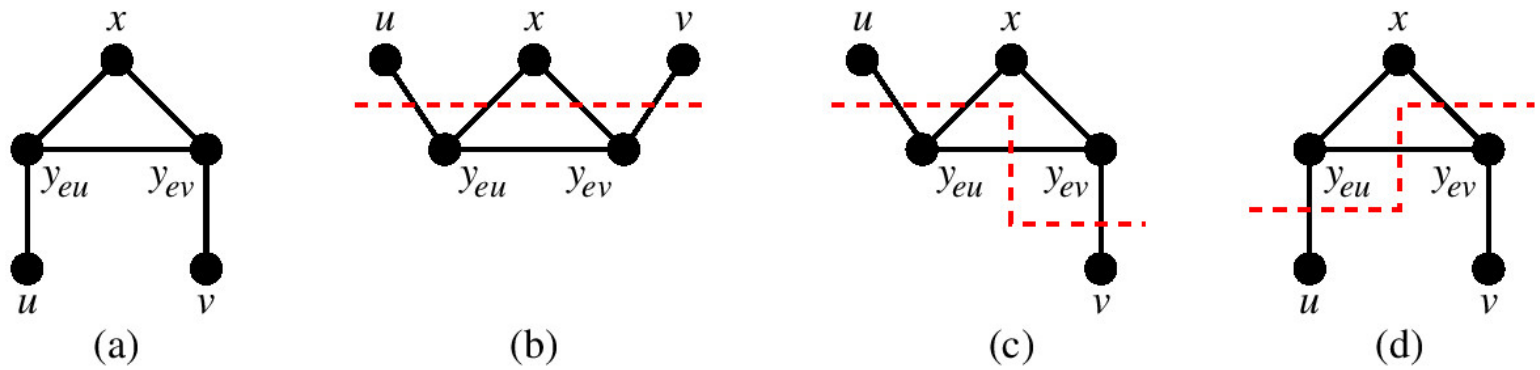
Conversely, suppose that A is a polynomial-time algorithm for solving the decision version. We first compute the number m of edges in G. Then, we repeatedly call A with k = m – 1, m – 2, m – 3, $\cdots$ until A says Yes. The running time of this optimization algorithm is m times the running time of A. So if A runs in polynomial time, so too does this optimization algorithm. You may use binary search to limit the increase of the running to a factor of log m only, but this is not a critical issue for the proof.

Note: Equivalence in the current context means this "if and only if" poly-time behavior.

(b) Prove that the decisional MAX-CUT problem is NP-complete.

Clearly, MAX-CUT $\in$ NP. To prove its NP-Hardness, we use reduction from INDEPENDENT-SET. Let $G = (V, E)$ and r constitute an instance of the INDEPENDENT-SET problem. We construct a graph $G' = (V', E')$ for MAX-CUT as follows. We include every vertex of V in V'. We add a new vertex x to V', and add x to all vertices in the copy of V in V'. For each edge $e = (u, v) \in E$, we introduce two new vertices y_{eu} and y_{ev} in V'. These new vertices and the vertices $x, u, v \in V'$ are connected as shown in Figure 137(a). Finally, take the cut size for G' as $k = r + 4|E|$.

Figure 137: An edge gadget for the reduction of INDEPENDENT-SET to MAXCUT



The property of the edge gagdet, that makes this construction work, is as follows. Consider a cut of G', and let $e = (u,v)$ be an edge of G. If one or both of u, v lie in the same part as x, then the gadget vertices y_{eu} and y_{ev} can be placed in appropriate parts of the cut such that the gadget for e contributes four edges to the cut (see Parts (b) and (c) of Figure 137). On the other hand, if both u and v lie on the other part of the cut as x, then no matter in which parts we put y_{eu} and y_{ev}, the edge gadget contributes at most three edges to the cut (Figure 137(d) shows one situation).

Now, suppose that $I \subseteq V$ is an idependent set of G. We produce a cut V_1', V_2' of G' of size exactly equal to $k = r + 4|E|$ as follows. We put x and all vertices of $V \setminus I$ in V_1'. We put all vertices of I in V_2'. Let $e = (u,v)$ be an edge of G. Since I is an independent set, both u and v do not belong to I and so not to V_2' as well. This implies that at least one of u and v, or possibly both, must be in the same part V_1' as x. Consequently, we can place y_{eu} and y_{ev} appropriately in the two parts so that the gadget corresponding to e contributes four edges to the cut. Finally, there are exactly r edges of the form (x,u), $u \in I$, in the cut. Thus, the cut is of the desired size.

Conversely, let V_1', V_2' be a cut of G' of size $k \geqslant r + 4|E|$. Suppose that $x \in V_1'$. Since an edge gadget can contribute at most four edges to a cut, there must exist $r' \geqslant r$ vertices $u \in V$ such that $u \in V_2'$ (consider the edges (x,u) in G'). If these r' vertices of V, that are in V_2', are already independent, we are done. So suppose that some edge $e = (u,v)$ between two of these r' vertices exists in E. The corresponding edge gadget contributes less than four edges to the cut. We can move one of the vertices and switch the side(s) of one or both of the gadget vertices y_{eu} and y_{ev} such that the cut size does not decrease. For example, in the situation of Figure 137(d), we simply transfer v from V_2' to V_1', so the edge gadget now contributes one more cut edge which compensates for the loss of the earlier cut edge (x,v). This process reduces r' by one. We continue doing these vertex switches in the cut until the vertices of V, that remain in V_2', become independent. Let ρ be this final value of r'. By the gadget property, we cannot have $\rho < r$.

3. MAX-3-CUT problem: Let G = (V,E) be an undirected graph, and k a positive integer. Decide whether V can be partitioned into three parts X, Y, Z such that the number of edges in E connecting vertices from different parts is $\geqslant$ k. Prove that MAX-3-CUT is NP-complete.

Clearly, MAX-3-CUT is in NP. To prove its NP-hard-ness, we use reduction from MAX-CUT. Let (G,r) be an instance of MAX-CUT. From this, we create an instance (G',k) for MAX-3-CUT as follows. First, make a copy of G to G' (both vertices and edges). Let $|V(G)| = n$. Add n new vertices $w_1, w_2, \cdots, w_n$ to G', and add an edge from each w_i to each old vertex v_j in the copy of G in G'. Finally, take $k = r + n^2$.

If G has a cut X,Y of size $\geqslant$ r, then consider the 3-cut X,Y,Z of G', where $Z = \{w_1, w_2, \cdots, w_n\}$.

Conversely, let X,Y,Z be a 3-cut of G' of size $\geqslant$ $k = r + n^2$. Let Z be the part containing the minimum number of old vertices v_j. But then, if some new vertex w_i is not in Z, then relocating w_i from X or Y to Z does not decrease the 3-cut size (w_i is connected to all old vertices). So we can assume that all w_i are in Z. Now, if Z contains one or more of the old vertices v_j, then relocating them from Z to X or Y (in any arbitrary manner) cannot again decrease the 3-cut size. So we have a 3-cut X,Y,Z of G' of size $\geqslant$ $r + n^2$ with $Z = \{w_1, w_2, \cdots, w_n\}$. But then, X,Y corresponds to a 2-cut of G of size $\geqslant$ r.

Remark: You can also make a reduction 3-COLOR $\leqslant$ MAX-3-CUT as $G \mapsto (G, |E(G)|)$.
G is 3-colorable if and only if it is tripartite.