Problem1 (NP-complete)

(1).Organize Problem

S (Total set of problem) = {1, 2, 3,…, i,…, n}

Where, $1 \leq i \leq n$

Score set $= \{s_1, s_1, \dots, s_i, \dots, s_n\}$

Score set can be partitioned into two sets A and $\overline{A}$ ($\overline{A} = S - A$), for example

$$\sum_{i \in A} s_i = \sum_{j \in \overline{A}} s_j$$

Goal: Show that SET-PARTITION is NP-Complete

The problem says I need separate the set of ice cream by two baskets named B_1 *and* B_2, which must have same score sum. Thus, I think it is two-partition problem, and I use A and $\overline{A}$ ($not\ A$) instead of B_1 *and* B_2. Now, I prove this problem is NP-Complete.

(2).Procedure

Step1. There is non-deterministic polynomial-time algorithm that solves A, i.e., $A \in NP$

Step2. Any NP-Complete Problem B can be reduced to A

Step3. The reduction of B to A works in Polynomial time

Step4. The original problem A has a solution if and only if B has a solution.

(3).Step1

SET-PARTITION $\in$ NP: Guess the two partitions and verify that the two have equal sums

(a).Prove the algorithm can be solved in polynomial time

This problem can be solved by dynamic programming, if the size of the set and the size of the sum of the integers in the set are not too big to render the storage requirements infeasible.

I use the set which I mentioned above, S = $\{x_1, x_2, x_3, \dots, x_i, \dots, x_n\}$. And K is the sum of all elements' score in S. that is $K = s_1 + \cdots + s_n$. I will build an algorithm that determines whether there is a subset of S that sums to $\left\lfloor \frac{K}{2} \right\rfloor$

Recurrence relation: I determine if there is a subset of S that sums to $\left\lfloor \frac{K}{2} \right\rfloor$. Let p(I,j) be True if a subset of $\{x_1, x_2, x_3, \dots, x_j\}$. sums to i and False otherwise. Then p($\left\lfloor \frac{K}{2} \right\rfloor$,N) is True if and only if there is a subset of S that sums to $\left\lfloor \frac{K}{2} \right\rfloor$. The goal of the algorithm will be to compute p($\left\lfloor \frac{K}{2} \right\rfloor$,N). Then I have recurrence relation:

$$p(i,j) = \begin{cases} True, & if\ either\ p(i, j-1) is\ True\ or\ p(i - x_j, j-1) \\ False, & otherwise \end{cases}$$

Then, the algorithm's running time is:

$$0\left(\frac{K}{2}\times n\right) = O(k \times n)$$

Thus, the running time is polynomial and this algorithm is NP.

(b).SUBSET-SUM:

Given a sequence $a_1, \dots, a_n, t$ of nonnegative integers, decide whether there is a subset $A \in \{1, \dots, n\}$ such that

$$\sum_{i\in A} a_i = t$$

(c).SET-PARTITION:

Given a sequence $a_1, \dots, a_n$ of nonnegative integers, decide whether there is a subset $A \in \{1, \dots, n\}$ such that

$$\sum_{i\in A} a_i = \sum_{j\in \bar{A}} a_j$$

(4).Step2

Reduction of SUBSET-SUM to SET-PARTITION: SUBSET-SUM is defined as follow: Given a set $X = \{a_1, \dots, a_n\}$ and a target number t of nonnegative integers, find a subset $Y \in X$ such that:

$$\sum_{i=1}^{n} a_i = k\ (sum\ of\ all\ elements\ in\ X)$$

$$\sum_{i\in Y} a_i = t\ (if\ i\ is\ belong\ to\ Y)$$

That is, the members of Y add up to exactly t, let k be the sum of member of X.

Feed $X' = X \cup \{k - 2t\} = 2k - 2t$ into SET-PARTITION. Accept if and only if SET-PARTITION accepts.

(5).Step3

This reduction clearly works in polynomial time.

(6).Step4

I will prove that $< X, t > \in$ SUBSET-SUM, iff $< X' > \in$ SET-PARTITION

Where sum(X') = 2k – 2t.

- If there exists a set of numbers in X that sum to t, then the remaining numbers in X sum to k- t. Therefore, there exists a partition of X' into two such that each partition sums to k- t.

- Let's say that there exists a partition of X' into two sets such that the sum over each set is k – t. One of these sets contains the number k-2t. Removing this number, I get a set of numbers whose sum is t, and all of these numbers are in X

Problem2 (NP-complete)

(1).Organize Problem

Given an undirected graph G = (V, E), where nodes represent towns and edges represent roads, and given a number k, is there a way to build k bunkers in k different towns so that every town either has its own bunker, or is connected by a direct road to a town that does have a bunker?

(2).Shows PYGMALION is in NP

A candidate solution is a subset of the vertices, $C \in V$. I must check the number of vertices $|C|$ doesn't exceed k, to verify that C is indeed a solution. Then, for each vertex in the graph, $v \in V$, I need to check if it is either in the given set $v \in C$, such as $O(|V|)$, or one of its edges has an endpoint in the set, which can be done in $O(|V| + |E|)$ by hashing the vertex set and then looping over the set of edges. This work will take most $O(|V| + |E|)$, which is the worst case. That is, the solution can be verified in poly-time. Thus, PYGMALION is in NP.

(3).Shows PYGMALION is in NP-Complete

I will use Vertex Cover

Step1. Show that Vertex-Cover(G, k) can be reduced to PYMALION(G', k'), and that the reduction can be done in polynomial time:

A vertex cover is set of vertices such that for every edge in the graph, at least one of edge's end points is in the vertex cover. In Vertex-Cover(G, k), I have a graph G = (V, E) and a target number k, and I need to find whether there exists is vertex cover of size at most k. Based on that, I can build the input for PYMALION(G', k'); here, k' = k. Now I will make new graph G' = (V', E'). First, G' has all the edges and vertices of G. In addition, for every edge {u, v}∈ E, I will create a new vertex w and new edges {u, w} and {w, v} and place them in G'.

Initial graph G must be copied for the reduction; $O(|V| + |E|)$. And for every edge, I need to add a new node ($O(|E|)$) and create two new edges ($O(|E|)$). Therefore, this takes poly-time: $O(|V| + |E|)$.

Step2. Vertex-Cover(G, k) has a solution, iff the corresponding PYGMALION(G', k') has a solution:

- **Show Vertex-Cover(G, k) to PYGMALION(G', k'):**
  Let $S \in V$ be a vertex cover in G. I will show S is a valid solution for G'. First notice that if S is a vertex cover, then this implies that for every edge in G, at least one of its endpoints is in S. Now consider some vertex x in G' such as $x \in V'$.

  (a). If x is an original vertex in G, then either $x \in S$ or there must exist some edge in G which connects x to some vertex $u \in S$. Thus, x either has a bunker ($x \in S$), or x is a connected to some vertex u which has a bunker ($u \in S$).

(b). If x is not in G, then it forms a triangle with two adjacent vertices u, v in G. The triangle is composed of three edges: (i, x), (x, v), and (u, v). Where, edge (u, v) is in G. Therefore, it must be covered by S which contain at least one of u or v. Vertex x is connected to by an edge to a vertex belonging to S. That is, x is connected to some vertex which has bunker.

Thus, if G has a vertex cover S, then S is also a solution to PYGMALION(G', k'), k =k' = $|S|$.

- **Show PYGMALION(G', k') to Vertex-Cover(G, k):**
  Let D be a solution for G' ($|D| = k'$). Consider first the vertices $w \in D$ which don't correspond to an original vertex in G. And w must be connected to two vertices u, v in G. Thus, I can replace w by either u or v, while maintaining the size and validity of the solution. This is because u, v, and w form a triangle, and w connects only to u and v. Do it for all vertices $w \in D$ that do not belong to the graph G, I can get a solution S ($|S| = |D| = k'$) which is a valid solution G', however that only consists of vertices in the original graph G. It makes sure that the edge (u, v) is covered in G by either u, v, because every edge (u, v) creates a vertex x that must be covered in G' by either u or v. Thus, all the additional vertices in G' are covered by the solution S. which means that all edges in G are covered by S. Therefore, if G' has a solution D, this can be converted, so a valid vertex cover for G of size at most k.

Step3. Thus, It is NP-Complete, because PYGMALION is in NP and has been shown to be at least as hard as Vertex-Cover.

Reference

http://ac.informatik.uni-freiburg.de/lak_teaching/ws11_12/combopt/notes/bin_packing.pdf

CSE6140 lecture slide 11 (NP-Completeness-3)