Project Euler Problem 260: Investigation and Solution

Aditya Bhimalingam

Problem 260. (Stone Game)
A game is played with three piles of stones and two players. At her turn, a player removes one or more stones from the piles. However, if she takes stones from more than one pile, she must remove the same number of stones from each of the selected piles. In other words, the player chooses some $N > 0$ and removes:

- N stones from any single pile
- N stones from each of any two piles (total of $2N$ stones)
- N stones from each of the three piles (total of $3N$ stones).

The winner is the player who removes the last stone(s).

A *winning configuration* is one where the first player can force a win. For example, $(0, 0, 13)$, $(0, 11, 11)$ and $(5, 5, 5)$ are winning configurations because the first player can immediately remove all stones.

A *losing configuration* is one where the second player can force a win, no matter what the first player does. For example, $(0, 1, 2)$ and $(1, 3, 3)$ are losing configurations: any legal move leaves a winning configuration for the opponent.

Consider all losing configurations (x_i, y_i, z_i) where $x_i \le y_i \le z_i \le 100$. It can be verified that

$$\sum (x_i + y_i + z_i) = 173\,895.$$

Find

$$\sum (x_i + y_i + z_i),$$

where the sum is taken over all losing configurations with

$$x_i \le y_i \le z_i \le 1000.$$

Restatement

We have a two-player impartial game with three piles (x, y, z) subject to $x \le y \le z$. A move consists of choosing one or more piles and removing the same positive number of stones from each chosen pile. A position is losing if the player to move has no winning strategy from it. We must compute the sum of all such losing positions with pile sizes bounded by $N = 1000$.

Background and Concepts

This problem lies within the theory of impartial combinatorial games. Standard tools include the *Sprague–Grundy theorem* and classification into P-positions (previous player wins, i.e. losing positions) and N-positions (next player wins, i.e. winning positions). For three-pile subtraction games, the challenge is to identify the set of P-positions efficiently.

Key definitions:

- **P-position:** A game state from which the previous player (i.e. the one who just moved) has a winning strategy, equivalently a losing position for the next player.
- **N-position:** A state from which the next player has a winning strategy.

Key Insights

The naive approach would test every triple (x, y, z) and recursively determine outcomes. This is infeasible for $N = 1000$ due to the exponential blowup. The solution is to exploit dynamic programming and monotonicity:

1. We impose $x \leq y \leq z$ to avoid symmetry redundancies.
2. We classify positions incrementally, beginning with $(0, 0, 0)$ as a terminal losing state as no moves can be made.
3. For each candidate triple, we test whether any legal move reaches a previously identified P-position. If so, the current state is N if not, it is P.
4. The legality of moves depends only on differences between coordinates. Therefore the state classification can be stored compactly in three two-dimensional tables:
   - `One[a,b]` records losing pairs where two coordinates are fixed e.g removing from one pile.
   - `Two[a,b]` records losing pairs arising from two-pile differences e.g removing from two piles.
   - `All[a,b]` records losing pairs arising when subtracting from all three piles e.g removing from all 3 piles.
5. When a new P-position (x, y, z) is found, the relevant table entries are updated to mark reachable N-states efficiently.

This table-based propagation eliminates redundant recomputation and reduces complexity from exponential to essentially cubic in N with low constant factors.

Algorithm Description

We describe the procedure formally.

Input: Maximum pile size N.

Output: $S(N)$, the sum of all P-positions with $x, y, z \leq N$.

Algorithm:

1. Initialise boolean arrays One, Two, All of dimension $(N + 1) \times (N + 1)$, filled with `True` (meaning not yet marked losing).
2. For $x = 0$ to N, for $y = x$ to N, for $z = y$ to N:
   - If any of $One(x, y)$, $One(x, z)$, $One(y, z)$ are already losing, continue.
   - If any of $Two(y - x, z)$, $Two(z - y, x)$, $Two(z - x, y)$ are losing, continue.
   - If $All(y - x, z - x)$ is losing, continue.
   - Otherwise (x, y, z) is a P-position and add $x+y+z$ to the sum and update the tables accordingly.
3. Return the cumulative sum.

Correctness Justification

Lemma 1. *If (x, y, z) can reach a known P-position by a legal move, then (x, y, z) is an N-position.*

Proof. This is the standard impartial game rule: from any N-position the next player moves to a P-position, ensuring victory. Conversely, from a P-position all moves go to N-positions. Thus the test conditions above exactly characterise losing states. □

Theorem 1. *The algorithm correctly enumerates all P-positions with coordinates bounded by N.*

Proof. By induction on $x + y + z$: the base case $(0, 0, 0)$ is losing. Suppose all positions with smaller total stones are classified correctly. When evaluating (x, y, z), we check how to reach earlier P-positions via the tables. If such a move exists, (x, y, z) is winning otherwise, no move leads to a losing state and thus (x, y, z) itself is losing. Therefore all states are classified correctly. □

Complexity Analysis

There are $O(N^3)$ triples with $x \leq y \leq z$. Each classification requires $O(1)$ checks due to the precomputed tables. Thus the overall runtime is $O(N^3)$, feasible for $N = 1000$ in optimised Python or near-instant in compiled languages. Memory usage is $O(N^2)$.

Results

- For $N = 10$, the algorithm yields $S(10) = 360$.
- For $N = 100$, the algorithm yields $S(100) = 173895$.
- For $N = 1000$, the algorithm yields

$$S(1000) = 167542057.$$

References

- Project Euler, Problem 260. `https://projecteuler.net/problem=260`.
- Peachnote mirror for statement access. `https://peachnote.com/euler/problems/260/`.
- T. Ferguson, *Game Theory*, notes on impartial games and Sprague–Grundy theory.