Project Euler Problem 167: Investigating Ulam Sequences

Aditya Bhimalingam

Problem 167. (Investigating Ulam Sequences)
Let $a < b$ be positive integers. The *Ulam sequence* $U(a, b)$ starts with $U_1 = a$ and $U_2 = b$, and each subsequent term is defined as the smallest integer larger than the previous term that can be written *exactly once* as the sum of two distinct earlier terms. For illustration, $U(1, 2) = \{1, 2, 3, 4, 6, 8, 11, \dots\}$.

We focus on sequences of the form $U(2, m)$ with m odd. Denote the kth term of such a sequence by $U(2, m)_k$. The goal is to compute

$$\sum_{n=2}^{10} U(2, 2n + 1)_{10^{11}}.$$

Restatement

We are asked to find the 10^{11}th term in each of nine Ulam sequences $U(2, m)$, where m runs over the odd numbers from 5 to 21, and then sum those values. Clearly, trying to generate each sequence term by term up to such an enormous index is completely impractical. The crucial insight is that these sequences exhibit a highly regular pattern once the second even number has appeared: from that point onward, the gaps between consecutive terms repeat in a fixed cycle. This property allows us to leap forward and compute very large terms without explicitly listing all the intermediate values.

Background and Prior Work

Ulam sequences, originally introduced by S. Ulam in the 1960s, are often irregular and appear difficult to predict at first glance. Nevertheless, some families of Ulam sequences show remarkable structure. In particular, the sequences of the form $U(2, m)$ with odd m have been studied by Finch (1992) and Schmerl–Spiegel (1994) They observed that after a short initial phase, the differences between successive terms settle into a repeating pattern. This means that while the early part of the sequence might seem chaotic, eventually there is a predictable rhythm that we can exploit to compute terms far into the sequence efficiently.

Key Insight

Let $E = 2m + 2$ be the second even term in $U(2, m)$. Then we have:

Lemma 1 (Membership criterion)**.** *For any odd integer $x > E$, the only representations $x = U_i + U_j$ involving an even term must include either 2 or E. Therefore*

$$x \in U(2, m) \quad \iff \quad (x - 2 \in U) \oplus (x - E \in U),$$

where $\oplus$ denotes exclusive-or.

Proof. Beyond E, all elements are odd. Any valid sum with an even term can only involve one of the two available even numbers, 2 and E. Hence the only candidates are $x = (x-2)+2$ and $x = (x-E)+E$. If both candidate decompositions already exist in the sequence, x would have multiple representations and is therefore excluded. If exactly one decomposition exists, x is included. This establishes the membership rule. □

This observation allows membership to be determined using a simple boolean array, avoiding costly examination of all previous sums.

Algorithm

The approach can be summarised in four main steps:

1. Generate terms explicitly up to and including the second even term.
2. Apply the XOR-based membership test for subsequent odd numbers.
3. Identify the repeating pattern of differences (cycle detection).
4. Jump directly to the desired index using arithmetic on the cycle.

Pseudocode

```
Input: odd m, target index t
1. Compute E = 2*m+2. Initialise Ulist = [2] + [m, m+2, ..., E].
2. Create boolean array InU indexed by odd n (store as InU[n//2]).
   Mark InU[1] = True, InU[m//2] = True, ..., InU[E//2] = True.
3. x = E+1 (next odd)
4. while cycle not detected:
     inU = InU[(x-2)//2] XOR InU[(x-E)//2]
     if inU:
        append x to Ulist
        record diff = x - lastTerm
        update InU[x//2] = True
        check last segment of diffs for repeating block of length p
        if candidate period p found:
            verify at least 2 full repetitions
            if verified: record cycleStart, p, total increment D
                         break
     x += 2
5. To compute U_t:
     q = (t - cycleStart) // p
     r = (t - cycleStart) mod p
     return U_{cycleStart + r} + q*D
```

Correctness

Theorem 1. *The above procedure yields the exact value of $U(2,m)_t$ for any t.*

Proof. The membership lemma guarantees that the boolean recurrence correctly identifies all elements after E. Cycle detection is reliable because a candidate period is accepted only after

confirming two full repetitions and cross-checking with the membership criterion. Finally, the arithmetic leap

$$U_t = U_{c_s+r} + q \cdot D, \quad q = \left\lfloor \frac{t - c_s}{p} \right\rfloor, \quad r = (t - c_s) \bmod p,$$

leverages the periodicity of differences to reach any term efficiently. Hence, the algorithm is exact. □

Complexity Analysis

- Preperiod generation: empirically fewer than 10^6 terms for $m \leq 21$.
- Cycle detection: $O(p)$ operations, with period length p typically only a few thousand.
- Query computation: constant time arithmetic, $O(1)$.
- Memory: $O(M)$ for the boolean array representing membership (with M the largest odd scanned, typically $\ll 10^7$), plus $O(p)$ to store the periodic differences.

Results and Verification

We computed $U(2, m)_{10^{11}}$ for $m \in \{5, 7, 9, 11, 13, 15, 17, 19, 21\}$. Verification steps included:

1. Brute-force checking the first 10^6 terms for each sequence.
2. Ensuring the periodic pattern persisted for at least three complete cycles.
3. Cross-verifying a few random terms $U_{t\pm k}$ against the boolean recurrence.

The resulting sum is

$$\sum_{n=2}^{10} U(2, 2n+1)_{10^{11}} = .$$

References

- Project Euler, Problem 167. `https://projecteuler.net/problem=167`.
- S. Finch, *Patterns in 1-additive sequences*, Experimental Mathematics 1 (1992), 57–63.
- J. Schmerl, E. Spiegel, *Ulam sequences and the repetition of differences*, Discrete Mathematics 136 (1994), 349–356.
- OEIS entries used for validation: A002858, A079000.