# Finding Square-Sum Sequences modulo m

Naing Tun and Patrick Ward

ILLINOIS WESLEYAN UNIVERSITY

## The Square-Sum Problem

Given an integer sequence from $1, 2, 3, \ldots, n$, can you rearrange *all* of the integers in a way that each consecutive sum is a square? Squares are: $1, 4, 9, 16, 25, 36, \ldots$ and so on.

**Example 1.** *: Consider the number from* 1 *to* 15*. The answer is:*

$$8, 1, 15, 10, 6, 3, 13, 12, 4, 5, 11, 14, 2, 7, 9$$

How do we solve this problem generally for all $n$, if there exists a solution?

This problem is originally introduced in the Numberphile video entitled *The Square-Sum Problem*. It uses a graph theory approach to find the solution.

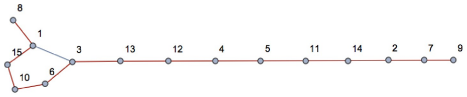



Figure 1: A graph depicting the a square-sum problem for $n = 15$ (Wolfram Alpha)

Symbolically, our rearranged sequence should satisfy the following equation:

$$x + y = a^2$$

where $x, y$ are two consecutive integers in the sequence and $a^2 = \{1, 4, 9, 16, 25, \ldots\}$.

## New Problem

Is it possible to arrange the first n integers in such a way that each consecutive sum is a square for a given modulus? The best way to understand the problem again is with graph theory.

**Definition 3.** *The graph $G_m(n)$ is a graph with vertices $[1, n]$. The vertices are connected with edges $(x, y)$ such that*

$$x + y \equiv a^2 \ (mod\ m),$$

*Where a is an integer between* 1 *and* $m$

The goal of this problem is to find the conditions for $m$ and $n$ for which the graph $G_m(n)$ contains a cycle which goes through every vertex exactly once. This type of cycle is called a Hamiltonian cycle.

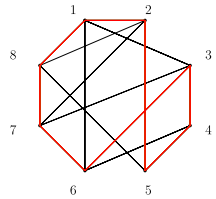



Figure 2: The $G_6(8)$ graph with a Hamiltonian cycle highlighted in red.

## Previous Work

- Isaac Dragomir inspired the project and has found cube sequences and 4th power sequences.
- In his book, *Things to Make and Do in the Fourth Dimension*, Matt Parker conjectures that a square sum sequence exists for all integers greater than 25.
- Charlie Turner computationally extended the search to 299.
- Bernardo Recamán presented computer evidence that for any k, a $k^{th}$ power sequence exists for large enough $n$.

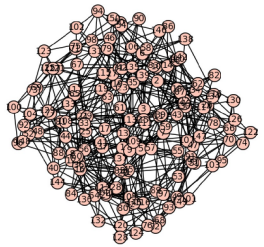


Figure 3: The square-sum problem for 150 vertices (Charlie Turner)

## When Does a Cycle Exist?

Let $G_m(n)$ be a graph with vertices $[1, n]$. The following table gives the conditions for $n$ such that $G_m(n)$ contains a Hamiltonian cycle.

| Modulus (m) | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| n such that $G_m(n)$ contains a Hamiltonian cycle | $n \in 0 \pmod 2$ | $\emptyset$ | $\emptyset$ | $n \in 0, 1 \pmod{10}$ | $n > 4$ | $n > 9$ | $n \in 0 \pmod{16}$ | $\emptyset$ | $n > 3$ |

*Proof.* (mod 2) An edge $(i, j)$ exists iff $i + j \equiv 1 \pmod 2$. From this, for every edge $(i, j)$, $i$ and $j$ are of opposite parity. If a Hamiltonian exists, then each odd vertex connects to two even vertices and each even vertex connects to two odd vertices. Therefore, the number of odd vertices equals the number of even vertices. This is true when the number of vertices is even.

When the number of vertices $n$ is even, the edges $\{(1, 2), (2, 3), (3, 4)...(n - 1, n), (n, 1)\} \in E(G_2(n))$ form a Hamiltonian cycle. □

## Bounds for the Degree of Vertices

**Definition 1.** *The* degree *$(D(v))$ of a vertex is the number of edges that connect to that vertex.*

**Example 2.** *The degree of vertex 1 in $G_6(8)$ (Figure 2) is 4.*

**Definition 2.** s(m) *gives the number integers that are perfect squares modulo m. Note that we do not count zero as a perfect square.*

**Example 3.** $s(6) = 3$ *because* $1^2 \equiv 1 \pmod 6$, $2^2 \equiv 4 \pmod 6$, $3^2 \equiv 3 \pmod 6$, $4^2 \equiv 2 \pmod 6$, $5 \equiv 1 \pmod 6$. *Notice that* $1, 4, 3$ *are the only unique squares modulo 6, so* $s(6) = 3$.

**Theorem 1.** *Let $v$ be a vertex in $G_m(n)$ with $n = mx + a$, $0 \leq a < m$. Then*

$$x \cdot s(m) - 1 \leq D(v) \leq (x + 1)s(m).$$

This tells us that the degree of a given vertex grows at the same rate as the number of vertices in a graph.

## Finding Hamiltonian Cycles Computationally.

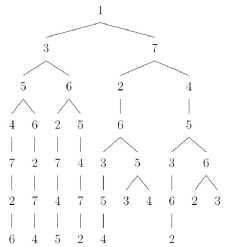
Brute-force backtracking algorithm takes $O(n \cdot n!)$ worst-case runtime, but we can optimize it into a faster $O(n!)$ runtime by not creating a branch for non-existent edge in the first place (brute-force creates branches for all vertices and checks linearly whether there is an edge for each branch).

```
bool hamiltonian_cycle(int start_node, int** ptr_adjMat, int num_nodes,
        bool visited[], int edge_count) {

    if (edge_count == num_nodes-1 && ptr_adjMat[start_node][0] == 1) {
        return true;
    }

    visited[start_node] = true;

    for (int col = 0; col < num_nodes; col++) {

        if (visited[col] == 0 && ptr_adjMat[start_node][col] == 1) {

            visited[col] = true;

            if (hamiltonian_cycle(col, ptr_adjMat, num_nodes, visited, edge_count+1)) {
                return true;
            }
            visited[col] = false;

        }
    }

    return false;
}
```

Optimized brute force Hamiltonian cycle finder

```
Enter num_nodes and mod value, each separated by a space: 7 7

residue for square mod 7: 1 4 2

Adjacency matrix:
0 0 1 0 0 0 1
0 0 0 0 0 1 1
1 0 0 0 1 1 0
0 0 0 0 1 0 1
0 0 1 1 0 1 0
0 1 1 0 1 0 0
1 1 0 1 0 0 0

1 conects to 3 7 (Degree: 2)
2 conects to 6 7 (Degree: 2)
3 conects to 1 5 6 (Degree: 3)
4 conects to 5 7 (Degree: 2)
5 conects to 3 4 6 (Degree: 3)
6 conects to 2 3 5 (Degree: 3)
7 conects to 1 2 4 (Degree: 3)

num of edges: 9

Hamiltonian cycle present: false
```

Information on $G_7(7)$




Tree diagram

Figure 4: Hamiltonian cycle finder for $G_7(7)$

If the graph has a Hamiltonian cycle, the algorithm will execute faster; but if it does not, the algorithm will be slower because it has to exhaust through all the possible paths.There is still a more-efficient algorithm for Hamiltonian cycle finder with $O(n^2 2^n)$ worst-case runtime, using dynamic programming.

Some interesting results for the case of odd primes $n \geq 11$ and $mod = n$:

1. There always exists a Hamiltonian cycle. The algorithm is tested for primes from 11 to 1000.
2. Brute force is already fast enough, even for large primes. It is tested for primes as large as 9973.
3. The algorithm performs a normal DFS search down the tree and backtracks when necessary. The backtracking process is an expensive part of the algorithm. However, in case of odd primes, backtracking will not occur for $\lceil n/2 \rceil$ level guaranteed.
4. For smaller primes, backtracking occurs more "immediately" compared to larger primes. For instance, if $n = 11$, backtracking occurs at $\lceil n/2 \rceil + 1$ level, but for larger primes such as $n = 67$, backtracking only occurs at $\lceil n/2 \rceil + 29$. Thus, $\lceil n/2 \rceil$ is more loose of a lower-bound as $n$ gets larger.