The Mathematician Educator
2023, Vol 4, No. 3, 201-218

Putnam Grasshopper Problem: Collaborative Problem Solving, Generalisations, and Computational Thinking

HO Weng Kin, TAY Eng Guan, TOH Pee Choon & TONG Cherng Luen
National Institute of Education
Nanyang Technological University, Singapore

This paper considers several generalisations of an interesting problem posed in the 2021 William Lowell Putnam Mathematical Competition. We describe how the authors embarked on a collaborative problem-solving journey which resulted in solving two of these generalisations and how computational thinking guided our approaches.

Keywords: collaborative problem-solving, generalisations, computational thinking

Introduction

The William Lowell Putnam Mathematical Competition is an annual mathematical competition for undergraduates enrolled at institutions of higher education across the United States and Canada. The first of the twelve problems of the 82nd edition (Ullman & Zeitz, 2022) held in December 2021, was the following:

> **A1**. A grasshopper starts at the origin in the coordinate plane and makes a sequence of hops. Each hop has length 5, and after each hop the grasshopper is at a point whose coordinates are both integers; thus, there are 12 possible locations for the grasshopper after the first hop. What is the smallest number of hops needed for the grasshopper to reach the point $(2021, 2021)$? (p. 705)

Two of the authors were introduced to the Putnam grasshopper problem[1] via a research project submitted by Pranjal Dasghosh, a secondary school student, for the Singapore Mathematics Project Festival 2023 (Dasghosh, 2023). Dasghosh considered the general problem of finding the minimum number of hops required to reach an arbitrary point. To this end, he wrote Python code and computed the solutions for a large number of points using exhaustive array search. In particular, Dasghosh's program showed that the point (1,0) (respectively, (0,1)) can be reached by a sequence of 3 hops, e.g., [(3, -4), (-5, 0), (3, 4)] (respectively, [(-4, 3), (0, -5), (4, 3)]). Consequently, he deduced that any point (x, y), where x, y are integers, is reachable by a finite sequence of hops:

$$(x, y) = x(1,0) + y(0,1) = x(3, -4) + x(-5,0) + x(3,4) + y(-4,3) + y(0, -5) + y(4,3).$$

[1] Another grasshopper problem – different from the Putnam one – appeared in the 2009 International Mathematical Olympiad (Kós, 2011)

Moreover, a primitive upper bound for the minimal number of hops is given by $3(x+y)$. From the data Dasghosh collected, he observed that trajectories with minimum hops often involve a sequence of $(3,4)$ and $(4,3)$ that approximates a straight line of gradient 1. This lead Dasghosh to investigate the number of minimum hops required to reach points (x,y) which lie on straight lines of the form $x = y + k$, where k is an integer. He further conjectured a closed form expression for the minimal number of hops but was unable to provide a proof. In fact, his conjectured expression holds for certain values of (x,y) on those lines $x = y + k$ for some integer k.

The first author was particularly impressed by Dasghosh's computational thinking – characterized by Decomposition, Pattern recognition, Abstraction and Algorithmic design (Papert, 1980; Wing, 2006) – as his main problem-solving paradigm to study this generalisation of the Putnam grasshopper problem. In the following sections, we describe how the authors embarked on a collaborative problem-solving journey which resulted in a solution for all values of (x,y), some nice generalisations of the problem and how computational thinking guided our approaches too. In the ensuing development, all ordered pairs[2] (x,y) have integer-valued components.

Generalised Putnam Grasshopper Problem

Let us first recall the solution to the original problem. The 12 possible hops that the grasshopper can make can be represented by the vectors $\pm(5,0), \pm(0,5), \pm(4,3), \pm(3,4), \pm(-3,4)$, and $\pm(-4,3)$. It is possible to use 578 hops to obtain

$$(2021,2021) = 288(4,3) + 288(3,4) + (5,0) + (0,5).$$

In order to show that the number 578 is the minimum, we consider the *Taxicab metric* which measures the distance between two points by the sum of the absolute values of the differences in each ordinate. In other words, the distance between the points (x_1, y_1) and (x_2, y_2) is given by $|x_1 - x_2| + |y_1 - y_2|$. Using this metric, the distance from $(2021,2021)$ to the origin is 4042. The maximum distance moved by each hop is 7 and since $577 \times 7 = 4039 < 4042$, at least 578 hops are necessary.

We were first interested in the different kinds of hops that would allow the grasshopper to reach any point on the coordinate plane. A discussion over coffee left us with some key ideas on which one of us, Peter (all names are pseudonyms) – a Number Theorist – then individually and completely solved the problem as follows.

One key feature of the problem is the type of hops allowed. For integers a and b, we say the grasshopper has a *basic hop* (a,b) if a hop it makes is any one of the eight vectors $\pm(a,b), \pm(b,a), \pm(a,-b)$, and $\pm(b,-a)$. Thus, the grasshopper in the original Putnam problem has basic hops $(3,4)$ and $(0,5)$. We now state the first generalisation of the problem.

> **Generalised Putnam Grasshopper Problem 1**. A grasshopper, with basic hops $(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)$, starts at the origin in the coordinate plane. Would it be able to reach any point (x,y) in the plane by a sequence of hops?

[2] An ordered pair with integer-valued components is also called a *lattice point*.

The solution of a special but important case of the above problem is stated and proved as a theorem below.

Theorem 1. Let a and b be positive integers. A grasshopper, with basic hop (a, b), can reach any point (x, y) from the origin by a sequence of hops if and only if a and b are relatively prime and have different parity.

Proof. Suppose that a and b are not relatively prime. This means that they have a common factor $k > 1$. Now suppose (x, y) is reached by a sequence of hops, (a_i, b_i). Since k is a common factor of each a_i and b_i, it must also be a common factor of x and y. However, the ordinates of the point (1,0), namely 0 and 1, have no common factors greater than 1. Thus, (1,0) cannot be reached. Now if a and b have the same parity, i.e., both are even or both are odd, adding any of the eight possible vectors to (0,0) will always result in a vector where the x and y-coordinates share the same parity. This again means that $(1,0)$ can never be reached.

Conversely, we can assume without loss of generality that a is odd and b is even. Given that a and b are relatively prime, *Bézout's lemma* states that there exist integers α and β such that $\alpha a + \beta b = 1$. From our assumption that a is odd and b is even, we conclude that α is odd. Note that we have

$$(\alpha - b)a + (\beta + a)b = \alpha a + \beta b = 1.$$

This means that if β was originally odd, we can replace it with a suitable $\beta_0 = \beta + a$ which is even, while $\alpha_0 = \alpha - b$ remains odd. In summary, we know there always exist α_0 and β_0 such that $\alpha_0 a + \beta_0 b = 1$ where α_0 is odd and β_0 is even. Thus

$$\begin{aligned}\frac{\alpha_0 + a}{2}(a, b) + \frac{\alpha_0 - a}{2}(a, -b) + \frac{\beta_0 + b}{2}(b, -a) + \frac{\beta_0 - b}{2}(b, a) \\ = (\alpha_0 a, ab) + (\beta_0 b, -ab) \\ = (\alpha_0 a + \beta_0 b, 0) = (1,0).\end{aligned}$$

So (1,0) can be reached. Analogously $(0,1), (-1,0), (0,-1)$ can also be reached and thus any arbitrary (x, y) can be reached. This concludes the proof. ☐

Minimality for Basic Hops $(3, 4)$ and $(0, 5)$

We then returned to consider a general form of the Putnam Grasshopper Problem, focusing on minimality of the number of hops.

> **Generalised Putnam Grasshopper Problem 2**. A grasshopper, with basic hops $(3,4), (0,5)$, starts at the origin in the coordinate plane. What is the minimum number of hops required for it to reach an arbitrarily given point (x, y)?

From Theorem 1, we see that the basic hop $(3,4)$ already suffice for the grasshopper to reach every point (x, y) in the coordinate plane. We define $m_{3,4}(x, y)$ as the minimum number of hops needed for the grasshopper, with basic hops $(3,4)$ and $(0,5)$, to reach (x, y) from the origin.

This second generalisation turned out to be much more difficult to deal with. As usual, we began our discussions over coffee and with some ideas, the most enthusiastic one among us, Fence – a computer scientist – proceeded to investigate the matter further by writing some programs in Python.

Fence began by analyzing Dasghosh's Python code. Because the Taxicab metric distance between the origin and the destination (x, y) is $x + y$, one adopts a greedy approach to reach this destination using those hops that result in maximum gain in the Taxicab distance via a single hop, i.e., $(3,4)$ and $(4,3)$. If you are lucky enough to hit the destination, then the minimum number of hops must be $\frac{x+y}{7}$ because using vectors other than these two would require more hops. Based on this consideration, Dasghosh used a theoretical estimate of $\left\lfloor\frac{x+y}{7}\right\rfloor$ for the minimum number of hops needed. He then employs a clever coding method to record the frequency of each of the twelve possible hops, i.e., [(0,5),(5,0),(3,4),(4,3),(0,-5),(-5,0),(-3,-4),(3,-4),(4,-3),(-3,4),(-4,3)], in that order using a 12-tuple. For example, the hops sequence [(3, 4), (-5, 0), (3, -4)] is coded as (0,0,1,0,0,1,0,1,0,0,0,0). All possible 12-tuples of non-negative integers whose component sum equal to the theoretical estimate $\left\lfloor\frac{x+y}{7}\right\rfloor$ are generated, and each of these possibilities is tested to check if it represents a sequence of hops that can reach (x, y). A positive test result then returns the desired sequence of hops in the 12-tuple form and its length; otherwise, the program increments the theoretical estimate by one and continues the trial-and-error by brute force within the while-loop.

Understanding program code written by another person is a key step to problem-solving through a computational paradigm as it uncovers the underlying logic and helps one solve the same problem more efficiently and completely by improving the code at hand. For Fence, this meant that he ran Dasghosh's program and found it slow even for small values of x and y. For instance, the destination point $(100,100)$ would initialize the theoretical estimate as 28 and execute a first check over an initial set of $\binom{39}{11} = 1676056044$ possible 12-tuples to see if any of these yield the corresponding vector sum of $(100,100)$; failing which, it enters into a recursion (upper-bounded by some arbitrary large number, say 10 000) by incrementing this theoretical estimate by one in each attempt to reach $(100,100)$. But alas, the program failed to terminate after waiting for more than 8 minutes.

Because an efficient code is much needed for the group to carry out further investigation, Fence quickly applied a simple adaptation of the *A^* search algorithm*, a well-known graph-traversal and pathfinding algorithm that is used in many fields of computer science due to its completeness, optimality and optimal efficiency (Russell & Norvig, 2020). Readers who have a predilection for program codes can find Fence's Python code in Appendix A, and the detailed explanation of the code in Appendix B.

Riding on the optimal efficiency of this A^*search algorithm, Fence wanted to replicate the results reported in Dasghosh (2023). Following closely to Dasghosh's line of investigation, Fence exploited his Python program as a "useful product" of computational thinking (Papert, 1996; Ho et al., 2019) to calculate the values of $m_{3,4}(x, y)$ for points lying on the lines $y = x + k$, where $k = 0,1,2, \ldots, 11$ within the range $0 \leq x \leq 50$. These values of $m_{3,4}(x, y)$ for $y = x$ and $y = x + 8$ (respectively, $y = x + 1$ and $y = x + 11$) are presented as polygonal line graphs in Figure 1 (respectively, Figure 2).

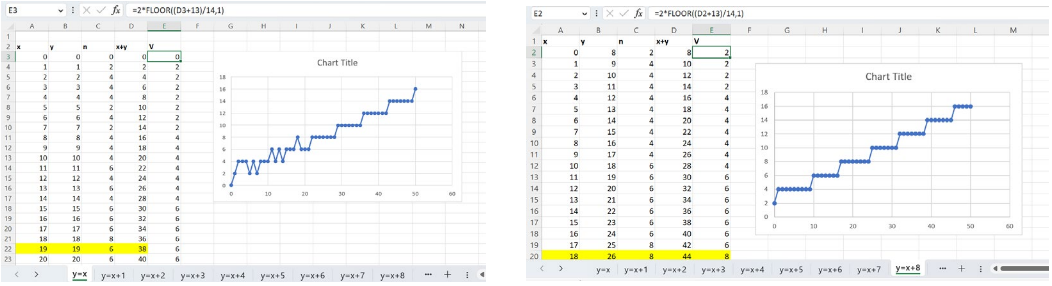

Figure 1. Values of $m_{3,4}(x,y)$ for points on $y = x$ and $y = x + 8$

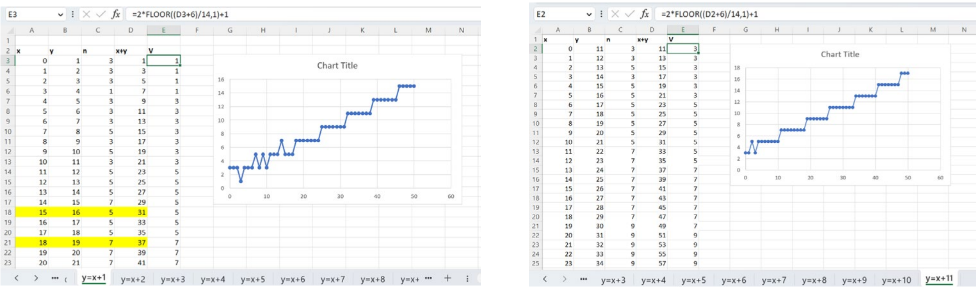

Figure 2. Values of $m_{3,4}(x,y)$ for points on $y = x + 1$ and $y = x + 11$

For each fixed k, the values of $m_{3,4}(x,y)$ (where $y = x + k$) are filled under the column label n. The polygonal line graph shows how $n = m_{3,4}(x,y)$ varies with $s = x + y$, for each point (x,y) on the line $y = x + k$. Invoking Pattern Recognition – one of the four pillars of computational thinking, Fence observed that the number patterns *eventually* emerge as monotonically increasing 'step-functions' with constant step-width of 7. By "eventually" it is meant that the regularity of the number patterns takes place only after a sufficiently large x. After some trial-and-error, Fence came up with a simple formula in terms of s, dependent only on the parity of k, to model the eventual values of $n = m_{3,4}(x,y)$. Here is how computational thinking has been involved so far: *algorithmic design* of the A* search scheme in the form of Python code helps generate the needed data, and *pattern recognition* leads to the following observation. For each non-negative integer k, there exists a positive integer j_k such that for all (x,y) lying on the line $y = x + k$, for all $x \geq j_k$,

$$m_{3,4}(x,y) = \begin{cases} 2\left\lfloor \dfrac{x+y+13}{14} \right\rfloor & \text{if } k \equiv 0 \pmod{2}; \\ 2\left\lfloor \dfrac{x+y+6}{14} \right\rfloor + 1 & \text{if } k \equiv 1 \pmod{2}. \end{cases}$$

From the polygonal line graphs, it appears that if $k \geq 6$, then j_k may be taken as $3k$. Furthermore, a simple identity connecting floor to ceiling, given by $\left\lfloor \frac{n}{m} \right\rfloor = \left\lceil \frac{n-m+1}{m} \right\rceil$, can be used to make each of the floor expressions neater, whence Conjecture 1:

Conjecture 1. For each non-negative integer k, there exists a positive integer j_k such that for all (x,y) lying on the line $y = x + k$, for all $x \geq j_k$,

$$m_{3,4}(x,y) = \begin{cases} 2\left\lceil \dfrac{x+y}{14} \right\rceil & \text{if } k \equiv 0 \pmod{2}; \\ 2\left\lceil \dfrac{x+y}{14} - \dfrac{1}{2} \right\rceil + 1 & \text{if } k \equiv 1 \pmod{2}. \end{cases}$$

When Fence explained his working and conjectures, another of the authors, Theo – a Graph Theorist – felt compelled to do some work following the ideas that arose from the discussion. He decided also to use induction but needed to "see" the patterns of $m_{3,4}(x, y)$ across areas of the coordinate plane rather than along lines of the form $y = x + k$ only. To this end, he independently worked on paper to obtain $m_{3,4}(x, y)$ for small values of x, i.e., $0 \leq x, y \leq 5$, and shared his hand-worked values and propositions with the group (see Figure 3).

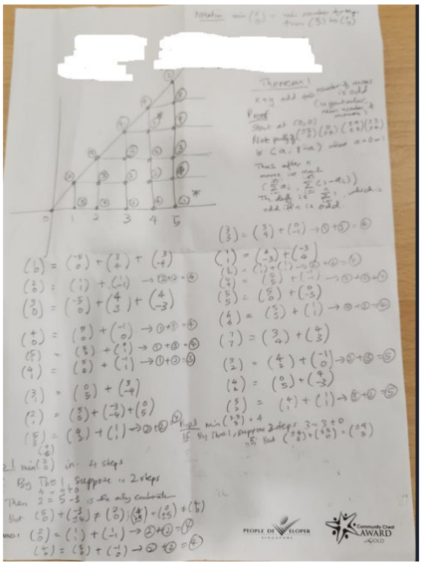

Figure 3. Theo's hand-worked values and some propositions

Concurrently, Peter also went about forging his analogue radar that was devised to inspect the 12 immediate neighbouring positions at any point (x, y) on his hand-crafted grid (Figure 4).

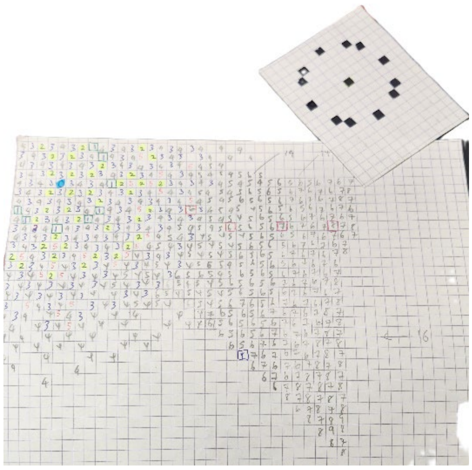

Figure 4. Peter's analogue radar and grid

Then both Theo and Peter moved from Concrete, through Pictorial, to Abstract as they carried out their independent investigations (Leong et al., 2015).

One important result about $m_{3,4}(x, y)$ concerning its parity was obtained from this initial pencil-and-paper working, which we explain below.

Theorem 2. Suppose $m_{3,4}(x,y) = k$, then $x + y$ is odd if and only if k is odd.

Proof. There exists a sequence of exactly k steps such that

$$(x,y) = \sum_{i=1}^{k} (a_i, b_i),$$

where each (a_i, b_i) is one of the 12 possible vectors arising from the basic hops $(3,4)$ and $(0,5)$. In each case $a_i + b_i$ is odd. Thus

$$x + y = \left(\sum_{i=1}^{k} a_i\right) + \left(\sum_{i=1}^{k} b_i\right) = \sum_{i=1}^{k} a_i + b_i \equiv k \pmod 2. \quad \square$$

Remark. Theorem 2 proves useful later when establishing minimality of the number of hops.

Peter then took the findings of Fence and Theo to the next level, and gave a complete solution to Problem 2 for points on the line $y = x$. Already from the solution of the original Putnam problem, we observe that $m_{3,4}(2021{,}2021) = 2\left\lceil \frac{2021}{7} \right\rceil$. Heuristically, we see that those two-hops sequence $(4,3), (3,4)$ resulting in the sum $(4,3) + (3,4) = (7,7)$ help get close to any point along the diagonal line $y = x$. So, a reasonable conjecture would be $m_{3,4}(x,x) \approx 2\left\lceil \frac{x}{7} \right\rceil$ for any positive integer x. Already in Conjecture 1, by specializing $y = x$, i.e., $k = 0$, Fence had guessed that there exists $j_k \in \mathbb{N}$ such that if $x \geq j_k$ then

$$2\left\lceil \frac{x + x}{14} \right\rceil = 2\left\lceil \frac{x}{7} \right\rceil.$$

In fact, Dasghosh (2023) also stated an equivalent conjecture (p. 8).

Theorem 3. Let $x \geq 19$ be a positive integer, then $m_{3,4}(x,x) = 2\left\lceil \frac{x}{7} \right\rceil$.

Proof. We write $x = 7k + r$, where r is the remainder modulo 7. When x is divisible by 7, $2\left\lceil \frac{x}{7} \right\rceil = 2k$; otherwise $2\left\lceil \frac{x}{7} \right\rceil = 2k + 2$. For each value of r, we construct a sequence of hops from the origin to (x,x), and these are listed in Table 1.

Table 1. *Sequences of hops from the origin to* $(7k + r, 7k + r)$

Value of r	Sequence
0	$k(4,3) + k(3,4)$
1	$k(4,3) + k(3,4) + (4,-3) + (-3,4)$
2	$(k-1)(4,3) + (k+1)(3,4) + (3,-4) + (0,5)$
3	$(k-1)(4,3) + (k-1)(3,4) + 2(5,0) + 2(0,5)$
4	$(k-3)(4,3) + (k+2)(3,4) + 2(5,0) + (0,5)$
5	$k(4,3) + k(3,4) + (5,0) + (0,5)$
6	$(k-2)(4,3) + (k+3)(3,4) + (5,0)$

Other than the case of $r = 0$, where only $2k$ hops are necessary, each of the other cases utilizes exactly $2k + 2$ hops. To show that these are indeed minimum, recall that the Taxicab distance from the origin to $(7k + r, 7k + r)$ is $14k + 2r$. When $r = 0$, using the maximum distance of 7 per hop, it is impossible to reach in fewer than $2k$ hops, making $2k$ the minimum. When $r > 0$, it is impossible to reach in $2k$ hops. Furthermore Theorem 2 informs us that the minimum number of hops must be even, thus making $2k + 2$ hops the minimum. A final remark concerns the requirement for $x \geq 19$. This was necessary for in the case of $r = 4$, we need to ensure that $k - 3$ is nonnegative. This completes our proof. □

For $x < 19$, the values of $m_{3,4}(x, x)$ are tabulated in Table 2.

Table 2. *Values of* $m_{3,4}(x, x)$ *for* $x < 19$.

$\boldsymbol{x}$	$\boldsymbol{m_{3,4}(x,x)}$	$\boldsymbol{x}$	$\boldsymbol{m_{3,4}(x,x)}$	$\boldsymbol{x}$	$\boldsymbol{m_{3,4}(x,x)}$
1	2	7	2	13	6
2	4	8	4	14	4
3	4	9	4	15	6
4	4	10	4	16	6
5	2	11	6	17	6
6	4	12	4	18	8

Fence surmised that there might be a formula for $m_{3,4}(x, y)$ for all points on the coordinate plane. Theo was not too sure, but he decided to investigate further by working out more values of $m_{3,4}(x, y)$ relying on a backward search algorithm performed on an Excel spreadsheet. The original hand-worked values for $0 \leq x, y \leq 5$ were inserted into the grid. Further values of $m_{3,4}(x, y)$ were obtained by adding 1 to the minimum value of all previous twelve cells that "could reach the point (x, y)" using one of the basic hops (3, 4) and (0, 5). Using the fill-handle in Excel, this backward search method literally spread the values of $m_{3,4}(x, y)$ from the originally hand-crafted 6×6 grid to a much larger region of 31×31, demonstrating the "spreading" power of electronic spreadsheets (Hvorecky & Trencansky, 1998). Figure 5 below displays the initial results.

Compared to Fence's Python code, working with spreadsheets gave better visuals for observing patterns in the cartesian plane. Though coding in spreadsheet seemed adhoc – notwithstanding some bugs (for example $m_{3,4}(18,18) = 8$; thankfully, none were fatal) – Theo's down-to-earth approach was in line with a "fail fast and learn quickly" (see for example, Lee, 2014) philosophy to computational problem-solving. Certain "best" values were noted, and those cells were then coloured; for example, $m_{3,4}(0,5k) = m_{3,4}(5k, 0) = k$ can be observed in the relevant (0, 5*k*) and (5*k*, 0) cells that were then coloured blue. Taking a cue from Fence's Python approach, best values along lines such as $y = \frac{4}{3}x + 5k$ and $x = \frac{4}{3}y + 5k$, for integer values of *k*, were also coloured. Theo visualized these lines as 'highways' via the $(3,4)$ or $(4,3)$ hops that get the grasshopper to quickly reach the blue (0, 5*k*) and (5*k*, 0) cells.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
0	0	3	4	3	4	1	2	5	2	5	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6
1	3	2	3	2	3	4	3	2	3	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7
2	4	3	4	3	2	3	4	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8
3	3	2	3	4	1	4	3	4	5	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7
4	4	3	2	1	4	3	4	3	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8
5	1	4	3	4	3	2	3	4	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7
6	2	3	4	3	4	3	4	3	2	5	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8
7	5	2	3	4	3	4	3	2	5	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7
8	2	3	4	5	2	3	2	5	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8
9	5	4	3	2	3	4	5	4	3	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7
10	2	5	4	3	4	3	4	5	4	5	4	3	4	5	6	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8
11	3	4	3	4	3	4	5	4	3	4	3	6	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7
12	4	3	4	3	4	5	4	3	4	3	4	5	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8
13	3	4	5	6	3	4	3	4	5	4	5	4	5	6	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7
14	4	5	4	3	4	5	4	5	4	5	6	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8
15	3	4	5	4	5	4	5	4	5	4	5	4	5	4	5	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7
16	4	5	4	5	4	5	4	5	4	5	4	5	4	5	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8
17	5	4	5	4	5	6	5	4	5	4	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9
18	4	5	6	5	4	5	4	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	8	7	8	7	8
19	5	6	5	4	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	8	7	8	7	8	9
20	4	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	8	7	8	7	8	9	8
21	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9
22	6	5	6	5	6	7	6	5	6	5	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8
23	5	6	7	6	5	6	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8	9
24	6	7	6	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8	9	8
25	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9
26	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8
27	7	6	7	6	7	8	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9
28	6	7	8	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	10
29	7	8	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	8	9	10	9
30	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	8	9	8	9	10	9	10

Figure 5. First spreadsheet of values of $m_{3,4}(x, y)$ for $0 \leq x \leq 30$ and $0 \leq y \leq 30$

Theo continued to stare occasionally at the spreadsheet and, for once, noticed that the coloured cells seemed to form "concentric arcs". He then "cleaned up" the colours, leaving only the "concentric" patterns as shown in Figure 6.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
0	0	3	4	3	4	1	2	5	2	5	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6
1	3	2	3	2	3	4	3	2	3	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7
2	4	3	4	3	2	3	4	3	4	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8
3	3	2	3	4	1	4	3	4	5	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7
4	4	3	2	1	4	3	4	3	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8
5	1	4	3	4	3	2	3	4	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7
6	2	3	4	3	4	3	4	3	2	5	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8
7	5	2	3	4	3	4	3	2	5	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7
8	2	3	4	5	2	3	2	5	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8
9	5	4	3	2	3	4	5	4	3	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7
10	2	5	4	3	4	3	4	5	4	5	4	3	4	5	6	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8
11	3	4	3	4	3	4	5	4	3	4	3	6	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7
12	4	3	4	3	4	3	4	3	4	3	4	5	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8
13	3	4	3	4	3	4	3	4	5	4	5	4	5	6	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7
14	4	5	4	3	4	5	4	5	4	5	6	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8
15	3	4	5	4	5	4	5	4	5	4	5	4	5	4	5	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7
16	4	5	4	5	4	5	4	5	4	5	4	5	4	5	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8
17	5	4	5	4	5	4	5	4	5	4	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9
18	4	5	4	5	4	5	4	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	8	7	8	7	8
19	5	6	5	4	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	8	7	8	7	8	9
20	4	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	8	7	8	7	8	9	8
21	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9
22	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8
23	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8	9
24	6	7	6	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8	9	8
25	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9
26	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8
27	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9
28	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	10
29	7	8	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	8	9	10	9
30	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	8	9	8	9	10	9	10

Figure 6. Second spreadsheet of values of $m_{3,4}(x, y)$ for $0 \leq x \leq 30$ and $0 \leq y \leq 30$ (note that we have left errors (non-fatal) in the values that we did not notice at that time)

This incidental discovery shed the insight that a Euclidean metric approach would perhaps yield a nice formula for $m_{3,4}(x, y)$ for sufficiently large values *x* and *y*. Theo explained his thinking to Peter. However, Peter was brutal in exposing the errors in Theo's computation. Peter suggested that the correct metric should be the Taxicab but acknowledged that nearer the axes, the Euclidean seemed to dominate. Theo suggested using both in some amalgam and countered Peter's objection of using two metrics by pointing out that we would not be claiming any "strong" properties of metric spaces.

Nonetheless, Theo persisted in working on the patterns afforded by the spreadsheet. A fifth iteration yielded Figure 7 and a comprehensive solution of Problem 2 that incorporated both metrics. The detailed proofs may not be suitable for this article as our main purpose is to demonstrate the workings of a collaborative problem-solving team and the insights offered by computational thinking and programming. We have instead submitted our results with full proofs to a mathematics journal (Tay et al., submitted). In this paper, we will just give a sketch of the approach and state the results. We need only investigate the positive quadrant of the Cartesian plane since by symmetry, results for the other quadrants follow. We first work out by hand and on a spreadsheet initial values of $m_{3,4}(x, y)$ for $0 \leq x \leq 35$ and $0 \leq y \leq 35$, and this time, present this table of values in the format of the first quadrant, i.e., increasing *y*-values (respectively, *x*-values) as one moves upward (respectively, rightward).

35	7	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	10	9	10	9	10	9	10	9	10	9	10	9	10	11	10	11	10	11	10
34	8	9	8	7	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	10	9	10	9	10	9	10	9	10	9	10	11	10	11	10	11
33	7	8	9	8	7	8	7	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	10	9	10	9	10	9	10	9	10	11	10	11	10
32	8	7	8	7	8	9	8	7	8	7	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	10	9	10	9	10	9	10	11	10	11
31	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	8	9	8	9	8	9	10	9	10	9	10	9	10	11	10
30	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	8	9	8	9	10	9	10	9	10	9	10	11
29	7	8	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	8	9	10	9	10	9	10	9	10
28	6	7	8	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	10	9	10	9	10	9
27	7	6	7	6	7	8	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	10	9	10	9	10
26	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	10	9	10	9
25	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	10	9	10
24	6	7	6	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8	9	8	9	8	9	10	9
23	5	6	7	6	5	6	5	6	7	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8	9	8	9	10	9	10
22	6	5	6	5	6	7	6	5	6	5	6	7	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8	9	8	9	10	9
21	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	6	7	8	7	8	7	8	7	8	9	8	9	8	9	10
20	4	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8	9	10	9
19	5	6	5	4	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8	9	10
18	4	5	6	5	4	5	4	5	6	5	6	5	6	7	6	5	6	5	8	7	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8	9
17	5	4	5	4	5	6	5	4	5	4	5	6	5	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8	9	10
16	4	5	4	5	4	5	4	5	4	5	4	5	4	5	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8	9
15	3	4	5	4	5	4	5	4	5	6	5	4	5	4	7	6	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8
14	4	5	4	3	4	5	4	5	4	5	6	5	4	5	4	7	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8	9
13	3	4	5	6	3	4	3	4	5	4	5	4	5	6	5	4	5	6	7	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8
12	4	3	4	3	4	5	4	3	4	3	6	5	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9
11	3	4	3	4	3	4	5	4	3	4	3	6	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8
10	2	5	4	3	4	3	4	5	4	5	4	3	6	5	6	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9
9	5	4	3	2	3	4	5	4	3	4	5	4	3	4	5	6	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8
8	2	3	4	5	2	3	2	5	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9
7	5	2	3	4	3	4	3	2	5	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8
6	2	3	4	3	4	3	4	3	2	5	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9
5	1	4	3	4	3	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9	8
4	4	3	2	1	4	3	4	3	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9
3	3	2	3	4	1	4	3	4	5	2	3	4	3	6	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8
2	4	3	4	3	2	3	4	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8	9
1	3	2	3	2	3	4	3	2	3	4	5	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7	8	9	8
0	0	3	4	3	4	1	2	5	2	5	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7	6	7	6	7	8	7	8	7
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Figure 7. Values of $m_{3,4}(x, y)$ for $0 \leq x \leq 35$ and $0 \leq y \leq 35$

We calculate $m_{3,4}(x, y)$ for $0 \leq x \leq y \leq 5$ as in Table 3 below. Note that $m_{3,4}(x, y) = m_{3,4}(|x|, |y|)$.

The minimality of the values in Table 3 is justified by using Theorem 2. For example, to show that $m_{3,4}(3,3) = 4$, we first use Theorem 2 to obtain $m_{3,4}(3,3) \geq 2$ since $m_{3,4}(x, y) = 0$ if and only if $(x, y) = (0,0)$. Now suppose $m_{3,4}(3,3) = 2$. We observe that to obtain 3 in the *y*-component within 2 hops, 3 = 3 + 0 is the only possibility. However, by running through all the possible sequences of hops with this restriction, we obtain $(\pm 4,3) + (\pm 5,0) \neq (3,3)$. Thus, by Theorem 2, since 3 + 3 is even, we must have $m_{3,4}(3,3) \geq 4$. A sequence of 4 hops, $(0,5) + (4, -3) + (-4, -3) + (3,4) = (3,3)$, shows that $m_{3,4}(3,3) = 4$.

Table 3. *Values of* $m_{3,4}(x, y)$ *for* $0 \leq x \leq y \leq 5$.

$(\boldsymbol{x}, \boldsymbol{y})$	**Path**	$\boldsymbol{m_5(x, y)}$	$(\boldsymbol{x}, \boldsymbol{y})$	**Path**	$\boldsymbol{m_5(x, y)}$
(0,1)	(0, –5)+(4,3)+(–4,3)	3	(2,2)	(1,1)+(1,1)	4
(1,1)	(4, –3)+(–3,4)	2	(2,4)	(5,0)+(–3,4)	2
(0,2)	(1,1)+(–1,1)	4	(2,3)	(3,4)+(3, –4)+(–4,3)	3
(0,3)	(0, –5)+(3,4)+(–3,4)	3	(2,5)	(3,4)+(–1,1)	3
(0,4)	(0,5)+(0, –1)	3	(3,3)	(3,4)+(0, –1)	4
(0,5)	(0,5)	1	(3,4)	(3,4)	1
(1,2)	(5,0)+(0,5)+(–4, –3)	3	(3,5)	(3,4)+(0,1)	4
(1,3)	(5,0)+(–4,3)	2	(4,4)	(3,4)+(1,0)	4
(1,4)	(0,5)+(1, –1)	3	(4,5)	(3,4)+(1,1)	3
(1,5)	(0,5)+(1,0)	3	(5,5)	(5,0)+(0,5)	2

The values $m_{3,4}(x, y)$ in the other cells in Figure 7 are obtained recursively by adding 1 to the minimum of the values of the cells that can reach (*x*, *y*) via (3, 4) and (5, 0) basic hops. For example,

$$m_{3,4}(1,7) = 1 + \min\{m_{3,4}(1,2), m_{3,4}(4,3), m_{3,4}(-2,3), m_{3,4}(5,4), m_{3,4}(-3,4)\}$$
$$= 1 + \min\{3,1,3,3,1\} = 2,$$
$$m_{3,4}(18,18) = 1 + \min\{m_{3,4}(13,18), m_{3,4}(18,13), m_{3,4}(15,14), m_{3,4}(14,15)\}$$
$$= 1 + \min\{7,7,7,7\} = 8, \text{ and}$$
$$m_{3,4}(22,22) = 1 + \min\{m_{3,4}(17,22), m_{3,4}(22,17), m_{3,4}(19,18), m_{3,4}(18,19)\}$$
$$= 1 + \min\{7,7,7,7\} = 8.$$

We then partitioned the positive coordinate plane into sets of the form:

$$[n] = \left\{ (x, y) : \begin{bmatrix} \left\lfloor \frac{x+3y}{15} \right\rfloor = n, & \text{if } y > \frac{4}{3}x; \\ \left\lfloor \frac{x+y}{7} \right\rfloor = n, & \text{if } \frac{3}{4}x \leq y \leq \frac{4}{3}x; \\ \left\lfloor \frac{y+3x}{15} \right\rfloor = n, & \text{if } y < \frac{3}{4}x. \end{bmatrix} \right\},$$

which correspond to the regions between the 'concentric arcs'. The final proof involved induction on [*n*].

The consolidated results for suitably large values of *x* and *y* are given in Theorems 4 and 5 below, whose proofs can be found in Tay et al. (submitted).

Theorem 4. For points (x, y) lying in the region $0 < x \leq y \leq \frac{4}{3}x$ and $x + y \geq 42$, we have

$$m_{3,4}(x, y) = \left\lceil \frac{4y - 3x}{7} \right\rceil + \left\lceil \frac{4x - 3y}{7} \right\rceil.$$

Theorem 5. For points (x, y) lying in the region $0 \leq \frac{4}{3}x \leq y$ and $x + 3y \geq 75$, we have

$$m_{3,4}(x, y) = \left\lfloor \frac{x + 3y}{15} \right\rfloor + \epsilon,$$

where $\epsilon = 0$ if $15 \mid x + 3y$. When $15 \nmid x + 3y$, define

$$W = \{(1, 5j - 1), (2, 5j - 2), (5, 5j - 3) : j \geq 6\},$$

then

$$\epsilon = \begin{cases} 1 & \text{if } (x, y) \notin W, x + y \not\equiv \left\lfloor \frac{x + 3y}{15} \right\rfloor \pmod 2; \\ 2 & \text{if } (x, y) \notin W, x + y \equiv \left\lfloor \frac{x + 3y}{15} \right\rfloor \pmod 2; \\ 3 & \text{if } (x, y) \in W. \end{cases}$$

The values of $m_{3,4}(x, y)$ not covered by Theorems 4 and 5 are tabulated in Table 4.

Table 4. *Values of $m_{3,4}(x, y)$ not covered by Theorems 4 and 5.*

24	6	7	6																		
23	5	6	7	6	5	6													7		
22	6	5	6	5	6	7	6	5	6									7	6	7	
21	5	6	5	6	5	6	5	6	5	6	5	6					7	6	7	6	7
20	4	5	6	5	6	5	6	5	6	5	6	5	6	5	6		6	7	6	7	6
19	5	6	5	4	5	6	5	6	5	6	5	6	5	6	5	6	5	6	7	6	
18	4	5	6	5	4	5	4	5	6	5	6	5	6	7	6	5	6	5	8		
17	5	4	5	4	5	4	5	4	5	4	5	4	5	6	5	6	5	6			
16	4	5	4	5	4	5	4	5	4	5	4	5	4	5	6	5	6				
15	3	4	5	4	5	4	5	4	5	6	5	4	5	4	7	6					
14	4	5	5	3	4	5	4	5	4	5	6	5	4	5	4						
13	3	4	5	6	3	4	3	4	5	4	5	4	5	6							
12	4	3	4	3	4	5	4	3	4	3	6	5	4								
11	3	4	3	4	3	4	5	4	3	4	3	6									
10	2	5	4	3	4	3	4	5	4	5	4										
9	5	4	3	2	3	4	5	4	3	4											
8	2	3	4	5	2	3	2	5	4												
7	5	2	3	4	3	2	3	2													
6	2	3	4	3	4	3	4														
5	1	4	3	4	3	2															
4	4	3	2	1	4																
3	3	2	3	4																	
2	4	3	4																		
1	3	2																			
0	0																				
y / x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Returning to Conjecture 1

Fence's investigation was fixated at looking at the behaviour of $m_{3,4}(x, y)$ along the lines $y = x + k$, where k is a non-negative integer. Since Theorems 3 and 4 completely answered Problem 2, it is only right (and interesting) to return to Conjecture 1 and show that it is just a subcase of these known results. Mathematically, this allows also the creation of new and somewhat surprising identities.

For each non-negative integer k, there exists a non-negative integer j_k such that the set $\{(x,y) \in \mathbb{Z}^2 : y = x + k, x \geq j_k\}$ of points on the line $y = x + k$ all lie in the region $0 < x \leq y \leq \frac{4}{3}x$ and $x + y \geq 42$. Indeed, if $0 \leq k \leq 5$, then $j_k = \left\lceil 21 - \frac{k}{2} \right\rceil$; else $j_k = 3k$. To prove Conjecture 1, it suffices to prove the proposition below and invoke Theorem 3.

Proposition 6. Let x and y be any integers. Define

$$m(x,y) = \left\lceil \frac{4y-3x}{7} \right\rceil + \left\lceil \frac{4x-3y}{7} \right\rceil \text{ and } n(x,y) = \begin{cases} 2\left\lceil \frac{x+y}{14} \right\rceil & \text{if } k \equiv 0 \pmod 2; \\ 2\left\lceil \frac{x+y}{14} - \frac{1}{2} \right\rceil + 1 & \text{if } k \equiv 1 \pmod 2). \end{cases}$$

Then $m(x,y) = n(x,y)$.

Proof. Note that $m(x,y) = -x + \left\lceil \frac{4y-3x+7x}{7} \right\rceil - y + \left\lceil \frac{4x-3y+7y}{7} \right\rceil = -(x+y) + 2\left\lceil \frac{4(x+y)}{7} \right\rceil$. Let $x + y = 14q + r$, where q is an integer and $r \in \{0,1,2,\ldots,13\}$. So,

$$m(x,y) = -(14q + r) + 2\left\lceil \frac{4(14q + r)}{7} \right\rceil = 2q + 2\left\lceil \frac{4r}{7} \right\rceil - r.$$

It remains to verify for each $r = 0,1,2,\ldots,13$ that $m(x,y) = n(x,y)$. The details are given in Table 5 below:

Table 5. *Verifying* $m(x,y) = n(x,y)$ *for* $r = 0,1,2,\ldots,13$.

r	$m(x,y)$	$n(x,y)$
0	$2q$	$2q$
7	$2q+1$	$2\left\lceil q + \frac{1}{2} - \frac{1}{2} \right\rceil + 1 = 2q + 1$
2, 4, 6, 8, 10, 12	$2q+2$	$2\left\lceil \frac{14q + r}{14} \right\rceil = 2q + 2$
1, 3, 5	$2q+1$	$2\left\lceil q + \frac{r}{14} - \frac{1}{2} \right\rceil + 1 = 2q + 1$
9, 11, 13	$2q+3$	$2q+3$

□

Summary for Pedagogical Considerations

The problem-solving journey described above suggests the following considerations for teachers. Firstly, look out for a problem which is rich enough for generalisations. Next, allow for free exploration of ideas alternating with a need for rigorous defence. In our journey, we find that the participants went on different paths but had to return to rigorously defend their approaches in front of the others. This led to a focusing of what was right and a pruning of what was wrong. A healthy attitude of critical judgment with acceptance of a rigorous defence is needed in the back-and-forth discussion of possible solutions. For example, Peter was initially extremely pessimistic that there would be a nice expression for $m_{3,4}(x,y)$. He was not convinced by Fence's conjecture because he had applied it to points near the y-axis and it did not work. He was only convinced after Theo had worked out everything rigorously.

Lester and Kehle (2003) defined problem solving as follows:

> Successful problem solving involves coordinating previous experiences, knowledge, familiar representations and patterns of inference, and intuition *in an effort* [emphasis added] to generate new representations and related patterns of inference that resolve some tension or ambiguity (i.e., lack of meaningful representations and supporting inferential moves) that prompted the original problem-solving activity. (p. 510)

Collaborative problem solving begins when all persons *make an effort* to start solving (a part of) the problem. We saw that the effort made by Fence in discussing the problem and initiating some preliminary results motivated his collaborators to build on his work. Unlike individual problem-solving, collaborative problem-solving allows another to carry on when one tires or runs out of ideas. This building upon one another's work makes the whole more than the sum of its parts. It is organic and not achieved through a mere distribution of work at the beginning of the project. Teachers should be aware of this feature of collaborative problem-solving that we experienced so that they would not 'structure' their student problem-solving groups in the usual distributed work manner and instead create opportunities for groups to discuss their work and findings at various stages.

Finally, we discuss here the computing approaches that were taken. The student used Python and an exhaustive search algorithm to obtain minimal values for many points. It was computationally inefficient, and the generalisation was inaccurate. Fence also used Python but in a more efficient manner and managed to formulate a far cleaner conjecture. The graphical representations and findings were clearer and encouraging enough for the others to pursue a more comprehensive solution. Theo used a spreadsheet and coded directly into the cells. It was good enough to give fast feedback, an aspect of computational thinking that is called "fail fast, learn quickly". The spreadsheet also afforded a stronger visualization of the minimal values and the easy insertion of colours brought up patterns that other programs would take some time to show. Recent studies have already recommended the use of electronic spreadsheets for developing computational thinking and for computational exploration of some school mathematics problems (Sanford, 2018; Ho et al, 2022), and this paper concurs with these.

References

Dasghosh, P. (2023). *Grasshoppers: A Case Study in Vectors and Number Sequences*. Unpublished manuscript.

Ho, W. K., Ghosh, J., Kaur, B., & Looi, C. K. (2022). Spreadsheets: A Technological Nexus between Big Ideas in Mathematics and Computational Thinking in Mathematics Classrooms. In Yang, W.-C., Majewski, M., Meade, D., & Ho, W. K., (Eds.), *Electronic Proceedings of the 27th Asian Technology Conference in Mathematics* (pp. 1–17). Prague, Czech Republic: Mathematics and Technology, LLC.

Ho, W. K., Looi, C. K., Huang, W., Seow, P. & Wu, L. (2019). Realizing Computational Thinking in the Mathematics Classroom: Bridging the Theory-Practice Gap. In Yang, W.-C., & Meade, D. B. (Eds.), *Proceedings of the 24th Asian Technology Conference in Mathematics* (pp. 35–49). Leshan, Sichuan, China: Mathematics and Technology, LLC.

Hvorecky, J., and Trencansky, I. (1998). Recursive Computations in Spreadsheets. In W.-C. Yang, K. Shirayanagi, S.-C. Chu, and G. Fitz-Gerald (Eds.), *Electronic Proceedings of the 3rd Asian Technological Conference in Mathematics*, 290 –298. Springer, Tokyo.

Kós, G. (2011). On the Grasshopper Problem with Signed Jumps. *The American Mathematical Monthly, 118*(10), 877–886. https://doi.org/10.4169/amer.math.monthly.118.10.877

Lee, H. L. (2014) Speech at Smart Nation launch 24 November 2014. Retrieved 8 Feb 2024: https://www.smartnation.gov.sg/media-hub/speeches/smart-nation-launch/

Leong, Y. H., Ho, W. K., & Cheng, L. P. (2015). Concrete-Pictorial-Abstract: Surveying its origins and charting its future. *The Mathematics Educator, 16*(1), 1–19.

Lester, F. K. & Kehle, P. (2003). From problem solving to modeling: The evolution of thinking about research on complex mathematical activity. In R. Lesh & H. M. Doerr (Eds.), *Beyond constructivism: Models and modeling perspectives on mathematics problem solving, learning, and teaching* (pp. 501-517). Erlbaum.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*, Basic Books, Inc. New York.

Papert, S. (1996). An Exploration in the Space of Mathematics Educations, *International Journal of Computers for Mathematical Learning, 1*(1), 95–123.

Russell, S. J., & Norvig, P. (2020). *Artificial intelligence: A modern approach.* (4th ed.). Boston: Pearson.

Sanford, J. (2018). Introducing Computational Thinking Through Spreadsheets, In M. S. Kline (Ed.), *Computational Thinking in the STEM disciplines.* Springer International Publishing AG, pp. 99–124.

Tay, E. G., Toh, P. C., Ho, W. K., & Tong, C. L. A general solution of the Putnam Grasshopper Problem. (Submitted.)

Ullman, D. H., & Zeitz, P. (2022). The Eighty-Second William Lowell Putnam Mathematical Competition. *The American Mathematical Monthly, 129*(8), 703–716. https://doi.org/10.1080/00029890.2022.2099714

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–35.

Authors:
HO, Weng Kin (wengkin.ho@nie.edu.sg),
National Institute of Education, Nanyang Technological University, Singapore,
1 Nanyang Walk, Singapore 637616

TAY, Eng Guan (engguan.tay@nie.edu.sg),
National Institute of Education, Nanyang Technological University, Singapore,
1 Nanyang Walk, Singapore 637616

TOH, Pee Choon (peechoon.toh@nie.edu.sg),
National Institute of Education, Nanyang Technological University, Singapore,
1 Nanyang Walk, Singapore 637616

TONG, Cherng Luen (chernglueen.tong@nie.edu.sg),
National Institute of Education, Nanyang Technological University, Singapore,
1 Nanyang Walk, Singapore 637616

Appendix A

Fence's Python code: an adaptation of A* search algorithm

```
import heapq
import math

# Define the possible hops
hops = [(5,0), (0,5), (-5,0), (0,-5), (3,4), (-3,4), (3,-4), (4,3), (-
4,3), (4,-3), (-3,-4), (-4,-3)]

# Define the heuristic function, which estimates the cost to reach the
goal from a given position
def heuristic(a, b):
    return math.sqrt((a[0] - b[0]) ** 2 + (a[1] - b[1]) ** 2)

# Define the function to find the least number of hops
def find_least_hops(start, goal, hops):
    queue = [(0, start, [])]
    visited = set()

    while queue:
        (cost, position, path) = heapq.heappop(queue)

        if position == goal:
            return cost, path

        if position not in visited:
            visited.add(position)

            for hop in hops:
                new_position = (position[0] + hop[0],
                                position[1] + hop[1])
                new_cost = cost + 1
                new_path = path + [hop]

                heapq.heappush(queue,
                       (new_cost + heuristic(new_position, goal),
                       new_position, new_path))

    return None

start = (0, 0)
goal  = (1, 0)

least_hops, path = find_least_hops(start, goal, hops)
print("The least number of hops is", len(path))
print("The sequence of hops is", path)
```

Appendix B

Detailed explanation of Python codes in Appendix A.

For each goal (x, y) the grasshopper is attempting to reach, define the vertices of the finite graph G to be all the lattice points (u, v) reachable by the grasshopper in $3(x + y)$ basic hops of $(3,4)$ and $(0,5)$, starting from the origin, and the edges to be those basic hops between vertices. We begin by defining the possible hops using a list `hop`:

```
# Define the possible hops
hops = [(5,0), (0,5), (-5,0), (0,-5), (3,4), (-3,4), (3,-4), (4,3), (-4,3),
(4,-3), (-3,-4), (-4,-3)]
```

These are the possible directions (hops) that can be taken on the grid. For example, `(5,0)` represents a move 5 units to the right, and `(3,4)` represents a move 3 units to the right and 4 units up.

The A* search algorithm makes use of a heuristic function whose role is to estimate the cost of reaching the goal from the start position; in this case, we choose the Euclidean distance between two points as the heuristic.

```
# Define the heuristic function, which estimates the cost to reach the goal
from a given position
def heuristic(a, b):
    return math.sqrt((a[0] - b[0]) ** 2 + (a[1] - b[1]) ** 2)
```

We then define the `find_least_hops` function.

```
# Define the function to find the least number of hops
def find_least_hops(start, goal, hops):
```

This function uses the A* search algorithm to find the number of hops from the `start` position to the `goal` position. It takes the staring position, the goal position, and the allowed hop as parameters. Typically, the A* search algorithm comprises the following:

1. The function uses a priority queue `heapq` to prioritize positions with lower total cost.
2. It initializes the queue with the starting position, cost and an empty path.
3. It iteratively pops the position with the lowest cost from the queue, checks if it is the goal position, and if not, expands the possible hops from that position.
4. The heuristic is then employed to prioritize positions that are closer to the goal.

```
start = (0, 0)
goal  = (1, 0)

least_hops, path = find_least_hops(start, goal, hops)
```

It initializes the start and goal positions and then calls the `find_least_hops` function to get the least number of hops and the path.

```
print("The least number of hops is", len(path))
print("The sequence of hops is", path)
```

We now open the bonnet to reveal the prioritizing mechanism used in the A* search algorithm. A routine initialization involves setting up the priority queue, traditionally called the *open list*, that holds tuples `(cost, position, path)`, where `cost` is the total cost, `position` is the current position, and `path` is the list of hops taken. The list `visited` archives all visited positions.

```
    queue = [(0, start, [])]
    visited = set()

    while queue:
        (cost, position, path) = heapq.heappop(queue)
```

The program contains one main loop:

```
    while queue:
        (cost, position, path) = heapq.heappop(queue)
```

This loop continues till the queue is empty. It then pops the element with the lowest cost (i.e., the path consisting of the least number of hops) from the priority queue.

A goal check is performed:

```
        if position == goal:
            return cost, path
```

More precisely, if the current position is the goal, the function returns the cost and path because this means this is a possible path that reaches the destination (x, y).

When this current position has not been visited, the program adds it to the list `visited`:

```
        if position not in visited:
            visited.add(position)
```

The function then iterates over the possible hops, calculates the new position, cost, and path, and adds the new state to the priority queue with the total cost plus the heuristic.

```
            for hop in hops:
                new_position = (position[0] + hop[0], position[1] + hop[1])
                new_cost = cost + 1
                new_path = path + [hop]

                heapq.heappush(queue, (new_cost + heuristic(new_position, goal),
                               new_position, new_path))
```

If the loops exhaust and no goal is found, it returns `None`.

```
    return None
```