Stimulating students' creativity with tasks solved using precomputation and visualization

Tomasz Kulczyński, Jakub Łącki, Jakub Radoszewski

University of Warsaw

24th July 2011

Introduction

The usual approaches of solving problems:

- making observations,
- guessing some properties,
- deducing solutions,
- ...

Introduction

The usual approaches of solving problems:

- making observations,
- guessing some properties,
- deducing solutions,
- ...

An alternative:

- use a computer,
- experiment,
- visualize,
- precompute,
- generate data and analyze it.

Number game

Two-player game

- state: a pair (x, y) of positive integers $(x, y \leq 10^9)$
- move
  1. a player subtracts 1 from one of the numbers
  2. then divides both numbers by their GCD
- the player who gets $(1, 1)$ loses the game

Number game

Two-player game

- state: a pair (x, y) of positive integers $(x, y \leq 10^9)$
- move
  1. a player subtracts 1 from one of the numbers
  2. then divides both numbers by their GCD
- the player who gets $(1, 1)$ loses the game

$$(4, 7) \rightarrow (2, 3) \rightarrow (1, 1)$$

Number game

Two-player game

- state: a pair (x, y) of positive integers $(x, y \leq 10^9)$
- move
  1. a player subtracts 1 from one of the numbers
  2. then divides both numbers by their GCD
- the player who gets $(1, 1)$ loses the game

$$(4, 7) \rightarrow (2, 3) \rightarrow (1, 1)$$

$$(4, 7) \rightarrow (3, 7) \rightarrow (1, 2) \rightarrow (1, 1)$$

Task

Task

Determine if a given position is winning.

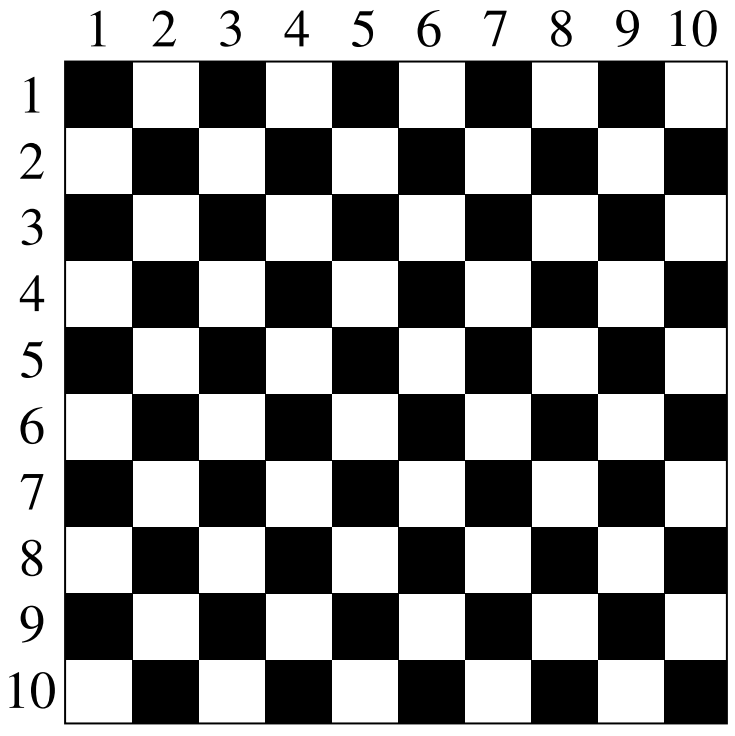
Task

Task

Determine if a given position is winning.

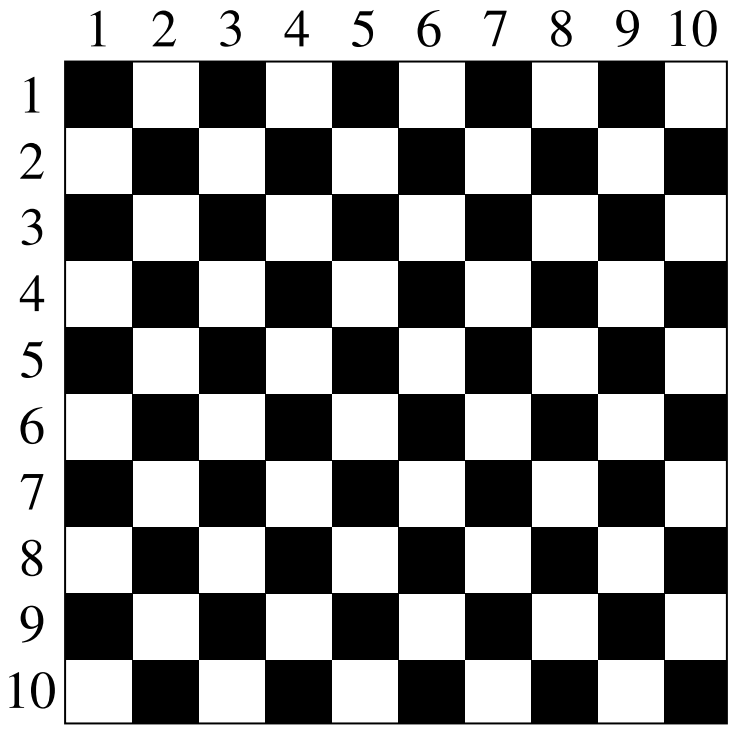
We can implement a simple DP algorithm (running in $O(n^2 \log n)$) to solve the problem for $i, j \leq n$.

Visualization



A cell $[i, j]$ is white if and only iff position (i, j) is winning.

Visualization



A cell $[i,j]$ is white if and only iff position (i,j) is winning.
We conjecture that (i,j) is losing if and only if i and j have the same parity.

Proof

We prove the claim by induction on $i + j$.

Proof

We prove the claim by induction on $i + j$.
Base case ($i = j = 1$) follows from rules. Now assume that the claim holds for all i', j', such that $i' + j' < i + j$.

Proof

We prove the claim by induction on $i + j$.
Base case ($i = j = 1$) follows from rules. Now assume that the claim holds for all i', j', such that $i' + j' < i + j$.

1. If both numbers have the same parity, then after the move they have different parity, so the position is losing.

Proof

We prove the claim by induction on $i + j$.
Base case ($i = j = 1$) follows from rules. Now assume that the claim holds for all i', j', such that $i' + j' < i + j$.

1. If both numbers have the same parity, then after the move they have different parity, so the position is losing.
2. If the parity is different, we subtract 1 from the even number. Hence, after the move both numbers are odd, so the opponent faces a losing position.

Antiprime numbers

An *antiprime number* is a positive integer for which the number of divisors increases to a record.

Antiprime numbers

An *antiprime number* is a positive integer for which the number of divisors increases to a record.

First few elements of the sequence are

$$1, 2, 4, 6, 12, 24, 36, 48, 60, 120, 180.$$

Antiprime numbers

An *antiprime number* is a positive integer for which the number of divisors increases to a record.

First few elements of the sequence are

$$1, 2, 4, 6, 12, 24, 36, 48, 60, 120, 180.$$

Task (8th POI)
Compute the greatest antiprime which is lower than c ($c \leq 2 \cdot 10^9$).

Precomputation

Inefficient solution:

- We iterate over consecutive positive integers $\leq M$ and count their divisors.
- This requires $O(M\sqrt{M})$ time.
- Using Eratosthenes' sieve, one can improve the complexity to $O(M \log M)$ time.

Precomputation

Inefficient solution:

- We iterate over consecutive positive integers $\leq M$ and count their divisors.
- This requires $O(M\sqrt{M})$ time.
- Using Eratosthenes' sieve, one can improve the complexity to $O(M \log M)$ time.

We discover that for $M = 10^6$ there are only 38 antiprimes.

Solution

A possible solution:

1. Implement the $O(M \log M)$ solution.
2. Run it for $M = 2 \cdot 10^9$.
3. Wait.
4. Hardcode the result into the program.

Possible improvements

The algorithm can be improved significantly.

- If $n = 2^{w_1} \cdot 3^{w_2} \cdot 5^{w_3} \cdot 7^{w_4} \cdot \ldots$ then n has $(w_1 + 1)(w_2 + 1)(w_3 + 1)(w_4 + 1) \ldots$ divisors.

Possible improvements

The algorithm can be improved significantly.

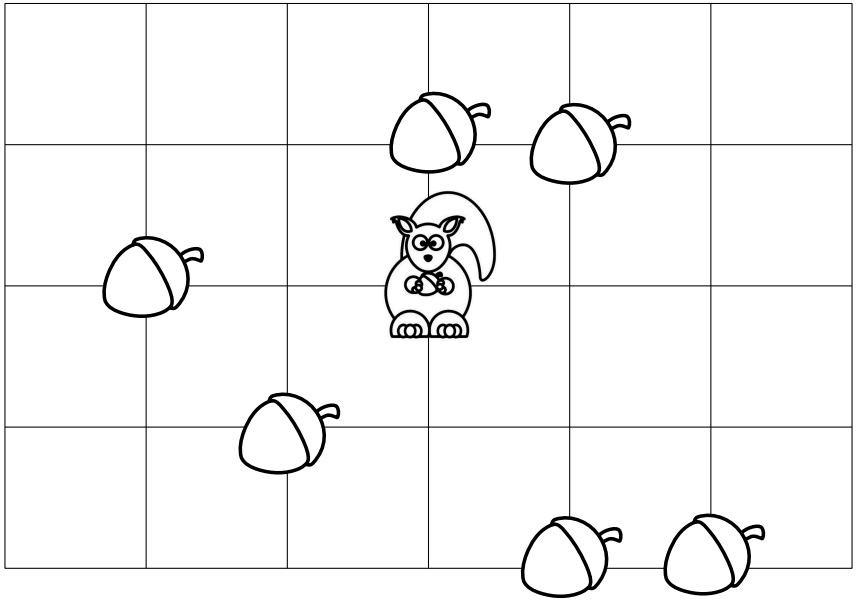
- If $n = 2^{w_1} \cdot 3^{w_2} \cdot 5^{w_3} \cdot 7^{w_4} \cdot \ldots$ then n has $(w_1 + 1)(w_2 + 1)(w_3 + 1)(w_4 + 1) \ldots$ divisors.
- We can only consider those numbers, where $w_1 \geq w_2 \geq w_3 \geq \ldots$.

Problem statement

Problem: A squirrel in a plane (ONTAK 2007)

Setting:

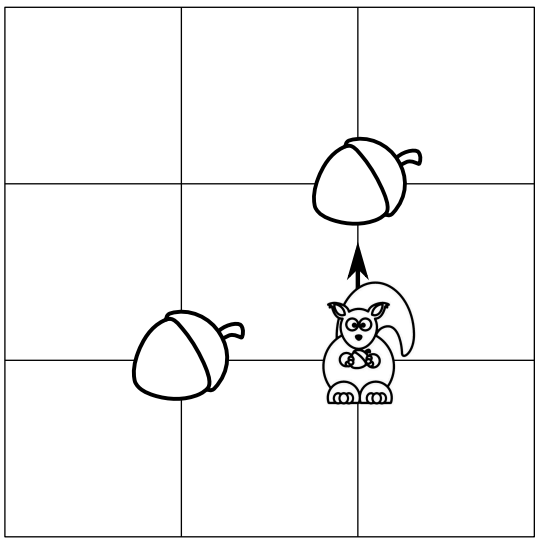
- infinite plane with $n \leq 7$ nuts,
- nuts placed at points (x_i, y_i) $-2 \leq x_i, y_i \leq 2$,
- a squirrel (initially at $(0, 0)$, facing north) collecting nuts.



Problem statement

Squirrel's moves:

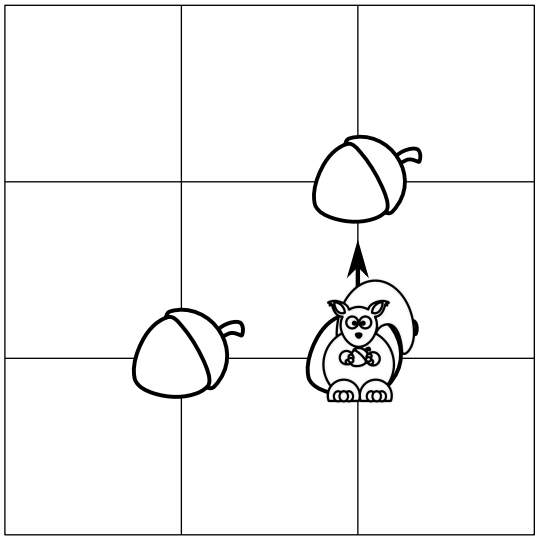
- If it stands on a nut, then it
  - picks it up,
  - turns 90° right,
  - moves 1 unit ahead.
- Otherwise, it
  - drops a nut,
  - turns 90° left,
  - moves 1 unit ahead.



Problem statement

Squirrel's moves:

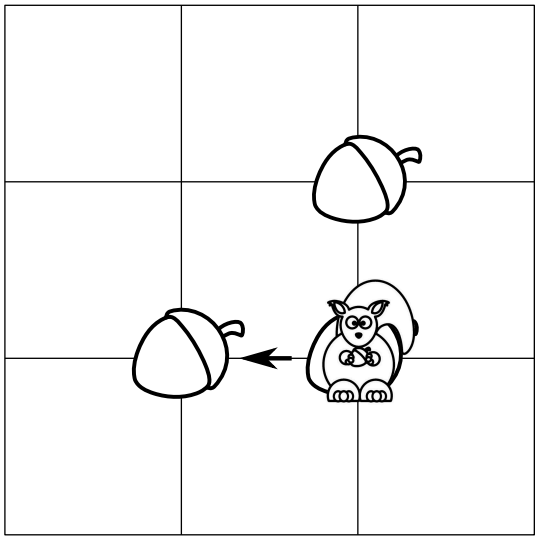
- If it stands on a nut, then it
  - picks it up,
  - turns 90° right,
  - moves 1 unit ahead.
- Otherwise, it
  - drops a nut,
  - turns 90° left,
  - moves 1 unit ahead.



Problem statement

Squirrel's moves:

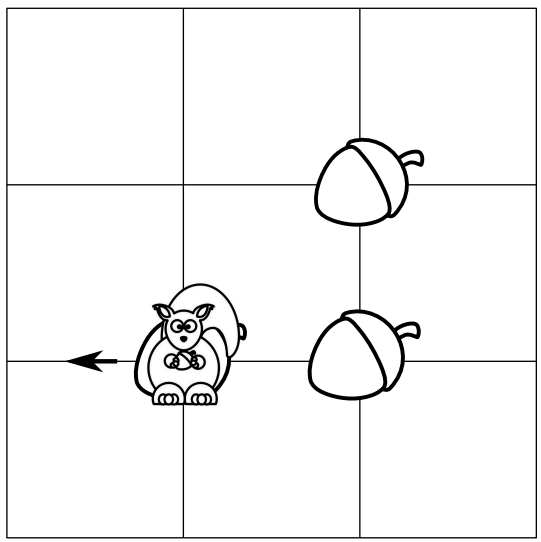
- If it stands on a nut, then it
  - picks it up,
  - turns 90° right,
  - moves 1 unit ahead.
- Otherwise, it
  - drops a nut,
  - turns 90° left,
  - moves 1 unit ahead.



Problem statement

Squirrel's moves:

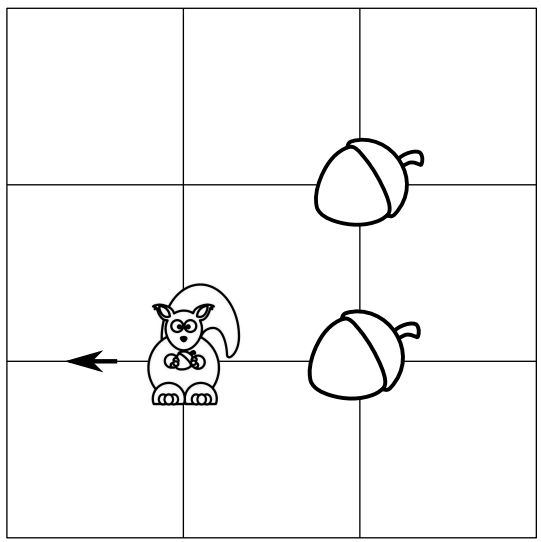
- If it stands on a nut, then it
  - picks it up,
  - turns 90° right,
  - moves 1 unit ahead.
- Otherwise, it
  - drops a nut,
  - turns 90° left,
  - moves 1 unit ahead.



Problem statement

Squirrel's moves:

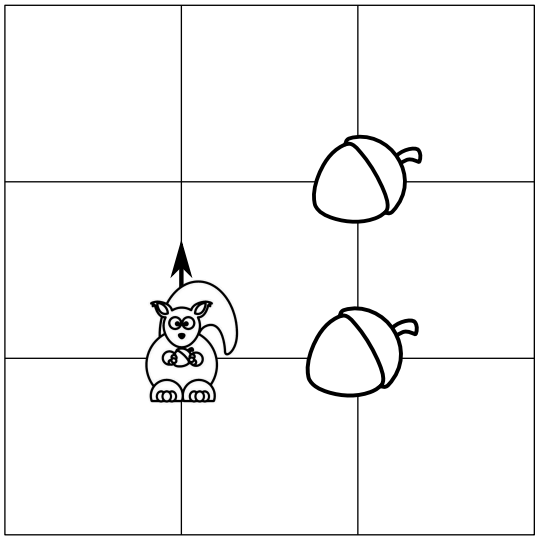
- If it stands on a nut, then it
  - picks it up,
  - turns 90° right,
  - moves 1 unit ahead.
- Otherwise, it
  - drops a nut,
  - turns 90° left,
  - moves 1 unit ahead.



Problem statement

Squirrel's moves:

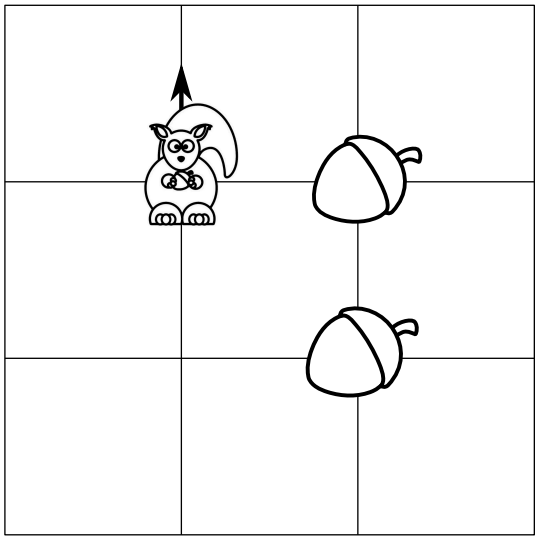
- If it stands on a nut, then it
  - picks it up,
  - turns 90° right,
  - moves 1 unit ahead.
- Otherwise, it
  - drops a nut,
  - turns 90° left,
  - moves 1 unit ahead.



Problem statement

Squirrel's moves:

- If it stands on a nut, then it
  - picks it up,
  - turns 90° right,
  - moves 1 unit ahead.
- Otherwise, it
  - drops a nut,
  - turns 90° left,
  - moves 1 unit ahead.



Task

Task

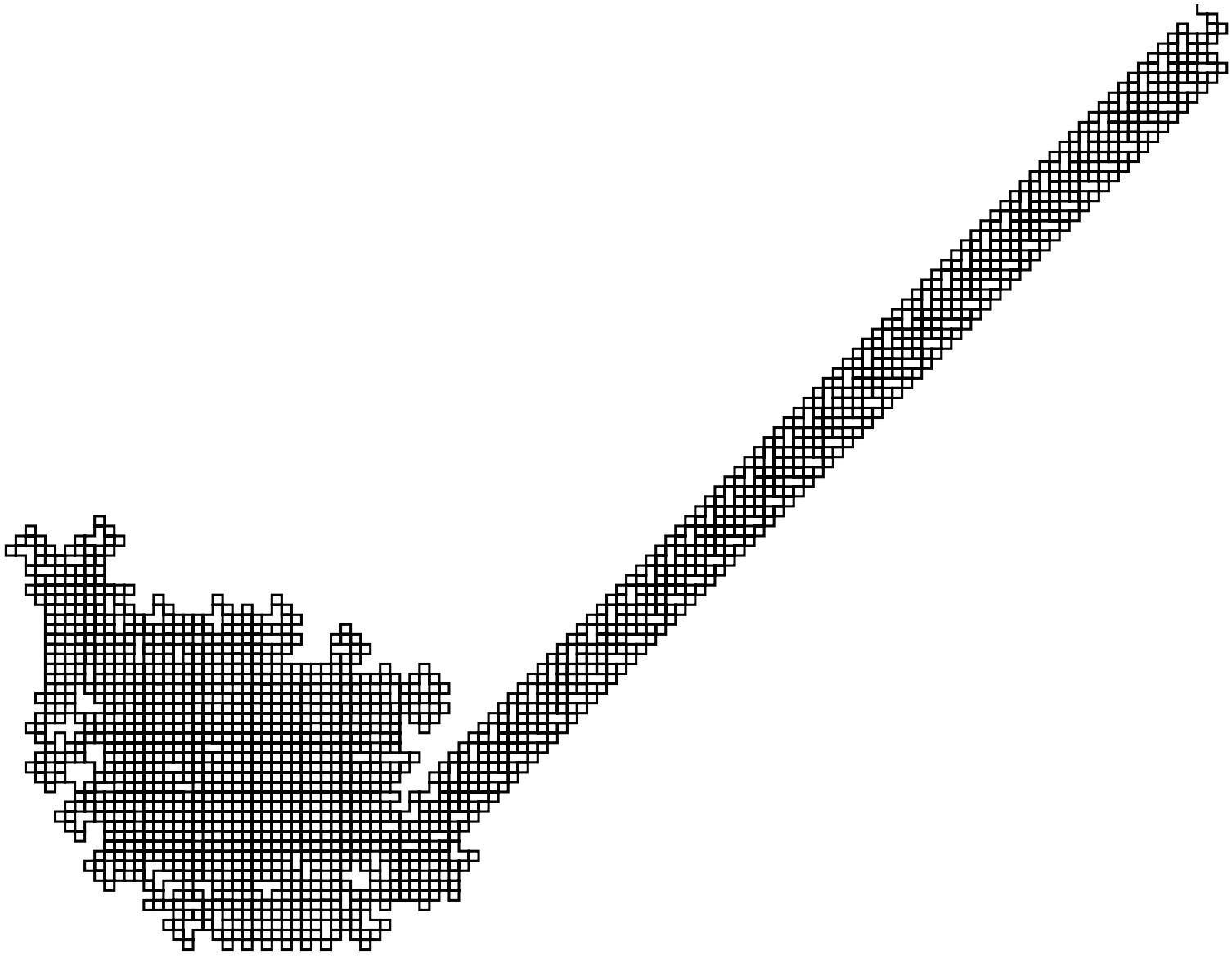
Compute the number of nuts after $t \leq 10^9$ moves.

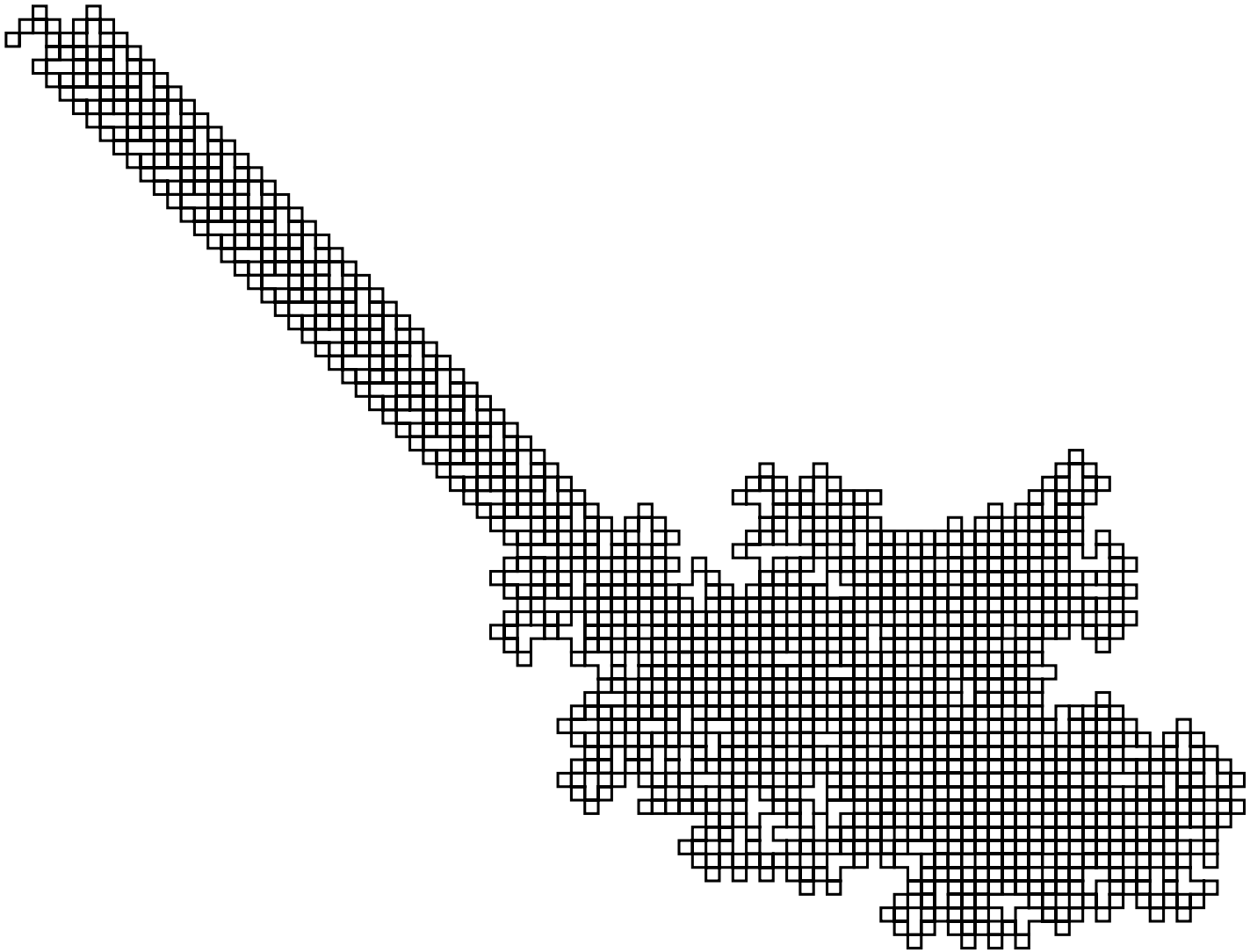
Task

Task

Compute the number of nuts after $t \leq 10^9$ moves.

We simulate several thousand steps of the squirrel for different inputs.





Conclusions

Observations:

- The squirrel repeats a sequence of 104 moves,
- during which 12 nuts are put on the ground.
- The cycle starts close to the origin.

Solution

- We simulate the moves of the squirrel.
- The simulation stops after x moves if the squirrel
  - has reached the distance of 200 units from the origin and
  - $t - x$ is divisible by 104.
- We output the current number of nuts + $\frac{t-x}{104} \cdot 12$.

Correctness

How do we know if it is correct?

Correctness

How do we know if it is correct?

Well, we can check all 726 206 initial configurations.

Stirling numbers of the second kind

$$\left\{ {n \atop m} \right\} = \text{ number of partitions of } \{1, 2, \ldots, n\} \text{ into } m \text{ nonempty sets.}$$

Stirling numbers of the second kind

$\left\{ {n \atop m} \right\}$ = number of partitions of $\{1, 2, \ldots, n\}$ into m nonempty sets.

E.g. $\left\{ {4 \atop 2} \right\} = 7$:

$$\{1,2,3\} \cup \{4\} \quad \{1,2,4\} \cup \{3\} \quad \{1,3,4\} \cup \{2\} \quad \{2,3,4\} \cup \{1\}$$

$$\{1,2\} \cup \{3,4\} \quad \{1,3\} \cup \{2,4\} \quad \{1,4\} \cup \{2,3\}$$

Problem description

Problem (ACM ICPC SWERC 2001)

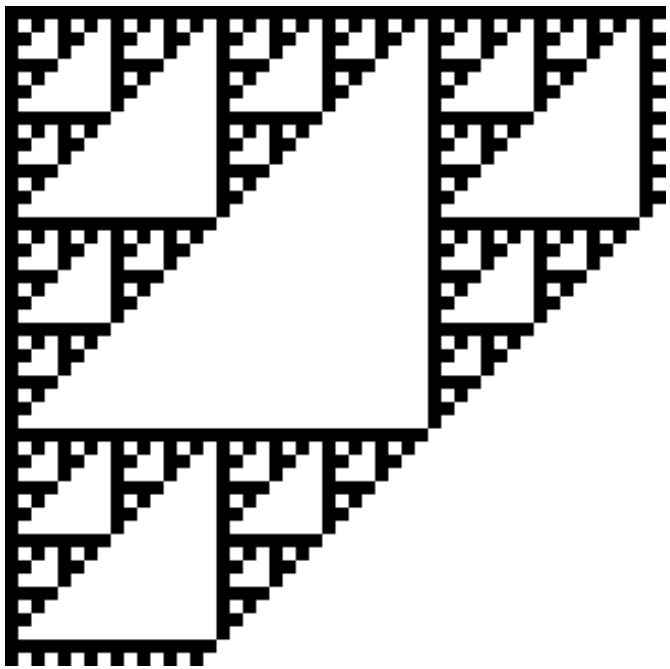
Knowing that:

$$\left\{ {n \atop m} \right\} = \begin{cases} 0 & \text{if } n = m = 0 \\ 1 & \text{if } n = 0 \wedge m > 0 \text{ or } m = 0 \wedge n > 0 \\ m\left\{ {n-1 \atop m} \right\} + \left\{ {n-1 \atop m-1} \right\} & \text{if } n > 0 \wedge m > 0 \end{cases}$$

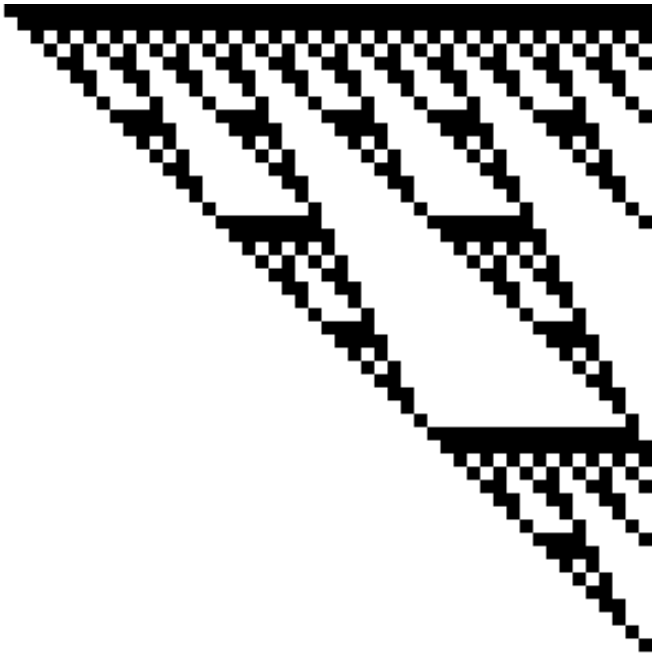
compute $\left\{ {n \atop m} \right\} \mod 2$ for $0 \leq m \leq n \leq 10^9$.

$$a[x][y] = \left\{ {x \atop y} \right\} \mod 2$$

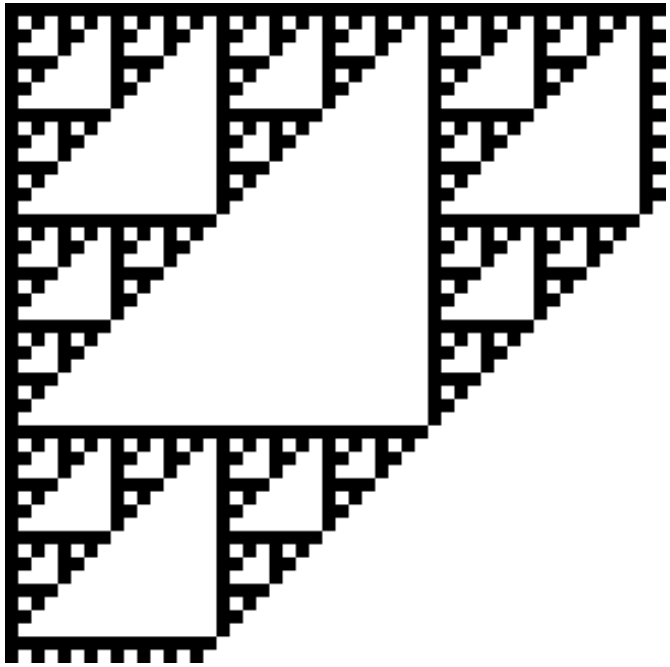
$$a[x][y] = \binom{x}{y} \mod 2$$



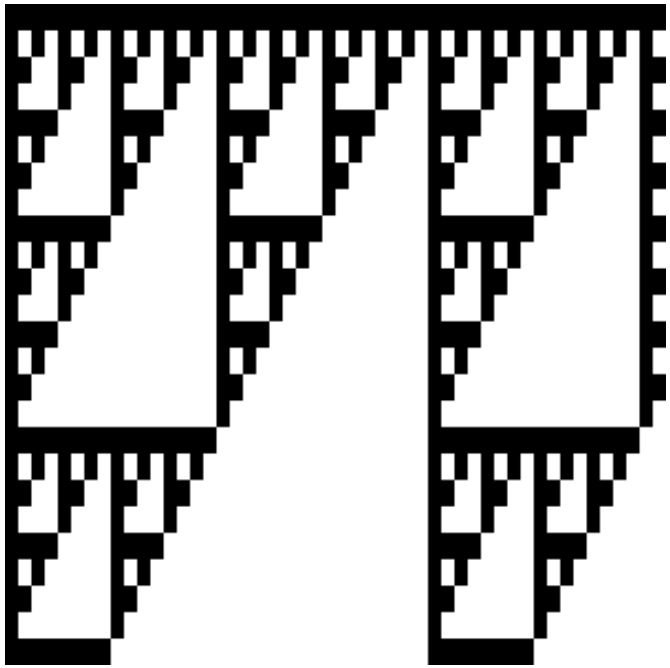
$$b[x][y] = 1 \iff x \& y = 0$$



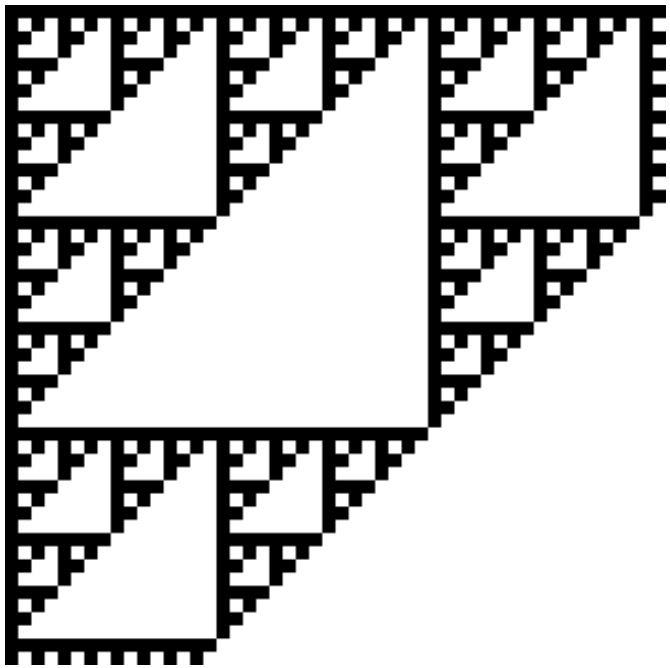
$$a[x][y-1] = \left\{ {x \atop y} \right\} \mod 2$$



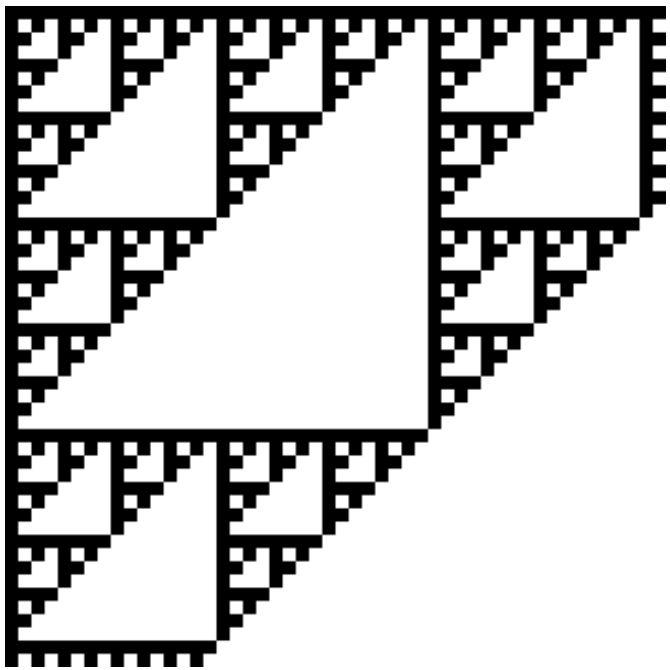
$$b[x][y] = 1 \iff x \& y = 0$$



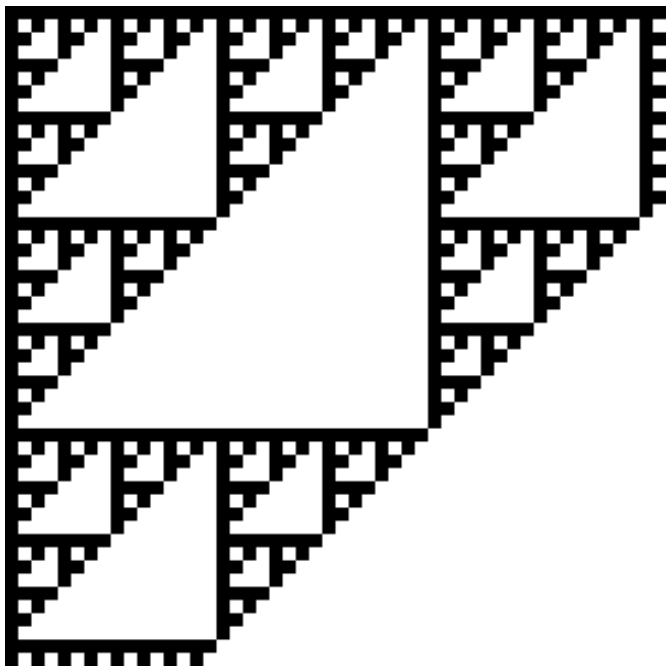
$$a[x-y][y-1] = \binom{x}{y} \mod 2$$



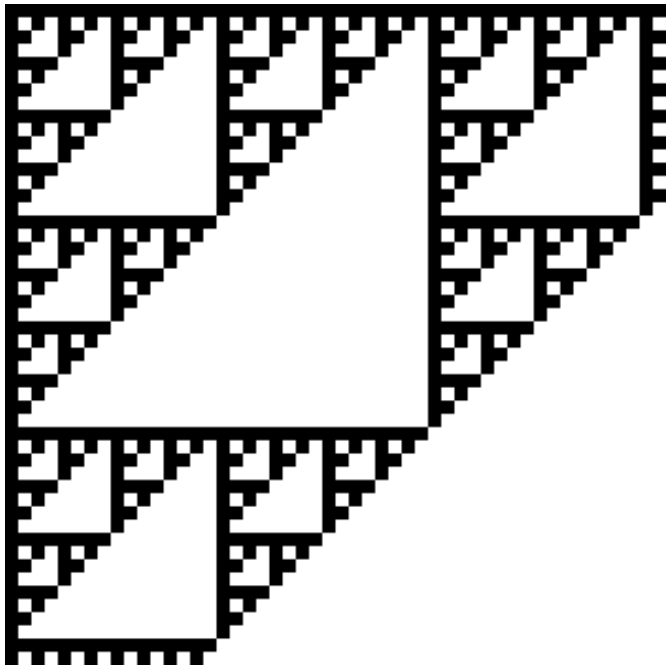
$$b[x][y] = 1 \iff x \& y = 0$$

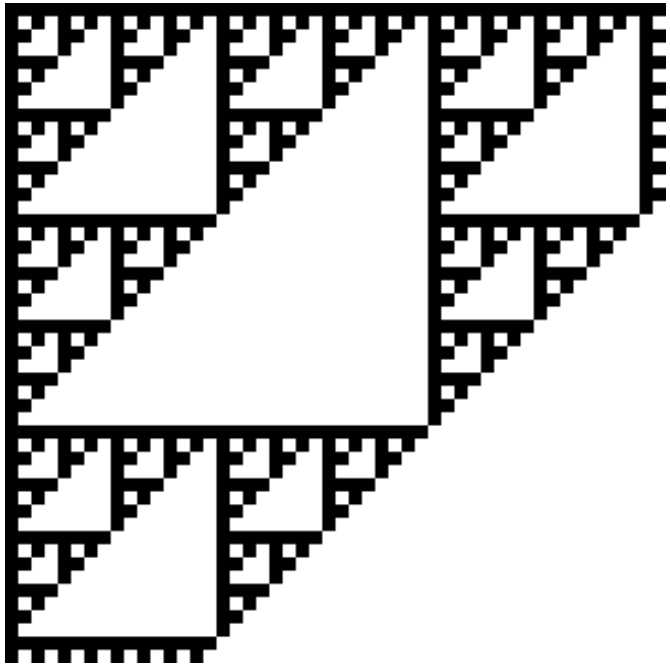


$$a[x-y][\lfloor \tfrac{y-1}{2} \rfloor] = \binom{x}{y} \mod 2$$



$$b[x][y] = 1 \iff x \& y = 0$$





$a[x-y][\lfloor \frac{y-1}{2} \rfloor] = \binom{x}{y} \mod 2$

$b[x][y] = 1 \iff x \& y = 0$

We have $a[x][y] = b[x][y]$.

Conclusions

From

- $a[x-y][\lfloor \frac{y-1}{2} \rfloor] = \binom{x}{y} \mod 2$,
- $b[x][y] = 1 \iff x \& y = 0$,
- $a[x][y] = b[x][y]$

Conclusions

From

- $a[x-y][\lfloor \frac{y-1}{2} \rfloor] = \left\{ {x \atop y} \right\} \mod 2$,
- $b[x][y] = 1 \iff x \& y = 0$,
- $a[x][y] = b[x][y]$

we get

$$\left\{ {x \atop y} \right\} \mod 2 = \left((x-y) \,\&\, \left\lfloor \frac{y-1}{2} \right\rfloor \right) == 0.$$

Summary

Solving tasks with a computer can:

- provide a deeper understanding for the student,
- develop engineering skills,
- stimulate creativity,
- teach students an alternative way of approaching problems,
- result in finding a better solution.