前述背景知識

Birthday Paradox

Birthday Paradox

在最糟的情況裡, 必須選 **367** 個人,
才能保証有人生日撞期.

但, 生日撞期這件事, 也許比我們想的更容易發生！

- 隨機挑出 N 個人,
  是否會有人的生日在同一天？

  - 沒有人的生日同一天的機率為

$$\tilde{p}(n) = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdots \cdot \frac{365-n+1}{365}$$

  - 至少有兩個人生日在同一天的機率

$$p(n) = 1 - \tilde{p}(n)$$

n	*p*(*n*)
1	0.0%
5	2.7%
10	11.7%
20	41.1%
23	50.7%
30	70.6%
40	89.1%
50	97.0%
60	99.4%
70	99.9%
75	99.97%
100	99.99997%
≥ 366	100%

Birthday Paradox

當 n = $O(\sqrt{k})$ 時, p(n) = $O(1)$!

- 從 1 ~ k 隨機挑出一個數字, 挑 N 次,

  至少有一次挑到重覆的數字的機率？

$$p(n) = 1 - \frac{k}{k} \cdot \frac{k-1}{k} \cdot \frac{k-2}{k} \cdots\cdots \frac{k-n+1}{k} = 1 - \prod_{1 \le j < n} \left(1 - \frac{j}{k}\right)$$

$$\approx 1 - \left(\frac{k-1}{k}\right)^{\frac{n(n-1)}{2}} \qquad \approx 1 - e^{-\frac{n(n-1)}{2k}}$$

代入 n = $\sqrt{k}$ 時, p(n) 約莫為 0.4 左右!

Pollard’s Rho 演算法

Pollard’s Rho Algorithm

- 給定一個合成數 N,

  Pollard’s Rho 演算法的目標, 是找出一個介於 1 與 N 之間的因數。

- 在有**完美的亂數序列**的情況下,

  Pollard’s Rho 演算法大約有 ½ 的機率,

  可以在 $O(\sqrt{N})$ 個回合內執行結束, 並且 output 一個 N 的因數。

註: 亦即, 執行的時間可能會超過 $O(\sqrt{N})$ 個回合, 也可能會失敗.

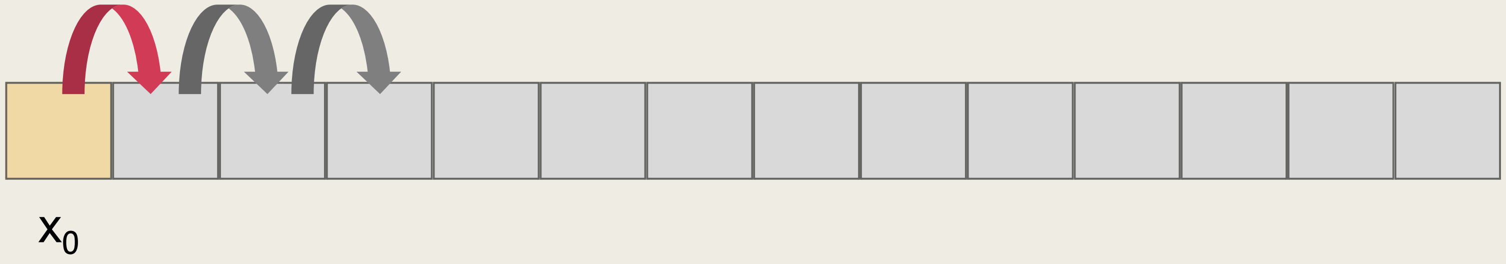
Pollard’s Rho Algorithm

- 此方法背後的原理很簡單！

- 假設我們有一個 **<u>完美的亂數序列</u>** X, 且 X 每一項的值只由前項決定。

  亦即,

$$x_k = g(x_{k-1}), \quad \text{for all } k \geq 1$$



Pollard's Rho Algorithm

- 假設我們有一個 **<u>完美的亂數序列</u>** X, 且 X 每一項的值只由前項決定。

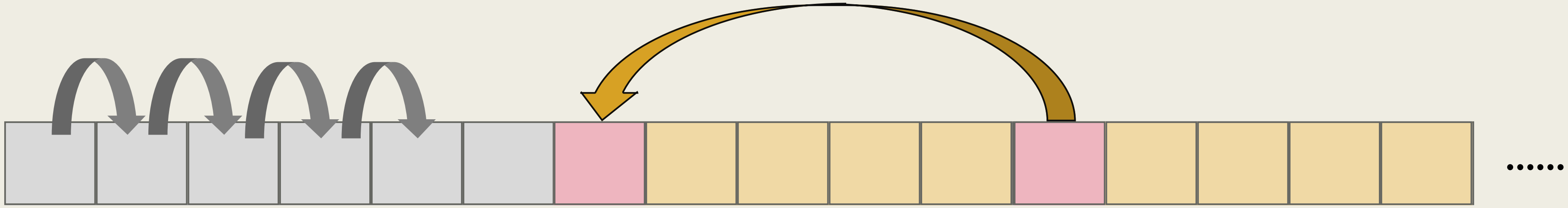
  亦即, $x_k = g(x_{k-1}),\ \text{for all } k \geq 1$

- 令 p 為 n 的任意一個因數.

  考慮序列的每一項 mod p 產生的結果。

由於 mod p 只有 p-1 種可能的結果,

因此，mod 後**序列的值必定會循環**！



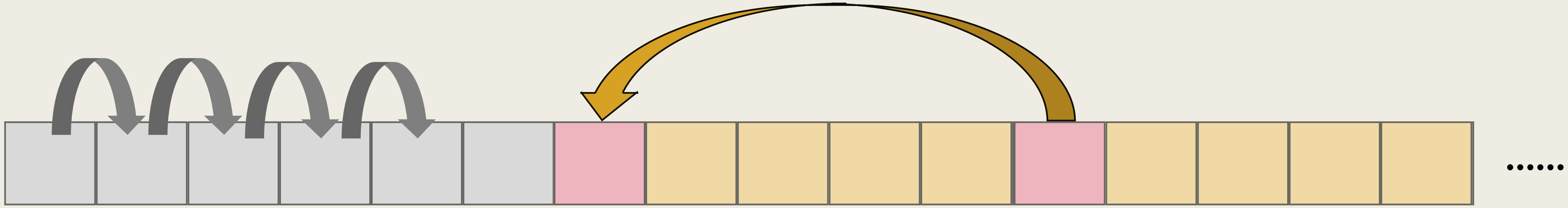
Pollard's Rho Algorithm

- 令 p 為 n 的任意一個因數.

  考慮序列的每一項 mod p 產生的結果。

由於 mod p 只有 p-1 種可能的結果,

因此，mod 後**序列的值必定會循環**！



因為兩項 mod p 後的結果相等

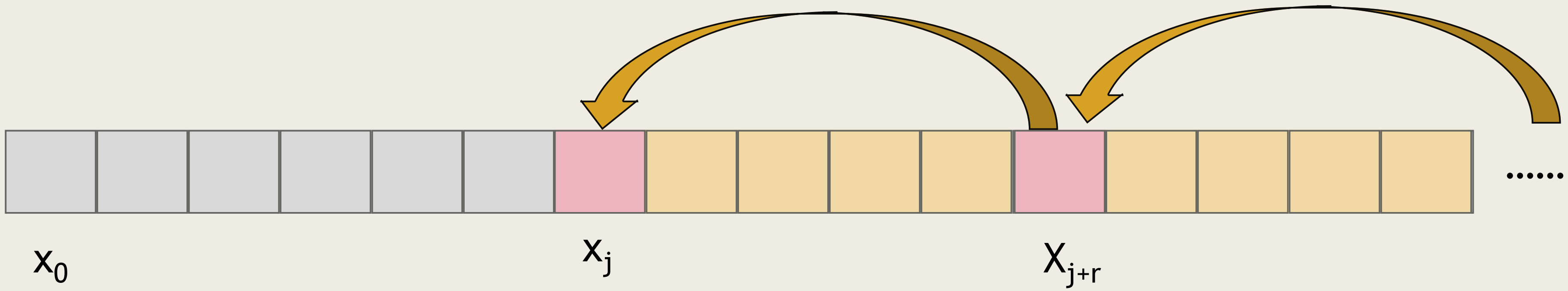
- 令 x_j 為循環的起點, r 為週期, 那麼

  對於所有的正整數 k， x_j 與 x_{j+k*r} 這兩項的差, 必定為 p 的倍數。

- 亦即，當 $\gcd\left(\left|x_j - x_{j+k\cdot r}\right|, n\right) \neq 1$ 以及 n 時, 即為題目所求答案。

如何找出序列的循環點？

- 令 x_j 為循環的起點, r 為週期



- **定理** (Floyd's Cycle Finding Algorithm)

對於任意的正整數 i,

$i = k \cdot r,\ \text{for some } k$ 的充分必要條件為 $x_i \equiv x_{2i}$

如何找出序列的循環點？

- 由定理, 可用 two pointer method 雙指標法來找循環點！

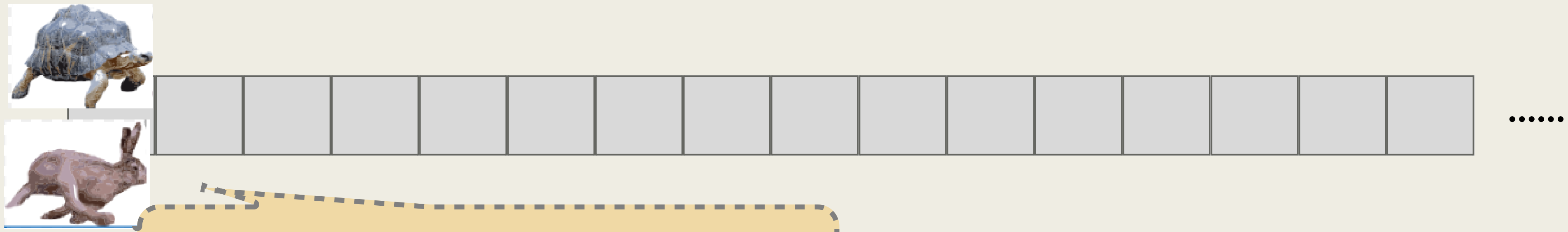
- **<u>定理</u>** <u>(Floyd’s Cycle Finding Algorithm)</u>

  對於任意的正整數 i,

  $i = k \cdot r, \ \text{for some } k$ 的充分必要條件為 $x_i \equiv x_{2i}$

如何找出序列的循環點？

- d 雙指標法！



- <u>定理 (Floyd's Cycle Finding Algorithm)</u>

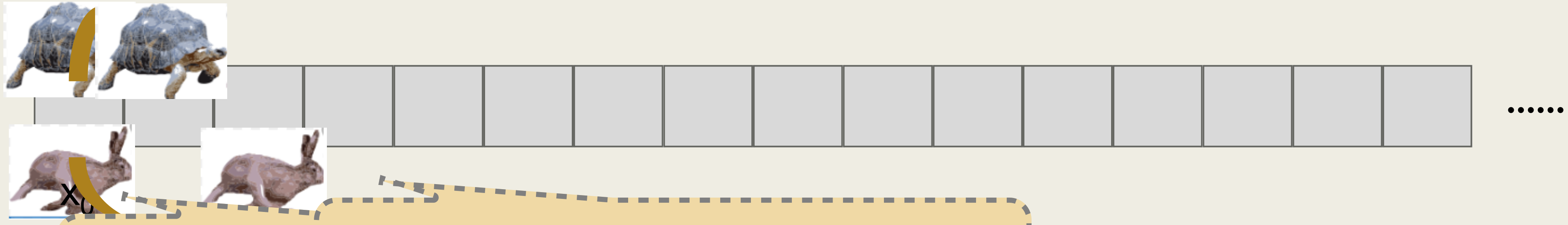
  對於任意的正整數 i,

  $i = k \cdot r,\ \text{for some } k$ 的充分必要條件為 $x_i \equiv x_{2i}$

如何找出序列的循環點？

- 指標法！

Tortoise pointer: 每次前進一格



Hare pointer: 每次前進兩格！

- <u>定理 (Floyd’s Cycle Finding Algorithm)</u>

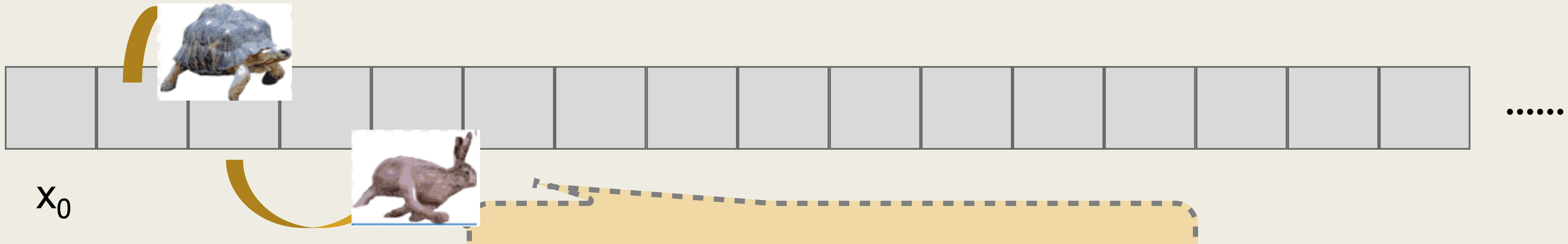
  對於任意的正整數 i,

  $i = k \cdot r,\ \text{for some } k$ 的充分必要條件為 $x_i \equiv x_{2i}$

如何找出序列的循環點？

- 由定理, 法！

Tortoise pointer: 每次前進一格



Hare pointer: 每次前進兩格！

- <u>定理 (Floyd's Cycle Finding Algorithm)</u>

  對於任意的正整數 i,

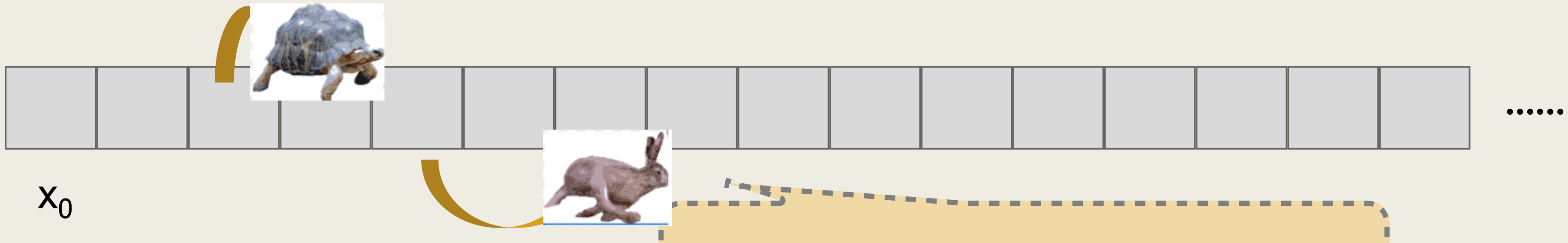
  $i = k \cdot r,\ \text{for some } k$ 的充分必要條件為 $x_i \equiv x_{2i}$

如何找出序列的循環點？

由 Floyd’s Cycle Finding 定理,
當兩 pointer 的值 mod p 相等時,
代表各自對應到序列循環點！

- 由定理, 可用

Tortoise pointer: 每次前進一格



Hare pointer: 每次前進兩格！

- <u>**定理** (Floyd’s Cycle Finding Algorithm)</u>

  對於任意的正整數 i,

  $i = k \cdot r,\ \text{for some } k$ 的充分必要條件為 $x_i \equiv x_{2i}$

需要的回合數

- 由於 X 為完美的亂數序列,
  每次 Tortoise pointer 與 Hare pointer 的值可視為隨機由 0~p-1 挑選出來
- 因此, 由 Birthday Paradox 可知,
  約有 ½ 的機率可在 $O(\sqrt{p})$ 個回合挑到同樣的數字。

Pollard’s Rho Algorithm

- 由前述, 方法的流程需要一個完美的亂數序列.
- 在此我們使用函數

$$g(x) = (x^2 + c) \mod n$$

作為 (偽)亂數產生器 Pseudorandom number generator。

雖不完美, 但堪用…