**Tuần 1:**

* Tóm tắt đề: Tìm số Lucas thứ n rồi module cho 109 + 7. Bắt đầu bằng **L1=2** và **L2=1**.
* Hướng giải:

1. Mức 1: Sinh tuyến tính từng số Lucas mà module đồng thời – Độ phức tạp: O(n).
2. Định nghĩa về dãy Lucas :

**Ln = Ln-1 + Ln-2**

1. Tính phân phối của module đối với phép nhân:

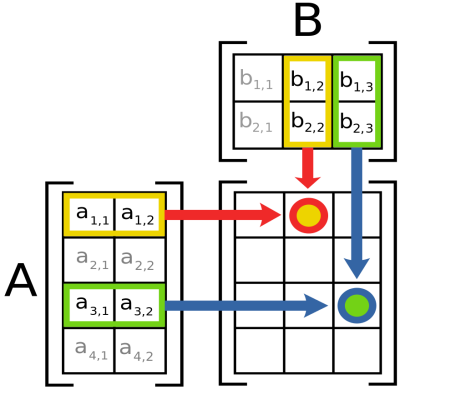
**( a + b ) % c = a % c + b % c**

Từ 1. và 2. ta có cách giải như sau: Ta sẽ chạy vòng for – với vòng thứ i để tính số Lucas thứ i mà module luôn:

**Li = ( Li-1 + Li-2 ) % mod**

[Code mẫu](Tuan1/Muc1.cpp)

1. Mức 2: Nhân ma trận + với tính lũy thừa bằng cách bình phương (exponentiation by squaring) - Độ phức tạp: O(min(log2n, MAX\_UNSIGNED\_LONG\_LONG)).
2. Định nghĩa về [nhân ma trận](https://vi.wikipedia.org/wiki/Ma_trận_(toán_học)) :



Cụ thể về cách lập trình nó bạn có thể tham khảo tại [đây](http://vnoi.info/wiki/algo/trick/matrix-multiplication).

Và ở bài Lucas này ta sẽ có như sau:

* 2 số bắt đầu có là **L1=2** và **L2=1** :
* Công thức quy hồi của Lucas là **Ln = Ln-1 + Ln-2** :
* Và khi đó sau mỗi lần nhân ma trận ta sẽ được cặp số Lucas tiếp theo:

Cặp số **L2** và **L3** :

Cặp số **L3** và **L4** :

Cặp số **L4** và **L5** :

Cặp số **L5** và **L6** :

Và thế là ta có dãy số Lucá từ 1 -> 6 là: 2, 1, 3, 4, 7, 11.

1. Tính [lũy thừa bằng cách bình phương](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_b%C3%ACnh_ph%C6%B0%C6%A1ng_v%C3%A0_nh%C3%A2n) ([exponentiation by squaring](https://en.wikipedia.org/wiki/Exponentiation_by_squaring)):

Cụ thể về cách lập trình nó bạn có thể tham khảo tại [đây](http://vnoi.info/wiki/translate/he/Number-Theory-3#thu%E1%BA%ADt-to%C3%A1n-chia-%C4%91%E1%BB%83-tr%E1%BB%8B).

Ví dụ:

Ta có: 1010 = 10102

Vậy ta có thể tính 210 = 28 x 22

Ta có: 1310 = 11012

Vậy ta có thể tính 210 = 28 x 24 x 21

Từ 1. và 2. ta có cách giải như sau: Tính nhanh bằng [lũy thừa bằng cách bình phương](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_b%C3%ACnh_ph%C6%B0%C6%A1ng_v%C3%A0_nh%C3%A2n) rồi lấy nó nhân với .

Ví dụ:

* Muốn có cặp số **L5** và **L6** ta phải nhân 4 lần ⬄ nhân với = =
* Sau đó nhân với

[Code mẫu](Tuan1/Muc2.cpp)

1. Mức 3: Xử lý số lớn – Độ phức tạp O(log2n)

Thích thì bạn có thể copy code Bignum vào hoặc chỉ cần code div – mod cho 2 là đủ

[Code mẫu](Tuan1/Muc3.cpp)