主題: Simulation

- What is simulation?
- Loop detection (I): Table lookup
- Loop detection (II): Search
- Speedup
- Event-point queue (not ready)
- 作業與自我挑戰

What is simulation?

Simulation: 按照一定的規則以程式逐步模擬並輸出最後結果

- 簡單題
- 整人題
- 有趣模擬題 √

Loop detection (I): Table lookup

- **範例: A. 202 Repeating Decimals**
  - 給一個分數 a / b，將這個分數化成循環小數的型式 (a, b are integers ≤ 3000)
- Example
  - 1/6 = 0.1(6) 括號內代表循環的部分
  - 5/7 = 0.(714285)
  - 300/31 = 9.(677419354838709)
  - 1/250 = 0.004(0)

Solution

- 照著除法計算的規則模擬
  - 利用整數除法和 mod 來將分數化為循環小數
- 問題: 如何偵測 "循環"?
- Table lookup: 分母是 b， mod b 只會出現 0 到 b – 1 的餘數，所以循環小數的位數最多只會有 b 位
  - 利用一個 table flag[b]，一開始全設為 – 1 (表尚未出現)
  - 當餘數為 i
    (a) 若 flag [i] = –1:
    餘數 i 第一次發生，將這個發生位置存入 flag [i]
    (b) 若 flag [i] ≠ –1:
    表 loop 發生
- Table lookup 只適用於可能的狀態數目在記憶體範圍內

Example

flag

0	1	2	3	4	5	6
–1	2	4	3	6	1	5

0.714 •••

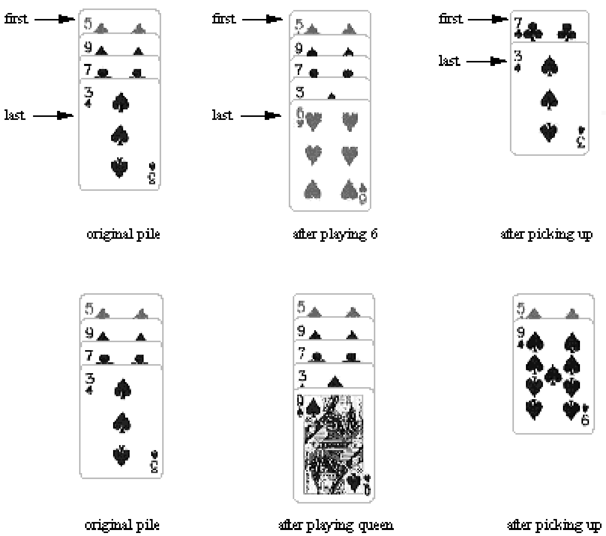
7) 5
0
5 0
4 9
1 0
7
3 0
2 8
2
•
•
•

5/7 = 0 ... 5 set flag[5]=1
50/7 = 7 ... 1 set flag[1]=2
10/7 = 1 ... 3 set flag[3]=3
30/7 = 4 ... 2 set flag[2]=4
20/7 = 2 ... 6 set flag[6]=5
60/7 = 8 ... 4 set flag[4]=6
40/7 = 5 ... 5 flag[5] =! –1

⇒ 0.(714285)

Loop detection (II): Search

- **範例: A.246 10-20-30**
  - 給一副牌，按照順序發成七疊
  - 在發了一張牌後，若有 (1)下面一張與最上面兩張 (2)下面兩張與最上面一張 (3)最上面連續三張加起來點數為 10、20 或 30，則把這三張牌收進手牌 (J, Q, K → 10)
  - 依 (1)(2)(3)順序檢查
  - 若某疊牌拿起三張後仍符合上述條件，則一直拿到不符合條件為止
  - 若有某一疊牌被完全拿光，則以後不在往這疊發牌
  - 請輸出最後結果: (1) win: 七疊牌都清空了
    (2) lose: 手牌空了
    (3) draw: 進入無窮迴圈



Method 1: Search in a data structure

- 可能的狀態 (states, 在此為所有可能牌面) 數目超過記憶體容量，不能使用 table lookup
  - How many states ???
- 利用一個 array 依續紀錄所有出現過的狀態
  - linear search: $O(m^2)$，m 是 loop 第一次產生的發牌次數

Example

S_1	S_2	S_3	S_4	S_5					

$S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5 \rightarrow S_3$ loop !!!

Method 1: (cont.)

insertion + search

- 利用一個 dynamic search tree 紀錄所有出現過的狀態
  - tree search: O(m lg m)

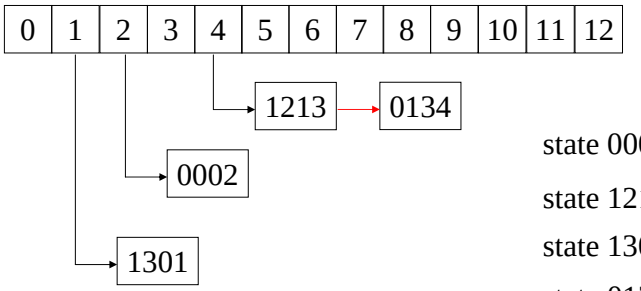
- 利用 hash table 來紀錄所有出現過的狀態

Hash

- 用一個 hash function h(s) 把 state s 轉換成一個範圍內的數字
- 每出現一個 state，就檢查已出現過且 hash value 相同的 states，看是否有重複出現
- 常用的 hash function: h(s) = g(s) mod p
  - g 為一個將 state 轉為 integer 的 function
  - p 為質數 ???
  - 以現有 storage 而言，取 $p \le 10^6$
  - collision resolved by chaining

Example: 0123 的所有4 位數組合

- $H(s_1s_2s_3s_4) = s_1\ s_2\ s_3\ s_4 \bmod 13$



state 0002, H(0002) = 2

state 1213, H(1213) = 4

state 1301, H(1301) = 1

state 0134, H(0134) = 4

Method 2: Search by simulation

- Assume that linear search is ok, which is of $O(m^2)$ time.
- Search by simulation
  - 每跑出一個新狀態，就從頭開始再跑一次看是否出現過同樣的狀態
  - 簡單、O(1) storage
  - $O(m^2)$，m 是 loop 第一次產生的發牌次數

Example: $S_1 \to S_2 \to S_3 \to S_4 \to S_5 \to S_3$

no no no no

$S_1 \to S_2 \to S_3 \to S_4 \to S_5 \to S_3$ loop !!!

$S_2 \to S_3 \to S_4$

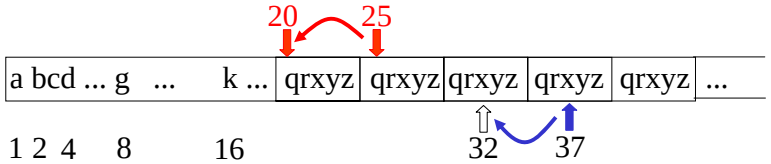
Pseudo code

```
state DispatchOneCard (state s);
  // 這 function 用來模擬從 state s 發一張牌後的情況

int check_draw (state current_state, int current_step)
  // 用模擬來檢查迴圈
{
state s;
    s = init_state;
    for (i = 1; i < current_step; i++){
          s = DispatchOneCard(s);
          if (s = = current_state) return (i); //a loop is found
    }
    return -1; //no loop is found
}
```


Method 3: exponential search

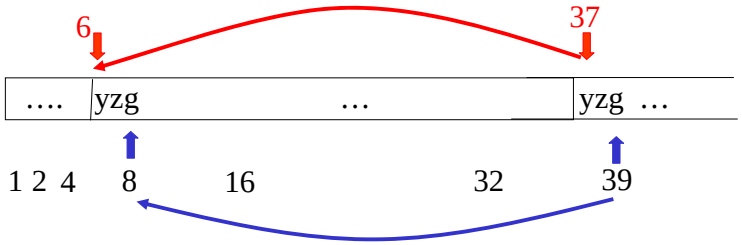
- 僅依續記錄 1, 2, 4, 8, 16, ... 出現的狀態



  - loop 在 17 與 32 之間，period = 5
  - 由 x = 17 與 y = 17 + 5 開始
    - x++; y++;
    - 檢查 state(x) = = state(y)?

- O(m lg m) time, O(lg m) storage

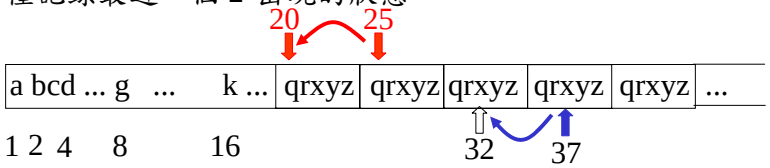
Method 3 (cont.)



- loop 在 5 與 8 之間，period = 31
- 由 x = 5 與 y = 5 + 31 開始
  - x++; y++;
  - 檢查 state(x) = = state(y)?

Further improvement

- 僅記錄最近一個 2^i 出現的狀態



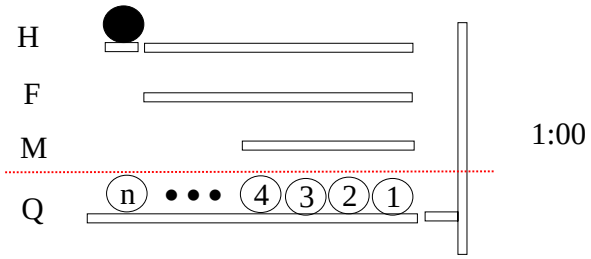
  - loop 在 1 與 32 之間，period = 5
  - 由 x = 1 與 y = 1 + 5 開始
    - x++; y++;
    - 檢查 state(x) = = state(y)?

- O(m) time, O(1) storage

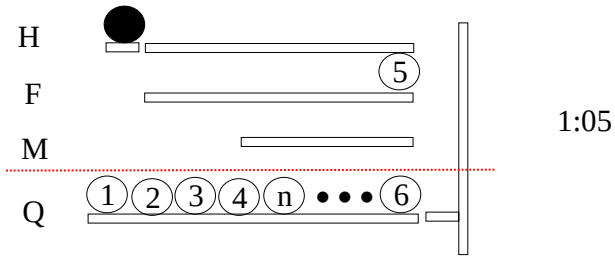
Speedup

- 很多模擬題目看起來不難，可是一步一步模擬會因為時間太長而不通過
  - 應事先評估一步一步模擬的時間是否會太長
  - 如果會的話要找出加速的方法

範例: A.239 Time and motion



- 一個時鐘是由下至上四個軌道 Q, M, F, H 所組成
- 開始時 Q 中有編號 1, 2, ..., n 的 n 個球, $27 \le n \le 7000$.



- M 表示 "分"，可放 4 個球，每一分鐘 Q 中會有一個球被提到 M 中，當第 5 顆想要進入時，第 5 顆會 "進位" 到 F 中，原來 4 顆會以相反順序掉回 Q 中
- F 表示 "5 分"，可放 11 個球，當第 12 顆想要進入時會 "進位" 到 H 中，原來 11 顆會以相反順序掉回 Q 中

- H 表示 "時"，開始時就有一顆額外固定的球，另外可放入 11 個球，當第 12 顆想要進入時會 "進位"，12 顆都以相反順序掉回 Q 中
- 輸出:
  - 每 12 個小時，所有球會回到 Q 中
  - 請問，多少天後 Q 中的球會回到原來 1, 2, ..., n 的順序
  - 輸出保證是 64-bit 整數

記得使用 64-bit 整數

Solution

- 簡單模擬題？
  - 逐步模擬不會過: $2^{64} \times 24 \times 60 \approx 10^{22}$
- 加速:
  - 先模擬 12 小時得到一個 permutation $\pi_{1/2}$
  - 利用 $\pi_{1/2}$ 得到一天的 permutation π_1
  - 利用 π_1 得到之後每一天的 permutation
  - $O(2^{64}) \approx 10^{19}$ (還是不會過)

- 進一步加速:
  - 利用 π 得到經過 k 天後第一棵球會回到第一個位置的 permutation π_k
  - 利用 π_k 得到每經過 k 天後的 permutation $\pi_{k \times i}$ 直到所有球回到原位置
  - $O(2^{64} / k)$ (不知道會不會過???)
- 進進一步加速 ???

Another Solution

- 事先計算出 27 到 7000 的答案 (偷跑!)
- code length ≤ 40 K, 只能存後面約 4K 答案
  - for small n: 直接計算
  - for large n: 查表回答: O(1)

作業與自我挑戰

- 作業
  - 練習題
    - A.246 10-20-30
      http://uva.onlinejudge.org/external/2/246.html
  - 挑戰題
    - A.305 Joseph (m would be 10^6)
      http://uva.onlinejudge.org/external/3/305.html
- 自我挑戰
  - A.296 Safebreaker
  - AF.2004. A Carl the Ant (no online judge)
  - A.180 Eeny Meeny (Very difficult!)
    http://uva.onlinejudge.org/external/1/180.html

Remark: A246 is much harder than A.305 !!

- 其它有趣的題目
  - A.10015 Joseph's Cousin (similar to A.305)
  - A.602 What day is it?
  - A.10500 Robot maps
  - A.131 The psychic poker player
  - A.457 Linear cellular automata
  - A.402 M*A*S*H
    - http://uva.onlinejudge.org/external/4/402.html
  - A.10315 Poker hands
    - http://uva.onlinejudge.org/external/103/10315.html