1. 了解動態陣列的原理之後，請推導新增元素時的複雜度。(malloc/new 的時間可以不考慮)

   (a) 請證明一個動態陣列若從初始狀態開始進行了 n 次的新增元素操作，總時間複雜度為 $O(n)$，空間複雜度也是 $O(n)$。

解答

證明分成兩步驟：

(1) 先考慮 $n = 2^k + 1$ 的情況。在新增第 n 個元素之前，陣列的 `size` $=$ `capacity` $= 2^k$，因此為了新增第 n 個元素，需要先擴張陣列。將當前花在「將資料從舊陣列複製到新陣列」的操作數加總，一共有

$$1 + 2 + 4 + 8 + \cdots + 2^k = 2^{k+1} - 1 = 2n - 3$$

而花在「將元素放到陣列尾端」的操作次數共恰好 n 次，因此總操作數為

$$(2n - 3) + n = 3n - 3 \in O(n)$$

使用到的空間為 2^{k+1}，可以得到

$$2^{k+1} = 2n - 2 \in O(n)$$

因此，時間複雜度和空間複雜度皆為 $O(n)$

(2) 接著考慮 $n \neq 2^k + 1$ 的情況，此時因為陣列在新增第 n 個元素之前不是滿的，所以只需要一個操作，將元素放到陣列尾端就好了。令 m 為滿足 $m = 2^k + 1, m < n$ 的最大可能值，也就是

$$m = 2^k + 1 < n < 2^{k+1} + 1$$

則新增 n 個元素就是「新增 m 個元素，之後再新增 $n - m$ 個元素」，而且最後的 $n - m$ 個新增操作不會改變陣列大小，操作數為

$$(3m - 3) + (n - m) = n + 2m - 3 < 3n - 3 \in O(n)$$

使用的空間為 2^{k+1}，可以得到

$$2^{k+1} < 2n - 2 \in O(n)$$

時間複雜度和空間複雜度也都是 $O(n)$

由 (1),(2)，證明了對所有 n，從初始狀態新增 n 個元素的時間複雜度和空間複雜度都是 $O(n)$。

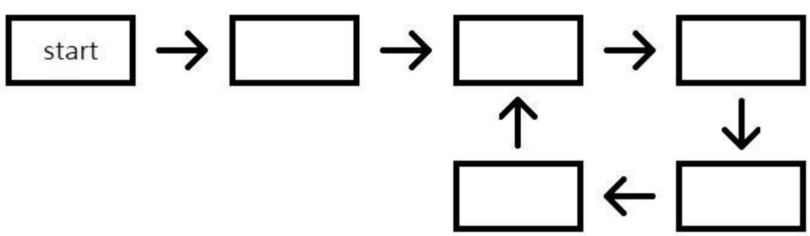
   (b) 請證明一個動態陣列若從初始狀態開始進行了 n 次的新增元素操作，但擴張陣

列時，大小不是增加到 `capacity` $\times 2$，而是 `capacity` $+1$，則總時間複雜度為 $O(n^2)$，空間複雜度是 $O(n)$。

> 解答
>
> 每次新增第 k 個元素時，都要宣告一塊記憶體，並將原本的 $k-1$ 個元素搬過去，再將第 k 個元素放到陣列尾端。因此，新增 n 個元素的時間複雜度為
> $$1+2+3+\cdots+n=\frac{n(n+1)}{2}\in O(n^2)$$
> 使用的陣列大小恰好為 n，空間複雜度為 $O(n)$。

2. 有 n 個節點，每個節點內只存著它的下一個節點。已知從 start 開始，每次走到當前節點的下一個節點，最終可以走過所有的節點，並且進入一個環。也就是說，這些節點形成的形狀就像字母 ρ 一樣。

```c
struct Node {
    struct Node* next;
};
struct Node* start;
```



請在不知道 n 的確切數值之下，找出這個環的長度。(意即，不能有如「從 start 開始走 n 步」的敘述，因為你並不知道 n 是多少)

(a) 請給出時間複雜度 $O(n)$，不限制空間大小的做法。(額外空間宣告在 Node 裡面或外面皆可)

> 解答
>
> 在每個節點內記錄一個變數 visit 表示是否走過，若 visit=True 表示走過，否則表示還沒走過，且初始值為 False。從起點開始往後走，每走到一個節點就將該點的 visit 設為 True，則在第 n 步時會發現下一個節點已經被拜訪過了。記錄該點，並繼續走且記錄走的步數，在 n 步之內一定會回到自己，如此就可以得到環的長度。此演算法在 $2n$ 步之內一定會停止，時間複雜度為 $O(n)$；在每個節點裡面加了一個變數 visit，空間複雜度為 $O(n)$。
>
> 很多同學的解法是在每個節點內紀錄走到的編號，需注意這樣子的額外空間複雜度會是 $O(n\log n)$，因為需要 n 個能表達數值 0 到 n 的變數。(當

然，這個解法還是滿足這題的要求）

(b) 給定 k，且假設環和起點的距離小於 k，請描述如何判斷環的大小是否小於 k，如果小於 k 的話還要給出環的大小。另外，限制時間複雜度為 $O(k)$，（額外空間的）空間複雜度為 $O(1)$。(環和起點的距離為 d 表示從起點開始至少走 d 步之後會進入環)

解答

一開始先從起點開始走 k 步並記錄該點，由題目假設，該點一定在環內。接著繼續走 k 步並記錄步數，若在 k 步內走回自己則表示環的大小小於 k，走的步數就是環的大小，否則表示環的長度大於 k。

(c) 令 k 從 1 開始，檢查 (b) 中提到的判斷是否成立，如果成立則可以得到環的長度，否則將 k 變成 2 倍，繼續判斷直到得到環的長度。請證明這個演算法一定會停止，並證明該演算法的時間複雜度為 $O(n)$，（額外空間的）空間複雜度為 $O(1)$。

解答

此演算法停止的條件為「起點到環的距離小於 k」且「環的大小小於 k」，當 k 大於 n 時，這兩個條件一定成立。每次的檢查為 $O(k)$，而 k 在倍增 $\lfloor \log n + 1 \rfloor$ 次之後一定會大於 n，因此這個演算法一定會停止。時間複雜度為 $1 + 2 + 4 + \cdots + 2^m = 2^{m+1} - 1 \in O(2^m)$，其中 2^m 為不小於 n 的最小 2 的冪次，因此 $2^m < 2n$，得到時間複雜度為 $O(2^m) \in O(2n) = O(n)$。演算法中只需要記錄走了 k 步之後所到達的 Node，以及一個記錄步數的變數，因此空間複雜度為 $O(1)$。

Note: (b) 跟 (c) 中不允許修改 Node 中的資料。
Hint: 縱使不能修改 Node 本身，但你可以記錄指標的值來記下某一個走過的點。

3. 以下為 2015 年資訊之芽入芽考的其中一題。

Description

由於円円嚴重缺乏運動，他的好朋友們幫他設計了一個好玩的跳格子遊戲，讓他在娛樂之餘還能順便活動身體，希望能讓他再長高一點 (雖然應該希望渺茫了)。

這個跳格子的遊戲是這樣的：一開始在地上畫出一條 $1 \times N$ 的方格圖，並且大小依序標上編號 $0 \sim (N-1)$。接著在每個格子裡面寫上一個數字 a_i，表示當円円跳到第 i 格之後，下一次就要跳到第 a_i 格。由於在格子內轉身不太方便，因此円円的好朋友們十分好心，填上數字時一定保證 $a_i \geq i$，也就是說，

円円只會往前跳或是待在原地，而不會往後跳。

現在已知円円一開始在第 x 格，請問他跳了 k 步之後會停在哪格呢？

Input

第一行為兩個正整數 N, Q，表示共有 N 個格子，且有 Q 次詢問。
第二行為 N 個整數，依序為 $a_0, a_1, \cdots, a_{N-1}$。
接著 Q 行，每行為兩個正整數 x, k，表示円円一開始在第 x 個格子，並且接著要跳 k 步。

- $0 \le x, k \le (N-1)$

(a) 如果對於每筆詢問 (x, k)，都一步一步的跳 (所謂的一步一腳印) 得到最後的答案，時間複雜度為何？(請以 N, Q 表示)

解答

每筆詢問都要花 $O(k) = O(N)$ 的時間回答，共有 Q 筆詢問，因此時間複雜度為 $O(QN)$。

(b) 若以 $a[i]$ 表示從第 i 格跳 1 步之後所到達的格子編號 (即題目中的 a_i)，請以 (a, i) 表示從第 i 格跳 2 步之後所到達的格子編號。

解答

$a[a[i]]$

(c) 承上題，若以 $b[i][j]$ 表示從第 i 格跳 2^j 步之後所到達的格子編號，請以 $(b, i, j-1)$ 表示 $b[i][j]$ 在 $j > 0$ 時的遞迴關係。
（解答的長度為 $O(1)$，例如：不能寫重複哪個式子 2^j 遍）

$$b[i][j] = \begin{cases} a[i] & \text{, if } j = 0 \\ ??? & \text{, if } j > 0 \end{cases}$$

解答

$b[b[i][j-1]][j-1]$

(d) 承上題，假設已經建好了 $b[i][j]$ 陣列，給定 x, k，請在 $O(\log k)$(或是 $O(\log N)$) 的時間內得到從 x 開始跳 k 步之後所在的格子編號。

解答

將 k 以二進位表示，則可以將 k 分解成最多 $\lceil \log_2 k \rceil$ 個 2 的冪次的和 (如 $21 = 2^4 + 2^2 + 2^0$)。因為 b 陣列已經建好了，所以對於每個分解出來的冪次 2^j，都可以用查表的方式在 $O(1)$ 時間得到跳 2^j 步之後會到哪裡，跳 k 步就只需要 $O(\log k)$ 的時間就可以完成。

(e) 承上題，請問上述演算法的時間複雜度為何？(請以 N, Q 表示)

解答

每筆詢問需要花 $O(\log k) = O(\log N)$ 時間，共有 Q 筆詢問，因此時間複雜度為 $O(Q \log N)$。