# LINKED LISTS

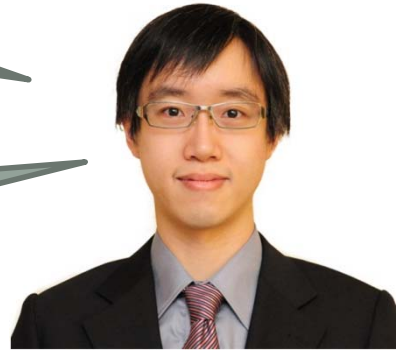
Michael Tsai

2010/10/15

下星期開始教室改至R101

作業二下星期四due

請多利用下周二三四的office hour



# 今日菜單

- 上次講得不好的Sparse Matrix Transpose方法
- 上次未講完的迷宮問題
- Linked List
- 應用
- 多項式
- 相等集合
- 很鬆的矩陣

# 來一個有趣的迷宮問題吧

- 迷宮: 0是路, 1是牆壁. 每一部可以往上、下、左、右和四個斜角方向走一步.
- 問題: 怎麼找出一條路從(0,0)走到(7,7)? (不一定要最短)
- 提示: 跟stack是朋友

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

# 走迷宮的時候，人要怎麼走?

- 最重要的時候, 是碰到岔路的時候
- 先記起來, 選其中一條走走看
- 如果碰壁了 (一直沒有走到終點), 就退回最後一次碰到的岔路, 換另外一條岔路
- 關鍵字: 最後一次碰到的岔路 (不是最先碰到的)
- 所以是先進後出→使用stack
- 關鍵字: 換另外一條岔路
- 要記得上一次走過哪一條路了

# 一些細節

- Q: 那麼, stack裡面要存什麼呢?

- A:
- "岔路"的地方的
- 座標, 也就是(row, col)
- 試過那些岔路了(試到八個方向的哪個方向了)

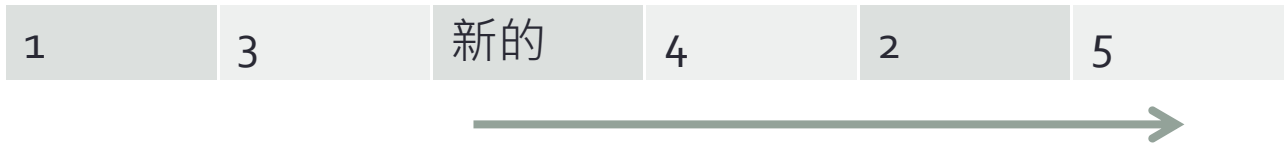
- Q: 要怎麼預防繞圈圈? (永遠出不來)
- A: 標示所有已經走過的地方, 走過就不用再走了
- <注意> 這是因為不用找"最短"的路

# 讓我們來寫algorithm

- 把(row_start, col_start, 第一個方向) 放入stack
- while(stack不是空的) {
  - 從stack拿出一組岔路點(row, col, dir)
    - while(還有別的dir還沒試) {
      - 將(row, col)往dir方向移動, 得到(row_n, col_n, dir).
      - 如果(row_n, col_n)就是終點, 則結束
      - 如果(row_n, col_n)不是牆壁且沒有來過 {
        - 標示(row_n, col_n)來過了
        - 把(row, col, dir的下一個方向)放入stack
        - row=row_n; col=col_n; dir=第一個方向;
      - }
    - }
- }

# 大家來吐Array的槽

- Array有什麼不好?
- 插入新element




- 刪除原本的element


- Time complexity= O(??)

# 新朋友: Linked List

- 要怎麼讓資料
- 1. 可以隨便亂排
- 2. 但是我們仍然知道他的順序?

- 答案:
- 資料亂排
- 但是, 另外存"下一個是誰"

| index | [0] | [1] | [2] | [3] | [4] |
|---|---|---|---|---|---|
| 資料 | 子毅 | 林潔 | 立中 | 婷尹 | 予迪 |
| 下一個是誰 | 4 | 0 | 1 | -1 | 3 |
| 開始 | 1 | | | | |

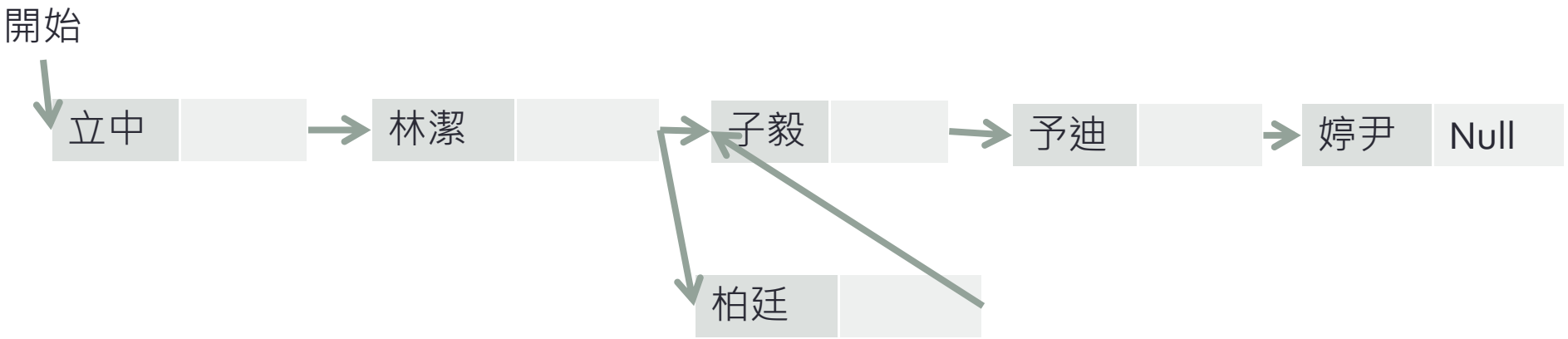
# 概念上，應該是長這樣


真正的樣子:

| index | [0] | [1] | [2] | [3] | [4] |
|---|---|---|---|---|---|
| 資料 | 子毅 | 林潔 | 立中 | 婷尹 | 予迪 |
| 下一個是誰 | 4 | 0 | 1 | -1 | 3 |

| 開始 | 2 |
|---|---|

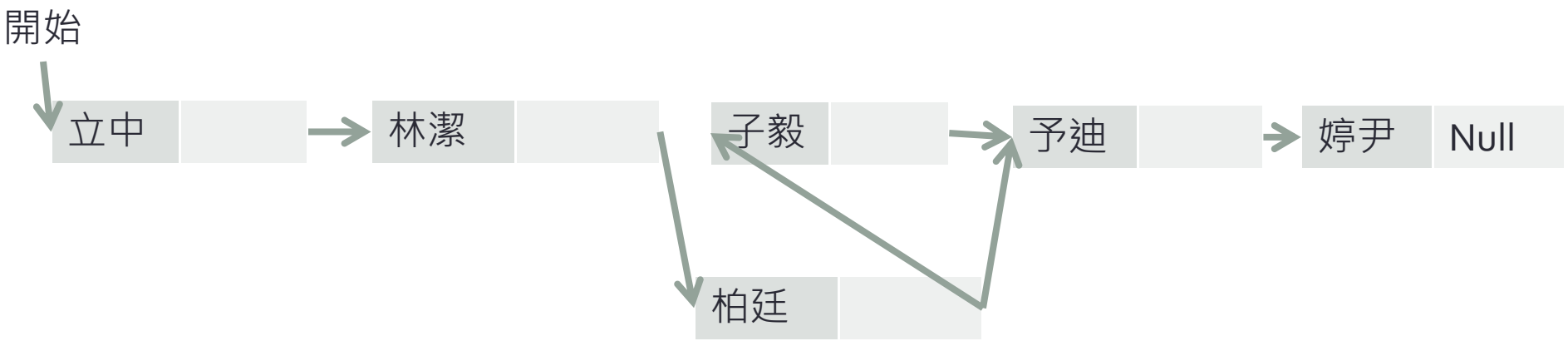
# 增加一個人?




| index | [0] | [1] | [2] | [3] | [4] | [5] |
|---|---|---|---|---|---|---|
| 資料 | 子毅 | 林潔 | 立中 | 婷尹 | 予迪 | 柏廷 |
| 下一個是誰 | 4 | 5 | 1 | -1 | 3 | 0 |

| 開始 | 2 |
|---|---|

# 刪掉一個人




| index | [0] | [1] | [2] | [3] | [4] | [5] |
|---|---|---|---|---|---|---|
| 資料 | 子毅 | 林潔 | 立中 | 婷尹 | 予迪 | 柏廷 |
| 下一個是誰 | 4 | 5 | 1 | -1 | 3 | 4 |

| 開始 | 2 |
|---|---|

# 來看一些code (用malloc/free)

- 怎麼宣告一個node的struct?

```
struct listNode {
  int data;
  struct listNode *link;
};
```

- 怎麼拿一個新的node?
- `listNode *new;`
- `new=(struct listNode*)malloc(sizeof(struct listNode));`

# 來看一些code

- new是指標, 指到一個struct listNode的變數.
- 那如果要拿這個變數裡面的data呢?
- 可以這樣寫:
- (*new).data
- 或者,
- new->data

- 要拿link呢?
- (*new).link
- 或者,
- new->link

# 來看一些code


- 假設start指到第一個struct listNode

- 那麼我要拿到551的下一個listNode的位址怎麼寫?
- start->link->link (連續技)

# 製造兩個node


- `struct listNode* start, *tmp;`

- `tmp=(struct listNode*)malloc(sizeof(struct listNode));`
- `tmp->data=551;`
- `tmp->link=NULL;`
- `start=tmp;`
- `tmp=(struct listNode*)malloc(sizeof(struct listNode));`
- `tmp->data=342;`
- `tmp->next=start;`
- `start=tmp;`

# 插入一個新node在某node後面

- `struct listNode *x; //`指到要插入的node的位置
- `struct listNode *new;`
- `new=(struct listNode*)malloc(sizeof(struct listNode));`
- `new->data=123;`
- 下一步呢? 先處理`new->link`還是`x->link`?
- `new->link=x->link;`
- `x->link=new;`

# 刪除某一個node

- `struct listNode *x; //`指到要刪除的node的位置
- `struct listNode *trail; //`指到x的前一個node的位置

- 分兩種狀況處理: x是頭, 還有x不是頭
- `if (trail)  //`x不是第一個node
  - `trail->link=x->link;`
- `else`
  - `start=x->link;`
- `free(x);`

# 印出整個list

- `struct listNode *tmp;`
- `for(tmp=start; tmp; tmp=tmp->next)`
  - `printf("%d ", start->data);`

# Stack & Queue

- 上次最後提到, 如果是一塊記憶體要放很多stack或queue
- 就很難做到很efficient
- 例如如果某一stack滿了, 就要把一堆資料往後擠
- 就不是O(1)了 T_T

- 解決: 跟Linked List當朋友

# Stack

- 要怎麼拿來當stack呢? (想想怎麼做主要的operation)
- push & pop
- 請一位同學上來講解☺

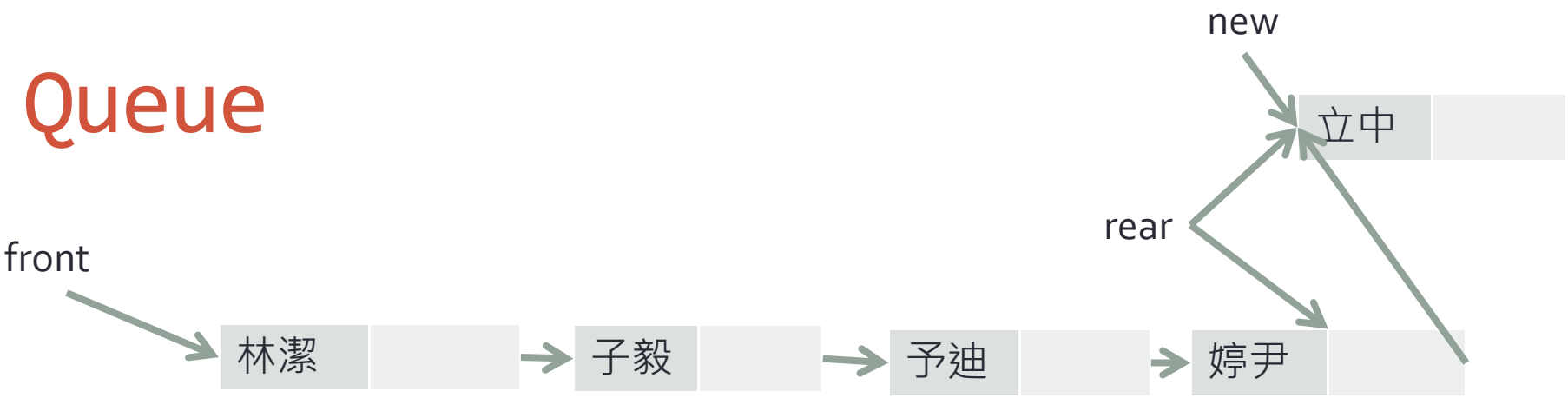

- 例: push("立中")
- start當作stack top
- 怎麼寫code?
- 那pop呢?

# Queue

- 類似stack的作法
- 不過頭尾都要有一個指標
- 從頭拿, 從尾放


- 怎麼拿? (delete)
- `struct listNode* tmp;`
- `tmp=front;`
- `front=front->link;`
- `tmp_data=tmp->data;`
- `free(tmp);`
- `return tmp_data;`

# Queue




- 那怎麼放?
- 假設new是指到新的node
- rear->link=new;
- new->link=NULL;
- rear=new;

# <動腦時間>

- 有沒有什麼是array比linked list好的?
- 什麼時候用array?
- 什麼時候用linked list?

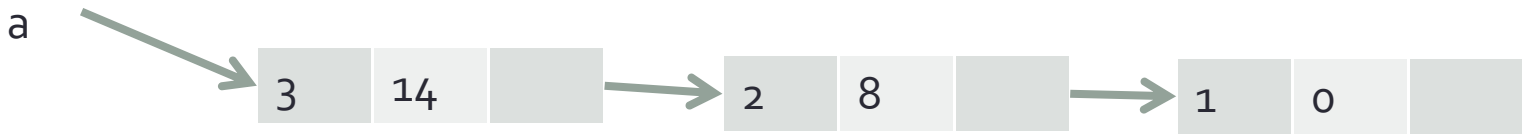
# 多項式

- 陰魂不散多項式

- 怎麼用linked list表示呢?

```
struct polyNode {
      int coef;
      int expon;
      struct polyNode *link;
};
```

例: a $= 3x^{14} + 2x^{8} + 1$

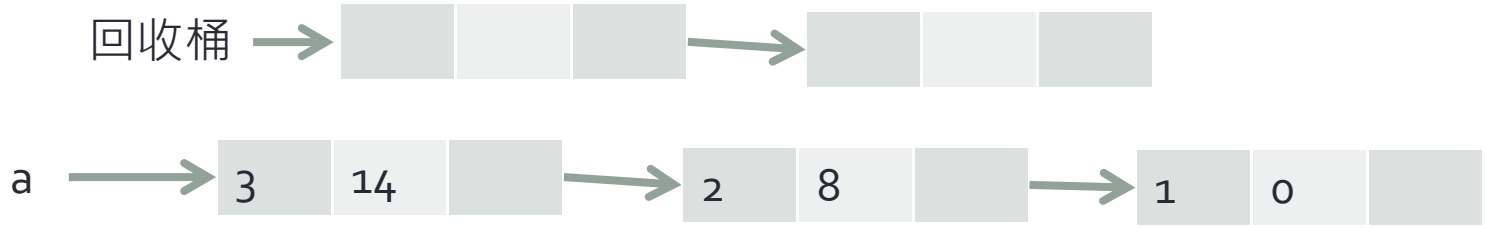


# 那麼，多項式加法?

- 挑戰! 直接看code!
- Time complexity = O(??)
- 那之前用array表示, 有什麼不好?
- 答案: 用完的比較好回收
- 不回收會怎樣?
- 怎麼回收? 請一位同學來解釋☺

# 回收很慢

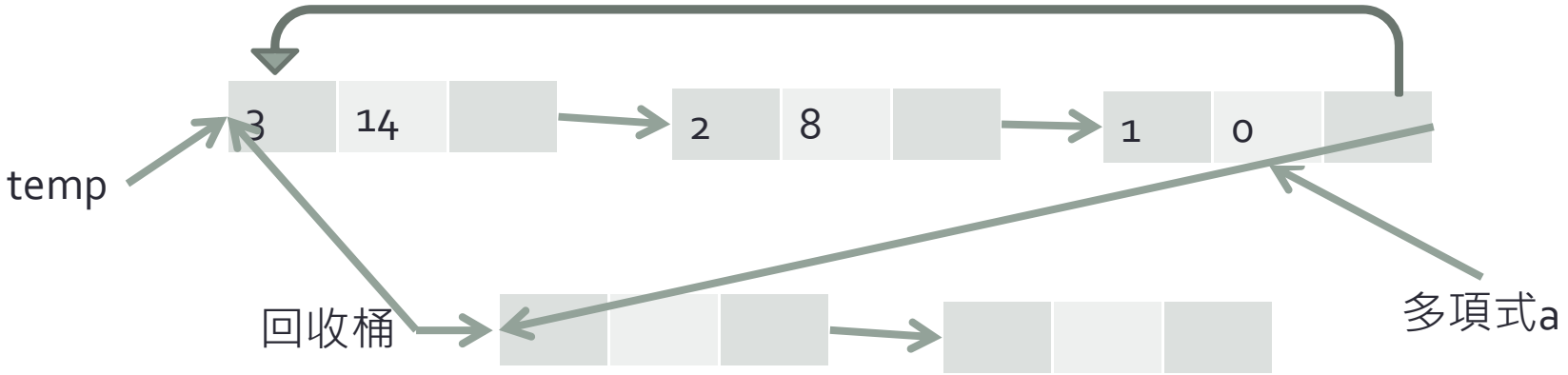
- 有幾項就要O(幾項)的時間
- 懶人方法: 丟到一個"回收桶", 之後需要的時候再撿出來
- 希望丟=O(1), 而且撿=O(1)
- 怎麼做?




- 關鍵:找尾巴很慢. (不是O(1))

# Circular List

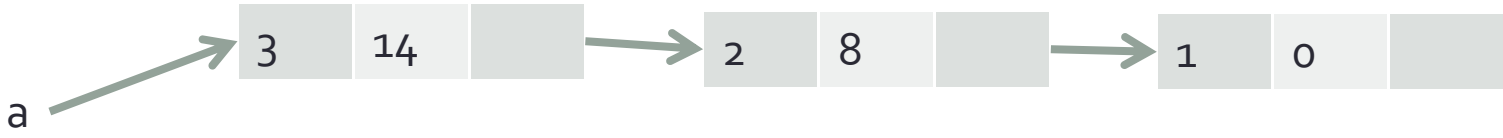
- “開始”的那個箭頭, 指在尾巴
- 最後一個node指回開頭
- 有什麼好處?
- 丟進回收桶=O(1) !!



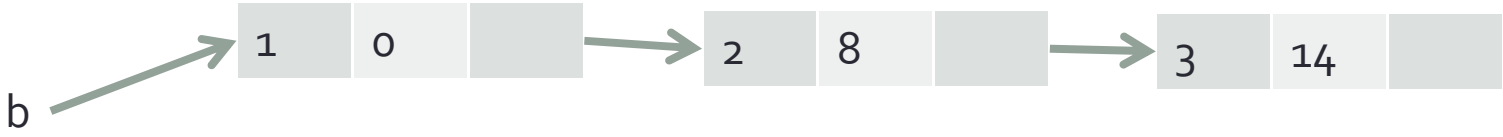

- 請看program 4.15 on p.169 (用circular list來做多項式加法)

# 來練習一些動作

- 反過來: 把




- 變成



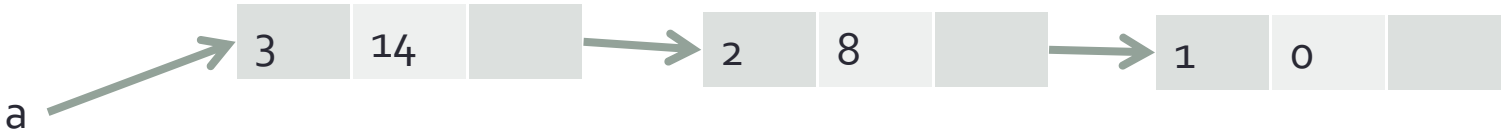

- 怎麼弄?
- 請同學上來解釋☺+ 用白板寫程式

# 練習題

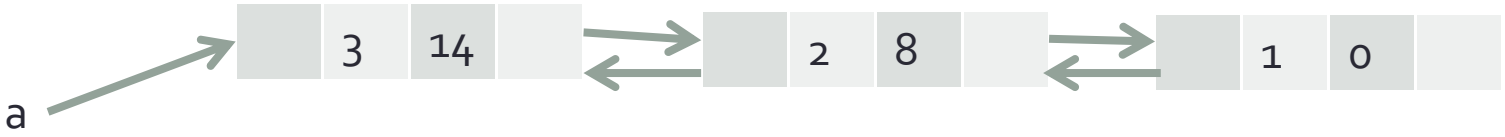
- 把list b接到list a後面 (chain)
- 把一個新的node加到circular list的最前面
- 找出長度(circular list)

# Singly v.s. doubly linked list

Singly linked list:




Doubly linked list:




- 有單就有雙
- 什麼時候需要用雙?
- Singly linked list只能往後, 不能往前 (要從最前面開始重新找)
- Doubly linked list用在常常需要"倒帶"的時候
- Doubly circular linked list???? (請一位同學來畫☺)

# 暫停!

- 來回想一下我們學了哪些種類的linked list

- Singly linked list
  - Circular
  - Non-circular (chain)
- Doubly linked list
  - Circular
  - Non-circular (chain)

# 應用題

- 相等集合 (Equivalence Classes)
- 很鬆的矩陣(又來了)

# 相等集合

- 1. $x \equiv x$
- 2. $x \equiv y$則$y \equiv x$
- 3. $x \equiv y$且$y \equiv z$, 則$x \equiv z$

- 問題: 給一批 $i \equiv j$的關係, 請列出所有不同集合的成員.
- 舉例:
- $0 \equiv 4,3 \equiv 1,6 \equiv 10,8 \equiv 9,7 \equiv 4,6 \equiv 8,3 \equiv 5,2 \equiv 11,11 \equiv 0$
- 則可以分成
- $\{0,2,4,7,11\}$; $\{1,3,5\}$; $\{6,8,9,10\}$

# 怎麼解呢?

- 幾個提示:
- 1. 用linked list的array記住每個數跟其他那些數"相等"
- 2. 印出來的時候記得那些數已經印過了(不要重複)
- 3. 利用stack的概念

- 黑板解釋 (做動畫會很困難Orz) program 4.22

- O(??)

# 很鬆的矩陣

- 這個給同學自行閱讀
- Section 4.7

- 到目前為止最"複雜"的資料結構

- 但是! 其實沒有很深的學問.

- <動腦時間> (自己回家想) 用這麼複雜的資料結構, 什麼時候適用?
- Space complexity = O(??)

# 周末愉快

- 下周開始可能教室更改到R101
- 作業二下周四5pm前交
- 有問題的話請多利用下周二三四的office hour
- 作業加分題:
- (1) 從開學到現在, 覺得哪個課題講得最好? 哪個課題最不清楚, 希望老師再講一次? (3%)
- (2) 對於老師到目前為止上課的方式, 提出一個具體的建議. (3%)