

QUEUE & STACK

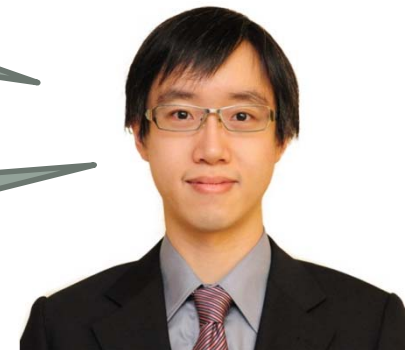
Michael Tsai

2010/10/8

作業一解答下星期一也會上線。

作業二今天會上網站。

各位同學寫作業一辛苦了。
請發表對這次作業的感想。



今日菜單

- 複雜的KMP例子
- 上次沒有講完的sparse matrix
- Stack
- Queue
- 計算機
- 迷宮



Failure function

- 另外一個定義(recursive)
- $f(j) = \begin{cases} -1, & \text{if } j = 0 \\ f^m(j-1) + 1 & \text{where } m \text{ is the least int } k \text{ for which } p_{f^k(j-1)+1} = p_j \\ -1, & \text{if there is no } k \text{ satisfying the above} \end{cases}$
- $f^1(j) = f(j)$ and $f^m(j) = f(f^{m-1}(j))$

	a	b	c	a	b	d	a	b	c	a	b	c
f												

	a	b	c	a	b	d	a	b	c	a	b	c
f	-1	-1	-1	0	1	-1	0	1	2	3	4	2

朋友三：矩陣

- $$\begin{bmatrix} -27 & 3 & 4 \\ 6 & 82 & -2 \\ 109 & -64 & 11 \end{bmatrix}$$

- 怎麼存起來?
- 用二維陣列, `a[row][col]` 來存
- 加法乘法都很容易

很鬆的矩陣

$$\bullet \begin{bmatrix} 15 & & & 22 & & -15 \\ & 11 & 3 & & & \\ & & & -6 & & \\ 91 & & & & & \\ & & 28 & & & \end{bmatrix}$$

- 浪費很多空間在存T_T
- 有沒有什麼節省空間的方法可以存?

很鬆的矩陣

$$\bullet \begin{bmatrix} 15 & & 22 & & -15 \\ & 11 & 3 & & \\ & & & -6 & \\ 91 & & & & \\ & & 28 & & \end{bmatrix}$$

- 存法:

index	0	1	2	3	4	5	6	7	8
Row	6	0	0	0	1	1	2	4	5
Col	6	0	3	5	1	2	3	0	2
value	8	15	22	-15	11	3	-6	91	28

空間省了，但是...

- 如果要做transpose...

$$\begin{bmatrix} 15 & & 22 & -15 \\ & 11 & 3 & \\ & & -6 & \\ 91 & & & \\ & & & 28 \end{bmatrix} \longrightarrow \begin{bmatrix} 15 & & & 91 \\ & 11 & & \\ & 3 & & 28 \\ 22 & & -6 & \\ -15 & & & \end{bmatrix}$$

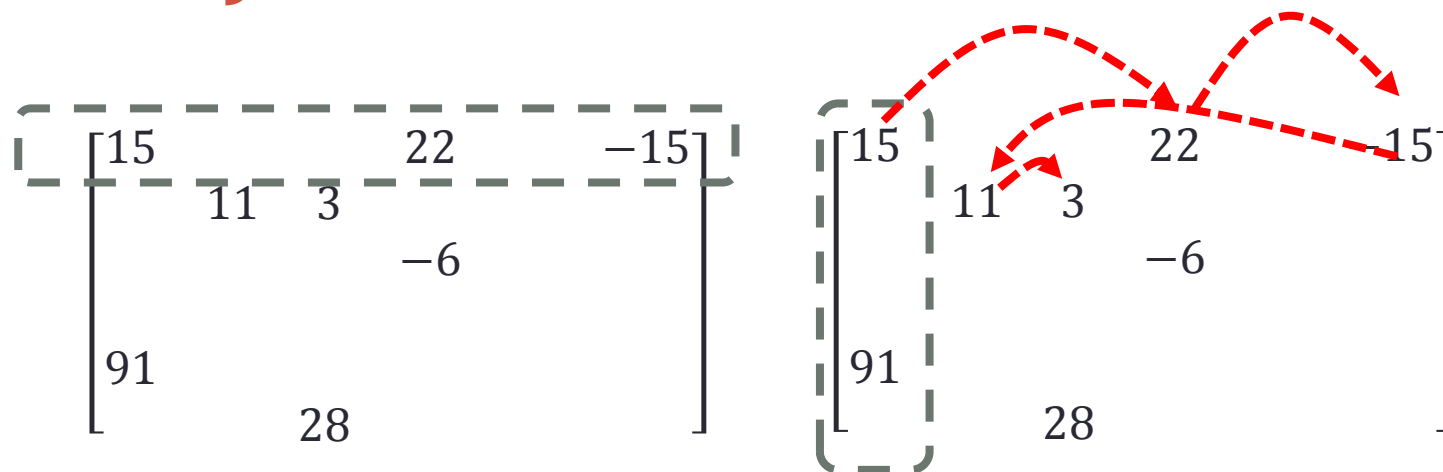
- 直接把row & col的值交換?
- 這樣就沒有照順序了 (以row為主)

index	0	1	2	3	4	5	6	7	8
Row	6	0	3	5	1	2	3	0	2
Col	6	0	0	0	1	1	2	4	5
value	8	15	22	-15	11	3	-6	91	28

<動腦時間>

- 怎麼用 $O(\text{columns} + \text{elements})$ 的時間做完transpose?
- 當不鬆的時候, 就變成 $O(\text{columns} + \text{elements})$
 $= O(\text{columns} + \text{columns} * \text{rows}) = O(\text{columns} * \text{rows})$
- 怎麼做? 當做回家習題☺ 請看program 2.9 (p.78)

那麼，矩陣乘法呢？



- 不順 (希望同樣column的會放在一起)
- 所以可以先Transpose

實際乘乘看

$$\begin{array}{cccc}
 \boxed{15} & & 22 & -15 \\
 & 11 & 3 & \\
 & & -6 & \\
 91 & & & \\
 & & 28 &
 \end{array}
 \quad
 \begin{array}{cccc}
 \boxed{15} & 11 & 3 & 22 & -15 \\
 & & & -6 & \\
 & & & & \\
 \boxed{91} & & & & \\
 & & & 28 &
 \end{array}$$

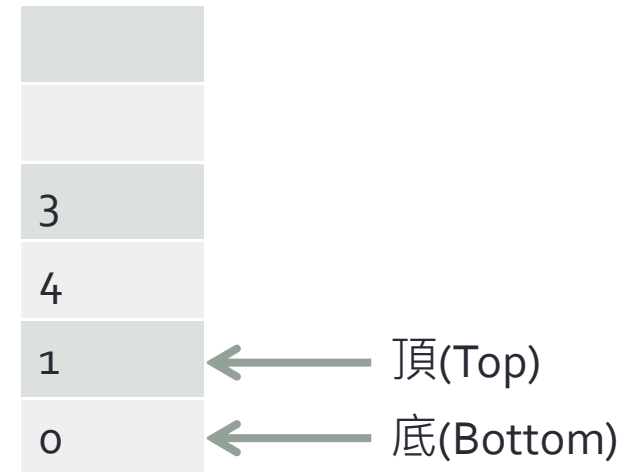
- 類似之前的“鬆多項式”加法
- 比較兩邊的row & col數
- $a < b \rightarrow a$ 移到下一項
- $a == b \rightarrow$ 可以相乘, 乘起來以後的值存起來
- $a > b \rightarrow b$ 移到下一項

看看程式

- program 2.10和2.11 (p.81)
- 這個程式比較複雜, 如果看不懂回家請再多花時間 😊
- Time complexity = $O(??)$
- 複雜度也很複雜, 請也多花時間

Stack

- Stack是蝦密碗糕?
- 一串有“順序”的列表
- 又叫Ordered List
- 有什麼特別的地方?
- 永遠從頂部拿與放
- 拿: Pop
- 放: Push
- 看一個例子
- 所以先放進去的, 會??出來?

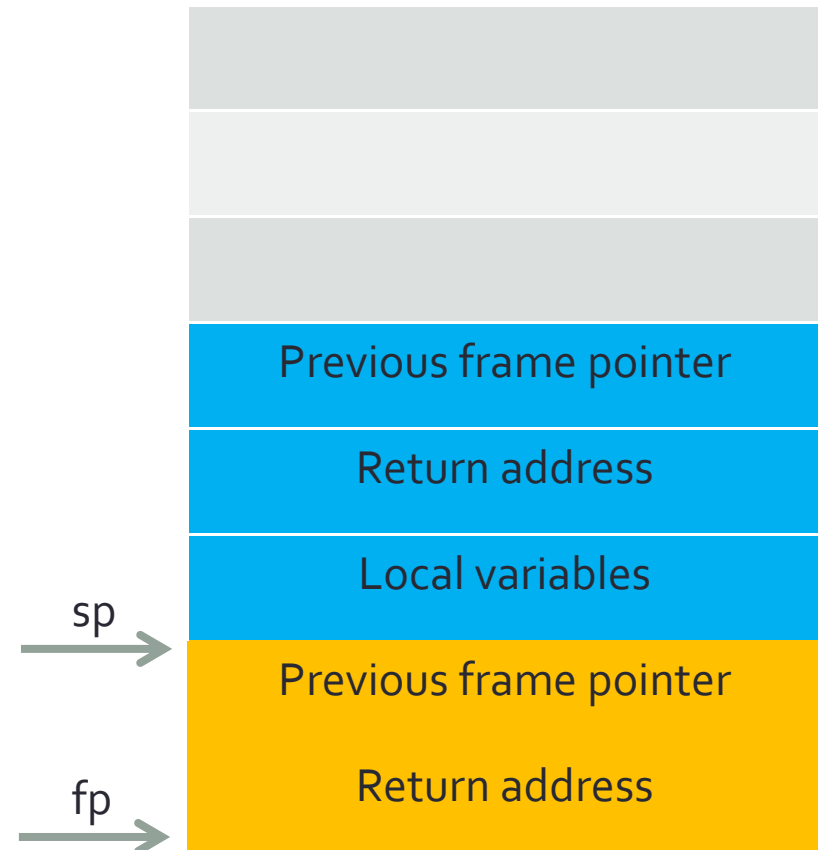


Stack

- 所以是一個先進後出(First-In-Last-Out)的資料結構
- 那麼, 支援那些動作呢? 各動作又需要什麼參數?
- 請同學們列舉:
 - 做一個Stack
 - 放(Push)
 - 拿(Pop)
 - 是不是空的?
 - 是不是滿了?

例子一：系統堆疊

- 被程式拿來儲存呼叫function的相關資訊
- 放什麼? 叫做activation record或stack frame
 - return address: 呼叫function的下一個指令應該去的地方
 - previous frame pointer: 指到前一個stack frame在stack裡面的位置(底)
 - local variables
- 看一個例子.
- 適不適用recursive call?
- Stack overflow?



怎麼implement一個stack?

- 請同學上來舉例說明☺
- 提示: Array這個好朋友又出現了
- 除了array以外, 還需要哪些東西?
 - 紀錄頂部的index
- 怎麼implement四個operations?
 - Push
 - Pop
 - 是不是空?
 - 是不是滿?

滿了怎麼辦?

- `int stack[MAX_SIZE];`
- 如果裡面有超過MAX_SIZE元素, 就滿了.T_T
- 如果一開始我用`stack=(int*)malloc(sizeof(int)*MAX_SIZE);`
- 如何把stack變大?

- 複習: `realloc`做什麼用的?

- `realloc`相關決定:
- 要`realloc`成多大?
- `realloc`完畢有沒有什麼要改的地方?

來一點複雜度分析

- 如果每次都realloc成原本的兩倍大
- capacity 存現在的stack大小
- `realloc(stack, 2*capacity*sizeof(int));`
- `capacity*=2;`
- 問題: push n 次的時間複雜度為多少?

來一點複雜度分析

- 問題: push n 次的時間複雜度為多少?
- 假設:
- Push一個element是 $O(1)$
- Allocate一個element是 $O(1)$
- 移動一個element是 $O(1)$

- 第一小問題: 如果目前capacity是 2^k , 花在realloc的時間是多少?
- 第一小問題的子問題: realloc是 $O(?)$
- Worst case: Allocate $2 * \text{capacity}$ 個element的位置, 然後移動capacity個element的資料到新地方
- 答案??

來一點複雜度分析

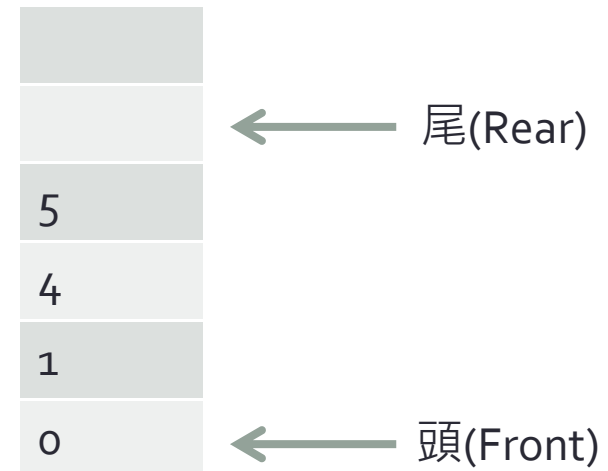
- 第一小問題: 如果目前capacity是 2^k , 花在realloc的時間是多少?
- 第一小問題的子問題: realloc是 $O(\text{capacity})$
- 那麼花在realloc的時間就是 $\sum_{i=1}^k 2^i = 2^{k+1} - 2$

來一點複雜度分析

- 第二小問題: 會使得目前的stack大小為capacity= 2^k , 應該已經push了幾次?
- 答案: 至少已經 $n = 2^{k-1}$ 次了.
- $2^{k+1} - 2 = 4n - 2 = O(n)$
- 喔耶.
- <動腦時間: 回家作業> 如果每次增加為原本capacity的c倍, $c > 1$, 那麼複雜度會有變化嗎? 為什麼?

Queue

- Queue是蝦密咚咚?
- 也是Ordered List
- 和Stack有什麼不一樣?
- 從尾(Rear)放, 從頭(Front)拿
- 拿: Delete
- 放: Add
- 所以先放進去的, 會??出來?



Queue

- 先放進去的, 會先出來
- 所以是First-In-First-Out (FIFO)

- 那麼, 支援那些動作呢? 各動作又需要什麼參數?
- 請同學們列舉:

- 做一個Queue
- 放(Add)
- 拿>Delete)
- 是不是空的?
- 是不是滿了?

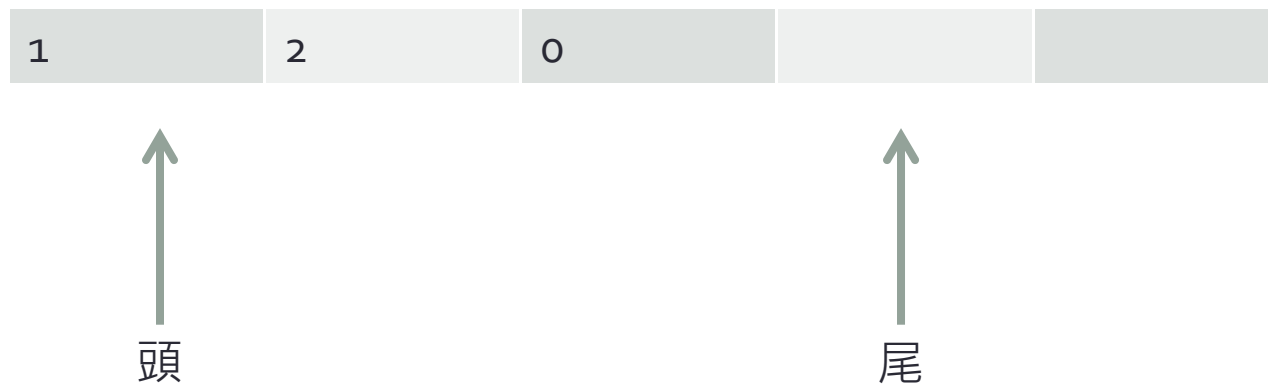
舉一個例子

- Job scheduling
- 如果沒有使用priority的概念的話
- 先來的job就先做 → FIFO的概念

- 問題:程式要求作業系統做一個job, 作業系統要照來的順序決定誰先使用資源.
- 作法:
- 程式發出要求的時候, 就放入queue中
- 資源空出來了, 就從queue拿出一個job來做
- 結果: FIFO
- <動腦時間> 如果放入一個stack會怎麼樣?

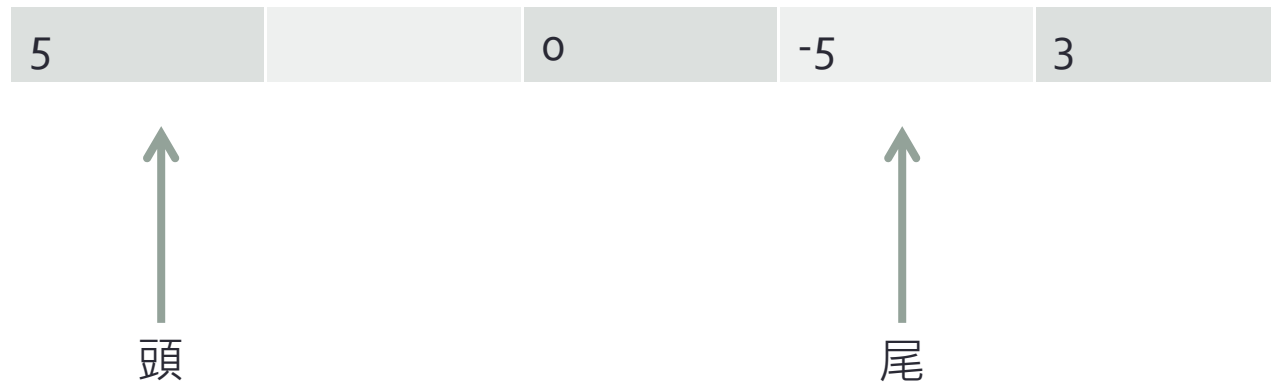
那麼，怎麼implement queue呢？

- 一樣是array的朋友
- 要記得頭跟尾的在array裡面的位置(index)
- 但是這次有點問題



- 滿了! 但是前面的空間沒有用到!
- 怎麼解決?

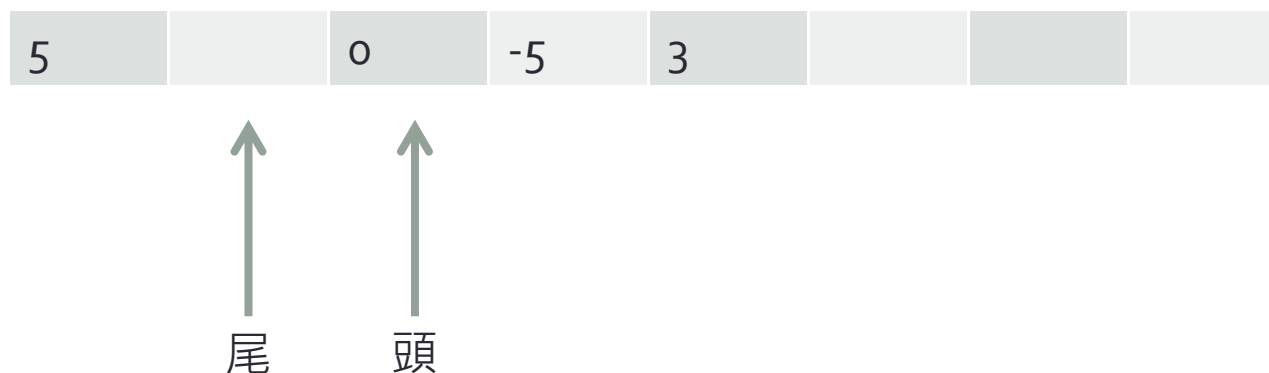
解決方式：頭尾相接



- 課本叫它circular queue
- 這樣就可以保證queue儲存的element數目最多可以跟array大小一樣
- 什麼時候是空?
- 什麼時候是滿?
- (能不能分辨空和滿?)

那麼，如果要使用動態大小呢？

- 一樣使用realloc來要到一塊更大的記憶體
- 然後還有什麼工作要做？
- realloc之後的樣子：看起來不太對



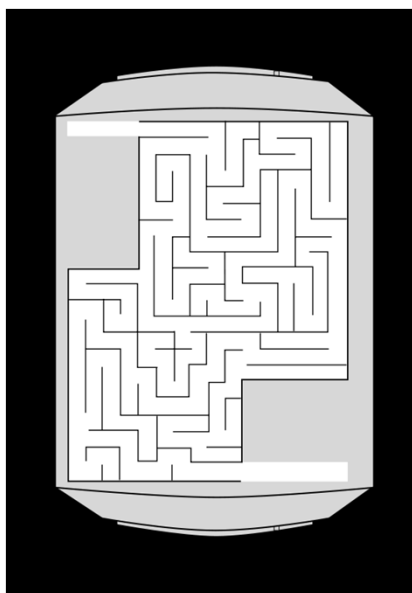
- 請一位同學上來講解怎麼修正☺

問題：很多stack用一個array來存

- 問題: 如果現在我們有一個array, 但是我希望用這個array存兩個stack, 請問要怎麼implement?
- 什麼時候stack滿了?
- push & pop的動作怎麼做?
- 問題: 那麼, 如果有n個stack, 又要怎麼存在一個array?
- 什麼時候stack滿了?
- 真的有滿嗎?

接下來講一些應用：

- 計算機問題
- 迷宮問題



應用一： 計算機

- 不是普通的計算機
- 題目: 如果打入一串如 $1+2*3-5/(4+5)/5$ 的算式, 請寫一個演算法來算出這串算式的結果.
- 怎麼寫呢? 好複雜T_T



先來看看算式長什麼樣子

- $1+2*3-5/(4+5)/5$
- 裡面有:
- Operand – 1, 2, 3, 5, 4, 5, etc.
- Operator – + * - /
- 括號– (,)

- 特色1: 左到右(left-to-right associativity)
- 特色2: 一般這種寫法叫做infix (operator夾在operand中間)
- 先後順序要operator去“比大小”
- 例如乘除是大, 加減是小, 那麼就先乘除後加減

別種寫法： postfix

- 把operator放到兩個operands的後面
- 例如 $2+3*4 \rightarrow 2\ 3\ 4\ *\ +$
- $a*b+5 \rightarrow ?$
- $(1+2)*7 \rightarrow ?$
- $a*b/c \rightarrow ?$
- $(a/(b-c+d))*(e-a)*c \rightarrow ?$
- $a/b-c+d* \rightarrow ?$
- $e-a*c \rightarrow ?$

Postfix有什麼好處?

- 沒有括號
- 用stack幫忙就可以很容易地算出結果!

- 例子: $6\ 2\ /\ 3\ -\ 4\ 2\ * +$
- 從左邊讀過去
- 讀到6, 放6進stack (stack: 6)
- 讀到2, 放2進stack (stack: 6 2)
- 讀到/, 取兩個operands (6和2), 算 $6/2$, 然後答案放回去stack (stack: 3)
- 讀到3, 放3進stack (stack: 3 3)
- 讀到-, 取兩個operands(3和3), 算 $3-3$, 然後答案放回去stack (stack: 0)
- 依此類推....(請同學上來繼續完成☺)

剩下來的“小問題”： infix to postfix

- “小問題”：怎麼把infix expression轉成postfix expression?
- 第一種方法：(適合紙上談兵)
 - 1. 把整個expression
 - 2. 把所有的operator都移到operand的後面去, 方便去除所有括號
 - 3. 去除所有括號
- 但是實作上要怎麼做呢? (不希望做兩次)

又是一個可以利用stack解決的問題

- 方法:
 - 1. 從左至右讀取expression
 - 2. 碰到operand就直接輸出
 - 3. 碰到operator時比較stack頂上的operator和目前讀到的operator哪一個比較“大” (precedence)
 - 如果目前讀到的operator比較大, 就把operator放到stack裡
 - 如果一樣大或者現在讀到的operator比較小, 就一直把stack裡面的operator拿出來印出來, 一直到stack是空的或者現在的operator比stack頂的operator大或者一樣
- 為什麼可以這樣做?
- 裡面括號需要先印出, 但是卻是後讀到
- 比較外面的先用stack記起來, FILO, 由內而外

例子一

- 請一位同學來解說 怎麼把 $a+b*c$ 利用前述方法轉換成postfix
😊
- 那麼, 如果碰到括號怎麼辦?

括號

- 括號內的有優先性
- 因此當碰到右括號的時候, 就立刻把stack中的東西一直拿出來直到碰到左括號為止
- 舉例: $a*(b+c)$
- 輸出的內容, stack內容(最右邊為top)
- a
- a, *
- a, *(
- ab, *(
- ab, *(+
- abc, *(+
- abc+*

還有一些小問題

- 小問題1: 左括號的“大小”到底是多少?
- 碰到左括號一定要放進去stack →此時要是最大的
- 左括號的下一個operator一定要可以放進去→此時要是最小的

- 結論:
- 左括號有兩種“大小”
- 在stack內的時候優先性為最小
- 在stack內的時候優先性為最大
- 其他operator的優先性則不管在stack內外都一樣

還有一些小問題

- Q: 如何確保stack空的時候, 第一次碰到的operator可放進去?
- A: 在stack底部放入一個虛擬operator帶有最低的優先性
- Q: 如何確保讀到expression最後的時候, 可以把stack裡面未取出的operator都拿出來?
- A: 字串最後可以加一個優先性最低的虛擬operator

例題

- $a+(b*c/(d-f)+e)$
- 用白板解說 (請同學? :P)
- <回家作業> 請各位看一下page 137的program 3.15 以了解怎麼寫infix \rightarrow prefix的轉換程式
- 作業預告: 會跟“prefix”相關
- 例子: $a*b+c \rightarrow +*abc$

最後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;
 - }
 - }
- }

休息時間



周末快樂~

- 作業二今天會出
- 參考同學們的意見, 這次份量會減少
- 請及早開始 (程式作業不簡單)

