



DATA STRUCTURE

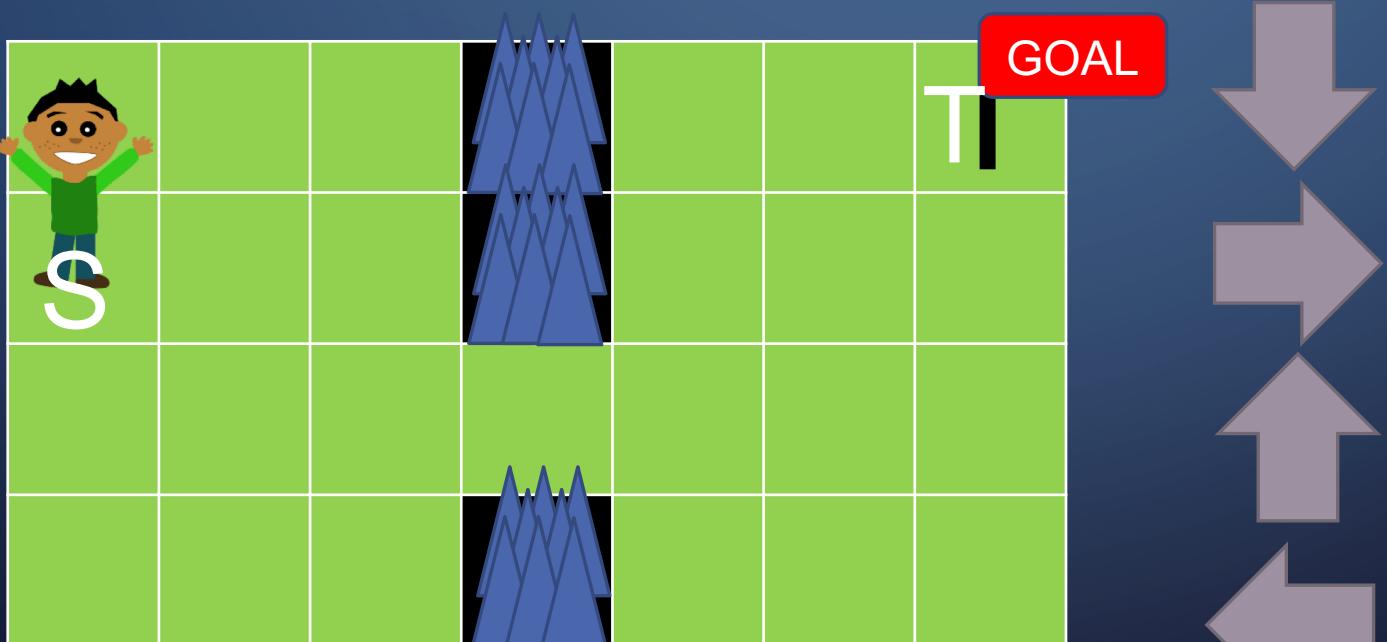
02. 基本程式設計(II)

迴圈與遞迴(ITERATION & RECURSION)

- 遞迴只得天上有，凡人應當用迴圈
- To iterate is human, to recursive, divine!

分而治之法(DIVIDE AND CONQUER)

- 小明又在山上迷路了，請問他該怎麼走到出口？

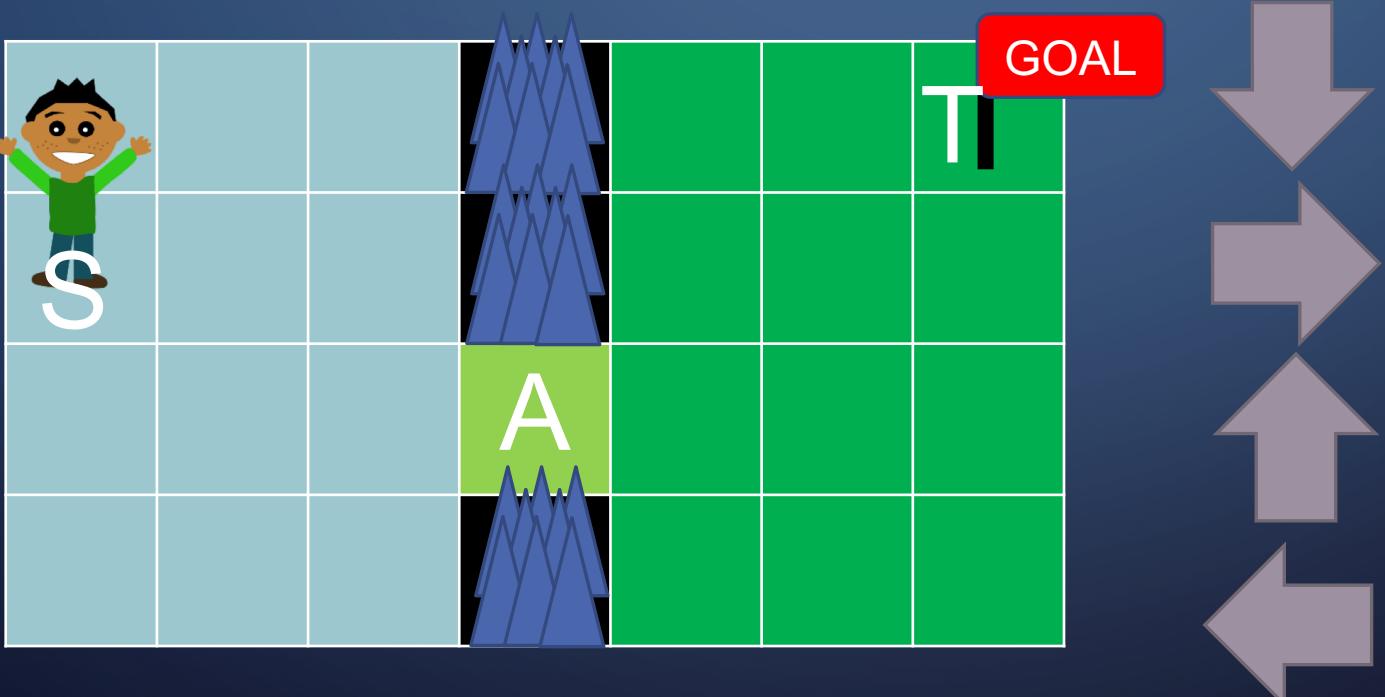


要找到一條路徑可以從S點走到T點...



分而治之法(DIVIDE AND CONQUER)

- 其實可以分成兩段!



先從S走到A，
再從A走到T也
可以達到目的!

分而治之法(DIVIDE AND CONQUER)

- 將一個大問題拆成兩個或多個小問題，這樣比較好解決
- 如果其中的小問題又可以再拆成更小的問題...
- 我可以設計一個演算法自己呼叫自己？？

自己呼叫自己 = 遞迴

- 繼續拿上次的例子
- 計算 $1+2+\dots+n$ 的總和
- 小明：我知道！ $n(1+n)/2$ ！



迴圈：重複直到某個條件

```
int x = 0;  
for(int i=0;i<n;i++)  
{  
    x = x + i;  
}  
printf("%d\n",n);
```

- 沒有不好，不過...

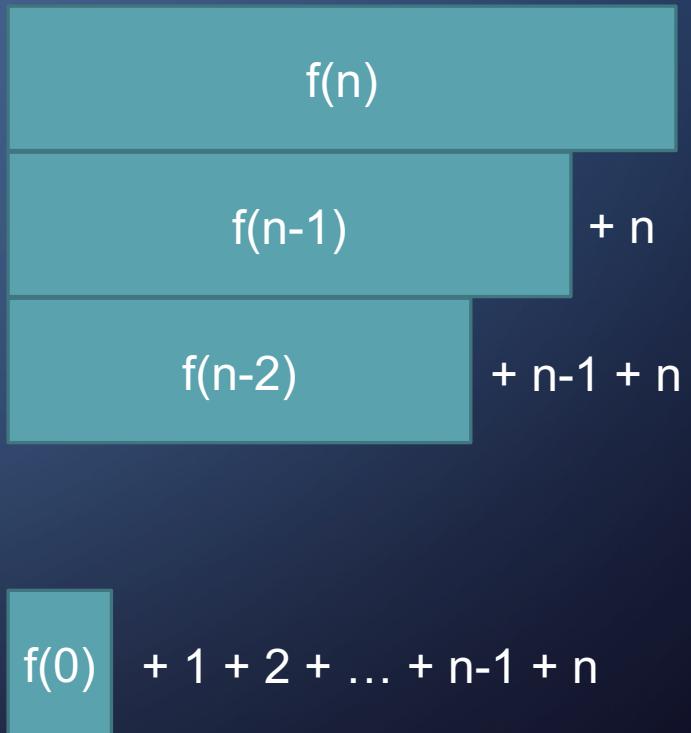


LET'S BE A DIVINE!

- 想一想，從1加到n這個問題，可不可以稍微的簡化？
 - 它會等於從1加到n-1，再加上n !!!
- 用數學式子表示： $f(n) = f(n-1) + n$!
 - 還要加上終止條件： $f(0) = 0$

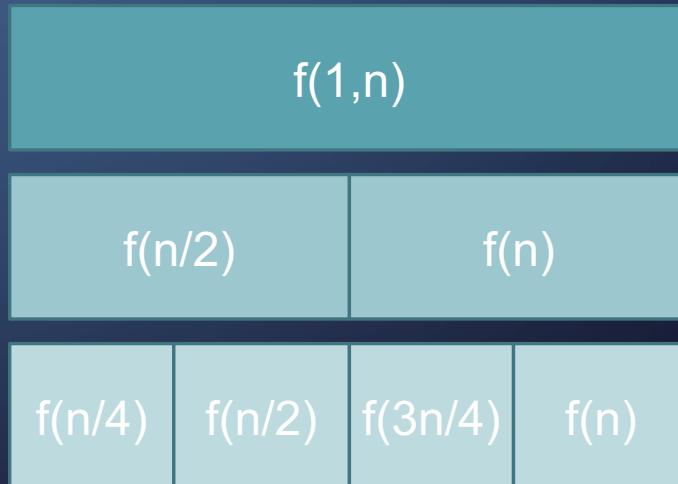
跨出天才的第一步!

- $$\begin{aligned}f(n) &= f(n-1) + n \\&= f(n-2) + n-1 + n \\&= f(n-3) + n-2 + n-1 + n \\&= \dots \\&= f(0) + 1 + 2 + \dots + n-1 + n \\&= 0 + 1 + 2 + \dots + n-1 + n\end{aligned}$$



跨出天才的第二步!

- $f(1,n) = f(1,n/2) + f(n/2,n)$
 $= f(1,n/4) + f(n/4,n/2) + f(n/2,3n/4) + f(3n/4,n)$
 $= \dots$



遞迴 VS 迴圈

優點：思考有時候比較方便

(1)可增加程式的可讀性。

缺點：

(1)需要花費較多的時間。

(2)利用暫存堆疊(Stack)的觀念，需要額外的儲存空間。

PRACTICE : 快速冪算法

- 練習使用遞迴
- 計算a的b次方
- 類似上週，但改用遞迴寫
- 上週我們先把b轉成二進位制，然後才一位一位去讀取，但其實可以把b看成停止條件，也就是
- 當 $b > 2$ 的時候....如果b會被整除？/如果b不會被整除？
- 問題：這樣真的有比較快嗎？為什麼沒有？怎樣才會有？

```
pow(3,19) 想求3的19次方  
=pow(3,9) * pow(3,9) * 3  
=pow(3,4) * pow(3,4) * 3 * .....  
= .....  
=?
```

ONLINE JUDGE

- 因為OJ不會知道你用甚麼語法寫，只會判斷結果正不正確
- 請上傳通過後在聊天室呼叫助教，由助教檢查確認通過