



# 動態規劃 II

(Dynamic Programming II)

# 2D / 0D DP

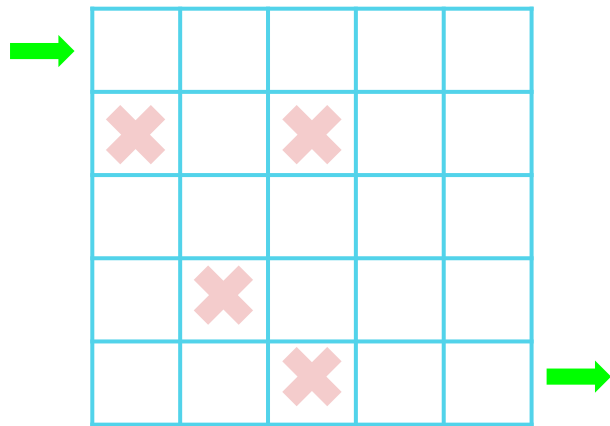


# Grid DP



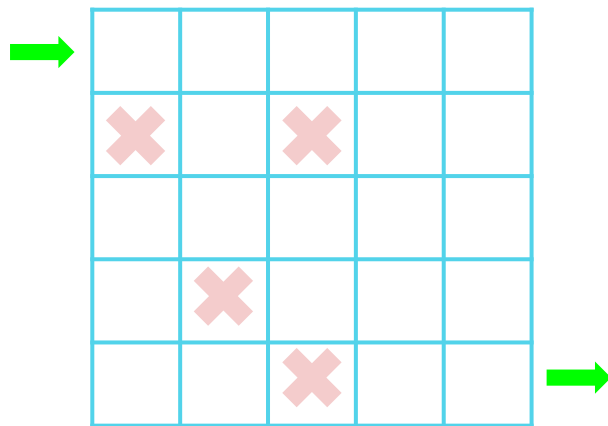
# Atcoder DP H. Grid 1

- 題目敘述
  - 輸入一個  $n * m$  大小的 Grid A, '.' 可以走, '#' 不能走
  - 只能往右或往下走
  - 求左上角到右下角有幾種走法
  - 答案可能很大, 請  $\text{mod } 10^9+7$  後再輸出
- 測資範圍
  - $n, m \leq 1000$



# Atcoder DP H. Grid 1

- 可以走到  $(i, j)$  的只有  $(i-1, j)$  與  $(i, j-1)$
- 狀態
  - $dp(i, j)$  從  $(0, 0)$  走到  $(i, j)$  的方法數
- 轉移
  - 不能走的格子  $dp(i, j) = 0$
  - 其他格子  $dp(i, j) = dp(i-1, j) + dp(i, j-1)$
- 答案
  - $dp(n-1, m-1)$

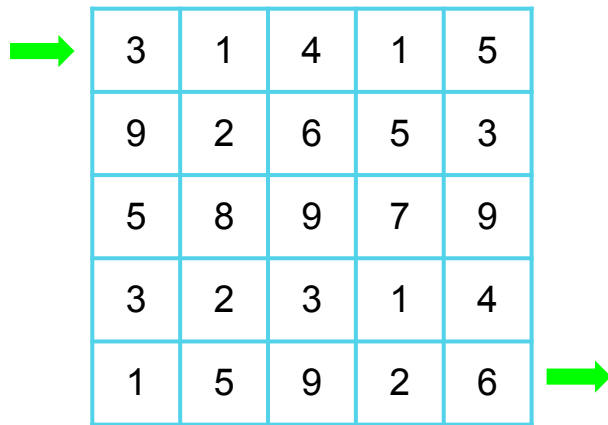


Grid 問題跟 1D / 0D 很像喔 ~



# LeetCode 64. Minimum Path Sum

- 題目敘述
  - 輸入一個  $n * m$  大小的 Grid A, 每格都有一個非負整數
  - 只能往右或往下走
  - 求左上角到右下角經過路線總和的最小值
- 測資範圍
  - $n, m \leq 1000$



A 5x5 grid of numbers. A green arrow points to the top-left cell (3), and another green arrow points from the bottom-right cell (6).

3	1	4	1	5
9	2	6	5	3
5	8	9	7	9
3	2	3	1	4
1	5	9	2	6



# LeetCode 64. Minimum Path Sum

- 可以走到  $(i, j)$  的只有  $(i-1, j)$  與  $(i, j-1)$
- 狀態
  - $dp(i, j)$  從  $(0, 0)$  走到  $(i, j)$  的最小總和
- 轉移
  - $dp(i, j) = A[i][j] + \min\{ dp(i-1, j), dp(i, j-1) \}$
- 答案
  - $dp(n-1, m-1)$



3	1	4	1	5
9	2	6	5	3
5	8	9	7	9
3	2	3	1	4
1	5	9	2	6



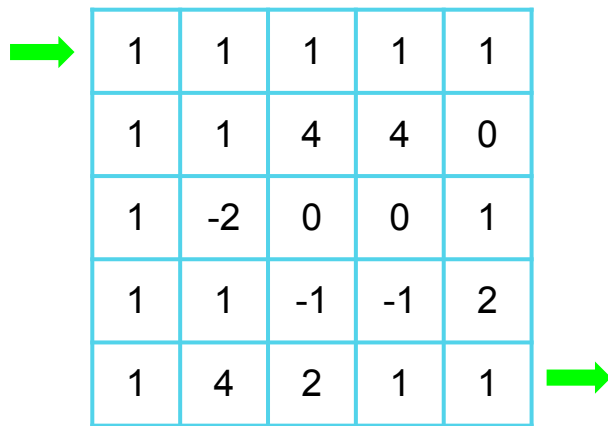
# LeetCode 1594. Maximum Non Negative Product in a Matrix

- 題目敘述

- 輸入一個  $n * m$  大小的 Grid A, 每格都有一個整數(可正可負)
- 只能往右或往下走
- 求左上角到右下角經過路線乘積的最大值

- 測資範圍

- $n, m \leq 15$
- $-4 \leq A[i] \leq 4$



1	1	1	1	1
1	1	4	4	0
1	-2	0	0	1
1	1	-1	-1	2
1	4	2	1	1





# LeetCode 1594. Maximum Non Negative Product in a Matrix

- 只記到  $(i, j)$  的最大值會出問題，因為會乘以負數，把最小值也記下來
- 狀態
  - $mx(i, j)$  : 走到  $(i, j)$  的最大值
  - $mn(i, j)$  : 走到  $(i, j)$  的最小值
- 轉移
  - $mx(i, j) = \max\{ mx(i-1, j) * A[i][j], mx(i, j-1) * A[i][j], mn(i-1, j) * A[i][j], mn(i, j-1) * A[i][j] \}$
  - $mn(i, j)$  類似
- 答案
  - $mx(n-1, m-1)$

1D 陣列版本要怎麼做呢？



## DP - Grid

- LeetCode 62. Unique Paths
- LeetCode 63. Unique Paths II
- LeetCode 64. Minimum Path Sum
- LeetCode 174. Dungeon Game
- LeetCode 1301. Number of Paths with Max Score
- LeetCode 1594. Maximum Non Negative Product in a Matrix



# LCS 問題



# LeetCode 1143. 最長共同子序列 (LCS)

- 題目敘述
  - 輸入兩個字串 A, B, 長度分別是 n, m
  - 找到 A, B 的最長共同子序列長度
- 測資範圍
  - $n, m \leq 1000$

Input

```
banana  
ababa
```

Output

```
3
```

baa, aaa 都是答案



# LeetCode 1143. 最長共同子序列 (LCS)

- 考慮兩個字串的最後一個字元是否可以配對
- 狀態
  - $dp(i, j)$  表示只看  $A[0 \sim i]$  與  $B[0 \sim j]$  的 LCS 長度
- 轉移
  - 若  $A[i] == B[j]$ 
    - $dp(i, j) = dp(i-1, j-1) + 1$
  - 若  $A[i] != B[j]$ 
    - $dp(i, j) = \max\{ dp(i, j-1), dp(i-1, j) \}$
- 答案
  - $dp(n-1, m-1)$



# LeetCode 1143. 最長共同子序列 (LCS)

	b	a	n	a	n	a
a						
b						
a						
b						
a						



# LeetCode 516. Longest Palindromic Subsequence

- 題目敘述
  - 輸入一個長度  $n$  的字串  $s$
  - 找到  $s$  的最長回文子序列長度
- 測資範圍
  - $n \leq 1000$

Input

baccda

Output

4

acca



# LeetCode 516. Longest Palindromic Subsequence

- 考慮每個連續區間的頭尾是否可以配對
- 狀態
  - $dp(l, r)$  表示只  $s[l \sim r]$  的 LPS 長度
- 轉移
  - 若  $s[l] == s[r]$ 
    - $dp(l, r) = dp(l + 1, r - 1) + 2$
  - 若  $s[l] != s[r]$ 
    - $dp(l, r) = \max\{ dp(l, r - 1), dp(l + 1, r) \}$
- 答案
  - $dp(0, n - 1)$





# LeetCode 1458. Max Dot Product of Two Subsequences

- 題目敘述
  - 輸入一個長度為  $n$  的整數陣列  $A$  與一個長度為  $m$  的整數陣列  $B$
  - 各找一個長度一樣的子序列, 讓 dot 總和最大化
- 測資範圍
  - $n, m \leq 500$

Input

```
4 3
2 1 -2 5
3 0 -6
```

Output

```
18
```

$$2 * 3 + (-2) * (-6) = 18$$



# LeetCode 1458. Max Dot Product of Two Subsequences

- 與 LCS 的狀態一樣，不過要注意兩個都不選的 case
- 狀態
  - $dp(i, j)$  表示只看  $A[0 \sim i]$  與  $B[0 \sim j]$  的答案，至少要選一個
- 轉移
  - $$dp(i, j) = \max \{ \begin{array}{l} dp(i-1, j-1) + A[i] * B[j], \\ A[i] * B[j], \\ dp(i-1, j-1), dp(i-1, j), dp(i, j-1) \end{array} \}$$
- 答案
  - $dp(n-1, m-1)$



## 題單 - LCS and Edit distance (2D/0D)

- LeetCode 1143. Longest Common Subsequence
- LeetCode 72. Edit Distance
- LeetCode 5. Longest Palindromic Substring
- LeetCode 392. Is Subsequence
- LeetCode 718. Maximum Length of Repeated Subarray
- LeetCode 32. Longest Valid Parentheses
- LeetCode 10. Regular Expression Matching
- LeetCode 1458. Max Dot Product of Two Subsequences



# 背包問題 (Knapsack Problem)



# 背包問題三部曲

- 輸入  $n$  種物品的重量與價值  $(w[i], v[i])$ , 背包容量上限是  $U$
  - 物品不能分割
  - 在重量總和不超過  $U$  的前提下, 讓背包內裝載的價值總和愈大愈好
  - 目標時間複雜度  $O(n*U)$
- 
- 0 / 1 背包
    - 每種物品只有一個
  - 無限背包
    - 每種物品有無限多個
  - 有限背包
    - 第  $i$  種物品有  $c[i]$



# 0 / 1 背包問題

- 題目敘述
  - 輸入  $n$  種物品的重量與價值 ( $w[i]$ ,  $v[i]$ ), 背包容量上限是  $U$
  - 每種物品只有一個
  - 在重量總和不超過  $U$  的情前提下, 讓背包內裝載的價值總和愈大愈好
- 測資範圍
  - $n, U \leq 1000$

Input	Output
4 10 2 3 3 1 5 2 7 6	9

(3, 1) + (7, 6)



# 0 / 1 背包問題

- 列舉重量總和，重量總和相同時可以直接比較價值
- 狀態
  - $dp(i, j)$  表示只物品  $0 \sim i$ ，重量總和為  $j$  的最大價值
- 轉移
  - $dp(i, j) = \max\{ \text{要用物品 } i, \text{不用物品 } i \}$
  - $dp(i, j) = \max\{ v[i] + dp(i - 1, j - w[i]), dp(i - 1, j) \}$
- 答案
  - $\max\{ dp(n-1, 0), dp(n-1, 1), dp(n-1, 2), \dots, dp(n-1, U) \}$



# 無限背包問題

- 題目敘述
  - 輸入  $n$  種物品的重量與價值  $(w[i], v[i])$ , 背包容量上限是  $U$
  - 每種物品有無限多個
  - 在重量總和不超過  $U$  的情前提下, 讓背包內裝載的價值總和愈大愈好
- 測資範圍
  - $n, U \leq 1000$

Input	Output
4 10 4 7 3 6 5 5 2 3	19

$(4, 7) + 2 * (3, 6)$





# 無限背包問題 - 方法一

- 狀態
  - $dp(i, j)$  表示只物品  $0 \sim i$  , 重量總和為  $j$  的最大價值
- 轉移
  - $dp(i, j) = \max\{ \text{物品 } i \text{ 拿 } 0 \text{ 個, } 1 \text{ 個, } 2 \text{ 個, } 3 \text{ 個, } \dots \}$
  - $dp(i, j) = \max\{ \begin{aligned} &dp(i - 1, j), \\ &dp(i - 1, j - w[i]) + v[i], \\ &dp(i - 1, j - 2 * w[i]) + 2 * v[i], \\ &\dots \end{aligned} \}$
- 答案
  - $\max\{ dp(n-1, 0), dp(n-1, 1), dp(n-1, 2), \dots, dp(n-1, U) \}$



# 無限背包問題 - 方法二

- 狀態
  - $dp(i, j)$  表示只物品  $0 \sim i$  , 重量總和為  $j$  的最大價值
- 轉移
  - $dp(i, j) = \max\{ \text{不拿物品 } i, \text{拿物品 } i \}$
  - $dp(i, j) = \max\{ dp(i - 1, j), dp(i, j - w[i]) + v[i] \}$
- 答案
  - $\max\{ dp(n-1, 0), dp(n-1, 1), dp(n-1, 2), \dots, dp(n-1, U) \}$



# 有限背包問題

- 題目敘述

- 輸入  $n$  種物品的重量與價值  $(w[i], v[i])$ , 背包容量上限是  $U$
- 第  $i$  種物品有  $c[i]$
- 在重量總和不超過  $U$  的情前提下, 讓背包內裝載的價值總和愈大愈好

- 測資範圍

- $n \leq 300$
- $U \leq 1000$

Input	Output
4 10	
4 7 2	
3 6 1	
5 5 4	
2 3 2	



# 有限背包問題 - 方法一

- 出現非常多次的東西就當作無限背包處理
- 其他的每一個當作 0 / 1 背包處理
- 物品數量可能變多  $U$  倍, 時間複雜度也是變  $U$  倍



## 有限背包問題 - 方法二

- 把重複的東西打包
- 有一種東西有 13 個, 把 13 分成  $1 + 2 + 4 + 6$
- 按照 1 個、2 個、4 個、6 個把東西打包
  
- 本來有  $c[i]$  個東西, 變成有  $\log(c[i])$  個東西, 用 0 / 1 背包處理
- 物品個數可能變成  $\log(U)$  倍, 時間複雜度也變成  $\log(U)$  倍

有限背包問題也可以做到  $O(n*U)$   
但是需要比較進階的技巧



# 背包問題變化問題

- 輸入  $n$  個數字總和為  $U$ , 將數字盡量平分成兩堆
- 找到重量總和剛好是  $U$  的最大價值
- 計算有幾種方法可以達到最大價值



## 題單 - 找零錢與背包問題

- LeetCode 322. Coin Change
- LeetCode 494. Target Sum
- LeetCode 474. Ones and Zeroes
- LeetCode 279. Perfect Squares
- LeetCode 322. Coin Change
- LeetCode 377. Combination Sum IV
- LeetCode 474. Ones and Zeroes
- LeetCode 494. Target Sum
- LeetCode 983. Minimum Cost For Tickets
- LeetCode 1049. Last Stone Weight II



## 題單 - 找零錢與背包問題

- LeetCode 1262. Greatest Sum Divisible by Three
- LeetCode 1449. Form Largest Integer With Digits That Add up to Target
- LeetCode 377. Combination Sum IV





# 輸出最佳解方案



# 輸出最佳解方案

- 如何證明一個陣列存在長度 10 個遞增子序列?  
=> 輸出一組答案
- 輸出一個 LIS 的方案
- 輸出一個 Grid 的走法
- 輸出一個 LCS 的方案
- 輸出一個背包問題最佳解拿了哪些物品
- 輸出字典順序最小的方法?



## 輸出一組答案 - LIS

a[i]	2	1	4	7	4	8	3	6	4	7
dp(i)	1	1	2	3	2	4	2	3	3	4



## 輸出一組答案 - LCS

	b	a	n	a	n	a
a	0	1	1	1	1	1
b	1	1	1	1	1	1
a	1	2	2	2	2	2
b	1	2	2	2	2	2
a	1	2	2	3	3	3



## 輸出一組答案 - 0/1 背包

sum	0	1	2	3	4	5	6	7	8	9	10
放入 2 (\$3)			3								
放入 3 (\$1)			3	1		4					
放入 5 (\$2)			3	1		4		5	3		9
放入 7 (\$6)			3	1		4		6	3	9	9

