基礎動態規劃

sam571128

October 25, 2021

開頭

假設你已經知道動態規劃 (DP) 是什麼了，建議可以去寫寫 AtCoder Educational DP Contest 的題目，而不會 DP 的人，我們今天會從基礎和經典題開始教

寫過費氏數列嗎

費氏數列

請輸出費氏數列第 n 項。

費氏數列為 $\{1, 1, 2, 3, 5, 8, \cdots\}$ 的數列

寫過費氏數列嗎

費氏數列

請輸出費氏數列第 n 項。

費氏數列為 $\{1, 1, 2, 3, 5, 8, \cdots\}$ 的數列

學過遞迴的人應該會覺得很簡單吧，就是學遞迴時的基本題!

寫過費氏數列嗎

```
int f(int x){
    if(x<=1) return 1;
    return f(x-1)+f(x-2);
}
```

遞迴實作費氏數列

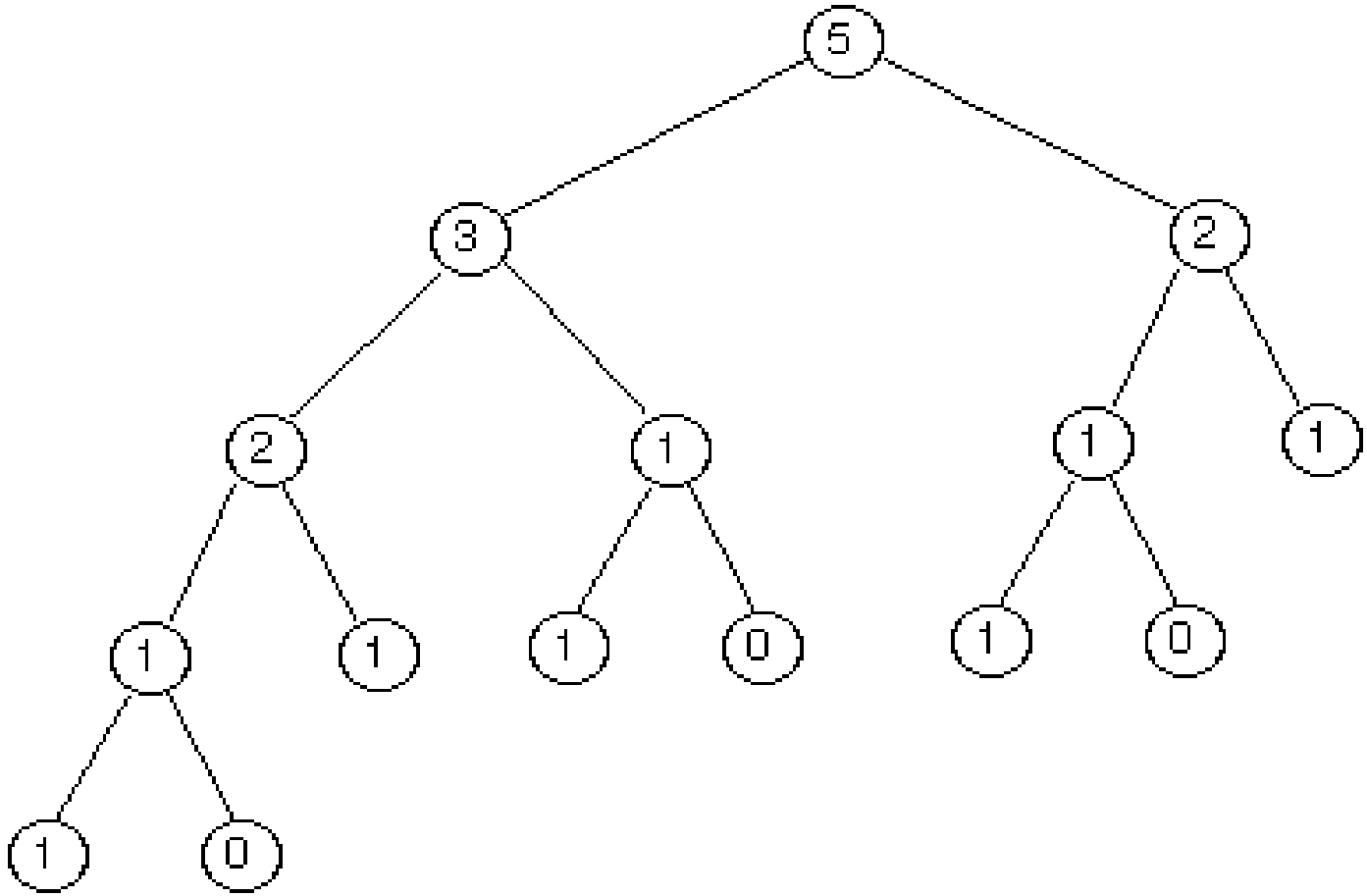
寫過費氏數列嗎

不過這樣的寫法，當 n 到 30 左右就不能在 1 秒內跑完了

寫過費氏數列嗎

實際分析看看他的時間複雜度，我們可以畫個圖來看看

寫過費氏數列嗎



費氏數列的遞迴樹

寫過費氏數列嗎

你會發現，每個節點大多都伸出兩隻手，我們可以粗略地說，這個寫法的時間複雜度會是 $O(2^n)$，不過實際上的複雜度是 $O(\phi^n)$，ϕ 是 $1.618\cdots$ 也就是黃金比例

寫過費氏數列嗎

不過，真的需要那麼久的時間才能完成這個問題嗎?

寫過費氏數列嗎

我們可以在遞迴的程式碼中加入幾行程式

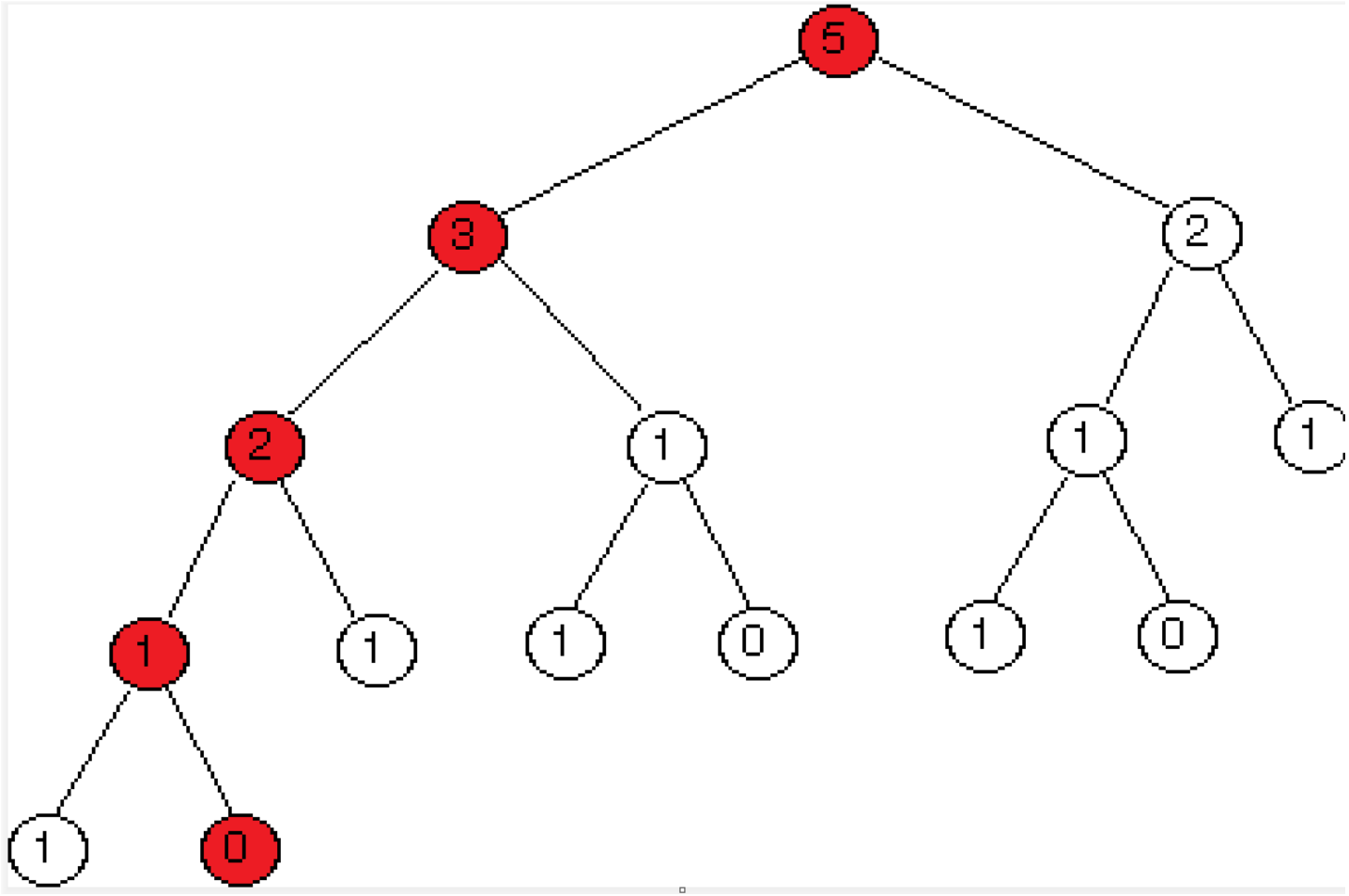
```
int dp[N];

int f(int x){
    if(dp[x]) return dp[x];
    if(x<=1) return dp[x] = 1;
    return dp[x] = f(x-1)+f(x-2);
}
```

也就是將走過的狀態儲存下來

寫過費氏數列嗎

這樣會快非常多! 而且時間複雜度是 $O(n)$ 的線性時間，讓我們一樣來看看遞迴樹

寫過費氏數列嗎



費氏數列的遞迴樹

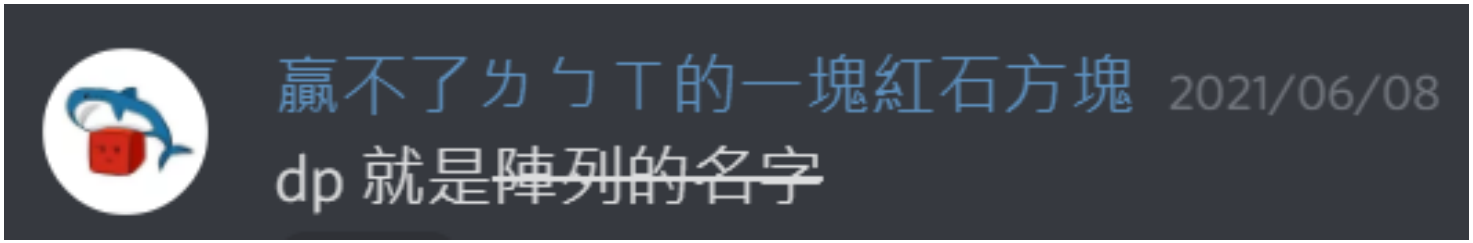
寫過費氏數列嗎

僅僅是將狀態儲存下來，就能夠有極大的差距，這就是「動態規劃」最基礎的概念。

什麼是動態規劃

所以說，動態規劃 (Dynamic Programming) 到底是什麼，哪裡動態？又哪裡在規劃了？

什麼是動態規劃



By 今年資奧女國手

什麼是動態規劃

不過要說 DP 到底是指什麼，其實也沒有人知道，但他就是一種概念，

只要有以下三種東西，我們大概就會稱其為 DP

動態規劃 (Dynamic Programming)

DP 可以做到的幾種事情

1. 儲存狀態
2. 解決重複子結構
3. 找最佳子問題

寫法

DP 根據寫法，分為 Top-Down 和 Bottom-Up，通常 Top-Down 是以遞迴的方式實作，而 Bottom-Up 則是以迴圈的方式實作。

寫法

```cpp
//Top-Down(遞迴)
int f(int x){
    if(dp[x]) return dp[x];
    if(x<=1) return dp[x] = 1;
    return dp[x] = f(x-1)+f(x-2);
}

//Bottom-Up(迴圈)
for(int i = 0;i <= n;i++){
    if(i <= 1) dp[i] = 1;
    else dp[i] = dp[i-1]+dp[i-2];
}
```

DP 的不同寫法

寫法

Bottom-Up 又根據寫法分為推和拉，這些東西都看個人習慣

重複子結構

走階梯 (1)

今天你在走階梯，你一步可以走一階或兩階，問當你走了 n 階時，總共有幾種不同可能的走法?

例如: 走 3 階，有 $\{1,1,1\},\{2,1\},\{1,2\}$ 三種走法

重複子結構

走階梯 (1)

今天你在走階梯，你一步可以走一階或兩階，問當你走了 n 階時，總共有幾種不同可能的走法?

例如: 走 3 階，有 $\{1,1,1\},\{2,1\},\{1,2\}$ 三種走法

我們可以假設 $dp[i]$ 代表走了 i 階時，有幾種走法

重複子結構

走階梯 (1)

今天你在走階梯，你一步可以走一階或兩階，問當你走了 n 階時，總共有幾種不同可能的走法?

例如: 走 3 階，有 $\{1,1,1\},\{2,1\},\{1,2\}$ 三種走法

那麼我們可以推出轉移式 $dp[i] = dp[i-1] + dp[i-2]$，也就是跟費氏數列一樣!

可以在 $O(n)$ 時間完成計算

重複子結構

走階梯 (2)

今天你在走階梯，你一步可以走 1 到 k 階，問當你走了 n 階時，總共有幾種不同可能的走法?

例如: 走 3 階，有 $\{1,1,1\},\{2,1\},\{1,2\}$ 三種走法

重複子結構

走階梯 (2)

今天你在走階梯，你一步可以走 1 到 k 階，問當你走了 n 階時，總共有幾種不同可能的走法?

例如: 走 3 階，可以走 1 到 3 階，有 $\{1,1,1\},\{2,1\},\{1,2\},\{4\}$ 四種走法

這個問題跟剛剛那個很像，只是有 k 個不同的轉移，

$dp[i] = dp[i-1] + dp[i-2] + \cdots + dp[i-k]$，時間複雜度為 $O(nk)$

重複子結構

走階梯 (2)

今天你在走階梯，你一步可以走 1 到 k 階，問當你走了 n 階時，總共有幾種不同可能的走法?

例如: 走 3 階，可以走 1 到 3 階，有 $\{1,1,1\},\{2,1\},\{1,2\},\{4\}$ 四種走法

這個問題跟剛剛那個很像，只是有 k 個不同的轉移，

$dp[i] = dp[i-1] + dp[i-2] + \cdots + dp[i-k]$，時間複雜度為 $O(nk)$

重複子結構

```cpp
//走階梯(2)
dp[0] = 1;
for(int i = 1;i <= n;i++){
    for(int j = 1;j <= k;j++){
        if(i-j>=0) dp[i] += dp[i-j];
    }
}
```

走階梯 (2)

DP 能做什麼?

DP 能做的事情非常多，以下舉幾種可以用 DP 解決的問題

1. 計數
2. 計算最大/最小的問題
3. Longest Common Subsequence (LCS)
4. Longest Increasing Subsequence (LIS)
5. 還有很多...

經典例題

最大子陣列和

給你一個 n 項的陣列，問整個陣列的最大和為多少?

範例: $\{1, 2, 3, -7, 5\}$ 的最大子陣列和為 6，出現在 $\{1, 2, 3\}$

經典例題

最大子陣列和

給你一個 n 項的陣列，問整個陣列的最大和為多少?

範例: $\{1, 2, 3, -7, 5\}$ 的最大子陣列和為 6，出現在 $\{1, 2, 3\}$

這個問題我們昨天也提過了，不過我們可以用 DP 的角度思考

經典例題

最大子陣列和

給你一個 n 項的陣列，問整個陣列的最大和為多少?

範例: $\{1, 2, 3, -7, 5\}$ 的最大子陣列和為 6，出現在 $\{1, 2, 3\}$

假設 $dp[i]$ 為結束在第 i 個位置的最大子陣列和

經典例題

最大子陣列和

給你一個 n 項的陣列，問整個陣列的最大和為多少?

範例: $\{1, 2, 3, -7, 5\}$ 的最大子陣列和為 6，出現在 $\{1, 2, 3\}$

那麼轉移式為 $dp[i] = \max(dp[i-1] + arr[i], arr[i])$，也就是要馬延續著上一個位置的最大連續和，要馬重頭開始做

經典例題

```
//最大子陣列和, dp[i]表示以第 i 個位置結尾的最大連續和
for(int i = 1;i <= n;i++){
    dp[i] = max(dp[i-1]+arr[i],arr[i]);
}
```

用 DP 實作最大子陣列和

經典例題

最長共同子序列 (LCS)

給你兩個字串 A, B，問這兩個字串的最長共同子序列為? 範例: $abcde$ 和 $adbec$ 的 LCS 長度為 3，字串為 abc

經典例題

最長共同子序列 (LCS)

給你兩個字串 A, B，問這兩個字串的最長共同子序列為? 範例: $abcde$ 和 $adbec$ 的 LCS 長度為 3，字串為 abc

這個問題也是經典的 DP 問題，設 $dp[i][j]$ 表示 A 的前 i 個字母與 B 的前 j 個字母的 LCS 長度

經典例題

最長共同子序列 (LCS)

給你兩個字串 A, B，問這兩個字串的最長共同子序列為? 範例: $abcde$ 和 $adbec$ 的 LCS 長度為 3，字串為 abc

$dp[0][0] = 0$，轉移式有以下幾種

$$dp[i][j] = \begin{cases} max(dp[i-1][j], dp[i][j-1]) \\ max(dp[i][j], dp[i-1][j-1]+1) \quad \text{when } A[i] = b[j] \end{cases}$$

經典例題

最長共同子序列 (LCS)

給你兩個字串 A, B，問這兩個字串的最長共同子序列為? 範例: $abcde$ 和 $adbec$ 的 LCS 長度為 3，字串為 abc

$dp[0][0] = 0$，轉移式有以下幾種

$$dp[i][j] = \begin{cases} max(dp[i-1][j], dp[i][j-1]) \\ max(dp[i][j], dp[i-1][j-1]+1) \quad \text{when } A[i] = b[j] \end{cases}$$

而答案為 $dp[n][m]$ (n 為 A 的長度，m 為 B 的長度)

經典例題

```cpp
//LCS
for(int i = 1;i <= n;i++){
    for(int j = 1;j <= m;j++){
        dp[i][j] = max(dp[i-1][j],dp[i][j-1]);

        if(a[i]==b[j]) dp[i][j] = max(dp[i][j],dp[i-1][j-1]+1);
    }
}
```

LCS (Longest Common Subsequence)

經典例題

Edit Distance

給你兩個字串 A, B，問這兩個字串的 Edit Distance 是多少?

兩個字串的 Edit Distance 為經過以下三種操作的最少步數:

1. 在 S 插入一個字母
2. 刪除 S 的一個字母
3. 替換 S 的一個字母

經典例題

Edit Distance

給你兩個字串 A, B，問這兩個字串的 Edit Distance 是多少?

兩個字串的 Edit Distance 為經過以下三種操作的最少步數:

1. 在 S 插入一個字母
2. 刪除 S 的一個字母
3. 替換 S 的一個字母

設 $dp[i][j]$ 表示 A 的前 i 個字母與 B 的前 j 個字母的 Edit Distance

經典例題

Edit Distance

給你兩個字串 A, B，問這兩個字串的 Edit Distance 是多少?

兩個字串的 Edit Distance 為經過以下三種操作的最少步數:

1. 在 S 插入一個字母
2. 刪除 S 的一個字母
3. 替換 S 的一個字母

經典例題

$dp[0][0] = 0$，轉移式有以下幾種

$$dp[i][j] = \min \begin{cases} min(dp[i-1][j]+1, dp[i][j-1]+1) \\ min(dp[i][j], dp[i-1][j-1]) & \text{when } A[i] = B[j] \\ dp[i-1][j-1]+1 \end{cases}$$

而答案為 $dp[n][m]$ (n 為 A 的長度，m 為 B 的長度)

經典例題

最長遞增子序列 (LIS)

給你一個 n 項的陣列，問這個陣列的 LIS 長度為多少?

範例: $\{1, 3, 2, 5, 8\}$ 的 LIS 為 $\{1, 3, 5, 8\}$，長度為 4

經典例題

最長遞增子序列 (LIS)

給你一個 n 項的陣列，問這個陣列的 LIS 長度為多少?

範例: $\{1, 3, 2, 5, 8\}$ 的 LIS 為 $\{1, 3, 5, 8\}$，長度為 4

設 $dp[i]$ 為以第 i 項結尾的最長遞增子序列，那麼有轉移式

$$dp[i] = \max \begin{cases} dp[j] + 1 & \text{if } i > j, a[i] > a[j] \\ 1 \end{cases}$$

經典例題

最長遞增子序列 (LIS)

給你一個 n 項的陣列，問這個陣列的 LIS 長度為多少?

範例: $\{1, 3, 2, 5, 8\}$ 的 LIS 為 $\{1, 3, 5, 8\}$，長度為 4

所以我們可以由 $i - 1$ 開始去尋找可以轉移的 j，並更新 $dp[i]$，這樣的時間複雜度為 $O(n^2)$

經典例題

最長遞增子序列 (LIS)

給你一個 n 項的陣列，問這個陣列的 LIS 長度為多少?

範例: $\{1, 3, 2, 5, 8\}$ 的 LIS 為 $\{1, 3, 5, 8\}$，長度為 4

不過實際上，這個問題可以做到 $O(n \log n)$，不過我們明天再講

經典例題

AtCoder DP Contest pC - Vocation

山姆有 n 天可以去度假，而他每天有三種事情可以做，並獲得歡樂

1. 去游泳。在第 i 天可以得到 a_i 點歡樂值
2. 去山上捕蟲。在第 i 天可以得到 b_i 點歡樂值
3. 在家寫作業。在第 i 天可以得到 c_i 點歡樂值

連續兩天做一樣的事情會使他感到無聊，因此他不會連續兩天做一樣的事，而山姆找你來幫他計算他最多可以得到多少歡樂

經典例題

設 $dp[i][0/1/2]$ 表示在第 i 天做第 j 件事情所能得到的最大歡樂總和。

而 $dp[0][0] = dp[0][1] = dp[0][2] = 0$

經典例題

設 $dp[i][0/1/2]$ 表示在第 i 天做第 j 件事情所能得到的最大歡樂總和。

而 $dp[0][0] = dp[0][1] = dp[0][2] = 0$ 而 $dp[0][0] = dp[0][1] = dp[0][2] = 0$

$$dp[i][0] = \max(dp[i-1][1] + a_i, dp[i-1][2] + a_i)$$

$$dp[i][1] = \max(dp[i-1][0] + b_i, dp[i-1][2] + b_i)$$

$$dp[i][2] = \max(dp[i-1][0] + c_i, dp[i-1][1] + c_i)$$

經典例題

設 $dp[i][0/1/2]$ 表示在第 i 天做第 j 件事情所能得到的最大歡樂總和。

而 $dp[0][0] = dp[0][1] = dp[0][2] = 0$

轉移式如下:

$$dp[i][0] = \max(dp[i-1][1] + a_i, dp[i-1][2] + a_i)$$

$$dp[i][1] = \max(dp[i-1][0] + b_i, dp[i-1][2] + b_i)$$

$$dp[i][2] = \max(dp[i-1][0] + c_i, dp[i-1][1] + c_i)$$

時間複雜度: $O(3n) = O(n)$

經典例題

從這些題目應該可以觀察到 dp 的神奇之處吧! 我們只要想到**一個狀態的設定**，並**列出轉移式**，一個問題就解決了!

DP 的步驟

以下為思考一個 DP 問題的做法:

1. 設計狀態
2. 列出轉移式

經典例題

換零錢

今天你有 n 塊錢，你想要把他們換成盡量少的零錢，而你所在的國家的金錢面額為 $a_1, a_2, \cdots, a_k$，問你最少能換成多少零錢?

經典例題

換零錢

今天你有 n 塊錢，你想要把他們換成盡量少的零錢，而你所在的國家的金錢面額為 $a_1, a_2, \cdots, a_k$，問你最少能換成多少零錢?

這個問題我們昨天有提過貪心的作法不見得會產生最佳解，而 DP 就能很穩定的產出我們所想要的最佳答案。

經典例題

換零錢

今天你有 n 塊錢，你想要把他們換成盡量少的零錢，而你所在的國家的金錢面額為 $a_1, a_2, \cdots, a_k$，問你最少能換成多少零錢?

設 $dp[i]$ 為最少能把 i 塊錢換成多少零錢

轉移式為: $dp[i] = \max_{1 \le j \le k} (dp[i - a_k] + 1)$

經典例題

換零錢

今天你有 n 塊錢，你想要把他們換成盡量少的零錢，而你所在的國家的金錢面額為 $a_1, a_2, \cdots, a_k$，問你最少能換成多少零錢?

設 $dp[i]$ 為最少能把 i 塊錢換成多少零錢

轉移式為: $dp[i] = \max_{1 \le j \le k} (dp[i - a_k] + 1)$

時間複雜度 $O(nk)$

經典例題

背包問題 (Knapsack Problem)

你有一個容量為 C 的背包，接下來有 n 個物品，每個物品有各自的價值與重量 $v[i], w[i]$，問你最多可以拿到多少總價值?

測資範圍: $1 \leq C \leq 5000, 1 \leq w[i] \leq 5000$

經典例題

背包問題 (Knapsack Problem)

你有一個容量為 C 的背包，接下來有 n 個物品，每個物品有各自的價值與重量 $v[i], w[i]$，問你最多可以拿到多少總價值?

測資範圍: $1 \leq C \leq 5000, 1 \leq w[i] \leq 5000$

這個問題我們之前提過，他是一個 **NP** 問題，當 n 很小時，我們可以利用位元枚舉，或 DFS 的方式找到答案。

經典例題

背包問題 (Knapsack Problem)

你有一個容量為 C 的背包，接下來有 n 個物品，每個物品有各自的價值與重量 $v[i], w[i]$，問你最多可以拿到多少總價值?

測資範圍: $1 \leq C \leq 5000$, $1 \leq w[i] \leq 5000$

不過當 C 有範圍限制時，我們可以在 $O(nC)$ 的時間做完

經典例題

背包問題 (Knapsack Problem)

你有一個容量為 C 的背包，接下來有 n 個物品，每個物品有各自的價值與重量 $v[i], w[i]$，問你最多可以拿到多少總價值?

測資範圍: $1 \leq C \leq 5000, 1 \leq w[i] \leq 5000$

設 $dp[i][j]$ 為對於前 i 個物品，拿了 j 的重量時，所得到的最大總價值

經典例題

背包問題 (Knapsack Problem)

你有一個容量為 C 的背包，接下來有 n 個物品，每個物品有各自的價值與重量 $v[i], w[i]$，問你最多可以拿到多少總價值?

測資範圍: $1 \leq C \leq 5000, 1 \leq w[i] \leq 5000$

設 $dp[i][j]$ 為對於前 i 個物品，拿了 j 的重量時，所得到的最大總價值

轉移式: $dp[i][j] = \max(dp[i-1][j-w[i]+v[i], dp[i-1][j])$

時間複雜度: $O(nC)$

暫時總結

今天講的 DP 都是比較經典的題目，基本上 DP 就是一種概念，各式各樣的題目都會出現 DP。學會 DP 的話，他可以幫助你對於一個問題有截然不同的思考方式，他的變化也很多，不過要說經典的題目的話，建議大家可以去寫寫 AtCoder Educational DP Contest 的前半。

而明天，我們會來講一些比較特殊的 DP，例如: LIS、二維 DP、位元 DP、區間 DP 等

LIS 的 $O(n \log n)$ 解

首先，我們要先來談談 LIS 的 $O(n \log n)$ 解，避免大家忘記 LIS 是什麼了，我們重新來看一次。

LIS 的 $O(n \log n)$ 解

最長遞增子序列 (LIS)

給你一個 n 項的陣列，問這個陣列的 LIS 長度為多少?

範例: $\{1, 3, 2, 5, 8\}$ 的 LIS 為 $\{1, 3, 5, 8\}$，長度為 4

LIS 的 $O(n \log n)$ 解

最長遞增子序列 (LIS)

給你一個 n 項的陣列，問這個陣列的 LIS 長度為多少?

範例: $\{1, 3, 2, 5, 8\}$ 的 LIS 為 $\{1, 3, 5, 8\}$，長度為 4

設 $dp[i]$ 為以第 i 項結尾的最長遞增子序列，那麼有轉移式

$$dp[i] = \max \begin{cases} dp[j] + 1 & \text{if } i > j, a[i] > a[j] \\ 1 \end{cases}$$

LIS 的 $O(n \log n)$ 解

最長遞增子序列 (LIS)

給你一個 n 項的陣列，問這個陣列的 LIS 長度為多少?

範例: $\{1, 3, 2, 5, 8\}$ 的 LIS 為 $\{1, 3, 5, 8\}$，長度為 4

所以說，我們現在的瓶頸在哪?

LIS 的 $O(n \log n)$ 解

最長遞增子序列 (LIS)

給你一個 n 項的陣列，問這個陣列的 LIS 長度為多少?

範例: $\{1, 3, 2, 5, 8\}$ 的 LIS 為 $\{1, 3, 5, 8\}$，長度為 4

當轉移式要我們從比 i 前面的元素繼續往後增加 LIS 的長度，我們會發現，有沒有更好的方式

LIS 的 $O(n \log n)$ 解

因此，我們這裡決定開一個額外的 vector，儲存一個遞增的陣列。然後我們可以對於每個元素，使用二分搜的方式來尋找這個元素往前的最長遞增子序列長度。

LIS 的 $O(n \log n)$ 解

由於我們要依序去做 n 個元素，每個元素做一次二分搜，最後的總時間為 $O(n \log n)$

```
vector<int> lis;
for(int i = 0;i < n;i++){
    int idx = lower_bound(lis.begin(),lis.end(),arr[i])-lis.begin();
    if(idx==n) lis.push_back(arr[i]);
    else lis[idx] = arr[i];
}
```

LIS 的 $O(n \log n)$ 解

LIS 的 $O(n \log n)$ 解

但這裡要提醒一下，vector 裡面儲存的不是原來的 LIS，但長度會是正確的長度。

二維偏序

APCS 2021/01 pD 飛黃騰達

飛黃是一種生物，活在二維座標平面上。

有隻特別的飛黃一開始在座標 $(0, 0)$ 的位置，而且你知道它只會往右上方移動，也就是移動的時只可以走到 x 座標跟 y 座標都不比原本小的位置。

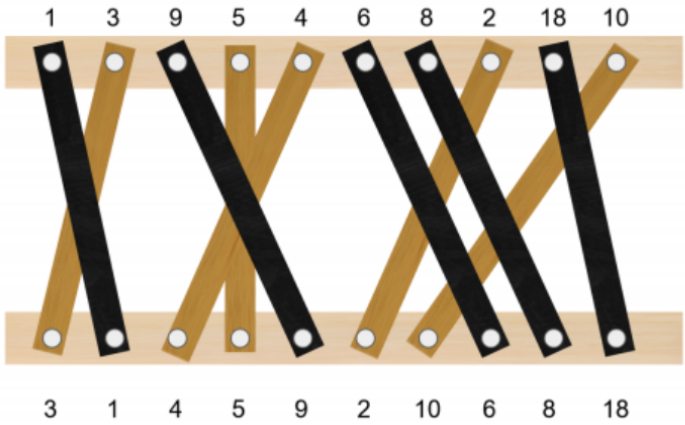
現在座標平面的第一象限上有 n 個位置有果實，給定這 n 個果實的座標，你想要知道這隻特別的飛黃最多可以吃到幾個果實（它必須移動到果實所在的座標才可以吃到果實）。

二維偏序

我們可以觀察到，這題的要求是 x 座標與 y 座標皆非嚴格遞增，那麼我們可以將所有點依照 x 座標進行排序，之後把每個點的 y 座標拿來做 LIS，即可完成這題

TOI 初選 2021 Problem C

方塊設計公司希望兩兩不相交的木板愈多愈好，請你寫一支程式來找出需要上漆的木板。以圖二為例，可選出 5 片木板上漆，一個可能的上漆方式如圖三。



圖三

TOI 初選 2021 Problem C

TOI 初選 2021 Problem C

TOI 初選 2021 pC

給你上下兩排的數字編號，你想要找到最多的木板對數，使得兩兩木板之間互不相交，並構造出字典序最小的一組解。

TOI 初選 2021 Problem C

這題由於木板編號不同，因此我們先將上排的數字依序改為 $\{1, 2, 3 \cdots, n\}$，下排根據上排修改後的數字進行修改。

TOI 初選 2021 Problem C

接著我們要來思考看看，最大字典序要怎麼做到?

TOI 初選 2021 Problem C

```
int dp[n], from[n];
for(int i = 0;i < n;i++){
    dp[i] = 1;
    for(int j = 0;j < i;j++){
        if(arr[j] < arr[i]){
            dp[i] = max(dp[i],dp[j]+1);
        }
    }
}
```

$O(n^2)$ 的 LIS

TOI 初選 2021 Problem C

```
int dp[n], from[n];
for(int i = 0;i < n;i++){
    dp[i] = 1;
    for(int j = 0;j < i;j++){
        if(arr[j] < arr[i]){
            dp[i] = max(dp[i],dp[j]+1);
            from[i] = j;
        }
    }
}
```

TOI 初選 2021 Problem C

```
int dp[n], nxt[n];
for(int i = n-1;i >= 0;i--){
    dp[i] = 1;
    nxt[i] = -1;
    for(int j = i+1;j < n;j++){
        if(arr[j] >= arr[i]) continue;
        if(dp[i]==dp[j]+1 && num[j] > num[nxt[i]]){
            nxt[i] = j;
        }else if(dp[i] < dp[j]+1){
            dp[i] = dp[j]+1;
            nxt[i] = j;
        }
    }
}
```

TOI 初選 2021 Problem C

那如何做到 $O(n \log n)$ 呢?

答案很簡單，同樣是由後往前做，我們在 vector 存的東西改為存 pair，記錄前一個數字和他的位置，這樣就可以解決這題了!

TOI 初選 2021 Problem C

題外話: 今年初選只要能夠解出這題，就能夠進入選訓營了

二維 dp

網格路徑數量

給你一個 $n \times m$ 的網格，今天你要從左上開始走，每一部只能往右或往下，問總共有幾種走法?

二維 dp

網格路徑數量

給你一個 $n \times m$ 的網格，今天你要從左上開始走，每一部只能往右或往下，問總共有幾種走法?

這題大家在國中的時候應該就在數學課本上看過類似的了吧

二維 dp

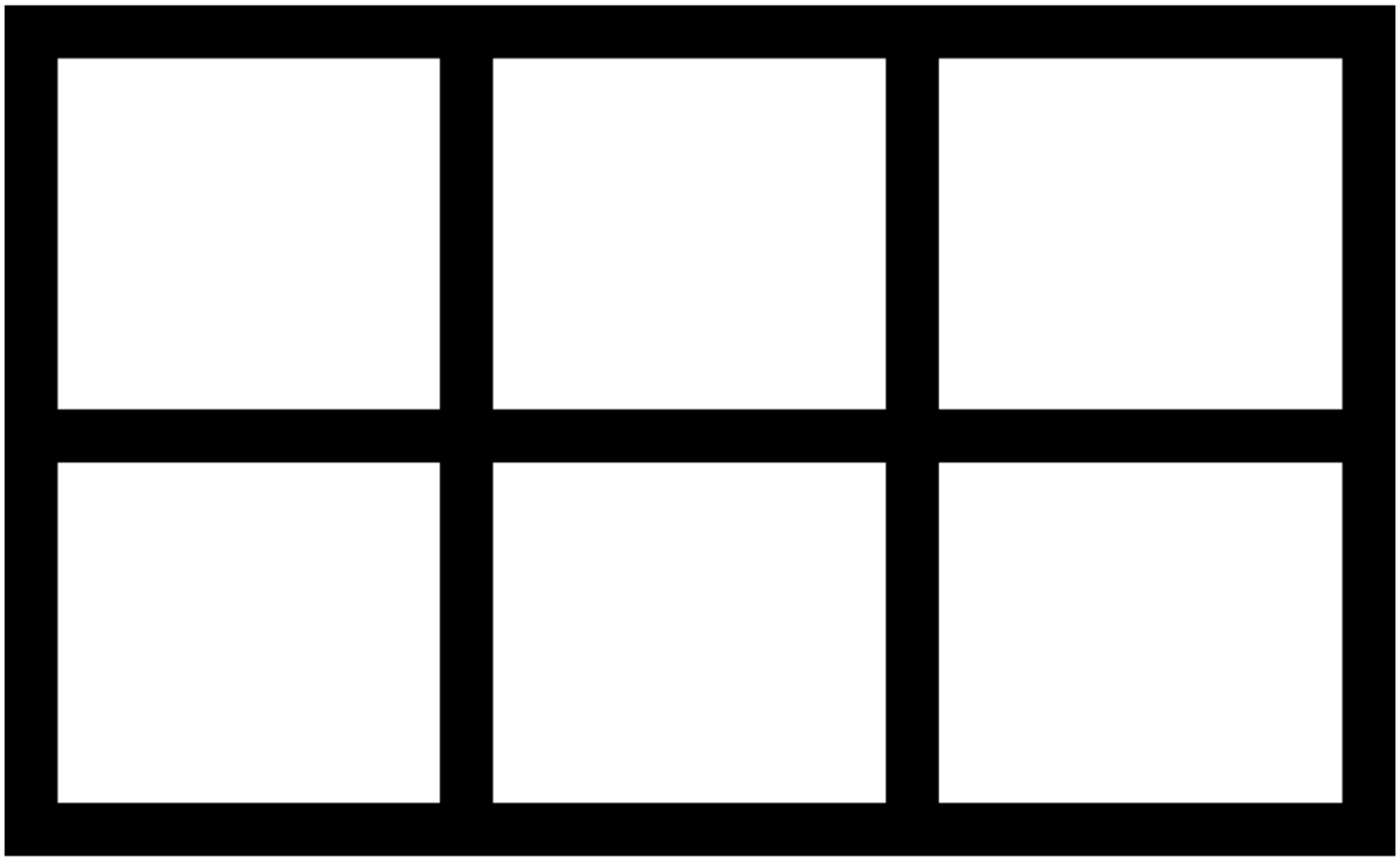
網格路徑數量

給你一個 $n \times m$ 的網格，今天你要從左上開始走，每一部只能往右或往下，問總共有幾種走法?

這題大家在國中的時候應該就在數學課本上看過類似的了吧

二維 dp

作法如下:



二維 dp

當你國中在算這種題目時，其實就是在做 DP 了!

二維 dp

設 $dp[i][j]$ 為走到第 (i, j) 時有幾種不同的走法，而轉移式即為

$$dp[i][j] = dp[i-1][j] + dp[i][j-1]$$

二維 dp

如果今天格子有分數，要算最大分數呢?

設 $dp[i][j]$ 為走到第 (i, j) 時有幾種不同的走法，而轉移式即為

$$dp[i][j] = dp[i-1][j] + dp[i][j-1] + val[i][j]$$

二維 dp

如果今天格子有障礙物，要做同樣的計算呢?

有障礙物的那幾格的 $dp[i][j]$ 設為 0 即可，畢竟走不到

雙人二維 dp

2019 北市賽 pE - 搜集寶藏

有兩個人從網格的左上角開始走，他們只能往右和下走，有的格子會有寶藏，他們希望兩個人共同拿到的寶藏數量最多，問他們最多可以拿到幾個寶藏?

雙人二維 dp

2019 北市賽 pE - 搜集寶藏

有兩個人從網格的左上角開始走，他們只能往右和下走，有的格子會有寶藏，他們希望兩個人共同拿到的寶藏數量最多，問他們最多可以拿到幾個寶藏?

同樣的問題，變成了兩個人，狀態要怎麼設計呢?

雙人二維 dp

2019 北市賽 pE - 搜集寶藏

有兩個人從網格的左上角開始走，他們只能往右和下走，有的格子會有寶藏，他們希望兩個人共同拿到的寶藏數量最多，問他們最多可以拿到幾個寶藏?

假設 $dp[i][x1][x2]$ 表示過了 i 步之後，第一個人走到 $(x1, i-x1)$, 第二個人走到 $(x2, i-x2)$ 時可以拿到的最多寶藏。

狀態的轉移式與只有一人時差不多

位元 dp

旅行推銷員問題 (Traveling Salesman Problem)

有一個旅行推銷員來到了一個國家，這個國家有 n $(n \leq 20)$ 個城市，他想要從第一個城市出發，恰好走過每個城市一次並回到第一個城市，給你每個城市間的距離，請問他最少要走多少距離才能完成他的旅行。

位元 dp

旅行推銷員問題 (Traveling Salesman Problem)

有一個旅行推銷員來到了一個國家，這個國家有 n $(n \leq 20)$ 個城市，他想要從第一個城市出發，恰好走過每個城市一次並回到第一個城市，給你每個城市間的距離，請問他最少要走多少距離才能完成他的旅行。

這個問題看起來十分困難，由於要恰好走過每個城市一次，因此，我們引入一種我們稱為位元 DP 的概念。

位元 dp

旅行推銷員問題 (Traveling Salesman Problem)

有一個旅行推銷員來到了一個國家，這個國家有 n $(n \leq 20)$ 個城市，他想要從第一個城市出發，恰好走過每個城市一次並回到第一個城市，給你每個城市間的距離，請問他最少要走多少距離才能完成他的旅行。

設 $dp[i][mask]$ 表示走到第 i 個城市時，mask 表示已經走過的城市 (以 0/1 表示)，所走的最小距離。

位元 dp

旅行推銷員問題 (Traveling Salesman Problem)

有一個旅行推銷員來到了一個國家，這個國家有 n $(n \leq 20)$ 個城市，他想要從第一個城市出發，恰好走過每個城市一次並回到第一個城市，給你每個城市間的距離，請問他最少要走多少距離才能完成他的旅行。

轉移式很簡單，$dp[i][mask] = \min_{i!=j, mask|i==0} (dp[j][mask \oplus (1 << i)])$

位元 dp

旅行推銷員問題 (Traveling Salesman Problem)

有一個旅行推銷員來到了一個國家，這個國家有 n $(n \leq 20)$ 個城市，他想要從第一個城市出發，恰好走過每個城市一次並回到第一個城市，給你每個城市間的距離，請問他最少要走多少距離才能完成他的旅行。

答案就是 $\max_{1 \leq i \leq n} (dp[i][(1 << n) - 1] + dis[i][1])$

區間 dp

AtCoder DP Contest pN - Slimes

有 n 隻史萊姆，他們有各自的大小 a_i，他們只能跟相鄰的史萊姆進行合併，當一隻大小為 a 的史萊姆和大小為 b 的史萊姆進行合併之後，要花費 $a+b$ 的時間，並在原地合併成一隻大小為 $a+b$ 的史萊姆。問最少要多少時間，這些史萊姆才能變為一隻史萊姆?

區間 dp

AtCoder DP Contest pN - Slimes

有 n 隻史萊姆，他們有各自的大小 a_i，他們只能跟相鄰的史萊姆進行合併，當一隻大小為 a 的史萊姆和大小為 b 的史萊姆進行合併之後，要花費 $a+b$ 的時間，並在原地合併成一隻大小為 $a+b$ 的史萊姆。問最少要多少時間，這些史萊姆才能變為一隻史萊姆?

區間 dp

AtCoder DP Contest pN - Slimes

有 n 隻史萊姆，他們有各自的大小 a_i，他們只能跟相鄰的史萊姆進行合併，當一隻大小為 a 的史萊姆和大小為 b 的史萊姆進行合併之後，要花費 $a+b$ 的時間，並在原地合併成一隻大小為 $a+b$ 的史萊姆。問最少要多少時間，這些史萊姆才能變為一隻史萊姆?

設 $dp[l][r]$ 表示將區間 $[l,r]$ 的史萊姆合併所需的最少時間。邊界條件:

$dp[i][i]=0$

區間 dp

AtCoder DP Contest pN - Slimes

有 n 隻史萊姆，他們有各自的大小 a_i，他們只能跟相鄰的史萊姆進行合併，當一隻大小為 a 的史萊姆和大小為 b 的史萊姆進行合併之後，要花費 $a+b$ 的時間，並在原地合併成一隻大小為 $a+b$ 的史萊姆。問最少要多少時間，這些史萊姆才能變為一隻史萊姆?

轉移式為 $dp[l][r] = \max_{l \le k \le r}(dp[l][k] + dp[k][r] + sum(l, r))$

區間 dp

AtCoder DP Contest pN - Slimes

有 n 隻史萊姆，他們有各自的大小 a_i，他們只能跟相鄰的史萊姆進行合併，當一隻大小為 a 的史萊姆和大小為 b 的史萊姆進行合併之後，要花費 $a+b$ 的時間，並在原地合併成一隻大小為 $a+b$ 的史萊姆。問最少要多少時間，這些史萊姆才能變為一隻史萊姆?

會發現我們的 code 裡面有一段在計算連續的區間和，因此，我們這裡引入一種稱為前綴和的東西

區間 dp

由於我們要找區間的總和 $a_l + a_{l+1} + \cdots + a_r$，我們可以使用一種特殊的東西，稱為前綴和，定義 $pref_i = a_1 + a_2 + \cdots + a_i$，要找區間 $[l, r]$ 的總和即為 $pref_r - pref_{l-1}$

區間 dp

AtCoder DP Contest pN - Slimes

有 n 隻史萊姆，他們有各自的大小 a_i，他們只能跟相鄰的史萊姆進行合併，當一隻大小為 a 的史萊姆和大小為 b 的史萊姆進行合併之後，要花費 $a+b$ 的時間，並在原地合併成一隻大小為 $a+b$ 的史萊姆。問最少要多少時間，這些史萊姆才能變為一隻史萊姆?

這樣做完之後，我們會去枚舉 l 和 r 還有他們之間的 k，因此時間複雜度是 $O(n^3)$

區間 dp

AtCoder DP Contest K - Stones

Alice 和 Bob 在玩一個遊戲，桌上會有 n 個石頭，他們在石頭上面寫下分數，並開始玩遊戲，在遊戲中，他們輪流進行，Alice 先拿，他們只能從最左和最右的石頭開始拿，問在兩人都走最佳策略時，Alice 和 Bob 分別可以拿到多少分數

區間 dp

AtCoder DP Contest K - Stones

Alice 和 Bob 在玩一個遊戲，桌上會有 n 個石頭，他們在石頭上面寫下分數，並開始玩遊戲，在遊戲中，他們輪流進行，Alice 先拿，他們只能從最左和最右的石頭開始拿，問在兩人都走最佳策略時，Alice 和 Bob 分別可以拿到多少分數

設 $dp[l][r]$ 表示先手在這個區間的石頭中，進行最佳策略可以拿到最多的分數

區間 dp

AtCoder DP Contest K - Stones

Alice 和 Bob 在玩一個遊戲，桌上會有 n 個石頭，他們在石頭上面寫下石頭的分數 a_i，並開始玩遊戲，在遊戲中，他們輪流進行，Alice 先拿，他們只能從最左和最右的石頭開始拿，問在兩人都走最佳策略時，Alice 和 Bob 分別可以拿到多少分數

轉移式為 $dp[l][r] =$

$$\max(a[l] + sum(l+1, r) - dp[l+1][r], a[r] + sum(l, r-1) - dp[l, r-1])$$

區間 dp

AtCoder DP Contest K - Stones

Alice 和 Bob 在玩一個遊戲，桌上會有 n 個石頭，他們在石頭上面寫下石頭的分數 a_i，並開始玩遊戲，在遊戲中，他們輪流進行，Alice 先拿，他們只能從最左和最右的石頭開始拿，問在兩人都走最佳策略時，Alice 和 Bob 分別可以拿到多少分數

使用前綴和計算 sum，時間複雜度 $O(n^2)$

背包問題

回想一下，背包問題是什麼?

背包問題

背包問題大致上的 Pattern

給你一個容量為 C 的背包，然後問拿物品可以拿到的最大價值

背包問題

不過，就只是拿物品這件事情，變化就可以有很多!

背包問題

背包問題的三大分類:

1. 0/1 背包問題
2. 無限背包問題
3. 有限背包問題

0/1 背包問題

0/1 背包問題，看到 0/1，你可能會以為跟二進位有關，不過並沒有。

0/1 背包問題

0/1 背包問題的題目大概會長這樣

0/1 背包問題

今天，你有一個容量為 C 的背包，桌上有 n 個物品，各自有自己的價值 v_i，每個物品只能拿一個或者不拿，問你最多可以拿多少價值?

0/1 背包問題

0/1 背包問題

今天，你有一個容量為 C 的背包，桌上有 n 個物品，各自有自己的價值 v_i，每個物品只能拿一個或者不拿，問你最多可以拿多少價值?

這樣的問題其實很簡單，我們前面也有說過，設 $dp[i][j]$ 表示考慮前 i 個物品，裝了 j 的重量時，最多可以拿多少價值。

0/1 背包問題

0/1 背包問題

今天，你有一個容量為 C 的背包，桌上有 n 個物品，各自有自己的價值 v_i，每個物品只能拿一個或者不拿，問你最多可以拿多少價值?

而轉移式十分簡單，也就是

$$dp[i][j] = max(dp[i-1][j-w[i]] + v[i], dp[i-1][j])$$

0/1 背包問題

寫法上就是兩個迴圈，就可以寫完了

```
for(int i = 1;i <= n;i++){
    for(int j = 0;j <= W;j++){
        //W 為物品重量的總和
        dp[i][j] = dp[i-1][j];
        if(j-w[i] >= 0) dp[i][j] = max(dp[i][j], dp[i-1][j-w[i]]+v[i]);
    }
}
```

0/1 背包問題

不過你應該會發現，我們其實在這個 code 當中，出現了一行 $dp[i][j] = dp[i-1][j]$，我們真的會需要開二維的陣列來存 dp 值嗎?

0/1 背包問題

答案是不用! 不過寫法上會有一點點的變化

```
for(int i = 1;i <= n;i++){
    for(int j = W;j >= 0;j--){
        //W 為物品重量的總和
        if(j-w[i] >= 0) dp[j] = max(dp[j], dp[j-w[i]]+v[i]);
    }
}
```

無限背包問題

下一個我們要講的東西是無限背包問題，實際上，他跟我們剛剛講的 0/1 背包問題非常相似。但所有物品都有無限量

無限背包問題

無限背包問題

今天，你有一個容量為 C 的背包，桌上有 n 種物品，各自有自己的重量與價值 v_i，每個物品都是無限量供應的，問你最多可以拿多少價值?

無限背包問題

不過要講無限背包問題聽起來有點抽象，我們這裡先來看看「換零錢」這個問題吧!

無限背包問題

換零錢

今天你有 K 塊錢，你想把他們換成 n 種面額的零錢，不過你希望能將這些錢換成盡量少的數量，問你最少可以將這些錢換成多少零錢。

無限背包問題

換零錢

今天你有 K 塊錢，你想把他們換成 n 種面額的零錢，不過你希望能將這些錢換成盡量少的數量，問你最少可以將這些錢換成多少零錢。

這個同樣的問題，我們不知道講了多少次了，不過，事實上，換零錢問題就是無限背包問題的另一種表示方式，不過物品的價值都是 1。

無限背包問題

換零錢

今天你有 K 塊錢，你想把他們換成 n 種面額的零錢 a_i，不過你希望能將這些錢換成盡量少的數量，問你最少可以將這些錢換成多少零錢。

設 $dp[i]$ 為最少可以將 i 塊錢換成多少零錢。轉移式為

$$dp[i] = \min_{1 \le j \le n} (dp[i - a[j]] + 1)$$

無限背包問題

寫法如下

```
for(int i = 1;i <= n;i++){
    for(int j = 0;j <= K;j--){
        if(j-a[i] >= 0) dp[j] = min(dp[j], dp[j-a[i]]+1);
    }
}
```

無限背包問題

有沒有發現這個寫法跟剛剛的 0/1 背包問題十分相似，甚至只是將迴圈反過來寫而已!

無限背包問題

```cpp
//0/1背包問題
for(int i = 1;i <= n;i++){
    for(int j = C;j >= 0;j--){
        if(j-w[i] >= 0) dp[j] = max(dp[j], dp[j-w[i]]+v[i]);
    }
}


//無限背包問題
for(int i = 1;i <= n;i++){
    for(int j = 0;j <= C;j++){
        if(j-w[i] >= 0) dp[j] = max(dp[j], dp[j-w[i]]+v[i]);
    }
}
```

有限背包問題

下一種背包問題是「有限背包問題」，看到有限，大概就可以知道，他實際上是與無限背包問題相反的一種題目，也就是物品數量是有限制的

有限背包問題

有限背包問題

今天，你有一個容量為 C 的背包，桌上有 n 種物品，各自有自己的價值 v_i，每個物品都有數量 m_i 的限制，問你最多可以拿多少價值?

有限背包問題

有限背包問題

今天，你有一個容量為 C 的背包，桌上有 n 種物品，各自有自己的重量 w_i 與價值 v_i，每個物品都有數量 m_i 的限制，問你最多可以拿多少價值?

這個問題想法也是一樣，同樣是設 $dp[i]$ 為拿了重量為 i 時，能夠拿到的最多價值是多少。

有限背包問題

有限背包問題

今天，你有一個容量為 C 的背包，桌上有 n 種物品，各自有自己的重量 w_i 與價值 v_i，每個物品都有數量 m_i 的限制，問你最多可以拿多少價值?

那我們可以把 m_i 個物品分別變成 m_i 個不同的物品，但他們的重量和價值都是相同的，這個問題就又變回 0/1 背包問題了，不過時間複雜度變為 $O(\sum m_i \times W)$

有限背包問題

寫法如下:

```
//有限背包問題
for(int i = 1;i <= n;i++){
    for(int k = 1;k <= m[i];k++){
        for(int j = C;j >= 0;j--){
            if(j-w[i] >= 0) dp[j] = max(dp[j], dp[j-w[i]]+v[i]);
        }
    }
}
```

有限背包問題

不過，要是物品的數量很大，那這樣做是不是很容易就會超時了! 因此，我們要來談談有限背包問題的加速!

有限背包問題

我們將物品數量拆成了 m_i 個不同的物品，不過，事實上我們可以拆成更少的物品來得到同樣的結果，而要做到這一點，我們只要將 m_i 拆成二進位即可，而時間複雜度變為 $O(\sum \log(m_i) \times W)$

有限背包問題

寫法如下:

```
//有限背包問題
for(int i = 1;i <= n;i++){
    for(int k = 1;k <= m[i];k *= 2){
        if(m[i]&k){
            for(int j = C;j >= 0;j--){
                if(j-w[i]*k >= 0) dp[j] = max(dp[j], dp[j-w[i]*k]+v[i]*k);
            }
        }
    }
}
```

有限背包問題

有限背包問題實際上有更快的方式，可以將時間複雜度降低為 $O(nW)$，不過他會用到斜率優化與單調對列的概念，會是我們之後才會講到的內容。

總結

基礎 DP 大概就講到這裡，下次我們會來講一些與競賽相關的數學技巧。