IOI-camp lecture Math

Meteor

January 29, 2019

Introduction

Introduction – 自我介紹

題目 (Roller Coaster Railroad, IOI 2016)

現在有 n 段雲霄飛車軌道，你要將這些軌道做排列，並用一些額外的煞車軌道連接他們。每段軌道有兩個值 s_i, t_i 表示進入這段軌道時車速不能超過 s_i，且出去這段軌道車速會變為 t_i。每一單位長的煞車軌道可以將車子減速一單位，請找一個將所有軌道都用到並合乎規定、且用的煞車軌道長度總合最短的方案。($n \leq 2 \cdot 10^5$)

Introduction – 自我介紹

題目 (Roller Coaster Railroad, IOI 2016)

現在有 n 段雲霄飛車軌道，你要將這些軌道做排列，並用一些額外的煞車軌道連接他們。每段軌道有兩個值 s_i, t_i 表示進入這段軌道時車速不能超過 s_i，且出去這段軌道車速會變為 t_i。每一單位長的煞車軌道可以將車子減速一單位，請找一個將所有軌道都用到並合乎規定、且用的煞車軌道長度總合最短的方案。$(n \leq 2 \cdot 10^5)$

Introduction – 自我介紹

1. 字串

Introduction – 自我介紹

1. 字串
2. 圖論、flow

Introduction – 自我介紹

1. 字串
2. 圖論、flow
3. 數學

Introduction – 自我介紹

1. 字串
2. 圖論、flow
3. 數學
4. 還是數學⋯

Introduction – 自我介紹

1. 字串
2. 圖論、flow
3. 數學
4. 還是數學⋯

資料結構	DP	幾何	圖論
DP	圖論		
geometry			
從枚舉到 K 短路			
nim	flow	圖論	dp
dp	flow		
圖論			

Introduction – 視力檢查

Introduction – 視力検査

- $>$

Introduction – 視力檢查

- $>$
- $\sqcup$

Introduction – 視力検査

- $>$
- $\sqcup$
- ϵ

Introduction – 視力検査

- $>$
- $\sqcup$
- ϵ
- $\lessgtr$

Introduction – 視力檢查

- $>$
- $\sqcup$
- ϵ
- $\lesseqgtr$
- $*$

Introduction – 數學

程式競賽中的數學：

1. 數學想法
2. 數學知識

Introduction – 數學想法

題目 (Increasing Numbers, AtCoder Grand Contest 011)

我們說一個數字是**遞增數**，如果他的任意兩個相鄰的位數 d_i, d_{i+1} 都滿足 $d_i \leq d_{i+1}$。給你一個數 N，請你把他寫成最少的遞增數的和。($\log_{10} N \leq 5 \cdot 10^5$)

Introduction – 數學想法

題目 (Increasing Numbers, AtCoder Grand Contest 011)

我們說一個數字是**遞增數**，如果他的任意兩個相鄰的位數 d_i, d_{i+1} 都滿足 $d_i \le d_{i+1}$。給你一個數 N，請你把他寫成最少的遞增數的和。($\log_{10} N \le 5 \cdot 10^5$)

最窩囊的殘局面世，解不出題 ~~輸子棋~~還要受窩囊氣，脾氣大的都不能看這~~盤棋~~道題。解讀街頭像棋殘局第167期

Introduction – 題解

1. 關鍵的第一步：

Introduction – 題解

1. 關鍵的第一步： 乘 9 。

Introduction – 題解

1. 關鍵的第一步： 乘 9 。
2. $111\ldots111 \times 9 = 999\ldots999 = 10^k - 1$

Introduction – 題解

1. 關鍵的第一步： 乘 9 。
2. $111\ldots111 \times 9 = 999\ldots999 = 10^k - 1$
3. 變成求位數和！

Introduction – 數學想法

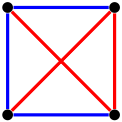
題目 (完全圖的分解)

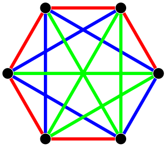
給你一個完全圖 K_{2n}，請你將所有邊分解成 n 個不相交的 Hamilton Path （通過所有點恰一次的鍊）。

Introduction – 數學想法

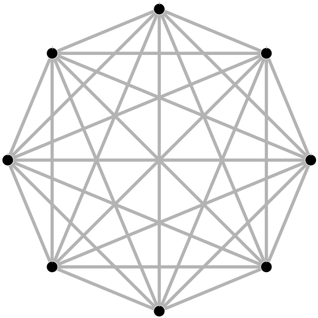
題目 (完全圖的分解)

給你一個完全圖 K_{2n}，請你將所有邊分解成 n 個不相交的 Hamilton Path （通過所有點恰一次的鍊）。





Introduction – 震驚！



Introduction – 數學知識

題目 (平方國的平方幣, TIOJ 1349)

給你一個正整數 n，請找出最小的 k，使得存在 k 個平方數 $a_1^2, a_2^2, \cdots, a_k^2$ 使得 $\sum a_i^2 = n$ 。$(n \leq 10^7)$

Introduction – 數學知識

題目 (平方國的平方幣, TIOJ 1349)

給你一個正整數 n，請找出最小的 k，使得存在 k 個平方數 $a_1^2, a_2^2, \cdots, a_k^2$ 使得 $\sum a_i^2 = n$ 。$(n \leq 10^7)$

- 一個很極端的「結論題」。

Introduction – 數學知識

題目 (平方國的平方幣, TIOJ 1349)

給你一個正整數 n，請找出最小的 k，使得存在 k 個平方數 $a_1^2, a_2^2, \cdots, a_k^2$ 使得 $\sum a_i^2 = n$ 。$(n \leq 10^7)$

- 一個很極端的「結論題」。
- 所有正整數都可以寫成 4 個平方數的和。

Introduction – 數學知識

題目 (平方國的平方幣, TIOJ 1349)

給你一個正整數 n，請找出最小的 k，使得存在 k 個平方數 $a_1^2, a_2^2, \cdots, a_k^2$ 使得 $\sum a_i^2 = n$ 。$(n \le 10^7)$

- 一個很極端的「結論題」。
- 所有正整數都可以寫成 4 個平方數的和。
- 太結論也不是很有趣……

Introduction – 數學知識真實案例

題目 (Little Artem and Graph, VK Cup 2016 Round 2, Div. 1 pF)

有一個圖是這樣生成的，從一個 k 個點的完全圖開始，每一次加入一個點，連接到 k 個已經在圖上的點。請計算這個圖的生成樹數量。$(n \leq 10000,\ k \leq 5)$

Introduction – 數學知識真實案例

題目 (Little Artem and Graph, VK Cup 2016 Round 2, Div. 1 pF)

有一個圖是這樣生成的，從一個 k 個點的完全圖開始，每一次加入一個點，連接到 k 個已經在圖上的點。請計算這個圖的生成樹數量。$(n \le 10000,\ k \le 5)$

- 本來是個 DP 題。

Introduction – 數學知識真實案例

題目 (Little Artem and Graph, VK Cup 2016 Round 2, Div. 1 pF)

有一個圖是這樣生成的，從一個 k 個點的完全圖開始，每一次加入一個點，連接到 k 個已經在圖上的點。請計算這個圖的生成樹數量。$(n \le 10000,\ k \le 5)$

- 本來是個 DP 題。
- 硬被玩成數學題！

Introduction – 怪怪的解法

Introduction – 怪怪的解法

1. 矩陣樹定理。

Introduction – 怪怪的解法

1. 矩陣樹定理。
2. Cayley–Hamilton theorem，最小多項式求行列式。

Introduction – 怪怪的解法

1. 矩陣樹定理。
2. Cayley–Hamilton theorem，最小多項式求行列式。
3. Berlekamp-Massey algorithm 求最小多項式。

Introduction – 怪怪的解法

1. 矩陣樹定理。
2. Cayley–Hamilton theorem，最小多項式求行列式。
3. Berlekamp-Massey algorithm 求最小多項式。
4. ???

Introduction – 怪怪的解法

1. 矩陣樹定理。
2. Cayley–Hamilton theorem，最小多項式求行列式。
3. Berlekamp-Massey algorithm 求最小多項式。
4. ???
5. Profit

Introduction – 怪怪的解法

1. 矩陣樹定理。
2. Cayley–Hamilton theorem，最小多項式求行列式。
3. Berlekamp-Massey algorithm 求最小多項式。
4. ???
5. Profit

You may double click into cells (or ctrl+click) to view the submissions history or hack the solution

Standings

#	Who	=	*	A 500	B 1000	C 1500	D 2000	E 3000	F 3000
1	anta	5680	-1	490 00:05	720 01:10	1272 00:38	1568 00:54		1680 01:50
2	wxhtxdy	4482		360 00:45	916 00:21	1446 00:09	1760 00:30		-2
3	Petr	4200		478 00:11	908 00:23	1254 00:41	1560 00:55	-2	
4	dotorya	4136		490 00:05	936 00:16	1134 00:36	1576 00:53	-1	
5	ikatanic	4080		490 00:05	928 00:18	1174 00:46	1488 01:04		

Introduction – 數學知識

題目 (經典問題)

給你 N 個點 (x_i, y_i) ，求一條線 $y = f(x)$ 使得

$$\sum_{i=1}^{N} (f(x_i) - y_i)^2$$

最小。

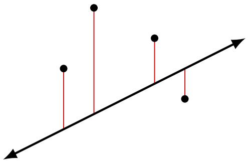
Introduction – 數學知識

題目 (經典問題)

給你 N 個點 (x_i, y_i) ，求一條線 $y = f(x)$ 使得

$$\sum_{i=1}^{N} (f(x_i) - y_i)^2$$

最小。



Introduction – 數學知識

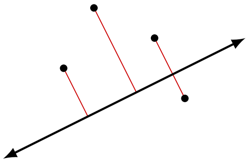
題目 (經典問題)

給你 N 個點 (x_i, y_i) ，求一條線 $y = f(x)$ 使得每個點到直線最短距離的平方和最小。

Introduction – 數學知識

題目 (經典問題)

給你 N 個點 (x_i, y_i) ，求一條線 $y = f(x)$ 使得每個點到直線最短距離的平方和最小。



微分方程

微分方程 – 定義

定義

一個微分方程是線性常微分方程，若且唯若以下幾點成立：

數論

數論 – 目標

題目 (An Easy Problem, NTUJ 1423)

給你等式 $a^b \equiv c \pmod d$ 中的其中 3 個，請計算出剩下的一個。(不同的子題有不同的範圍)

題目 (An Easy Problem – Subtask #1, NTUJ 1423)

給你 a, b，求最大的 m 使得 $a \equiv b \pmod{m}$。$(a, b \leq 10^{12})$

題目 (An Easy Problem – Subtask #1, NTUJ 1423)

給你 a, b，求最大的 m 使得 $a \equiv b \pmod{m}$。$(a, b \leq 10^{12})$

定義

$$a \equiv b \pmod{m} \iff a - b \mid m$$

數論 – 基礎

題目 (An Easy Problem – Subtask #2, NTUJ 1423)

給你 a, b, m，求 $c \equiv a^b \pmod{m}$。($0 \le a < m \le 10^9$, $b \le 10^{12}$)

數論 – 基礎

題目 (An Easy Problem – Subtask #2, NTUJ 1423)

給你 a, b, m，求 $c \equiv a^b \pmod{m}$。($0 \leq a < m \leq 10^9$, $b \leq 10^{12}$)

快速冪 ($\mathcal{O}(\log n)$)：

- 如果 $b = 2b'$，則 $a^b \equiv \left(a^{b'}\right)^2 \pmod{m}$。

數論 – 基礎

題目 (An Easy Problem – Subtask #2, NTUJ 1423)

給你 a, b, m，求 $c \equiv a^b \pmod{m}$。($0 \leq a < m \leq 10^9$, $b \leq 10^{12}$)

快速冪 ($\mathcal{O}(\log n)$)：

- 如果 $b = 2b'$，則 $a^b \equiv \left(a^{b'}\right)^2 \pmod{m}$。
- 如果 $b = 2b' + 1$，則 $a^b \equiv a \cdot \left(a^{b'}\right)^2 \pmod{m}$。

題目 (An Easy Problem – Subtask #2, NTUJ 1423)

給你 a, b, m，求 $c \equiv a^b \pmod{m}$。$(0 \le a < m \le 10^9,\ b \le 10^{12})$

快速冪 ($\mathcal{O}(\log n)$)：

- 如果 $b = 2b'$，則 $a^b \equiv \left(a^{b'}\right)^2 \pmod{m}$。
- 如果 $b = 2b' + 1$，則 $a^b \equiv a \cdot \left(a^{b'}\right)^2 \pmod{m}$。

```
long long fpow(long long a, long long b, long 1ong m) [
    if (!a) return 1;
    int ret = fastpow(a*a, b/2, m);
    if (b&1) (ret *= b) %= m;
   return ret; \\ return a**b % m
}
```

數論 – 基礎

題目 (An Easy Problem – Subtask #3, NTUJ 1423)

給你 a, c, m，求 b 使得 $a^b \equiv c \pmod{m}$。
($a, b, m \leq 10^9$, m 是質數)

題目 (An Easy Problem – Subtask #3, NTUJ 1423)

給你 a, c, m，求 b 使得 $a^b \equiv c \pmod{m}$。
$(a, b, m \leq 10^9$, m 是質數$)$

$$a^{xk+y} \equiv c \pmod{m} \iff a^{xk} \equiv ca^{-y} \pmod{m}$$

題目 (An Easy Problem – Subtask #3, NTUJ 1423)

給你 a, c, m，求 b 使得 $a^b \equiv c \pmod{m}$。
$(a, b, m \leq 10^9$, m 是質數$)$

$$a^{xk+y} \equiv c \pmod{m} \iff a^{xk} \equiv ca^{-y} \pmod{m}$$

1. 取 $k \triangleq \lfloor \sqrt{m} \rfloor$。
2. 找 $\{a^{xk}\}$, $\{ca^{-y}\}$ 有沒有一樣的元素。

數論 – 基礎

題目 (An Easy Problem – Subtask #3, NTUJ 1423)

給你 a, c, m，求 b 使得 $a^b \equiv c \pmod{m}$。
$(a, b, m \leq 10^9$, m 是質數$)$

$$a^{xk+y} \equiv c \pmod{m} \iff a^{xk} \equiv ca^{-y} \pmod{m}$$

1. 取 $k \triangleq \lfloor\sqrt{m}\rfloor$。
2. 找 $\{a^{xk}\}$, $\{ca^{-y}\}$ 有沒有一樣的元素。

問題

怎麼求出 $a^{-1} \bmod m$？

數論 – 模逆元

也就是要找 b 使得

$$ab \equiv 1 \pmod{m}$$

數論 – 模逆元

也就是要找 b 使得

$$ab \equiv 1 \pmod{m} \implies \exists k',\, ab = k'm + 1 \implies \exists k,\, ba + km = 1$$

數論 – 模逆元

也就是要找 b 使得

$$ab \equiv 1 \pmod{m} \implies \exists k',\, ab = k'm + 1 \implies \exists k,\, ba + km = 1$$

定理 ($ax + by = 1$ 有解的條件)

$$ax + by = 1 \text{ 有解} \iff \gcd(a, b) = 1$$

數論 – 模逆元

也就是要找 b 使得

$$ab \equiv 1 \pmod m \implies \exists k',\ ab = k'm + 1 \implies \exists k,\ ba + km = 1$$

定理 ($ax + by = 1$ 有解的條件)

$$ax + by = 1 \text{ 有解} \iff \gcd(a, b) = 1$$

1. 假設 $a = kb + r,\ 0 \le r < b$，並且我們已經知道

$$bx' + ry' = 1 \tag{1}$$

數論 – 模逆元

也就是要找 b 使得

$$ab \equiv 1 \pmod{m} \implies \exists k',\, ab = k'm + 1 \implies \exists k,\, ba + km = 1$$

定理 ($ax + by = 1$ 有解的條件)

$$ax + by = 1 \text{ 有解} \iff \gcd(a, b) = 1$$

1. 假設 $a = kb + r,\ 0 \leq r < b$，並且我們已經知道
$$bx' + ry' = 1 \tag{1}$$
2. 把 $r = a - kb$ 代入 (1) 我們得到 $y'a + (x' - ky')b = 1$。

數論 – 模逆元

定理 (模逆元存在的條件)

模 m 下 a^{-1} 存在 $\iff \gcd(a,m)=1$。

定理 (模逆元存在的條件)

模 m 下 a^{-1} 存在 $\iff \gcd(a,m)=1$。

我們把在模 m 下與 m 互質的所有數的集合稱作模 m 的**乘法群**，寫作

$$(\mathbb{Z}/m\mathbb{Z})^{\times} \triangleq \{a \mid 0 \leq a < m,\ \gcd(a,m)=1\}$$

數論 – 模逆元

定理 (模逆元存在的條件)

模 m 下 a^{-1} 存在 $\iff$ $\gcd(a, m) = 1$。

我們把在模 m 下與 m 互質的所有數的集合稱作模 m 的**乘法群**，寫作

$$(\mathbb{Z}/m\mathbb{Z})^\times \triangleq \{a \mid 0 \le a < m, \gcd(a, m) = 1\}$$

問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 長什麼樣子？

數論 – 乘法群

問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 有多少個元素？

問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 有多少個元素？

也就是有多少個 $0 \le a < m$ 使得 $\gcd(a, m) = 1$？

數論 – 乘法群

問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 有多少個元素？

也就是有多少個 $0 \le a < m$ 使得 $\gcd(a, m) = 1$？

定義

$$\varphi(m) = \#\{a \mid 0 \le a < m,\ \gcd(a, m) = 1\}$$

數論 – 乘法群

問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 有多少個元素？

也就是有多少個 $0 \leq a < m$ 使得 $\gcd(a, m) = 1$？

定義

$$\varphi(m) = \#\{a \mid 0 \leq a < m,\ \gcd(a, m) = 1\}$$

顯然對於質數，$\varphi(p) = p - 1$。

數論 – 乘法群

問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 有多少個元素？

也就是有多少個 $0 \le a < m$ 使得 $\gcd(a, m) = 1$？

定義

$$\varphi(m) = \#\{a \mid 0 \le a < m,\ \gcd(a, m) = 1\}$$

顯然對於質數，$\varphi(p) = p - 1$。
那對於 $\gcd(p, q) = 1$，$\varphi(pq)$ 呢？

數論 – 韓信點兵

「相傳漢高祖劉邦有天趁喝酒的時候問大將軍韓信統御兵士多少，韓信答說，每 3 人一列餘 1 人、4 人一列餘 2 人、5 人一列餘 4 人。劉邦與張良都算不出來，以為韓信兵很多，嚇到吃手手。」

數論 – 韓信點兵

「相傳漢高祖劉邦有天趁喝酒的時候問大將軍韓信統御兵士多少，韓信答說，每 3 人一列餘 1 人、4 人一列餘 2 人、5 人一列餘 4 人。劉邦與張良都算不出來，以為韓信兵很多，嚇到吃手手。」

其實就是要解

$$\begin{cases} x \equiv 1 \pmod{3} \\ x \equiv 2 \pmod{4} \\ x \equiv 4 \pmod{5} \end{cases}$$

數論 – 中國剩餘定理

定理 (中國剩餘定理)

如果 $m_1, m_2, \ldots, m_n$ 互質，則

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

有解，

數論 – 中國剩餘定理

定理 (中國剩餘定理)

如果 $m_1, m_2, \ldots, m_n$ 互質，則

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad\vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

有解，且解為

$$x \equiv a_1 t_1 M_1 + a_2 t_2 M_2 + \cdots + a_n t_n M_n \pmod{m_1 m_2 \cdots m_n}$$

其中 $M_i \triangleq \prod_j m_j / m_i = \prod_{j \neq i} m_j$, $t_i \triangleq M_i^{-1} \bmod m_i$。

數論 – 中國剩餘定理

定理 (中國剩餘定理)

如果 $m_1, m_2, \ldots, m_n$ 互質，則

$$\psi(x) = (x \bmod m_1, x \bmod m_2, \ldots, x \bmod m_k)$$

是一個一對一且滿射的函數，且

$$\mathbb{Z}/m\mathbb{Z} \cong \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$$

數論 – 中國剩餘定理

定理 (中國剩餘定理)

如果 $m_1, m_2, \ldots, m_n$ 互質，則

$$\psi(x) = (x \bmod m_1, x \bmod m_2, \ldots, x \bmod m_k)$$

是一個一對一且滿射的函數，且

$$\mathbb{Z}/m\mathbb{Z} \cong \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$$

```
std::tuple<T, U, ...>
```

數論 – Euler φ

定理

假設 n, m 互質，則 $\varphi(nm) = \varphi(n)\varphi(m)$。

數論 – Euler φ

定理

假設 n, m 互質，則 $\varphi(nm) = \varphi(n)\varphi(m)$。

證明：

1. 因為 $\gcd(a, nm) = 1$
   $\iff$ $\gcd(a, n) = 1$ 且 $\gcd(a, m) = 1$。

數論 – Euler φ

定理

假設 n, m 互質，則 $\varphi(nm) = \varphi(n)\varphi(m)$。

證明：

1. 因為 $\gcd(a, nm) = 1$
$\iff \gcd(a, n) = 1$ 且 $\gcd(a, m) = 1$。
2. 每個滿足 $\gcd(x, n) = 1$ 且 $\gcd(y, m) = 1$ 的數對 (x, y) 由中國剩餘定理又可以對回模 nm 下的一個數。

數論 – Euler φ

定理

假設 n, m 互質，則 $\varphi(nm) = \varphi(n)\varphi(m)$。

證明：

1. 因為 $\gcd(a, nm) = 1$
   $\iff \gcd(a, n) = 1$ 且 $\gcd(a, m) = 1$。
2. 每個滿足 $\gcd(x, n) = 1$ 且 $\gcd(y, m) = 1$ 的數對 (x, y) 由中國剩餘定理又可以對回模 nm 下的一個數。
3. 這樣的數對有 $\varphi(n)\varphi(m)$ 個。

數論 – Euler φ

- $\varphi(p) = p - 1$。

數論 – Euler φ

- $\varphi(p) = p - 1$。
- $\varphi(p^k) = p^{k-1}(p - 1)$。

數論 – Euler φ

- $\varphi(p) = p - 1$。
- $\varphi(p^k) = p^{k-1}(p-1)$。

定理

如果 $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$，則

$$\begin{aligned}\varphi(n) &= p_1^{\alpha_1 - 1}(p_1 - 1) p_2^{\alpha_2 - 1}(p_2 - 1) \cdots p_k^{\alpha_k - 1}(p_k - 1) \\ &= n \prod_{p \mid n} \left(1 - \frac{1}{p}\right)\end{aligned}$$

數論 – 題目

題目 (Cool lucky function, NTU final 2014)

給你 n 個正整數 $a_1, a_2, \ldots, a_n$，問你這些數字之中是否存在兩個互質的數對。$(2 \leq n \leq 10^5,\ a_i \leq 10^6)$

數論 – 趣事

題目 (數學歸納法考題)

令 $x_1 = 1,\ x_{n+1} = 2x_n + 1$，請證明 $x_n \le 2^n - 1$。

題目 (數學歸納法考題)

令 $x_1 = 1,\ x_{n+1} = 2x_n + 1$，請證明 $x_n \le 2^n - 1$。

證明：

- $n = 1$ 時顯然成立。

數論 – 趣事

題目 (數學歸納法考題)

令 $x_1 = 1,\ x_{n+1} = 2x_n + 1$，請證明 $x_n \leq 2^n - 1$。

證明：

- $n = 1$ 時顯然成立。
- 假設 $n = k$ 時成立，
  $x_{k+1} = 2x_k + 1 \leq 2 \cdot (2^k - 1) + 1 = 2^{k+1} - 1$。

數論 – 趣事

題目 (數學歸納法考題)

令 $x_1 = 1,\ x_{n+1} = 2x_n + 1$，請證明 $x_n \leq 2^n - 1$。

證明：

- $n = 1$ 時顯然成立。
- 假設 $n = k$ 時成立，
  $x_{k+1} = 2x_k + 1 \leq 2 \cdot (2^k - 1) + 1 = 2^{k+1} - 1$。
- 由數學歸納法證畢。

數論 – 趣事

結果老師打錯一個字：

題目 (數學歸納法考題 – 錯誤版)

令 $x_1 = 1, x_{n+1} = 2x_n + 1$，請證明 $x_n \leq 2^n + 1$。

數論 – 趣事

結果老師打錯一個字：

題目 (數學歸納法考題 – 錯誤版)

令 $x_1 = 1,\ x_{n+1} = 2x_n + 1$，請證明 $x_n \leq 2^n + 1$。

證明：

- $n = 1$ 時顯然成立。

數論 – 趣事

結果老師打錯一個字：

題目 (數學歸納法考題 – 錯誤版)

令 $x_1 = 1, x_{n+1} = 2x_n + 1$，請證明 $x_n \leq 2^n + 1$。

證明：

- $n = 1$ 時顯然成立。
- 假設 $n = k$ 時成立，
  $x_{k+1} = 2x_k + 1 \leq 2 \cdot (2^k + 1) + 1 = 2^{k+1} + 3 \not\leq 2^{k+1} + 1$。

結果老師打錯一個字：

題目 (數學歸納法考題 – 錯誤版)

令 $x_1 = 1,\ x_{n+1} = 2x_n + 1$，請證明 $x_n \leq 2^n + 1$。

證明：

- $n = 1$ 時顯然成立。
- 假設 $n = k$ 時成立，
  $x_{k+1} = 2x_k + 1 \leq 2 \cdot (2^k + 1) + 1 = 2^{k+1} + 3 \not\leq 2^{k+1} + 1$。
- ?????

數論 – 題目

有時候題目變難，反而好做！

數論 – 題目

有時候題目變難，反而好做！

題目 (Cool lucky function – Hard)

給你 n 個正整數 $a_1, a_2, \ldots, a_n$，問你這些數字之中有多少對互質的數對。($2 \leq n \leq 10^5$, $a_i \leq 10^6$)

數論 – 題目

有時候題目變難，反而好做！

題目 (Cool lucky function – Hard)

給你 n 個正整數 $a_1, a_2, \ldots, a_n$，問你這些數字之中有多少對互質的數對。$(2 \leq n \leq 10^5,\ a_i \leq 10^6)$

排容：令 $c_k \triangleq \#\{(a,b) : k \mid \gcd(a,b)\}$。

數論 – 題目

有時候題目變難，反而好做！

題目 (Cool lucky function – Hard)

給你 n 個正整數 $a_1, a_2, \ldots, a_n$，問你這些數字之中有多少對互質的數對。$(2 \le n \le 10^5,\ a_i \le 10^6)$

排容：令 $c_k \triangleq \#\{(a,b) : k \mid \gcd(a,b)\}$。

要算的是

$$\frac{n(n-1)}{2} - \sum_{p \text{ prime}} c_p + \sum_{p,q \text{ prime}} c_{pq} - \cdots$$

問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 長什麼樣子？

問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 長什麼樣子？

我們只要會 $(\mathbb{Z}/p^k\mathbb{Z})^\times$ 就可以了，其中 p 是質數。

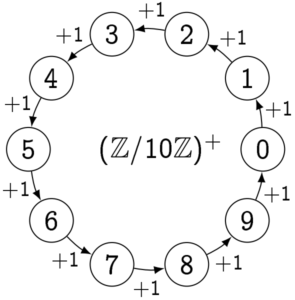
問題

$(\mathbb{Z}/m\mathbb{Z})^\times$ 長什麼樣子？

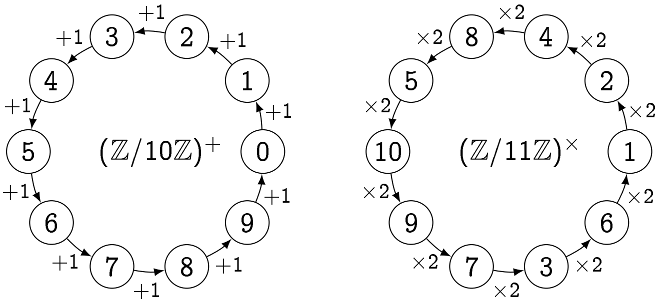
我們只要會 $(\mathbb{Z}/p^k\mathbb{Z})^\times$ 就可以了，其中 p 是質數。

先看 $(\mathbb{Z}/p\mathbb{Z})^\times$。

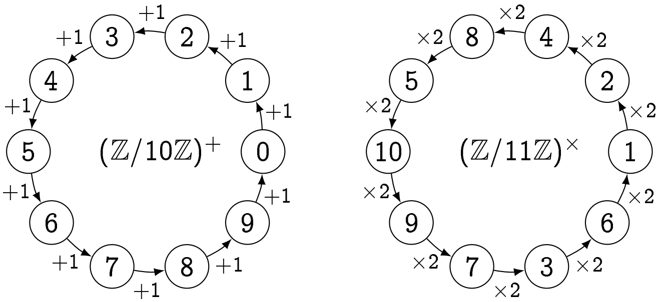
數論 – 乘法群



數論 – 乘法群


$(\mathbb{Z}/10\mathbb{Z})^+$
0
1
2
3
4
5
6
7
8
9
+1
$(\mathbb{Z}/11\mathbb{Z})^\times$
1
2
4
8
5
10
9
7
3
6
×2

數論 – 乘法群



$\Longrightarrow$ 模 11 下的乘法等同於模 10 下的加法。

定義 (原根)

如果 a^k 遍歷所有 $(\mathbb{Z}/m\mathbb{Z})^\times$ 下的元素，我們就說 a 是一個**原根**。

數論 – 原根

定義 (原根)

如果 a^k 遍歷所有 $(\mathbb{Z}/m\mathbb{Z})^\times$ 下的元素，我們就說 a 是一個**原根**。

定理 (原根存在的條件)

原根存在若且唯若 $m = 1, 2, 4, p^k, 2p^k$。

數論 – Euler 定理

定理 (Euler 定理)

- 對於質數 p，$a^{p-1} \equiv 1 \pmod{p}$。
- 對於任何數 m，$a^{\varphi(m)} \equiv 1 \pmod{m}$。

數論 – Euler 定理

定理 (Euler 定理)

- 對於質數 p，$a^{p-1} \equiv 1 \pmod p$。
- 對於任何數 m，$a^{\varphi(m)} \equiv 1 \pmod m$。

$$a \cdot a^{\varphi(m)-1} \equiv 1 \pmod m \implies a^{-1} \equiv a^{\varphi(m)-1} \pmod m$$

數論 – RSA

1. 選兩個大質數 p, q。

數論 – RSA

1. 選兩個大質數 p, q。
2. 令 $r = \varphi(pq) = (p-1)(q-1)$。

數論 – RSA

1. 選兩個大質數 p, q。
2. 令 $r = \varphi(pq) = (p-1)(q-1)$。
3. 選 e 使得 $\gcd(e, r) = 1$，此時 e 在模 r 下有反元素 d。

數論 – RSA

1. 選兩個大質數 p, q。
2. 令 $r = \varphi(pq) = (p-1)(q-1)$。
3. 選 e 使得 $\gcd(e, r) = 1$，此時 e 在模 r 下有反元素 d。
4. 加密 $x \to x^e$。

數論 – RSA

1. 選兩個大質數 p, q。
2. 令 $r = \varphi(pq) = (p-1)(q-1)$。
3. 選 e 使得 $\gcd(e, r) = 1$，此時 e 在模 r 下有反元素 d。
4. 加密 $x \to x^e$。
5. 解密 $(x^e)^d \to x^{ed} \equiv x \pmod{pq}$。

數論 – RSA

1. 選兩個大質數 p, q。
2. 令 $r = \varphi(pq) = (p-1)(q-1)$。
3. 選 e 使得 $\gcd(e, r) = 1$，此時 e 在模 r 下有反元素 d。
4. 加密 $x \to x^e$。
5. 解密 $(x^e)^d \to x^{ed} \equiv x \pmod{pq}$。

- 我們無法快速分解因數。

數論 – RSA

1. 選兩個大質數 p, q。
2. 令 $r = \varphi(pq) = (p-1)(q-1)$。
3. 選 e 使得 $\gcd(e, r) = 1$，此時 e 在模 r 下有反元素 d。
4. 加密 $x \to x^e$。
5. 解密 $(x^e)^d \to x^{ed} \equiv x \pmod{pq}$。

- 我們無法快速分解因數。
- 但可以快速判斷質數！

數論

競賽常用的演算法：

數論

競賽常用的演算法：

- 判斷 n 是不是質數：Miller Rabin $\implies \mathcal{O}(\log^{\mathcal{O}(1)}(n))$
- 因數分解 n ：Pollard's rho $\implies \mathcal{O}(n^{1/4})$

數論 – 找循環節

題目 (找循環節)

給你一個函數 f 和 x_0，對於所有 $i \geq 1$ 令 $x_i \triangleq f(x_{i-1})$，找出 $i \neq j$ 使得 $x_i = x_j$。

數論 – 找循環節

題目 (找循環節)

給你一個函數 f 和 x_0，對於所有 $i \geq 1$ 令 $x_i \triangleq f(x_{i-1})$，找出 $i \neq j$ 使得 $x_i = x_j$。

1. 令 $x = y = x_0$。

數論 – 找循環節

題目 (找循環節)

給你一個函數 f 和 x_0，對於所有 $i \geq 1$ 令 $x_i \triangleq f(x_{i-1})$，找出 $i \neq j$ 使得 $x_i = x_j$。

1. 令 $x = y = x_0$。
2. 每次 $x \leftarrow f(x)$, $y \leftarrow f(f(y))$。

數論 – 找循環節

題目 (找循環節)

給你一個函數 f 和 x_0，對於所有 $i \geq 1$ 令 $x_i \triangleq f(x_{i-1})$，找出 $i \neq j$ 使得 $x_i = x_j$。

1. 令 $x = y = x_0$。
2. 每次 $x \leftarrow f(x)$, $y \leftarrow f(f(y))$。
3. 有循環節的話總是會找到。

數論 – 生日悖論

現在有 70 個人，生日都在 $3 \cdot 365$ 天的範圍內。假設每個人的生日是獨立的，有 99% 的機率兩個人同年同月同日生！

數論 – 生日悖論

現在有 70 個人，生日都在 $3 \cdot 365$ 天的範圍內。假設每個人的生日是獨立的，有 99% 的機率兩個人同年同月同日生！

假設有 n 個人，隨機選 m 個位置。

數論 – 生日悖論

現在有 70 個人，生日都在 $3 \cdot 365$ 天的範圍內。假設每個人的生日是獨立的，有 99% 的機率兩個人同年同月同日生！

假設有 n 個人，隨機選 m 個位置。
任兩個人同個位置的機率是 $1/m$。

數論 – 生日悖論

現在有 70 個人，生日都在 $3 \cdot 365$ 天的範圍內。假設每個人的生日是獨立的，有 99% 的機率兩個人同年同月同日生！

假設有 n 個人，隨機選 m 個位置。
任兩個人同個位置的機率是 $1/m$。
令 X 表示有多少對人是同一個位置

$$\implies \mathbf{E}[X] = \frac{n(n-1)}{2}\frac{1}{m}$$

數論 – 生日悖論

現在有 70 個人，生日都在 $3 \cdot 365$ 天的範圍內。假設每個人的生日是獨立的，有 99% 的機率兩個人同年同月同日生！

假設有 n 個人，隨機選 m 個位置。
任兩個人同個位置的機率是 $1/m$。
令 X 表示有多少對人是同一個位置

$$\implies \mathbf{E}[X] = \frac{n(n-1)}{2}\frac{1}{m}$$

$n^2 \approx m \implies n = \mathcal{O}(\sqrt{m})$ 時就有高機率相撞！

數論 – 生日悖論應用

用 Hash 解以下兩題：

題目

給你 n 個字串，問其中有沒有字串和 A 相等。

數論 – 生日悖論應用

用 Hash 解以下兩題：

題目

給你 n 個字串，問其中有沒有字串和 A 相等。

題目

給你 n 個字串，問其中有沒有兩個字串相等。

數論 – 生日悖論應用

用 Hash 解以下兩題：

題目

給你 n 個字串，問其中有沒有字串和 A 相等。

題目

給你 n 個字串，問其中有沒有兩個字串相等。

Hash 的值分別要超過 $\mathcal{O}(n), \mathcal{O}(n^2)$。

期望值有一個好性質：

定理

如果 X, Y 是兩個隨機變數，不論 X, Y 是否獨立，都有

$$\mathbf{E}[aX + Y] = a\mathbf{E}[X] + \mathbf{E}[Y]$$

數論 – 期望值

期望值有一個好性質：

定理

如果 X, Y 是兩個隨機變數，不論 X, Y 是否獨立，都有

$$\mathbf{E}[aX + Y] = a\mathbf{E}[X] + \mathbf{E}[Y]$$

這常是解題的關鍵

題目 (Graph Game, Codeforces 235D)

現在有一個遊戲：

題目 (Graph Game, Codeforces 235D)

現在有一個遊戲：

1. 一開始的分數是 0，並且有一個 n 個點的樹。

數論 – 期望值

題目 (Graph Game, Codeforces 235D)

現在有一個遊戲：

1. 一開始的分數是 0，並且有一個 n 個點的樹。
2. 每次從剩下的點中隨機且等機率的選出一個點 v，並把分數加上 v 所在的連通塊的大小，且把 v 和與 v 相鄰的邊全部刪掉。

題目 (Graph Game, Codeforces 235D)

現在有一個遊戲：

1. 一開始的分數是 0，並且有一個 n 個點的樹。
2. 每次從剩下的點中隨機且等機率的選出一個點 v，並把分數加上 v 所在的連通塊的大小，且把 v 和與 v 相鄰的邊全部刪掉。
3. 一直進行到圖上沒有點為止。

數論 – 期望值

題目 (Graph Game, Codeforces 235D)

現在有一個遊戲：

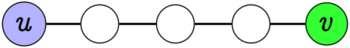
1. 一開始的分數是 0，並且有一個 n 個點的樹。
2. 每次從剩下的點中隨機且等機率的選出一個點 v，並把分數加上 v 所在的連通塊的大小，且把 v 和與 v 相鄰的邊全部刪掉。
3. 一直進行到圖上沒有點為止。

問你得到的分數的期望值。

數論 – 期望值

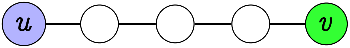
改求 u 被拔掉時 u, v 還連通的機率。

改求 u 被拔掉時 u, v 還連通的機率。



數論 – 期望值

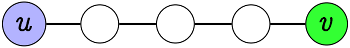
改求 u 被拔掉時 u, v 還連通的機率。



等價於 u 是這些點中第一個被拔掉的點

數論 – 期望值

改求 u 被拔掉時 u, v 還連通的機率。



等價於 u 是這些點中第一個被拔掉的點 。
$\Longrightarrow$ $p = 1/n$，n 是 u, v 間（包含）有幾個點。

數論 – Pollard's rho

```
int f(int x, int n) { return (x*x + 2) % n }
int pollard_rho(int n) {
    int xi, xj;
    int i = 1, j = 1;
    xi = xj = 2;
    while (true) {
        j++;
        xi = f(xi, n);
        xj = f(f(xj, n));
        int d = __gcd(abs(xi - xj), n);
        if (d != 1) return d;
    }
}
```

數論 – Pollard's rho

假設 $n = pq$

數論 – Pollard's rho

假設 $n = pq$

1. 用偽隨機函數 f 生成序列 $x_0, x_1, \ldots$ 。

數論 – Pollard's rho

假設 $n = pq$

1. 用偽隨機函數 f 生成序列 $x_0, x_1, \ldots$ 。
2. 當產生的數 $n \equiv \sqrt{p}$ 時應該就會有 $x_i \equiv x_j \pmod{p}$，也就是 $x_i \pmod{p}$ 在循環了。

數論 – Pollard's rho

假設 $n = pq$

1. 用偽隨機函數 f 生成序列 $x_0, x_1, \ldots$。
2. 當產生的數 $n \equiv \sqrt{p}$ 時應該就會有 $x_i \equiv x_j \pmod p$，也就是 $x_i \pmod p$ 在循環了。
3. 令 $g \triangleq \gcd(x_i - x_j, n)$，因為 p, q 互質不太可能也剛好 $x_i \equiv x_j \pmod q$，所以 g 高機率會是一個真因數。

數論 – Final

題目 (An Easy Problem – Subtask #4, NTUJ 1423)

給你 b, c, p，請你求 a 使得 $a^b \equiv c \pmod p$。
$(b, c, p \leq 10^9$, p 是數且 $\gcd(b, p-1) \leq 10^5)$

數論 – Final

題目 (An Easy Problem – Subtask #4, NTUJ 1423)

給你 b, c, p，請你求 a 使得 $a^b \equiv c \pmod{p}$。
($b, c, p \leq 10^9$, p 是數且 $\gcd(b, p-1) \leq 10^5$)

請看講義⋯

組合

組合 – 一些基本題目

1. $x_1 + x_2 + \cdots + x_n = m$，且每個 $x_i \geq 0$ 的方法數：

組合 – 一些基本題目

1. $x_1 + x_2 + \cdots + x_n = m$，且每個 $x_i \geq 0$ 的方法數：

$$\Longrightarrow \binom{m+n-1}{m}$$

組合 – 一些基本題目

1. $x_1 + x_2 + \cdots + x_n = m$，且每個 $x_i \geq 0$ 的方法數：

$$\Longrightarrow \binom{m+n-1}{m}$$

2. 用 1×2 瓷磚（可旋轉）排滿 $2 \times n$ 長方形的方法數：

組合 – 一些基本題目

1. $x_1 + x_2 + \cdots + x_n = m$，且每個 $x_i \geq 0$ 的方法數：
$$\Longrightarrow \binom{m+n-1}{m}$$

2. 用 1×2 瓷磚（可旋轉）排滿 $2 \times n$ 長方形的方法數：
$\texttt{Fib}_n = \texttt{Fib}_{n-1} + \texttt{Fib}_{n-2}$, $\texttt{Fib}_1 = 1$, $\texttt{Fib}_0 = 0$

組合 – 一些基本題目

1. $x_1 + x_2 + \cdots + x_n = m$，且每個 $x_i \geq 0$ 的方法數：
$$\Longrightarrow \binom{m+n-1}{m}$$

2. 用 1×2 瓷磚（可旋轉）排滿 $2 \times n$ 長方形的方法數：
$\mathtt{Fib}_n = \mathtt{Fib}_{n-1} + \mathtt{Fib}_{n-2}$, $\mathtt{Fib}_1 = 1$, $\mathtt{Fib}_0 = 0$

問題

[redacted] 有O(1)的演算法 https://openhome.cc/...... /Algorit....../FibonacciNumber.htm

OPENHOME.CC

費式數列

組合 – 費氏數列

$$\begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix}$$

組合 – 費氏數列

$$\begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix}$$

如果求的是模一個數 $m \implies$ 快速冪：$\mathcal{O}(\log n)$。

組合 – 一些基本題目

3. 把正 $n+2$ 邊型三角分割的方法數：

組合 – 一些基本題目

3. 把正 $n+2$ 邊型三角分割的方法數：卡特蘭數

$$C_{m+1} = \sum_k C_k C_{m-k}, \;\; C_0 = 1$$

組合 – 一些基本題目

3. 把正 $n+2$ 邊型三角分割的方法數：卡特蘭數

$$C_{m+1}=\sum_k C_kC_{m-k},\ \ C_0=1$$

4. $n!$ 種 1 到 n 的排列 p，滿足 $p_i \neq i, \forall i$ 的錯排數：

組合 – 一些基本題目

3. 把正 $n+2$ 邊型三角分割的方法數：卡特蘭數

$$C_{m+1} = \sum_k C_k C_{m-k}, \quad C_0 = 1$$

4. $n!$ 種 1 到 n 的排列 p，滿足 $p_i \neq i, \forall i$ 的錯排數：

$$\sum_{k=0}^{n} (-1)^k \binom{n}{k} (n-k)!$$

組合 – 一些基本題目

5. n 個點可以構成的生成樹的個數，點視為相異的：

組合 – 一些基本題目

5. n 個點可以構成的生成樹的個數，點視為相異的：n^{n-2}

組合 – 一些基本題目

5. n 個點可以構成的生成樹的個數，點視為相異的：n^{n-2}
6. 給 n 個數字 $A = \{a_1, a_2, \ldots, a_n\}$，有多少個 A 的子集 B 使得 $\bigoplus_{x \in B} x = 0$，$\bigoplus$ 表示 bit XOR：

組合 – 一些基本題目

5. n 個點可以構成的生成樹的個數，點視為相異的：n^{n-2}
6. 給 n 個數字 $A=\{a_1,a_2,\ldots,a_n\}$，有多少個 A 的子集 B 使得 $\bigoplus_{x\in B} x = 0$，$\bigoplus$ 表示 bit XOR：

$$2^{n-\mathtt{rank}(A)}$$

組合 – 目標

題目 (經典問題)

一個項鍊由 N 個寶石串成，有 M 種不同的寶石，並且因為項鍊是環形的，所以旋轉相同視為相同的。請問有多少種不同的項鍊？

組合 – 目標

題目 (經典問題)

一個項鍊由 N 個寶石串成，有 M 種不同的寶石，並且因為項鍊是環形的，所以旋轉相同視為相同的。請問有多少種不同的項鍊？

1. 有很多**物體**構成一個集合 X。

組合 – 目標

題目 (經典問題)

一個項鍊由 N 個寶石串成，有 M 種不同的寶石，並且因為項鍊是環形的，所以旋轉相同視為相同的。請問有多少種不同的項鍊？

1. 有很多**物體**構成一個集合 X。
2. 有一些**旋轉** G。

組合 – 目標

題目 (經典問題)

一個項鍊由 N 個寶石串成，有 M 種不同的寶石，並且因為項鍊是環形的，所以旋轉相同視為相同的。請問有多少種不同的項鍊？

1. 有很多**物體**構成一個集合 X。
2. 有一些**旋轉** G。
3. 旋轉相同視為相同，也就是旋轉會把物件劃分成許多等價類。

組合 – Burnside

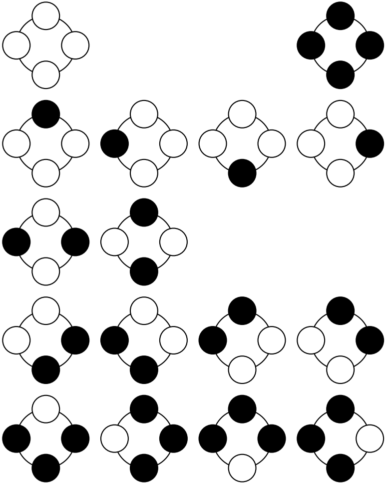


Figure: $M = 2$, $N = 2$ 的例子

組合 – Burnside

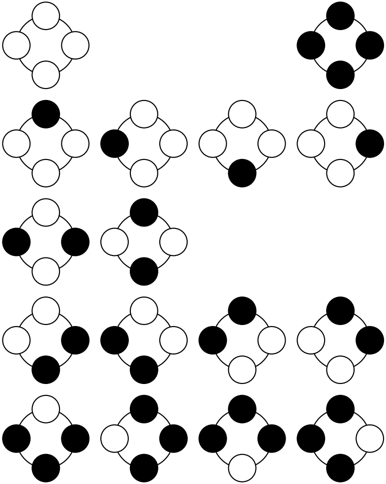


Figure: $M = 2$, $N = 2$ 的例子

- 物體有 $|X| = 2^4 = 16$ 種。

組合 – Burnside

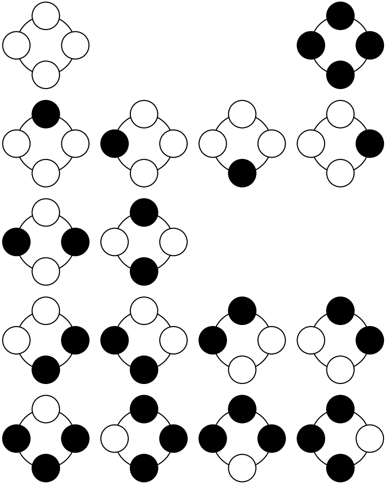


Figure: $M = 2$, $N = 2$ 的例子

- 物體有 $|X| = 2^4 = 16$ 種。
- 令 a 為旋轉 90°，則 $G = \{1, a, a^2, a^3\}$。

組合 – Burnside

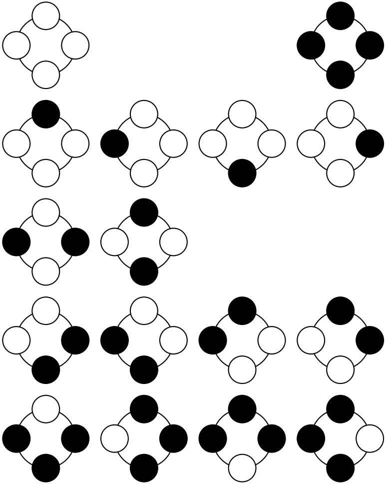


Figure: $M = 2,\ N = 2$ 的例子

- 物體有 $|X| = 2^4 = 16$ 種。
- 令 a 為旋轉 90°，則 $G = \{1, a, a^2, a^3\}$。
- 最後答案是 5。

定義

1. $G_x \triangleq \{g \in G : gx = x\}$，也就是固定 x 下，所有不會動到 x 的作用。

定義

1. $G_x \triangleq \{g \in G : gx = x\}$，也就是固定 x 下，所有不會動到 x 的作用。
2. $X^g \triangleq \{x \in X : gx = x\}$，也就是固定一個作用 g 下的**不動點**。

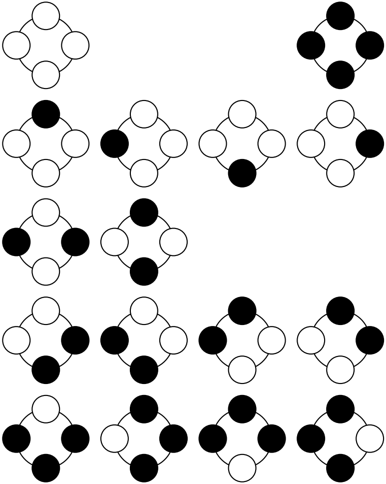
定義

1. $G_x \triangleq \{g \in G : gx = x\}$，也就是固定 x 下，所有不會動到 x 的作用。
2. $X^g \triangleq \{x \in X : gx = x\}$，也就是固定一個作用 g 下的**不動點**。
3. $Gx \triangleq \{gx : g \in G\}$，也被稱作是 x 在 G 下的**軌道**。

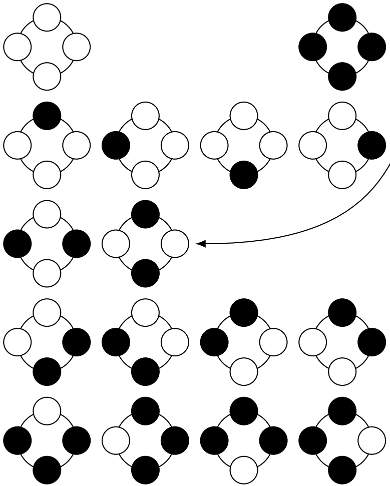
定義

1. $G_x \triangleq \{g \in G : gx = x\}$，也就是固定 x 下，所有不會動到 x 的作用。
2. $X^g \triangleq \{x \in X : gx = x\}$，也就是固定一個作用 g 下的**不動點**。
3. $Gx \triangleq \{gx : g \in G\}$，也被稱作是 x 在 G 下的**軌道**。
4. $X/G \triangleq \{Gx : x \in X\}$，也就是 X 在 G 下所有的軌道。

組合 – 目標



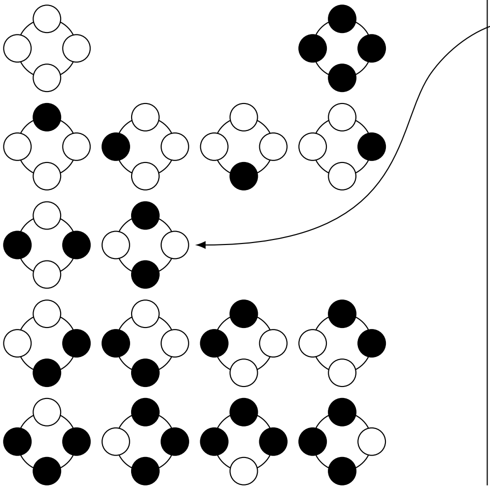
組合 – 目標



令 x 為箭頭所指的物體，
a 為旋轉 90°。

- $G_x = ?$

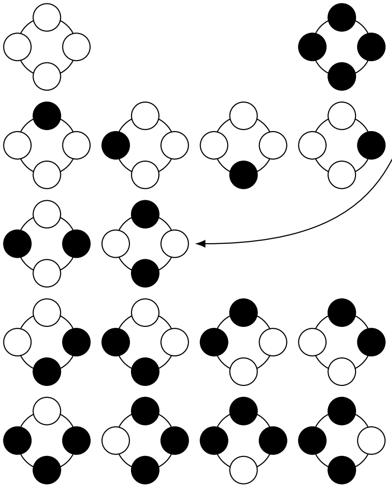
組合 – 目標



令 x 為箭頭所指的物體，a 為旋轉 90°。

- $G_x = \{1, a^2\}$

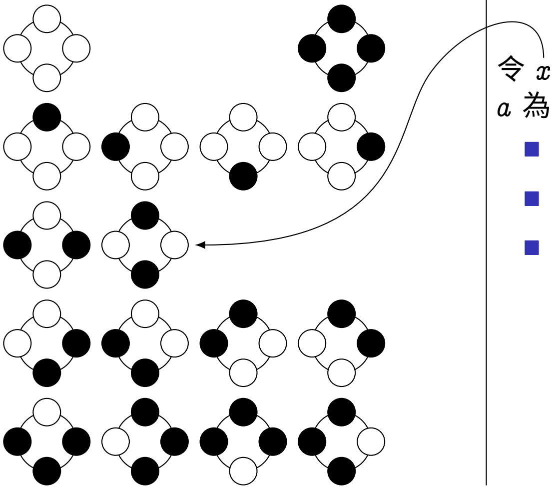
組合 – 目標



令 x 為箭頭所指的物體，a 為旋轉 90°。

- $G_x = \{1, a^2\}$
- $X^{a^2} = ?$

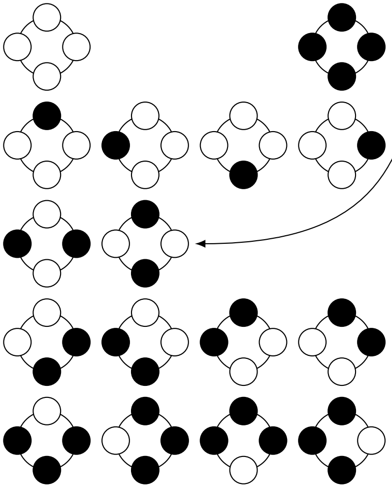
組合 – 目標



令 x 為箭頭所指的物體，
a 為旋轉 90°。

- $G_x = \{1, a^2\}$
- $X^{a^2} = ?$
- $G_x = ?$

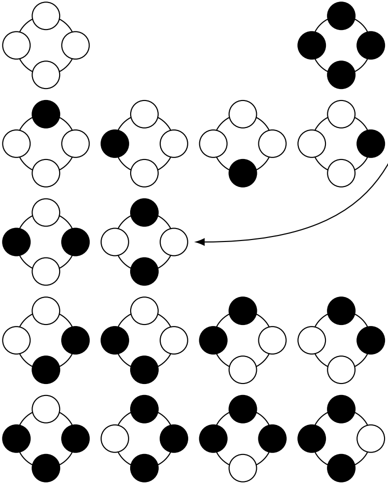
組合 – 目標



令 x 為箭頭所指的物體，a 為旋轉 90°。

- $G_x = \{1, a^2\}$
- $X^{a^2} = ?$
- $G_x = ?$
- $x^G = ?$

組合 – 目標



令 x 為箭頭所指的物體，a 為旋轉 90° 。

- $G_x = \{1, a^2\}$
- $X^{a^2} = ?$
- $G_x = ?$
- $x^G = ?$
- $X/G = ?$

組合 – 目標

定理 (Burnside lemma)

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

組合 – 目標

定理 (Burnside lemma)

$$|X/G| = \frac{1}{|G|}\sum_{g\in G}|X^g|$$

也就是說要算把旋轉相同視為相同的個數：

組合 – 目標

定理 (Burnside lemma)

$$|X/G| = \frac{1}{|G|}\sum_{g\in G}|X^g|$$

也就是說要算把旋轉相同視為相同的個數：

1. 對於每個旋轉 g，計算在這個旋轉下的不動點的個數 $|X^g|$。

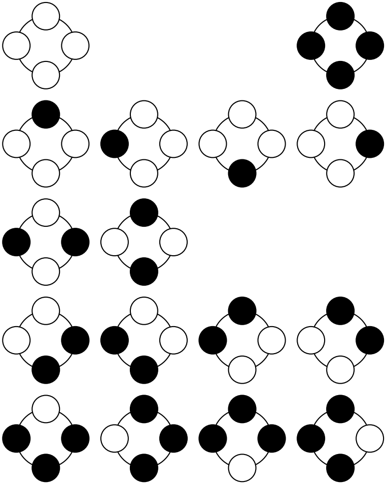
組合 – 目標

定理 (Burnside lemma)

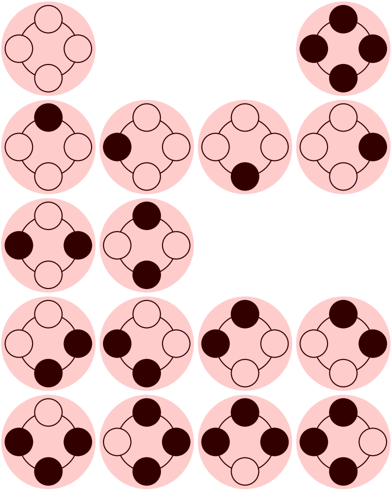
$$|X/G| = \frac{1}{|G|}\sum_{g\in G}|X^g|$$

也就是說要算把旋轉相同視為相同的個數：

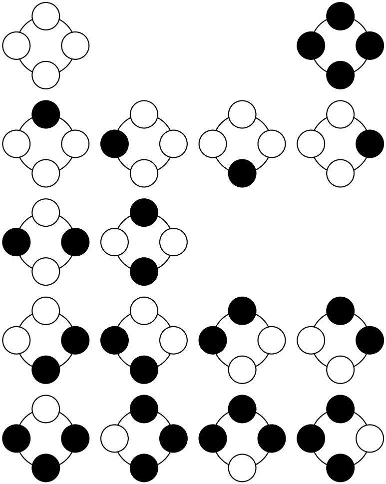
1. 對於每個旋轉 g，計算在這個旋轉下的不動點的個數 $|X^g|$。
2. 把這些數字加起來除以 $|G|$ 就是答案。



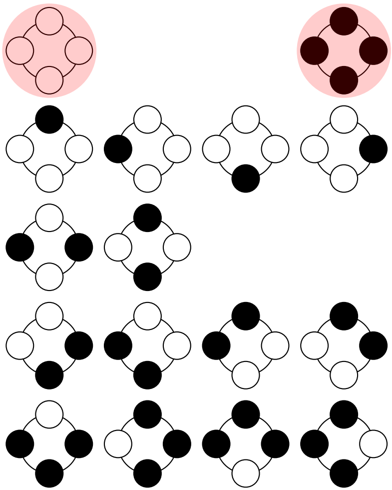
1 $\left|X^1\right| =$



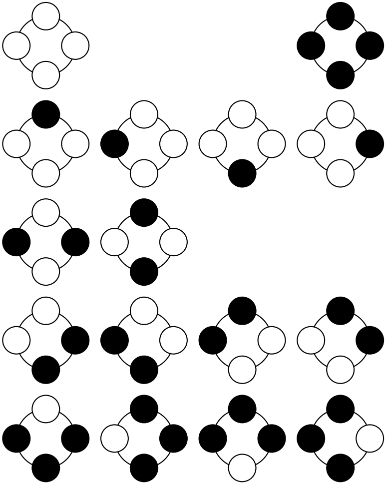
1 $\left|X^1\right| = 16$



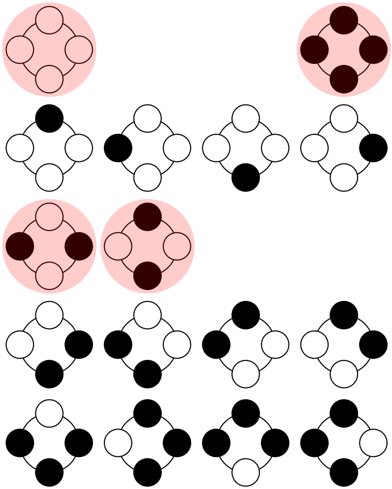
1. $|X^1| = 16$
2. $|X^a| =$



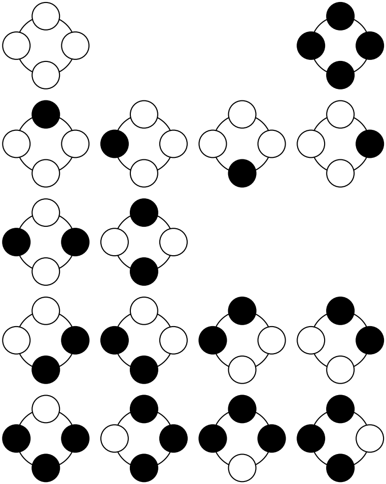
1. $\left|X^1\right| = 16$
2. $\left|X^a\right| = 2$



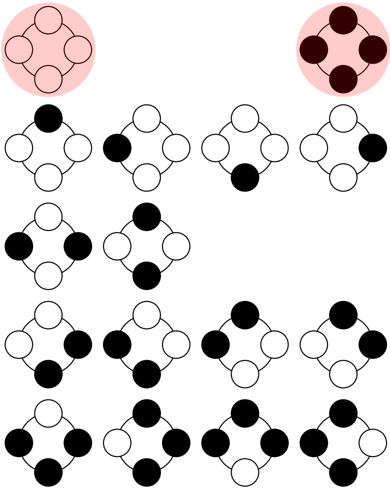
1. $\left|X^1\right| = 16$
2. $\left|X^a\right| = 2$
3. $\left|X^{a^2}\right| =$



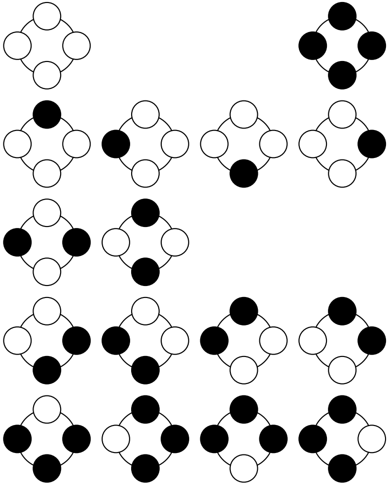
1. $\left|X^{1}\right| = 16$
2. $\left|X^{a}\right| = 2$
3. $\left|X^{a^{2}}\right| = 4$



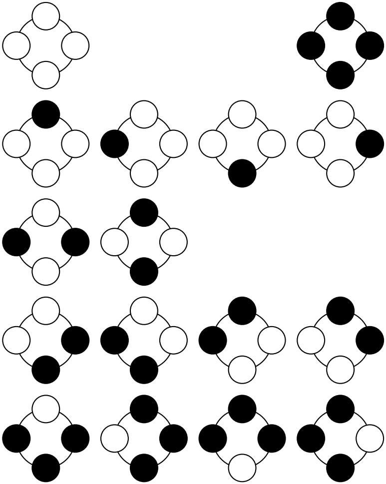
1. $|X^1| = 16$
2. $|X^a| = 2$
3. $|X^{a^2}| = 4$
4. $|X^{a^3}| =$



1. $\left|X^{1}\right| = 16$
2. $\left|X^{a}\right| = 2$
3. $\left|X^{a^2}\right| = 4$
4. $\left|X^{a^3}\right| = 2$



1. $\left|X^1\right| = 16$
2. $\left|X^a\right| = 2$
3. $\left|X^{a^2}\right| = 4$
4. $\left|X^{a^3}\right| = 2$



1. $\left|X^{1}\right| = 16$
2. $\left|X^{a}\right| = 2$
3. $\left|X^{a^2}\right| = 4$
4. $\left|X^{a^3}\right| = 2$

$$\frac{16+2+4+2}{4} = 6$$

證明：

$$\frac{1}{|G|}\sum_{g\in G}|X^g|$$

證明：

$$\frac{1}{|G|}\sum_{g\in G}|X^g| = \frac{1}{|G|}\#\{(x,g) \mid x \in X, g \in G, xg = g\}$$

證明：

$$
\begin{aligned}
\frac{1}{|G|}\sum_{g\in G}|X^g| &= \frac{1}{|G|}\#\{(x,g)\mid x\in X, g\in G, xg=g\} \\
&= \frac{1}{|G|}\sum_{x\in X}|G_x|
\end{aligned}
$$

證明：

$$
\begin{aligned}
\frac{1}{|G|}\sum_{g\in G}|X^g| &= \frac{1}{|G|}\#\{(x,g) \mid x\in X, g\in G, xg=g\} \\
&= \frac{1}{|G|}\sum_{x\in X}|G_x| \\
&= \frac{1}{|G|}\sum_{x\in X}\frac{|G|}{|Gx|}
\end{aligned}
$$

證明：

$$
\begin{aligned}
\frac{1}{|G|}\sum_{g\in G}|X^g| &= \frac{1}{|G|}\#\{(x,g)\mid x\in X, g\in G, xg=g\} \\
&= \frac{1}{|G|}\sum_{x\in X}|G_x| \\
&= \frac{1}{|G|}\sum_{x\in X}\frac{|G|}{|Gx|} \\
&= \sum_{Gx\in X/G}\ \sum_{x\in Gx}\frac{1}{|Gx|} = \sum_{Gx\in X/G} 1
\end{aligned}
$$

證明：

$$
\begin{aligned}
\frac{1}{|G|}\sum_{g\in G}|X^g| &= \frac{1}{|G|}\#\{(x,g)\mid x\in X, g\in G, xg=g\}\\
&= \frac{1}{|G|}\sum_{x\in X}|G_x|\\
&= \frac{1}{|G|}\sum_{x\in X}\frac{|G|}{|Gx|}\\
&= \sum_{Gx\in X/G}\ \sum_{x\in Gx}\frac{1}{|Gx|} = \sum_{Gx\in X/G} 1\\
&= |X/G|
\end{aligned}
$$

C++ 心得分享

C++ 心得分享 – Initializer list & for range

```cpp
int a[3] = {};
int a[3] = {0}; // in C
int b[3] = {1, 2, 3};
vector<int> C = {1, 2, 3};
pair<int, int> p = {2, 4};
```

C++ 心得分享 – Initializer list & for range

```cpp
int a[3] = {};
int a[3] = {0}; // in C
int b[3] = {1, 2, 3};
vector<int> C = {1, 2, 3};
pair<int, int> p = {2, 4};
```

```cpp
vector<int> V = {1, 2, 3};
for (auto x: V) {}
```

C++ 心得分享 – Some tricks

```cpp
#define int long long

int32_t main() {
}
```

C++ 心得分享 – Some tricks

```cpp
#define int long long

int32_t main() {
}
```

```cpp
__int128 a;
long double b;
__float128 c;
```

C++ 心得分享 – Some tricks

```cpp
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef tree<int, null_type, less<int>,
    rb_tree_tag, tree_order_statistics_node_update> set_t;
typedef cc_hash_table<int, int> umap_t;
```

C++ 心得分享 – Some tricks

```cpp
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef tree<int, null_type, less<int>,
    rb_tree_tag, tree_order_statistics_node_update> set_t;
typedef cc_hash_table<int, int> umap_t;
```

```cpp
priority_queue<int, greater<int>> pq;
```

C++ 心得分享 – Some tricks

```cpp
int _A[MX];
int *A = _A + MX/2; // A[-MX/2]
```

C++ 心得分享 – Some tricks

```
int _A[MX];
int *A = _A + MX/2; // A[-MX/2]
```

```
for (int i = 0; i < (1<<n); i++) {
    // for j ⊊ i
    for (int _j = i; _j; _j = (_j-1) & i) { j = i ^ _j }
}
```

C++ 心得分享 – Some tricks

```
int _A[MX];
int *A = _A + MX/2; // A[-MX/2]
```

```
for (int i = 0; i < (1<<n); i++) {
    // for j ⊊ i
    for (int _j = i; _j; _j = (_j-1) & i) { j = i ^ _j }
}
```

```
hypot(a, b) // sqrt(a*a + b*b); Since c++11
clamp(x, lo, hi) // min(max(x, lo), hi) Since c++17
```

C++ 心得分享 – Some tricks

```
while (l < r) {
    int mid = (l+r) / 2;
    bool flag;
    if (...) {
        flag = true;
        goto loop_end;
    }
loop_end:
    if (...) l = mid+1;
    else r = mid;
}
```

C++ 心得分享 – ???

```cpp
while (l < r) {
    int mid = (l+r) / 2;
    try {
        if (...) throw true;
    } catch (bool x) {
        if (...) l = mid+1;
        else r = mid;
    }
}
```

C++ 心得分享 – ???

```cpp
while (l < r) {
    int mid = (l+r) / 2;
    try {
        if (...) throw true;
    } catch (bool x) {
        if (...) l = mid+1;
        else r = mid;
    }
}
```

很慢…

C++ 心得分享 – Lambda

```
while (l < r) {
    int mid = (l+r) / 2;
    bool flag = []() {
        if (...) return true;
    } ();
    if (...) l = mid+1;
    else r = mid;
}
```

C++ 心得分享 – Lambda

```cpp
sort(begin(vec), end(vec),
     [](int a, int b) { return a > b; });
```

C++ 心得分享 – Lambda

```
sort(begin(vec), end(vec),
     [](int a, int b) { return a > b; });
```

```
[=x, &y] (int a, bool b) { ... };
```

```cpp
ios_base::sync_with_stdio(0);
cin.tie(0); // speed up

cout << x << endl; // endl is slow!!
#define endl '\n'
```

C++ 心得分享 – C++ IO

```
ios_base::sync_with_stdio(0);
cin.tie(0); // speed up

cout << x << endl; // endl is slow!!
#define endl '\n'
```

```
for (int i=0; i<vec.size(); i++)
    cout << v[i] << " \n"[i == ((int)vec.size() - 1)];
```

C++ 心得分享 – 大絕招

```cpp
#pragma GCC optimize ("O3")
```

C++ 心得分享 – 大絕招

```cpp
#pragma GCC optimize ("O3")
```

其他的還有 SIMD 等等。

C++ 心得分享 – Increase stack

```cpp
#include <sys/resource.h>
void increase_stack_size() {
    const rlim_t ks = 64*1024*1024;
    struct rlimit rl;
    int res = getrlimit(RLIMIT_STACK, &rl);
    if (res == 0){
        if (rl.rlim_cur < ks){
            rl.rlim_cur = ks;
            res=setrlimit(RLIMIT_STACK, &rl);
        }
    }
}
```

C++ 心得分享 – ???

題目

給你 (x_1, y_1), (x_2, y_2)，請你計算他們的曼哈頓距離。輸入：一行四個 $\pm 10^9$ 內整數用空白格開。

C++ 心得分享 – ???

題目

給你 (x_1, y_1), (x_2, y_2)，請你計算他們的曼哈頓距離。輸入：一行四個 $\pm 10^9$ 內整數用空白格開。

```cpp
#include<bits/stdc++.h>
using namespace std;
int x1, y1, x2, y2;

int main() {
    cin >> x1 >> y1 >> x2 >> y2;
    cout << abs(x1 - x2) + abs(y1 - y2) << endl;
}
```

C++ 心得分享 – static

```
static int C; // (1)
static int f() { // (2)
    static A[30]; // (3)
}
```

C++ 心得分享 – static

```cpp
static int C; // (1)
static int f() { // (2)
    static A[30]; // (3)
}
```

```
> g++ -O2 -Wall -Wextra -Wshadow -o a a.cpp
```

看過很神奇的寫法：

```
struct Tree {
    Tree *l, *r;
    void walk() {
        if (this == NULL) return;
        l->walk();
        r->walk();
    }
}
```

看過很神奇的寫法：

```
struct Tree {
    Tree *l, *r;
    void walk() {
        if (this == NULL) return;
        l->walk();
        r->walk();
    }
}
```

這是 Undefined behavior，非常不推薦！

C++ 心得分享 – Undefined behavior

1. Signed integer overflow

C++ 心得分享 – Undefined behavior

1. Signed integer overflow
2. Out of bound

1. Signed integer overflow
2. Out of bound
3. Uninitialized scalar

C++ 心得分享 – Undefined behavior

1. Signed integer overflow
2. Out of bound
3. Uninitialized scalar
4. Infinite loop without side-effects

C++ 心得分享 – 找 Bug

```cpp
int minus(int a, int b) {
    return a - b;
}
int input() {
    int t;
    cin << t;
    return t;
}
int main() {
    int a, b;
    cout << minus(input(), input()) << endl;
}
```

C++ 心得分享 – 找 Bug

$$\mathbf{E}[t_{\text{AC}}] \;= t_{\text{寫 Code}}$$

C++ 心得分享 – 找 Bug

$$\mathbf{E}[t_{\text{AC}}] = t_{\text{寫 Code}} + p_{\text{出 bug}}\left(\mathbf{E}[t_{\text{你 debug}}]\right.$$

C++ 心得分享 – 找 Bug

$$\mathbf{E}[t_{\text{AC}}] = t_{\text{寫 Code}} + p_{\text{出 bug}}\left(\mathbf{E}[t_{\text{你 debug}}] + p_{\text{你 de 不出來}}\mathbf{E}[t_{\text{隊友 debug}}]\right)$$

C++ 心得分享 – 找 Bug

$$\mathbf{E}[t_{\text{AC}}] = t_{\text{寫 Code}} + p_{\text{出 bug}}\left(\mathbf{E}[t_{\text{你 debug}}] + p_{\text{你 de 不出來}}\mathbf{E}[t_{\text{隊友 debug}}]\right)$$

- 變數名稱取好一點。

C++ 心得分享 – 找 Bug

$$\mathbf{E}[t_{\text{AC}}] = t_{\text{寫 Code}} + p_{\text{出 bug}}\left(\mathbf{E}[t_{\text{你 debug}}] + p_{\text{你 de 不出來}}\mathbf{E}[t_{\text{隊友 debug}}]\right)$$

- 變數名稱取好一點。
- 空白多用一點。

C++ 心得分享 – 找 Bug

$$\mathbf{E}[t_{\text{AC}}] = t_{\text{寫 Code}} + p_{\text{出 bug}}\left(\mathbf{E}[t_{\text{你 debug}}] + p_{\text{你 de 不出來}}\mathbf{E}[t_{\text{隊友 debug}}]\right)$$

- 變數名稱取好一點。
- 空白多用一點。

```
LL mx[ 3 ] = {};
for( size_t i = 1 ; i < f[ 1 ].size() ; i ++ )
    mx[ 1 ] = max( mx[ 1 ] , f[ 1 ][ i ] - f[ 1 ][ i - 1 ] );
```

捲積

捲積 – 目的

形如

$$c_k = \sum_{k=f(i,j)} a_i b_j$$

的和，應用：

捲積 – 目的

形如

$$c_k = \sum_{k=f(i,j)} a_i b_j$$

的和，應用：

- 多項式乘法

捲積 – 目的

形如

$$c_k = \sum_{k=f(i,j)} a_i b_j$$

的和，應用：

- 多項式乘法
- 大數乘法

捲積 – 目的

形如

$$c_k = \sum_{k=f(i,j)} a_i b_j$$

的和，應用：

- 多項式乘法
- 大數乘法
- 組合計數 DP

捲積 – 目的

形如

$$c_k = \sum_{k=f(i,j)} a_i b_j$$

的和，應用：

- 多項式乘法
- 大數乘法
- 組合計數 DP

一般時間複雜度 $\mathcal{O}(n^2)$。

捲積 – 多項式乘法

令

$$f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1},$$

捲積 – 多項式乘法

令

$$f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1},$$
$$g(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1},$$

捲積 – 多項式乘法

令

$$
\begin{aligned}
f(x) &= a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}, \\
g(x) &= b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}, \\
f(x)g(x) &= c_0 + c_1 x + \cdots + c_{2n-1} x^{2n-1}
\end{aligned}
$$

捲積 – 多項式乘法

令

$$
\begin{aligned}
f(x) &= a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}, \\
g(x) &= b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}, \\
f(x)g(x) &= c_0 + c_1 x + \cdots + c_{2n-1} x^{2n-1}
\end{aligned}
$$

捲積 – 多項式乘法

令

$$
\begin{aligned}
f(x) &= a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}, \\
g(x) &= b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}, \\
f(x)g(x) &= c_0 + c_1 x + \cdots + c_{2n-1} x^{2n-1}
\end{aligned}
$$

$$
c_k = \sum_{k=i+j} a_i b_j
$$

捲積 – Karatsuba algorithm

假設 f, g 皆為 $2n-1$ 次的多項式，也就是

$$f(x) \triangleq a_0 + a_1 x + \cdots + a_{2n-1} x^{2n-1}$$

捲積 – Karatsuba algorithm

假設 f, g 皆為 $2n-1$ 次的多項式，也就是

$$f(x) \triangleq a_0 + a_1 x + \cdots + a_{2n-1} x^{2n-1}$$

令

$$f_1(x) \triangleq a_0 + a_1 x + \cdots + a_{n-1} x^n$$
$$f_2(x) \triangleq a_n + a_{n+1} x + \cdots + a_{2n-1} x^n$$

捲積 – Karatsuba algorithm

假設 f, g 皆為 $2n-1$ 次的多項式，也就是

$$f(x) \triangleq a_0 + a_1 x + \cdots + a_{2n-1} x^{2n-1}$$

令

$$\begin{aligned} f_1(x) &\triangleq a_0 + a_1 x + \cdots + a_{n-1} x^n \\ f_2(x) &\triangleq a_n + a_{n+1} x + \cdots + a_{2n-1} x^n \end{aligned}$$

可以知道 $f(x) = f_1(x) + x^n f_2(x)$，類似的定義 g_1, g_2 使得 $g(x) = g_1(x) + x^n g_2(x)$，則

$$\begin{aligned} fg &= f_1 g_1 + x^n (f_1 g_2 + f_2 g_1) + x^{2n} f_2 g_2 \\ &= f_1 g_1 + x^n \big((f_1 + f_2)(g_1 + g_2) - f_1 g_1 - f_2 g_2 \big) + x^{2n} f_2 g_2 \end{aligned}$$

捲積 – Karatsuba algorithm

假設 f, g 皆為 $2n-1$ 次的多項式，也就是

$$f(x) \triangleq a_0 + a_1 x + \cdots + a_{2n-1} x^{2n-1}$$

令

$$f_1(x) \triangleq a_0 + a_1 x + \cdots + a_{n-1} x^n$$
$$f_2(x) \triangleq a_n + a_{n+1} x + \cdots + a_{2n-1} x^n$$

可以知道 $f(x) = f_1(x) + x^n f_2(x)$，類似的定義 g_1, g_2 使得 $g(x) = g_1(x) + x^n g_2(x)$，則

$$\begin{aligned} fg &= f_1 g_1 + x^n (f_1 g_2 + f_2 g_1) + x^{2n} f_2 g_2 \\ &= {\color{red} f_1 g_1} + x^n \left({\color{green}(f_1 + f_2)(g_1 + g_2)} - {\color{red} f_1 g_1} - {\color{blue} f_2 g_2}\right) + x^{2n} {\color{blue} f_2 g_2} \end{aligned}$$

捲積 – Karatsuba algorithm

只要算三個長度一半的多項式乘積。

捲積 – Karatsuba algorithm

只要算三個長度一半的多項式乘積。

複雜度：

$$T(n) = 3T(n/2) + \mathcal{O}(n) \implies T(n) = \mathcal{O}\left(n^{\log_2 3}\right)$$

捲積 – Karatsuba algorithm

只要算三個長度一半的多項式乘積。

複雜度：

$$T(n) = 3T(n/2) + \mathcal{O}(n) \implies T(n) = \mathcal{O}\left(n^{\log_2 3}\right)$$

實作上要跑的快常數小要注意：

- 如果用 `std::vector` 請用 `std::vector::reserve`。
- 還是建議自己弄個 buffer。

捲積 – 摸都嗨呀苦

n 個點 (x, y) 決定了一個多項式，我們的夢想是這樣的：

捲積 – 摸都嗨呀苦

n 個點 (x, y) 決定了一個多項式，我們的夢想是這樣的：

1. 用 $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$ 表示 f。

捲積 – 摸都嗨呀苦

n 個點 (x, y) 決定了一個多項式，我們的夢想是這樣的：

1. 用 $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$ 表示 f。
2. 同樣的用 $(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_n, g(x_n))$ 表示 g。

捲積 – 摸都嗨呀苦

n 個點 (x, y) 決定了一個多項式，我們的夢想是這樣的：

1. 用 $(x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_n, f(x_n))$ 表示 f。
2. 同樣的用 $(x_1, g(x_1)), (x_2, g(x_2)), \dots, (x_n, g(x_n))$ 表示 g。
3. 那 $(x_1, f(x_1)g(x_1)), \dots, (x_n, f(x_n)g(x_n))$ 就表示 $f \cdot g$。
   只要 n 次乘法！

捲積 – 摸都嗨呀苦

n 個點 (x, y) 決定了一個多項式，我們的夢想是這樣的：

1. 用 $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$ 表示 f。
2. 同樣的用 $(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_n, g(x_n))$ 表示 g。
3. 那 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 就表示 $f \cdot g$。
   只要 n 次乘法！
4. 再反推的 $f \cdot g$。

捲積 – 摸都嗨呀苦

n 個點 (x, y) 決定了一個多項式，我們的夢想是這樣的：

1. 用 $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$ 表示 f。
2. 同樣的用 $(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_n, g(x_n))$ 表示 g。
3. 那 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 就表示 $f \cdot g$。
只要 n 次乘法！
4. 再反推的 $f \cdot g$。

光這一步就要 $O(n^2)$！

捲積 – 摸都嗨呀苦

n 個點 (x, y) 決定了一個多項式，我們的夢想是這樣的：

1. 用 $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$ 表示 f。
2. 同樣的用 $(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_n, g(x_n))$ 表示 g。
3. 那 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 就表示 $f \cdot g$。只要 n 次乘法！
4. 再反推的 $f \cdot g$。

光這一步就要 $O(n^2)$！

找的 $x_1, x_2, \ldots, x_n$ 要夠好。

捲積 – 離散傅立葉變換

找 $x_k = \omega_n^k$, $0 \le k < n$，其中 $\omega_n \triangleq \mathrm{e}^{2\pi\mathrm{i}/n}$ 。

捲積 – 離散傅立葉變換

找 $x_k = \omega_n^k,\ 0 \le k < n$，其中 $\omega_n \triangleq \mathrm{e}^{2\pi\mathrm{i}/n}$ 。
現在
$f(x_k) = a_0 + a_1\omega_n^k + a_2\omega_n^{2k} + \cdots + a_{n-1}\omega_n^{(n-1)k} = \sum_{i=0}^{n-1} a_i\omega_n^{ik}$ 。

捲積 – 離散傅立葉變換

找 $x_k = \omega_n^k$, $0 \le k < n$，其中 $\omega_n \triangleq \mathrm{e}^{2\pi\mathrm{i}/n}$ 。
現在
$f(x_k) = a_0 + a_1\omega_n^k + a_2\omega_n^{2k} + \cdots + a_{n-1}\omega_n^{(n-1)k} = \sum_{i=0}^{n-1} a_i\omega_n^{ik}$ 。

也可寫成

$$
\begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{(n-1)} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{(2n-2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{(2n-2)} & \cdots & \omega_n^{(n^2-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}
$$

捲積 – 離散傅立葉變換

找 $x_k = \omega_n^k$, $0 \le k < n$，其中 $\omega_n \triangleq e^{2\pi i/n}$ 。
現在
$f(x_k) = a_0 + a_1\omega_n^k + a_2\omega_n^{2k} + \cdots + a_{n-1}\omega_n^{(n-1)k} = \sum_{i=0}^{n-1} a_i\omega_n^{ik}$ 。

也可寫成

$$\begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{(n-1)} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{(2n-2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{(2n-2)} & \cdots & \omega_n^{(n^2-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

把 $(f(x_0), \ldots, f(x_{n-1}))$ 稱作 $A = (a_0, \ldots, a_{n-1})$ 的**離散傅立葉變換** $\mathcal{F}^n(A)$ 。

捲積 – 離散傅立葉變換

$$
\begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{m-1}) \\ \vdots \\ f(x_n) \end{bmatrix}
=
\begin{bmatrix}
1 & 1 & 1 & 1 & \cdots & 1 & 1 \\
1 & \omega_n & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{(n-2)} & \omega_n^{(n-1)} \\
1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(n-2)} & \omega_n^{2(n-1)} \\
1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(n-2)} & \omega_n^{3(n-1)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & \omega_n^{(m-1)} & \omega_n^{2(m-1)} & \omega_n^{3(m-1)} & \cdots & \omega_n^{(m-1)(n-2)} & \omega_n^{(m-1)(n-1)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)(n-2)} & \omega_n^{(n-1)(n-1)}
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{bmatrix}
$$

捲積 – 離散傅立葉變換

$$\begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{m-1}) \\ \vdots \\ f(x_n) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \ldots & \omega_n^{(n-2)} & \omega_n^{(n-1)} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \ldots & \omega_n^{2(n-2)} & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \ldots & \omega_n^{3(n-2)} & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \omega_n^{(m-1)} & \omega_n^{2(m-1)} & \omega_n^{3(m-1)} & \ldots & \omega_n^{(m-1)(n-2)} & \omega_n^{(m-1)(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \ldots & \omega_n^{(n-1)(n-2)} & \omega_n^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{bmatrix}$$

$$\mathcal{F}(A)_k$$

捲積 – 離散傅立葉變換

$$\begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{m-1}) \\ \vdots \\ f(x_n) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & \omega_n & \omega_m & \omega_n^3 & \cdots & \omega_m^{(m-1)} & \omega_n^{(n-1)} \\ 1 & \omega_n^2 & \omega_m^2 & \omega_n^6 & \cdots & \omega_m^{2(m-1)} & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_m^3 & \omega_n^9 & \cdots & \omega_m^{3(m-1)} & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \omega_n^{(m-1)} & \omega_m^{(m-1)} & \omega_n^{3(m-1)} & \cdots & \omega_m^{(m-1)(m-1)} & \omega_n^{(m-1)(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)(n-2)} & \omega_n^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{bmatrix}$$

$$\mathcal{F}(A)_k = \mathcal{F}(E)_k$$

偶數項

捲積 – 離散傅立葉變換

$$
\begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{m-1}) \\ \vdots \\ f(x_n) \end{bmatrix}
=
\begin{bmatrix}
1 & 1 & 1 & 1 & \cdots & 1 & 1 \\
1 & \omega_n & \omega_m & \omega_n^3 & \cdots & \omega_m^{(m-1)} & \omega_n^{(n-1)} \\
1 & \omega_n^2 & \omega_m^2 & \omega_n^6 & \cdots & \omega_m^{2(m-1)} & \omega_n^{2(n-1)} \\
1 & \omega_n^3 & \omega_m^3 & \omega_n^9 & \cdots & \omega_m^{3(m-1)} & \omega_n^{3(n-1)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & \omega_n^{(m-1)} & \omega_m^{(m-1)} & \omega_n^{3(m-1)} & \cdots & \omega_m^{(m-1)(m-1)} & \omega_n^{(m-1)(n-1)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)(n-2)} & \omega_n^{(n-1)(n-1)}
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{bmatrix}
$$

$$\mathcal{F}(A)_k = \mathcal{F}(E)_k$$

偶數項

捲積 – 離散傅立葉變換

$$\begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{m-1}) \\ \vdots \\ f(x_n) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 & 1 \\ 1 & \omega_n \cdot 1 & \omega_m & \omega_n \cdot \omega_m^1 & \dots & \omega_m^{(m-1)} & \omega_n \cdot \omega_m^{(m-1)} \\ 1 & \omega_n^2 \cdot 1 & \omega_m^2 & \omega_n^2 \cdot \omega_m^2 & \dots & \omega_m^{2(m-1)} & \omega_n^2 \cdot \omega_m^{(m-1)} \\ 1 & \omega_n^3 & \omega_m^3 & \omega_n^9 & \dots & \omega_m^{3(m-1)} & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \omega_n^{(m-1)} & \omega_m^{(m-1)} & \omega_n^{3(m-1)} & \dots & \omega_m^{(m-1)(m-1)} & \omega_n^{(m-1)(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \dots & \omega_n^{(n-1)(n-2)} & \omega_n^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{bmatrix}$$

$$\mathcal{F}(A)_k = \mathcal{F}(E)_k + \omega_n^k \mathcal{F}(O)_k$$

偶數項

奇數項

捲積 – 離散傅立葉變換

$$F(A)_k = \sum_{j=0}^{n-1} \omega_n^{jk} A_j$$

捲積 – 離散傅立葉變換

$$
\begin{aligned}
F(A)_k &= \sum_{j=0}^{n-1} \omega_n^{jk} A_j \\
&= \sum_{j=0}^{n-1} \mathrm{e}^{2\pi \mathrm{i} jk/n} A_j
\end{aligned}
$$

捲積 – 離散傅立葉變換

$$\begin{aligned} F(A)_k &= \sum_{j=0}^{n-1} \omega_n^{jk} A_j \\ &= \sum_{j=0}^{n-1} \mathrm{e}^{2\pi \mathrm{i} jk/n} A_j \\ &= \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i} \cdot (2j) \cdot k/n} A_{2j} \ + \ \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i} \cdot (2j+1) \cdot k/n} A_{2j+1} \end{aligned}$$

捲積 – 離散傅立葉變換

$$
\begin{aligned}
F(A)_k &= \sum_{j=0}^{n-1} \omega_n^{jk} A_j \\
&= \sum_{j=0}^{n-1} \mathrm{e}^{2\pi \mathrm{i} jk/n} A_j \\
&= \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i} \cdot (2j) \cdot k/n} A_{2j} \quad + \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i} \cdot (2j+1) \cdot k/n} A_{2j+1} \\
&= \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i} jk/m} E_j \qquad + \mathrm{e}^{2\pi \mathrm{i} k/n} \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i} jk/m} O_j
\end{aligned}
$$

捲積 – 離散傅立葉變換

$$
\begin{aligned}
F(A)_k &= \sum_{j=0}^{n-1} \omega_n^{jk} A_j \\
&= \sum_{j=0}^{n-1} \mathrm{e}^{2\pi \mathrm{i} jk/n} A_j \\
&= \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i}\cdot(2j)\cdot k/n} A_{2j} &&+ \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i}\cdot(2j+1)\cdot k/n} A_{2j+1} \\
&= \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i} jk/m} E_j &&+ \mathrm{e}^{2\pi \mathrm{i} k/n} \sum_{j=0}^{m-1} \mathrm{e}^{2\pi \mathrm{i} jk/m} O_j \\
&= \mathcal{F}(E)_{k \bmod m} &&+ \mathrm{e}^{2\pi \mathrm{i} k/n} \mathcal{F}(O)_{k \bmod m}
\end{aligned}
$$

捲積 – 離散傅立葉變換

複雜度：

$$T(n) = 2T(n/2) + \mathcal{O}(n)$$

捲積 – 離散傅立葉變換

複雜度：

$$T(n) = 2T(n/2) + \mathcal{O}(n)$$

$$\implies T(n) = \mathcal{O}(n \log n)$$

捲積 – 離散傅立葉變換

前面的「夢想」：

3. 那 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 就表示 $f \cdot g$。
   只要 n 次乘法！

捲積 – 離散傅立葉變換

前面的「夢想」：

3. 那 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 就表示 $f \cdot g$。
   只要 n 次乘法！

其怪？$f \cdot g$ 不是應該是 $2n+1$ 次多項式嗎？

捲積 – 離散傅立葉變換

前面的「夢想」：

3 那 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 就表示 $f \cdot g$。只要 n 次乘法！

其怪？$f \cdot g$ 不是應該是 $2n+1$ 次多項式嗎？

我們其實是計算

$$c_k \triangleq \sum_{k \equiv i+j \bmod n} a_i b_j$$

適當的在前面補 0 還是可以計算多項式乘法！

捲積 – 離散傅立葉變換

1. 用 $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$ 表示 f。
2. 用 $(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_n, g(x_n))$ 表示 g。
3. 計算 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 表示 $f \cdot g$。
4. 反推 $f \cdot g$。

捲積 – 離散傅立葉變換

1. 計算 $\mathcal{F}(A)$。 ($\mathcal{O}(n\log n)$ ✓)
2. 用 $(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_n, g(x_n))$ 表示 g。
3. 計算 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 表示 $f \cdot g$。
4. 反推 $f \cdot g$。

捲積 – 離散傅立葉變換

1. 計算 $\mathcal{F}(A)$。 ($\mathcal{O}(n \log n)$ ✓)
2. 計算 $\mathcal{F}(B)$。 ($\mathcal{O}(n \log n)$ ✓)
3. 計算 $(x_1, f(x_1)g(x_1)), \ldots, (x_n, f(x_n)g(x_n))$ 表示 $f \cdot g$。
4. 反推 $f \cdot g$。

捲積 – 離散傅立葉變換

1. 計算 $\mathcal{F}(A)$。 ($\mathcal{O}(n \log n)$ ✓)
2. 計算 $\mathcal{F}(B)$。 ($\mathcal{O}(n \log n)$ ✓)
3. 計算 $\mathcal{F}(C) = \mathcal{F}(A) \odot \mathcal{F}(B)$。 ($\mathcal{O}(n)$ ✓)
4. 反推 $f \cdot g$。

捲積 – 離散傅立葉變換

1. 計算 $\mathcal{F}(A)$。 ($\mathcal{O}(n \log n)$ ✓)
2. 計算 $\mathcal{F}(B)$。 ($\mathcal{O}(n \log n)$ ✓)
3. 計算 $\mathcal{F}(C) = \mathcal{F}(A) \odot \mathcal{F}(B)$。 ($\mathcal{O}(n)$ ✓)
4. 計算 $C = \mathcal{F}^{-1}(\mathcal{F}(C))$。 ?

捲積 – 離散傅立葉變換

1. 計算 $\mathcal{F}(A)$。 ($\mathcal{O}(n \log n)$ ✓)
2. 計算 $\mathcal{F}(B)$。 ($\mathcal{O}(n \log n)$ ✓)
3. 計算 $\mathcal{F}(C) = \mathcal{F}(A) \odot \mathcal{F}(B)$。 ($\mathcal{O}(n)$ ✓)
4. 計算 $C = \mathcal{F}^{-1}(\mathcal{F}(C))$。 ?

定理 (離散傅立葉變換逆變換)

如果 $\mathcal{F}(A) = B$，則

$$A_k = \frac{1}{n} \sum_{j=0}^{n-1} \omega_n^{-jk} B_j$$

捲積 – 離散傅立葉變換

1. 計算 $\mathcal{F}(A)$。 ($\mathcal{O}(n\log n)$ ✓)
2. 計算 $\mathcal{F}(B)$。 ($\mathcal{O}(n\log n)$ ✓)
3. 計算 $\mathcal{F}(C)=\mathcal{F}(A)\odot\mathcal{F}(B)$。 ($\mathcal{O}(n)$ ✓)
4. 計算 $C=\mathcal{F}^{-1}(\mathcal{F}(C))$。 ?

定理 (離散傅立葉變換逆變換)

如果 $\mathcal{F}(A)=B$，則

$$A_k=\frac{1}{n}\sum_{j=0}^{n-1}\omega_n^{-jk}B_j$$

和正變換一樣，只要將 $\omega\to\omega^{-1}$

捲積 – 離散傅立葉變換

1. 計算 $\mathcal{F}(A)$。 $(\mathcal{O}(n\log n)\ \checkmark)$
2. 計算 $\mathcal{F}(B)$。 $(\mathcal{O}(n\log n)\ \checkmark)$
3. 計算 $\mathcal{F}(C) = \mathcal{F}(A) \odot \mathcal{F}(B)$。 $(\mathcal{O}(n)\ \checkmark)$
4. 計算 $C = \mathcal{F}^{-1}(\mathcal{F}(C))$。 $(\mathcal{O}(n\log n)\ \checkmark)$

定理 (離散傅立葉變換逆變換)

如果 $\mathcal{F}(A) = B$，則

$$A_k = \frac{1}{n}\sum_{j=0}^{n-1} \omega_n^{-jk} B_j$$

捲積 – 離散傅立葉變換

和 Karatsuba algorithm 一樣，要實作上效率高需要巧思。

```
void fft(int n, cplx a[], bool inv=false)
{
    int i = 0;
    for (int j = 1; j < n - 1; j++) {
        for (int k = n >> 1; k > (i ^= k); k >>= 1);
        if (j < i) swap(a[i], a[j]);
    }
    // ...
```

捲積 – 離散傅立葉變換

和 Karatsuba algorithm 一樣，要實作上效率高需要巧思。

```
void fft(int n, cplx a[], bool inv=false)
{
    int i = 0;
    for (int j = 1; j < n - 1; j++) {
        for (int k = n >> 1; k > (i ^= k); k >>= 1);
        if (j < i) swap(a[i], a[j]);
    }
    // ...
```

Question: 這個 i 依序是多少？

捲積 – 離散傅立葉變換

```
double d = inv ? 1 : -1;
for (int m = 2; m <= n; m <<= 1) {
    int mh = m >> 1;
    for (int j = 0; j < mh; j++) {
        cplx w = exp(cplx(0, d * PI * j / mh));
        for (int k = j; k < n; k += m) {
            int l = k + mh;
            cplx x = a[k] - w*a[l];
            a[k] = a[k] + w*a[l];
            a[l] = x;
        }
    }
}
```

捲積 – 另一個問題

題目 (經典問題)

有 n 個技能 m 個人，每個人都會這 n 個技能中的某一些。如果 B 會的所有技能 A 也會，那我們就說 A 完全贏過 B。對每個人輸出他完全贏過多少人（包含自己）。$(1 \leq n \leq 25,\ m \leq 10^6)$

[1]引用自 2017 年 IOI-Camp 捲積講義

捲積 – 另一個問題

題目 (經典問題)

有 n 個技能 m 個人，每個人都會這 n 個技能中的某一些。如果 B 會的所有技能 A 也會，那我們就說 A 完全贏過 B。對每個人輸出他完全贏過多少人（包含自己）。$(1 \leq n \leq 25,\ m \leq 10^6)$

你說這還不簡單：

```
for (int i = 0; i < (1<<n); i++) {
    for (int j: j | i == i) {
        dp[i] += cnt[j];
    }
}
```

[1]引用自 2017 年 IOI-Camp 捲積講義

捲積 – 另一個問題

題目 (經典問題)

有 n 個技能 m 個人，每個人都會這 n 個技能中的某一些。如果 B 會的所有技能 A 也會，那我們就說 A 完全贏過 B。對每個人輸出他完全贏過多少人（包含自己）。$(1 \leq n \leq 25,\ m \leq 10^6)$

你說這還不簡單：

```
for (int i = 0; i < (1<<n); i++) {
    for (int j: j | i == i) {
        dp[i] += cnt[j];
    }
}
```

「你開心地想要直接揍他，一揍下去不得了痛死了，你像成龍一樣甩手時發現 $n = 25$。」[1]

[1]引用自 2017 年 IOI-Camp 捲積講義

捲積 – 另一個問題

```
for (int k = 0; k < n; k++) {
    for (int i = 0; i < (1<<n); i++) {
        dp[k+1][i] = dp[k][i]; // 約定 dp[0][i] = cnt[i]
        if (i & (1 << k))
            dp[k+1][i] += dp[k][i ^ (1<<k)];
    }
}
```

捲積 – 另一個問題

```
for (int k = 0; k < n; k++) {
    for (int i = 0; i < (1<<n); i++) {
        dp[k+1][i] = dp[k][i]; // 約定 dp[0][i] = cnt[i]
        if (i & (1 << k))
            dp[k+1][i] += dp[k][i ^ (1<<k)];
    }
}
```

`dp[k+1][i]` 表示「假設你會的技能的集合為 i，則你完全贏過，且你多會的技能只有在前 $k+1$ 個技能當中」的人數。

捲積 – 另一個問題

```
for (int k = 0; k < n; k++) {
    for (int i = 0; i < (1<<n); i++) {
        dp[k+1][i] = dp[k][i]; // 約定 dp[0][i] = cnt[i]
        if (i & (1 << k))
            dp[k+1][i] += dp[k][i ^ (1<<k)];
    }
}
```

`dp[k+1][i]` 表示「假設你會的技能的集合為 i，則你完全贏過，且你多會的技能只有在前 $k+1$ 個技能當中」的人數。

1. 如果他會技能 k 但你會 $\Longrightarrow$ 他不會更前面的技能 $\Longrightarrow$ 在 `dp[k][i]` 算過。

捲積 – 另一個問題

```
for (int k = 0; k < n; k++) {
    for (int i = 0; i < (1<<n); i++) {
        dp[k+1][i] = dp[k][i]; // 約定 dp[0][i] = cnt[i]
        if (i & (1 << k))
            dp[k+1][i] += dp[k][i ^ (1<<k)];
    }
}
```

`dp[k+1][i]` 表示「假設你會的技能的集合為 i，則你完全贏過，且你多會的技能只有在前 $k+1$ 個技能當中」的人數。

1. 如果他會技能 k 但你會 $\Longrightarrow$ 他不會更前面的技能 $\Longrightarrow$ 在 `dp[k][i]` 算過。
2. 如果他不會技能 i 但你會 $\Longrightarrow$ 在 `dp[k-1][i]` 算過

捲積 – 另一個變換

考慮多項式

$$F(x_1, \ldots, x_n) = \sum_{I=\{i_1,\ldots,i_k\}\subseteq[n]} a_I x_{i_1} x_{i_2} \cdots x_{i_k} = \sum_{I\subseteq[n]} a_I x_I$$

a_I 表示會的技能的集合為 I 的人數。

捲積 – 另一個變換

考慮多項式

$$F(x_1, \ldots, x_n) = \sum_{I=\{i_1,\ldots,i_k\}\subseteq[n]} a_I x_{i_1} x_{i_2} \cdots x_{i_k} = \sum_{I\subseteq[n]} a_I x_I$$

a_I 表示會的技能的集合為 I 的人數。

我們最後會得到一個新的多項式：

$$\tilde{F}(x_1, \ldots, x_n) \triangleq \sum_{I\subseteq[n]} \left(\sum_{J\subseteq I} a_J \right) x_I$$

捲積 – 又一個捲積

題目 (Two Swords Style, Weekly Training Farm 16 pB)

給你 $a_0, a_1, \ldots, a_{2^n-1}$，$b_0, b_1, \ldots, b_{2^n-1}$，請你求出

$$c_k \triangleq \sum_{k=i\,|\,j} a_i b_j$$

對所有 $0 \leq k < 2^n$，其中 $|$ 表示 bit OR。$(n \leq 25)$

捲積 – 又一個捲積

題目 (Two Swords Style, Weekly Training Farm 16 pB)

給你 $a_0, a_1, \ldots, a_{2^n-1}$，$b_0, b_1, \ldots, b_{2^n-1}$，請你求出

$$c_k \triangleq \sum_{k=i\,|\,j} a_i b_j$$

對所有 $0 \le k < 2^n$，其中 $|$ 表示 bit OR。($n \le 25$)

硬揍下去 $\mathcal{O}(2^{2n}) \implies$ 痛的不得了。

捲積 – 做法

1. 把每個數看成集合：如果 x 在 $i_1, i_2, \ldots, i_k$ 的 bit 為 1，則對應到 $\{i_1, i_2, \ldots, i_k\}$。

捲積 – 做法

1. 把每個數看成集合：如果 x 在 $i_1, i_2, \ldots, i_k$ 的 bit 為 1，則對應到 $\{i_1, i_2, \ldots, i_k\}$。
2. 我們改成計算

$$\tilde{c}_x = \sum_{y \subseteq x} c_y$$

捲積 – 做法

1. 把每個數看成集合：如果 x 在 $i_1, i_2, \ldots, i_k$ 的 bit 為 1，則對應到 $\{i_1, i_2, \ldots, i_k\}$。
2. 我們改成計算
$$\tilde{c}_x = \sum_{y \subseteq x} c_y$$
3. 容易知道
$$\tilde{c}_x = \left(\sum_{y \subseteq x} a_y\right)\left(\sum_{y \subseteq x} b_y\right)$$

捲積 – 做法

1. 把每個數看成集合：如果 x 在 $i_1, i_2, \ldots, i_k$ 的 bit 為 1，則對應到 $\{i_1, i_2, \ldots, i_k\}$。
2. 我們改成計算
$$\tilde{c}_x = \sum_{y \subseteq x} c_y$$
3. 容易知道
$$\tilde{c}_x = \left(\sum_{y \subseteq x} a_y\right)\left(\sum_{y \subseteq x} b_y\right)$$
4. 再用排容從 $\tilde{c}$ 得出 c。

捲積 – 另一種看法

假設我們多項是的變數在 $\mathbb{F}_2$ 下，也就是 $x_i^2 = x_i$。令

$$F(x) = \sum a_I x_I, \ G(x) = \sum a_I x_I$$

捲積 – 另一種看法

假設我們多項是的變數在 $\mathbb{F}_2$ 下，也就是 $x_i^2 = x_i$。令

$$F(x) = \sum a_I x_I, \; G(x) = \sum a_I x_I$$

我們要求的其實就是

$$F(x)G(x) = \sum_K \sum_{I \mid J = K} a_I b_J x_K$$

捲積 – 另一種看法

1. 令
$$F(x) = \sum a_I x_I, \ G(x) = \sum b_I x_I$$

捲積 – 另一種看法

1. 令

$$F(x) = \sum a_I x_I,\ G(x) = \sum b_I x_I$$

2. 換成點值，$F(y)$ 表示「如果 $t \in y$ 則 $x_t = 1$，否則 $x_t = 0$」代入後的值。

$$F(x) \longleftrightarrow (F(0), F(1), \ldots, F(2^n - 1))$$

捲積 – 另一種看法

1. 令
$$F(x) = \sum a_I x_I,\ G(x) = \sum b_I x_I$$
2. 換成點值，$F(y)$ 表示「如果 $t \in y$ 則 $x_t = 1$，否則 $x_t = 0$」代入後的值。
$$F(x) \longleftrightarrow (F(0), F(1), \ldots, F(2^n - 1))$$
3. 計算點值相乘
$$(F(0)G(0), F(1)G(1), \ldots, F(2^n - 1)G(2^n - 1))$$

捲積 – 另一種看法

1. 令
$$F(x) = \sum a_I x_I,\ G(x) = \sum b_I x_I$$
2. 換成點值，$F(y)$ 表示「如果 $t \in y$ 則 $x_t = 1$，否則 $x_t = 0$」代入後的值。
$$F(x) \longleftrightarrow (F(0), F(1), \ldots, F(2^n - 1))$$
3. 計算點值相乘
$$(F(0)G(0), F(1)G(1), \ldots, F(2^n - 1)G(2^n - 1))$$
4. 反推回去 $F(x)G(x)$。

捲積 – 另一種看法

點值其實等於

$$F(x) = \sum_{y \subseteq x} a_y$$

捲積 – 另一種看法

點值其實等於

$$F(x) = \sum_{y \subseteq x} a_y$$

有

$$\sum_{I \subseteq [n]} F(x) \tilde{x}_I = \sum_{I \subseteq [n]} \left(\sum_{J \subseteq I} a_J \right) \tilde{x}_I$$

捲積 – 另一種看法

點值其實等於

$$F(x) = \sum_{y \subseteq x} a_y$$

有

$$\sum_{I \subseteq [n]} F(x) \tilde{x}_I = \sum_{I \subseteq [n]} \left(\sum_{J \subseteq I} a_J \right) \tilde{x}_I$$

這就是我們在「技能問題」中說的變換！

捲積 – 另一種看法

逆變換也很簡單：

```
for (int k = n-1; k >= 0; k--) {
for (int i = (1<<n)-1; i >= 0; i--) {
        dp[k][i] = dp[k+1][i];
        if (i & (1 << k))
            dp[k][i] -= dp[k][i ^ (1<<k)];
    }
}
```

捲積 – 舉一反三

題目 (經典問題)

給你 $a_0, a_1, \ldots, a_{2^n-1}$，$b_0, b_1, \ldots, b_{2^n-1}$，請你求出

$$c_k \triangleq \sum_{k=i\oplus j} a_i b_j$$

對所有 $0 \le k < 2^n$，其中 $\oplus$ 表示 bit XOR。($n \le 25$)

捲積 – 舉一反三

剛剛我們需要 $x^2 = x$，所以代值 $\{0, 1\}$。

捲積 – 舉一反三

剛剛我們需要 $x^2 = x$，所以代值 $\{0, 1\}$。

現在我們需要 $x = x^{-1}$，代值 $\{-1, 1\}$。
也就是說我們改求點值

$$F(x) \longleftrightarrow (F(0), F(1), \dots, F(2^n - 1))$$

捲積 – 舉一反三

剛剛我們需要 $x^2 = x$，所以代值 $\{0, 1\}$。

現在我們需要 $x = x^{-1}$，代值 $\{-1, 1\}$。
也就是說我們改求點值

$$F(x) \longleftrightarrow (F(0), F(1), \ldots, F(2^n - 1))$$

其中 $F(y)$ 表示「如果 $t \in y$ 則 $x_t = -1$，否則 $x_t = 1$」代入後的值。

捲積 – 舉一反三

令 $F(x) = F_0(x_2, x_3, \ldots, x_{n-1}) + F_1(x_2, x_3, \ldots, x_{n-1})x_1$，且 $I' = I \setminus \{1\}$

捲積 – 舉一反三

令 $F(x) = F_0(x_2, x_3, \ldots, x_{n-1}) + F_1(x_2, x_3, \ldots, x_{n-1})x_1$，且 $I' = I \setminus \{1\}$

$$\tilde{F}(x) = \sum_{I \subseteq [n]} F(I)x_I$$

捲積 – 舉一反三

令 $F(x) = F_0(x_2, x_3, \ldots, x_{n-1}) + F_1(x_2, x_3, \ldots, x_{n-1})x_1$，且 $I' = I \setminus \{1\}$

$$\begin{aligned}\tilde{F}(x) &= \sum_{I \subseteq [n]} F(I)x_I \\ &= \sum_{1 \notin I} F(I)x_I \qquad\qquad + \sum_{1 \in I} F(I)x_I\end{aligned}$$

捲積 – 舉一反三

令 $F(x) = F_0(x_2, x_3, \ldots, x_{n-1}) + F_1(x_2, x_3, \ldots, x_{n-1})x_1$，且 $I' = I \setminus \{1\}$

$$
\begin{aligned}
\tilde{F}(x) &= \sum_{I \subseteq [n]} F(I)x_I \\
&= \sum_{1 \notin I} F(I)x_I + \sum_{1 \in I} F(I)x_I \\
&= \sum_{1 \notin I} (F_0(I') + F_1(I'))x_I + \sum_{1 \in I} (F_0(I') - F_1(I'))x_I
\end{aligned}
$$

捲積 – 舉一反三

令 $F(x) = F_0(x_2, x_3, \ldots, x_{n-1}) + F_1(x_2, x_3, \ldots, x_{n-1})x_1$，且 $I' = I \setminus \{1\}$

$$
\begin{aligned}
\tilde{F}(x) &= \sum_{I \subseteq [n]} F(I)x_I \\
&= \sum_{1 \notin I} F(I)x_I + \sum_{1 \in I} F(I)x_I \\
&= \sum_{1 \notin I} (F_0(I') + F_1(I'))x_I + \sum_{1 \in I} (F_0(I') - F_1(I'))x_I \\
&= \sum_{1 \notin I} (F_0 + F_1)(I')x_I + \sum_{1 \in I} (F_0 - F_1)(I')x_I
\end{aligned}
$$

捲積 – 舉一反三

令 $F(x) = F_0(x_2, x_3, \ldots, x_{n-1}) + F_1(x_2, x_3, \ldots, x_{n-1})x_1$，且 $I' = I \setminus \{1\}$

$$\begin{aligned}
\tilde{F}(x) &= \sum_{I \subseteq [n]} F(I) x_I \\
&= \sum_{1 \notin I} F(I) x_I + \sum_{1 \in I} F(I) x_I \\
&= \sum_{1 \notin I} (F_0(I') + F_1(I')) x_I + \sum_{1 \in I} (F_0(I') - F_1(I')) x_I \\
&= \sum_{1 \notin I} (F_0 + F_1)(I') x_I + \sum_{1 \in I} (F_0 - F_1)(I') x_I
\end{aligned}$$

遞迴下去 $T(n) = 2T(n) + \mathcal{O}(n) \implies T(n) = \mathcal{O}(n \log n)$

捲積 – 習題

這個變換又叫作 Walsh–Hadamard transform。

捲積 – 習題

這個變換又叫作 Walsh–Hadamard transform。

習題

1. 找出 Walsh–Hadamard transform 的逆變換。
2. 寫出 in-place 的 Walsh–Hadamard transform。

線性規劃

線性規劃 – 例子

- 你一天總共會做四件事情，分別是「打程式競賽」、「打手遊」、「吃飯」和「睡覺」。假設你分別在這件事情上花了 x_1, x_2, x_3, x_4 小時。

線性規劃 – 例子

- 你一天總共會做四件事情，分別是「打程式競賽」、「打手遊」、「吃飯」和「睡覺」。假設你分別在這件事情上花了 x_1, x_2, x_3, x_4 小時。
- 一天有 24 小時 $\implies x_1 + x_2 + x_3 + x_4 \geq 0$。

線性規劃 – 例子

- 你一天總共會做四件事情，分別是「打程式競賽」、「打手遊」、「吃飯」和「睡覺」。假設你分別在這件事情上花了 x_1, x_2, x_3, x_4 小時。
- 一天有 24 小時 $\implies x_1 + x_2 + x_3 + x_4 \geq 0$。
- 有些行動會消耗/補充體力 $\implies -2x_1 - x_2 + 3x_3 + x_4 \geq 3$。

線性規劃 – 例子

- 你一天總共會做四件事情，分別是「打程式競賽」、「打手遊」、「吃飯」和「睡覺」。假設你分別在這件事情上花了 x_1, x_2, x_3, x_4 小時。
- 一天有 24 小時 $\implies x_1 + x_2 + x_3 + x_4 \geq 0$。
- 有些行動會消耗/補充體力 $\implies -2x_1 - x_2 + 3x_3 + x_4 \geq 3$。
- 有些行動會花錢 $\implies x_2 + 2x_3 \leq 6$。

線性規劃 – 例子

- 你一天總共會做四件事情，分別是「打程式競賽」、「打手遊」、「吃飯」和「睡覺」。假設你分別在這件事情上花了 x_1, x_2, x_3, x_4 小時。
- 一天有 24 小時 $\implies x_1 + x_2 + x_3 + x_4 \geq 0$。
- 有些行動會消耗/補充體力 $\implies -2x_1 - x_2 + 3x_3 + x_4 \geq 3$。
- 有些行動會花錢 $\implies x_2 + 2x_3 \leq 6$。
- 你要最大化爽度 $\implies$ maximize: $2x_1 + x_2$。

線性規劃 – 例子

$$
\begin{array}{lrcrcrcrcl}
\text{maximize} & 2x_1 & + & x_2 & & & & & & \\
\text{subject to} & x_1 & + & x_2 & + & x_3 & + & x_4 & \leq & 24 \\
 & -2x_1 & - & x_2 & + & 3x_3 & + & x_4 & \geq & 3 \\
 & & & x_2 & + & 2x_3 & & & \leq & 6
\end{array}
$$

線性規劃 – 例子

$$
\begin{array}{lrcrcrcrcl}
\text{maximize} & 2x_1 & + & x_2 & & & & & & \\
\text{subject to} & x_1 & + & x_2 & + & x_3 & + & x_4 & \leq & 24 \\
& -2x_1 & - & x_2 & + & 3x_3 & + & x_4 & \geq & 3 \\
& & & x_2 & + & 2x_3 & & & \leq & 6 \\
& & & \color{red}{x_1,} & & \color{red}{x_2,} & & \color{red}{x_3,\ x_4} & \color{red}{\geq} & \color{red}{0}
\end{array}
$$

線性規劃 – 標準型式

$$
\begin{array}{rrcl}
\text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\
\text{subject to} & A\boldsymbol{x} & \leq & \boldsymbol{b} \\
 & \boldsymbol{x} & \geq & \boldsymbol{0}
\end{array}
$$

線性規劃 – 標準型式

$$
\begin{array}{lrcl}
\text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\
\text{subject to} & A\boldsymbol{x} & \leq & \boldsymbol{b} \\
 & \boldsymbol{x} & \geq & \boldsymbol{0}
\end{array}
$$

- 原本的問題是要最小化 $\implies$ 將其取負號。

線性規劃 – 標準型式

$$
\begin{array}{lrcl}
\text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\
\text{subject to} & A\boldsymbol{x} & \leq & \boldsymbol{b} \\
 & \boldsymbol{x} & \geq & \boldsymbol{0}
\end{array}
$$

- 原本的問題是要最小化 $\Longrightarrow$ 將其取負號。
- 等式限制 $\Longrightarrow$ 拆成 $\geq$, $\leq$ 兩個不等式。

線性規劃 – 標準型式

$$
\begin{array}{lrcl}
\text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\
\text{subject to} & A\boldsymbol{x} & \leq & \boldsymbol{b} \\
 & \boldsymbol{x} & \geq & \boldsymbol{0}
\end{array}
$$

- 原本的問題是要最小化 $\implies$ 將其取負號。
- 等式限制 $\implies$ 拆成 $\geq, \leq$ 兩個不等式。
- 「大於等於」 $\implies$ 取負號後即變為「小於等於」。

線性規劃 – 標準型式

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\ \text{subject to} & A\boldsymbol{x} & \leq & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & \boldsymbol{0} \end{array}$$

- 原本的問題是要最小化 $\Longrightarrow$ 將其取負號。
- 等式限制 $\Longrightarrow$ 拆成 $\geq$, $\leq$ 兩個不等式。
- 「大於等於」 $\Longrightarrow$ 取負號後即變為「小於等於」。
- 本來某個 x_i 沒有 $x_i \geq 0$ 的限制 $\Longrightarrow$ 替換 $x_i = x_i' - x_i''$ 並設下 $x_i', x_i'' \geq 0$ 的限制即可。

線性規劃 – 標準型式

$$
\begin{array}{lrcrcrcrcl}
\text{maximize} & 2x_1 & + & x_2 & & & & & & \\
\text{subject to} & x_1 & + & x_2 & + & x_3 & + & x_4 & \leq & 24 \\
 & -2x_1 & - & x_2 & + & 3x_3 & + & x_4 & \geq & 3 \\
 & & & x_2 & + & 2x_3 & & & \leq & 6 \\
 & & & & & x_1,\ x_2,\ x_3, & & x_4 & \geq & 0
\end{array}
$$

線性規劃 – 標準型式

$$
\begin{array}{rrcrcrcrcl}
\text{maximize} & 2x_1 & + & x_2 & & & & & & \\
\text{subject to} & x_1 & + & x_2 & + & x_3 & + & x_4 & \leq & 24 \\
 & 2x_1 & + & x_2 & - & 3x_3 & - & x_4 & \leq & -3 \\
 & & & x_2 & + & 2x_3 & & & \leq & 6 \\
 & & & & & x_1, & x_2, & x_3, \; x_4 & \geq & 0
\end{array}
$$

線性規劃 – Slack form

現在對於一個限制

$$\sum_j a_{i,j} x_j \leq b_i$$

定義 Slack variable

$$x \triangleq b_i - \sum_j a_{i,j} x_j$$

線性規劃 – Slack form

現在對於一個限制

$$\sum_j a_{i,j} x_j \le b_i$$

定義 Slack variable

$$x \triangleq b_i - \sum_j a_{i,j} x_j$$

這個變數描述這個不等式差多少變等式。

線性規劃 – Slack form

現在對於一個限制

$$\sum_j a_{i,j} x_j \leq b_i$$

定義 Slack variable

$$x \triangleq b_i - \sum_j a_{i,j} x_j$$

這個變數描述這個不等式差多少變等式。

原不等式等價於 $x \geq 0$

線性規劃 – Slack form

$$
\begin{array}{rrcrcrcrcr}
\text{maximize} & 2x_1 & + & x_2 & & & & & & \\
\text{subject to} & x_1 & + & x_2 & + & x_3 & + & x_4 & \leq & 24 \\
& 2x_1 & + & x_2 & - & 3x_3 & - & x_4 & \leq & -3 \\
& & & x_2 & + & 2x_3 & & & \leq & 6 \\
& & & & & x_1, & x_2, & x_3, \; x_4 & \geq & 0
\end{array}
$$

線性規劃 – Slack form

$$
\begin{array}{ll}
\text{maximize} & 2x_1 + x_2 + 0 \\
\text{subject to} & \begin{array}{rcrcrcrcrcr}
x_5 & = & 24 & - & x_1 & - & x_2 & - & x_3 & - & x_4 \\
x_6 & = & -3 & - & 2x_1 & - & x_2 & + & 3x_3 & + & x_4 \\
x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
\mathbf{x} & \geq & \mathbf{0} & & & & & & & &
\end{array}
\end{array}
$$

線性規劃 – Slack form

$$
\begin{aligned}
\text{maximize} \quad & c_1 x_1 + \cdots + c_n x_n + c_0 \\
\text{subject to} \quad & x_{n+1} = b_{n+1} - \sum_{j=1}^{n} a_{n+1,j}\, x_j \\
& x_{n+2} = b_{n+2} - \sum_{j=1}^{n} a_{n+2,j}\, x_j \\
& \quad \vdots \\
& x_{n+m} = b_{n+m} - \sum_{j=1}^{n} a_{n+m,j}\, x_j \\
& x_i \geq 0 \qquad \forall i \in [1, m+n]
\end{aligned}
$$

線性規劃 – Slack form

$$
\begin{aligned}
\text{maximize} \quad & c_1 x_1 + \cdots + c_n x_n + c_0 \\
\text{subject to} \quad & x_{n+1} = b_{n+1} - \sum_{j=1}^{n} a_{n+1,j}\, x_j \\
& x_{n+2} = b_{n+2} - \sum_{j=1}^{n} a_{n+2,j}\, x_j \\
& \quad \vdots \\
& x_{n+m} = b_{n+m} - \sum_{j=1}^{n} a_{n+m,j}\, x_j \\
& x_i \geq 0 \qquad \forall i \in [1, m+n]
\end{aligned}
$$

- 左邊的 $x_{n+1}, \ldots, x_{n+m}$ 叫作**基礎變數** B。

線性規劃 – Slack form

$$
\begin{aligned}
\text{maximize} \quad & c_1 x_1 + \cdots + c_n x_n + c_0 \\
\text{subject to} \quad & x_{n+1} = b_{n+1} - \sum_{j=1}^{n} a_{n+1,j}\, x_j \\
& x_{n+2} = b_{n+2} - \sum_{j=1}^{n} a_{n+2,j}\, x_j \\
& \vdots \\
& x_{n+m} = b_{n+m} - \sum_{j=1}^{n} a_{n+m,j}\, x_j \\
& x_i \geq 0 \qquad \forall i \in [1, m+n]
\end{aligned}
$$

- 左邊的 $x_{n+1}, \ldots, x_{n+m}$ 叫作**基礎變數** B。
- 右邊的 $x_1, \ldots, x_n$ 叫作**非基礎變數** N。

線性規劃 – Slack form

$$
\begin{aligned}
\text{maximize} \quad & c_1 x_1 + \cdots + c_n x_n + c_0 \\
\text{subject to} \quad & x_{n+1} = b_{n+1} - \sum_{j=1}^{n} a_{n+1,j}\, x_j \\
& x_{n+2} = b_{n+2} - \sum_{j=1}^{n} a_{n+2,j}\, x_j \\
& \quad \vdots \\
& x_{n+m} = b_{n+m} - \sum_{j=1}^{n} a_{n+m,j}\, x_j \\
& x_i \geq 0 \qquad \forall i \in [1, m+n]
\end{aligned}
$$

- 左邊的 $x_{n+1}, \ldots, x_{n+m}$ 叫作**基礎變數** B。
- 右邊的 $x_1, \ldots, x_n$ 叫作**非基礎變數** N。
- 把所有 $x_i, i \in N$ 設成 0。

線性規劃 – Slack form

$$
\begin{aligned}
\text{maximize} \quad & c_1 x_1 + \cdots + c_n x_n + c_0 \\
\text{subject to} \quad & x_{n+1} = b_{n+1} - \sum_{j=1}^{n} a_{n+1,j}\, x_j \\
& x_{n+2} = b_{n+2} - \sum_{j=1}^{n} a_{n+2,j}\, x_j \\
& \quad \vdots \\
& x_{n+m} = b_{n+m} - \sum_{j=1}^{n} a_{n+m,j}\, x_j \\
& x_i \geq 0 \qquad \forall i \in [1, m+n]
\end{aligned}
$$

- 左邊的 $x_{n+1}, \ldots, x_{n+m}$ 叫作**基礎變數** B。
- 右邊的 $x_1, \ldots, x_n$ 叫作**非基礎變數** N。
- 把所有 $x_i, i \in N$ 設成 0。
- 對應到一組解 $x_i = b_i$, $i \in B$。

線性規劃 – Slack form

$$
\begin{aligned}
\text{maximize} \quad & c_1x_1 + \cdots + c_nx_n + c_0 \\
\text{subject to} \quad & x_{n+1} = b_{n+1} - \sum_{j=1}^{n} a_{n+1,j}\, x_j \\
& x_{n+2} = b_{n+2} - \sum_{j=1}^{n} a_{n+2,j}\, x_j \\
& \vdots \\
& x_{n+m} = b_{n+m} - \sum_{j=1}^{n} a_{n+m,j}\, x_j \\
& x_i \geq 0 \qquad \forall i \in [1, m+n]
\end{aligned}
$$

- 左邊的 $x_{n+1}, \ldots, x_{n+m}$ 叫作**基礎變數** B。
- 右邊的 $x_1, \ldots, x_n$ 叫作**非基礎變數** N。
- 把所有 $x_i, i \in N$ 設成 0。
- 對應到一組解 $x_i = b_i$, $i \in B$。
- 這個解是**可行解** $\iff \boldsymbol{b} \geq \mathbf{0}$。

線性規劃 – Slack form

$$
\begin{aligned}
\text{maximize} \quad & c_1 x_1 + \cdots + c_n x_n + c_0 \\
\text{subject to} \quad & x_{n+1} = b_{n+1} - \sum_{j=1}^{n} a_{n+1,j}\, x_j \\
& x_{n+2} = b_{n+2} - \sum_{j=1}^{n} a_{n+2,j}\, x_j \\
& \vdots \\
& x_{n+m} = b_{n+m} - \sum_{j=1}^{n} a_{n+m,j}\, x_j \\
& x_i \geq 0 \qquad \forall i \in [1, m+n]
\end{aligned}
$$

- 左邊的 $x_{n+1}, \ldots, x_{n+m}$ 叫作**基礎變數** B。
- 右邊的 $x_1, \ldots, x_n$ 叫作**非基礎變數** N。
- 把所有 $x_i, i \in N$ 設成 0。
- 對應到一組解 $x_i = b_i$, $i \in B$。
- 這個解是**可行解** $\iff \boldsymbol{b} \geq \boldsymbol{0}$。
- 如果 $\boldsymbol{c} \leq \boldsymbol{0}$，這個解是**最佳解**。

線性規劃 – Pivoting

假設我們的基礎解是一個可行解。

線性規劃 – Pivoting

假設我們的基礎解是一個可行解。

$$
\begin{array}{ll}
\text{maximize} & 2x_1 + x_2 + 0 \\
\text{subject to} & \begin{array}{rcrcrcrcrcr}
x_5 & = & 24 & - & x_1 & - & x_2 & - & x_3 & - & x_4 \\
x_6 & = & -3 & - & 2x_1 & - & x_2 & + & 3x_3 & + & x_4 \\
x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
\boldsymbol{x} & \geq & \mathbf{0} & & & & & & & &
\end{array}
\end{array}
$$

線性規劃 – Pivoting

假設我們的基礎解是一個可行解。

$$
\begin{array}{ll}
\text{maximize} & 2x_1 + x_2 + 0 \\
\text{subject to} & \begin{array}{rcrcrcrcrcr}
x_5 & = & 24 & - & x_1 & - & x_2 & - & x_3 & - & x_4 \\
x_6 & = & 3 & - & 2x_1 & - & x_2 & + & 3x_3 & + & x_4 \\
x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
\mathbf{x} & \geq & \mathbf{0} & & & & & & & &
\end{array}
\end{array}
$$

線性規劃 – Pivoting

假設我們的基礎解是一個可行解。

$c_i \geq 0 \implies$ 增加 x_i 會更好。

$$
\begin{array}{ll}
\text{maximize} & 2x_1 + x_2 + 0 \\
\text{subject to} & \begin{array}{rcrcrcrcrcr}
x_5 & = & 24 & - & x_1 & - & x_2 & - & x_3 & - & x_4 \\
x_6 & = & 3 & - & 2x_1 & - & x_2 & + & 3x_3 & + & x_4 \\
x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
\mathbf{x} & \geq & \mathbf{0} & & & & & & & &
\end{array}
\end{array}
$$

線性規劃 – Pivoting

假設我們的基礎解是一個可行解。

$c_i \geq 0 \implies$ 增加 x_i 會更好。

$$
\begin{array}{lrcrcrcrcrcr}
\text{maximize} & & & & & 2x_1 & + & x_2 & + & 0 & & \\
\text{subject to} & x_5 & = & 24 & - & x_1 & - & x_2 & - & x_3 & - & x_4 \\
 & x_6 & = & 3 & - & 2x_1 & - & x_2 & + & 3x_3 & + & x_4 \\
 & x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
 & \mathbf{x} & \geq & \mathbf{0} & & & & & & & &
\end{array}
$$

線性規劃 – Pivoting

假設我們的基礎解是一個可行解。

$c_i \geq 0 \implies$ 增加 x_i 會更好。

$$
\begin{array}{ll}
\text{maximize} & 2x_1 + x_2 + 0 \\
\text{subject to} & \begin{array}{rcrcrcrcrcr}
x_5 & = & 24 & - & x_1 & - & x_2 & - & x_3 & - & x_4 \\
x_6 & = & 3 & - & 2x_1 & - & x_2 & + & 3x_3 & + & x_4 \\
x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
\mathbf{x} & \geq & \mathbf{0} & & & & & & & &
\end{array}
\end{array}
$$

可以增加多少？

1. x_5: $24/1 = 24$。
2. x_6: $3/2 = 1.5$。

線性規劃 – Pivoting

假設我們的基礎解是一個可行解。

$c_i \geq 0 \implies$ 增加 x_i 會更好。

$$
\begin{array}{ll}
\text{maximize} & 2x_1 + x_2 + 0 \\
\text{subject to} & \begin{array}{rcrcrcrcrcr}
x_5 & = & 24 & - & x_1 & - & x_2 & - & x_3 & - & x_4 \\
x_6 & = & 3 & - & 2x_1 & - & x_2 & + & 3x_3 & + & x_4 \\
x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
\mathbf{x} & \geq & \mathbf{0} & & & & & & & &
\end{array}
\end{array}
$$

可以增加多少？

1. x_5: $24/1 = 24$。
2. x_6: $3/2 = 1.5$。

$$x_1 = \frac{3}{2} - \frac{1}{2}x_6 - \frac{1}{2}x_2 + \frac{3}{2}x_3 + \frac{1}{2}x_4$$

帶回每個式子。

線性規劃 – Pivoting

$$
\begin{array}{rrcrcrcrcrcr}
\text{maximize} & \multicolumn{11}{l}{3 - x_6 + 3x_3 + x_4} \\
\text{subject to} & x_5 & = & \frac{45}{2} & + & \frac{1}{2}x_6 & - & \frac{1}{2}x_2 & - & \frac{5}{2}x_3 & - & \frac{1}{2}x_4 \\
& x_1 & = & \frac{3}{2} & - & \frac{1}{2}x_6 & - & \frac{1}{2}x_2 & + & \frac{3}{2}x_3 & + & \frac{1}{2}x_4 \\
& x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
& x & \geq & \mathbf{0} & & & & & & & &
\end{array}
$$

線性規劃 – Pivoting

$$\begin{array}{ll}
\text{maximize} & 3 - x_6 + 3x_3 + x_4 \\
\text{subject to} & \begin{array}{rcrcrcrcrcr}
x_5 & = & \frac{45}{2} & + & \frac{1}{2}x_6 & - & \frac{1}{2}x_2 & - & \frac{5}{2}x_3 & - & \frac{1}{2}x_4 \\
x_1 & = & \frac{3}{2} & - & \frac{1}{2}x_6 & - & \frac{1}{2}x_2 & + & \frac{3}{2}x_3 & + & \frac{1}{2}x_4 \\
x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
x & \geq & \mathbf{0} & & & & & & & &
\end{array}
\end{array}$$

把這個操作叫作 Pivoting。

線性規劃 – Pivoting

$$
\begin{array}{ll}
\text{maximize} & 3 - x_6 + 3x_3 + x_4 \\
\text{subject to} & \begin{array}{rclclclclcl}
x_5 & = & \frac{45}{2} & + & \frac{1}{2}x_6 & - & \frac{1}{2}x_2 & - & \frac{5}{2}x_3 & - & \frac{1}{2}x_4 \\
x_1 & = & \frac{3}{2} & - & \frac{1}{2}x_6 & - & \frac{1}{2}x_2 & + & \frac{3}{2}x_3 & + & \frac{1}{2}x_4 \\
x_7 & = & 6 & & & - & x_2 & - & 2x_3 & & \\
x & \geq & \mathbf{0} & & & & & & & &
\end{array}
\end{array}
$$

把這個操作叫作 Pivoting。做完 Pivoting，係數、N 和 B 會變，但與原本的等價。

線性規劃 – Simplex

$$
\begin{array}{lrcl}
\text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\
\text{subject to} & \boldsymbol{x}_B & = & \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_N \\
 & \boldsymbol{x} & \geq & \boldsymbol{0}
\end{array}
$$

線性規劃 – Simplex

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\ \text{subject to} & \boldsymbol{x}_B & = & \boldsymbol{b} - A\boldsymbol{x}_N \\ & \boldsymbol{x} & \geq & \mathbf{0} \end{array}$$

1. 透過 Pivoting 讓 $\boldsymbol{b} \geq \mathbf{0}$

線性規劃 – Simplex

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\ \text{subject to} & \boldsymbol{x}_B & = & \boldsymbol{b} - A\boldsymbol{x}_N \\ & \boldsymbol{x} & \geq & \mathbf{0} \end{array}$$

1. 透過 Pivoting 讓 $\boldsymbol{b} \geq \mathbf{0}$
2. 透過 Pivoting 讓 $\boldsymbol{c} \leq \mathbf{0}$

線性規劃 – Simplex

$$
\begin{array}{lrcl}
\text{maximize} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} & & \\
\text{subject to} & \boldsymbol{x}_B & = & \boldsymbol{b} - A\boldsymbol{x}_N \\
& \boldsymbol{x} & \geq & \boldsymbol{0}
\end{array}
$$

1. 透過 Pivoting 讓 $\boldsymbol{b} \geq \boldsymbol{0}$
2. 透過 Pivoting 讓 $\boldsymbol{c} \leq \boldsymbol{0}$

分別為 Simplex 的 phase 1, phase 2。

線性規劃 – Simplex phase 1

1. 找一個 $x_s \in N$ 使得其對應的係數 $c_s > 0$。

線性規劃 – Simplex phase 1

1. 找一個 $x_s \in N$ 使得其對應的係數 $c_s > 0$。
2. 對所有 $x_j \in B$ 如果 $A_{j,s} \geq 0$ 則 $\delta_j \triangleq b_j / A_{j,s}$ 否則 $\delta_j \triangleq \infty$。

線性規劃 – Simplex phase 1

1. 找一個 $x_s \in N$ 使得其對應的係數 $c_s > 0$。
2. 對所有 $x_j \in B$ 如果 $A_{j,s} \geq 0$ 則 $\delta_j \triangleq b_j / A_{j,s}$ 否則 $\delta_j \triangleq \infty$。
3. 如果 $\min \delta_j = \infty$ 則回傳此線性規劃**無界**，否則找 x_t 使得 δ_t 最小。

線性規劃 – Simplex phase 1

1. 找一個 $x_s \in N$ 使得其對應的係數 $c_s > 0$。
2. 對所有 $x_j \in B$ 如果 $A_{j,s} \geq 0$ 則 $\delta_j \triangleq b_j / A_{j,s}$ 否則 $\delta_j \triangleq \infty$。
3. 如果 $\min \delta_j = \infty$ 則回傳此線性規劃**無界**，否則找 x_t 使得 δ_t 最小。
4. 對 x_s, x_t 做 Pivoting，得到新的 slack form。

線性規劃 – Simplex phase 1

1. 找一個 $x_s \in N$ 使得其對應的係數 $c_s > 0$。
2. 對所有 $x_j \in B$ 如果 $A_{j,s} \geq 0$ 則 $\delta_j \triangleq b_j / A_{j,s}$ 否則 $\delta_j \triangleq \infty$。
3. 如果 $\min \delta_j = \infty$ 則回傳此線性規劃**無界**，否則找 x_t 使得 δ_t 最小。
4. 對 x_s, x_t 做 Pivoting，得到新的 slack form。
5. 重複以上步驟直到 $c_s \leq \mathbf{0}$。

線性規劃 – Simplex phase 1

1. 找一個 $x_s \in N$ 使得其對應的係數 $c_s > 0$。有多個找 s 最小的。
2. 對所有 $x_j \in B$ 如果 $A_{j,s} \geq 0$ 則 $\delta_j \triangleq b_j / A_{j,s}$ 否則 $\delta_j \triangleq \infty$。
3. 如果 $\min \delta_j = \infty$ 則回傳此線性規劃**無界**，否則找 x_t 使得 δ_t 最小。有多個找 t 最小的。
4. 對 x_s, x_t 做 Pivoting，得到新的 slack form。
5. 重複以上步驟直到 $c_s \leq \mathbf{0}$。

Bland's rule 保證遍歷過的 Slack form 不會重複。
$\Longrightarrow$ 複雜度 $\binom{n+m}{m}$。

線性規劃 – Simplex phase 1

1. 找一個 $x_s \in N$ 使得其對應的係數 $c_s > 0$。有多個找 s 最小的。
2. 對所有 $x_j \in B$ 如果 $A_{j,s} \geq 0$ 則 $\delta_j \triangleq b_j / A_{j,s}$ 否則 $\delta_j \triangleq \infty$。
3. 如果 $\min \delta_j = \infty$ 則回傳此線性規劃**無界**，否則找 x_t 使得 δ_t 最小。有多個找 t 最小的。
4. 對 x_s, x_t 做 Pivoting，得到新的 slack form。
5. 重複以上步驟直到 $c_s \leq \mathbf{0}$。

Bland's rule 保證遍歷過的 Slack form 不會重複。
$\Longrightarrow$ 複雜度 $\binom{n+m}{m}$。實際上遠比這個好。

線性規劃 – Simplex phase 2

如何找一組可行解？

線性規劃 – Simplex phase 2

如何找一組可行解？

$$
\begin{array}{lrcl}
\text{maximize} & & & -x_0 \\
\text{subject to} & x_{n+1} & = & b_{n+1} - \sum_{j=1}^{n} a_{n+1,j}x_j + x_0 \\
& x_{n+2} & = & b_{n+2} - \sum_{j=1}^{n} a_{n+2,j}x_j + x_0 \\
& & \vdots & \\
& x_{n+m} & = & b_{n+m} - \sum_{j=1}^{n} a_{n+m,j}x_j + x_0 \\
& x_i & \geq & 0 \qquad \forall i \in [1, m+n]
\end{array}
$$

線性規劃 – Simplex phase 2

可以證明如果 b_t 是最小的，將 x_0, x_t 做 Pivoting 後所有 $b_i \geq 0$。

線性規劃 – Simplex phase 2

可以證明如果 b_t 是最小的，將 x_0, x_t 做 Pivoting 後所有 $b_i \geq 0$。

$\implies$ 用 Phase 2 解！

線性規劃 – Simplex phase 2

可以證明如果 b_t 是最小的，將 x_0, x_t 做 Pivoting 後所有 $b_i \geq 0$。

$\Longrightarrow$ 用 Phase 2 解！

1. 將 slack form 加入變數 x_0 並把目標函數換成 $-x_0$。

線性規劃 – Simplex phase 2

可以證明如果 b_t 是最小的，將 x_0, x_t 做 Pivoting 後所有 $b_i \geq 0$。

$\implies$ 用 Phase 2 解！

1. 將 slack form 加入變數 x_0 並把目標函數換成 $-x_0$。
2. 令 $b_t \triangleq \min b_i$，對 x_0 和 x_t 做一次 pivoting。

線性規劃 – Simplex phase 2

可以證明如果 b_t 是最小的，將 x_0, x_t 做 Pivoting 後所有 $b_i \geq 0$。

$\Longrightarrow$ 用 Phase 2 解！

1. 將 slack form 加入變數 x_0 並把目標函數換成 $-x_0$。
2. 令 $b_t \triangleq \min b_i$，對 x_0 和 x_t 做一次 pivoting。
3. 用 phase 2 的方法解出這個線性規劃的最佳解，在過程中每做一次 pivoting 時也順便將原目標函數做變數變換。

線性規劃 – Simplex phase 2

可以證明如果 b_t 是最小的，將 x_0, x_t 做 Pivoting 後所有 $b_i \geq 0$。

$\Longrightarrow$ 用 Phase 2 解！

1. 將 slack form 加入變數 x_0 並把目標函數換成 $-x_0$。
2. 令 $b_t \triangleq \min b_i$，對 x_0 和 x_t 做一次 pivoting。
3. 用 phase 2 的方法解出這個線性規劃的最佳解，在過程中每做一次 pivoting 時也順便將原目標函數做變數變換。
4. 如果修改後的線性規劃解出的最佳解不為 0 則回傳**無解**。

線性規劃 – Simplex phase 2

可以證明如果 b_t 是最小的，將 x_0, x_t 做 Pivoting 後所有 $b_i \geq 0$。

$\implies$ 用 Phase 2 解！

1. 將 slack form 加入變數 x_0 並把目標函數換成 $-x_0$。
2. 令 $b_t \triangleq \min b_i$，對 x_0 和 x_t 做一次 pivoting。
3. 用 phase 2 的方法解出這個線性規劃的最佳解，在過程中每做一次 pivoting 時也順便將原目標函數做變數變換。
4. 如果修改後的線性規劃解出的最佳解不為 0 則回傳**無解**。
5. 否則有解，現在如果 $x_0 \in B$，則隨便找一個 $x_t \in N$ 滿足 $a_{0,t} \neq 0$。對 x_0 和 x_t 做 pivoting 後用 phase 2 解。

線性規劃 – 對偶

回顧一下之前的例子：

	爽度	時間	體力	金錢
打程式競賽	$+2$	-1	-2	0
打手遊	$+1$	-1	-1	-1
吃飯	0	-1	$+2$	-2
睡覺	0	-1	$+1$	0

線性規劃 – 對偶

回顧一下之前的例子：

	爽度	時間	體力	金錢
打程式競賽	$+2$	-1	-2	0
打手遊	$+1$	-1	-1	-1
吃飯	0	-1	$+2$	-2
睡覺	0	-1	$+1$	0

最佳解為打程式競賽 9 小時、不玩手遊、吃飯 3 小時然後睡覺 12 小時。

線性規劃 – 對偶

- 你遇到了一個惡魔要跟用滿足度交易「時間」、「體力」與「金錢」。

線性規劃 – 對偶

- 你遇到了一個惡魔要跟用滿足度交易「時間」、「體力」與「金錢」。
- 雙方都不想吃虧。假設「時間」、「體力」與「金錢」分別值 y_1，y_2，y_3 的滿足度。

線性規劃 – 對偶

- 你遇到了一個惡魔要跟用滿足度交易「時間」、「體力」與「金錢」。
- 雙方都不想吃虧。假設「時間」、「體力」與「金錢」分別值 y_1，y_2，y_3 的滿足度。
- 惡魔想要付你越少滿足度越好。

線性規劃 – 對偶

- 你遇到了一個惡魔要跟用滿足度交易「時間」、「體力」與「金錢」。
- 雙方都不想吃虧。假設「時間」、「體力」與「金錢」分別值 y_1，y_2，y_3 的滿足度。
- 惡魔想要付你越少滿足度越好。
- 你花 1 單位的時間和 2 單位的體力打程式競賽，本來就可以獲得 2 單位的滿足度了，因此

$$y_1 + 2y_2 \geq 2$$

線性規劃 – 對偶

- 你遇到了一個惡魔要跟用滿足度交易「時間」、「體力」與「金錢」。
- 雙方都不想吃虧。假設「時間」、「體力」與「金錢」分別值 y_1，y_2，y_3 的滿足度。
- 惡魔想要付你越少滿足度越好。
- 你花 1 單位的時間和 2 單位的體力打程式競賽，本來就可以獲得 2 單位的滿足度了，因此

$$y_1 + 2y_2 \geq 2$$

- 類似考慮其他行動還有 3 條等式。

線性規劃 – 對偶

$$
\begin{array}{rl}
\text{maximize} & 2x_1 + x_2 \\
\text{subject to} & x_1 + x_2 + x_3 + x_4 \le 24 \\
& 2x_1 + x_2 - 3x_3 - x_4 \le -3 \\
& x_2 + 2x_3 \le 6 \\
& x_1,\ x_2,\ x_3,\ x_4 \ge 0
\end{array}
$$

$$
\begin{array}{rl}
\text{minimize} & 24y_1 - 3y_2 + 6y_3 \\
& y_1 + 2y_2 \ge 1 \\
& y_1 + y_2 + y_3 \ge 1 \\
\text{subject to} & y_1 - 3y_2 + 2y_3 \ge 0 \\
& y_1 - y_2 \ge 0 \\
& y_1,\ y_2,\ y_3 \ge 0
\end{array}
$$

線性規劃 – 對偶

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} \quad & A^{\mathrm{T}}\boldsymbol{y} \geq \boldsymbol{c} \\ & \boldsymbol{y} \geq \boldsymbol{0} \end{aligned}$$

線性規劃 – 對偶

$$
\begin{array}{ll}
\text{maximize} & \displaystyle\sum_{(u_i,v_j)\in E} x_{i,j} \\
\text{subject to} & \displaystyle\sum_{j:(u_i,v_j)\in E} x_{i,j} \le 1, \;\; \forall\, u_i \in V \\
& \displaystyle\sum_{i:(u_i,v_j)\in E} x_{i,j} \le 1, \;\; \forall\, v_j \in V \\
& \boldsymbol{x} \ge \mathbf{0}
\end{array}
$$

線性規劃 – 對偶

$$\begin{aligned} \text{maximize} \quad & \sum_{(u_i,v_j)\in E} x_{i,j} \\ \text{subject to} \quad & \sum_{j:(u_i,v_j)\in E} x_{i,j} \le 1, \quad \forall\, u_i \in V \\ & \sum_{i:(u_i,v_j)\in E} x_{i,j} \le 1, \quad \forall\, v_j \in V \\ & \boldsymbol{x} \ge \mathbf{0} \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & \sum_i y_i + \sum_j y'_j \\ \text{subject to} \quad & y_i + y'_j \ge 1, \quad \forall\, (u_i, v_j) \in E \\ & \boldsymbol{y} \ge \mathbf{0} \end{aligned}$$

線性規劃 – 對偶

定理

- 對偶問題的對偶即為原問題。
- 如果 $\bar{x}, \bar{y}$ 分別是原問題和對偶問題的一組最佳解，則

$$\boldsymbol{c}^{\mathsf{T}}\bar{\boldsymbol{x}} = \boldsymbol{b}^{\mathsf{T}}\bar{\boldsymbol{y}}$$

定理 (Dual slackness)

$\bar{x}, \bar{y}$ 是原問題和對偶問題的一組最佳解若且唯若對每個 $i \in [1, n]$，

$$x_i = 0 \quad \text{和} \quad \sum_{j=0}^{m} a_{j,i} y_j = c_i$$

中至少有一者成立，且對所有 $j \in [1, m]$，

$$y_j = 0 \quad \text{和} \quad \sum_{i=0}^{n} a_{i,j} x_i = b_j$$

中至少有一者成立。

線性規劃 – 對偶

證明： 由

$$\boldsymbol{c}^\top \bar{\boldsymbol{x}} \overset{(1)}{\leq} (A^\top \bar{\boldsymbol{y}})^\top \bar{\boldsymbol{x}} = \bar{\boldsymbol{y}}^\top (A\bar{\boldsymbol{x}}) \overset{(2)}{\leq} \bar{\boldsymbol{y}}^\top \boldsymbol{b} = \bar{\boldsymbol{b}}^\top \boldsymbol{y}$$

線性規劃 – 對偶

證明： 由

$$\boldsymbol{c}^{\mathsf{T}}\bar{\boldsymbol{x}} \overset{(1)}{\leq} (A^{\mathsf{T}}\bar{\boldsymbol{y}})^{\mathsf{T}}\bar{\boldsymbol{x}} = \bar{\boldsymbol{y}}^{\mathsf{T}}(A\bar{\boldsymbol{x}}) \overset{(2)}{\leq} \bar{\boldsymbol{y}}^{\mathsf{T}}\boldsymbol{b} = \bar{\boldsymbol{b}}^{\mathsf{T}}\boldsymbol{y}$$

等式成立若且唯若 (1) 和 (2) 的等式都成立，從 (1) 我們得出

$$\sum_{i=0}^{n}\left(-c_i + \sum_{j=0}^{m} a_{j,i}y_j\right)x_i = 0$$

線性規劃 – 對偶

證明： 由

$$\mathbf{c}^{\mathsf{T}}\bar{\boldsymbol{x}} \overset{(1)}{\leq} (A^{\mathsf{T}}\bar{\boldsymbol{y}})^{\mathsf{T}}\bar{\boldsymbol{x}} = \bar{\boldsymbol{y}}^{\mathsf{T}}(A\bar{\boldsymbol{x}}) \overset{(2)}{\leq} \bar{\boldsymbol{y}}^{\mathsf{T}}\boldsymbol{b} = \bar{\boldsymbol{b}}^{\mathsf{T}}\boldsymbol{y}$$

等式成立若且唯若 (1) 和 (2) 的等式都成立，從 (1) 我們得出

$$\sum_{i=0}^{n}\left(-c_i + \sum_{j=0}^{m} a_{j,i}y_j\right)x_i = 0$$

但因為 $\left(-c_i + \sum_{j=0}^{m} a_{j,i}y_j\right)$ 和 x_i 皆大於等於 0，因此

$$x_i = 0 \ \text{ 或 } \ \sum_{j=0}^{m} a_{j,i}y_j = c_i, \ \forall i$$

線性規劃 – 例題

題目 (Flood in Gridland, ICPC Dhaka Regional 2014)

在一個 $m \times n$ 的格子上，每個格子不是無盡的深淵，就是一個高為 $h_{i,j}$ 的土地。你現在可以做兩種操作：

線性規劃 – 例題

題目 (Flood in Gridland, ICPC Dhaka Regional 2014)

在一個 $m \times n$ 的格子上，每個格子不是無盡的深淵，就是一個高為 $h_{i,j}$ 的土地。你現在可以做兩種操作：

1. 將某一列土地的高度全部增加 x ($x \geq 0$)。

線性規劃 – 例題

題目 (Flood in Gridland, ICPC Dhaka Regional 2014)

在一個 $m \times n$ 的格子上，每個格子不是無盡的深淵，就是一個高為 $h_{i,j}$ 的土地。你現在可以做兩種操作：

1. 將某一列土地的高度全部增加 x ($x \geq 0$)。
2. 將某一行土地的高度全部減少 x ($x \geq 0$)。

線性規劃 – 例題

題目 (Flood in Gridland, ICPC Dhaka Regional 2014)

在一個 $m \times n$ 的格子上，每個格子不是無盡的深淵，就是一個高為 $h_{i,j}$ 的土地。你現在可以做兩種操作：

1. 將某一列土地的高度全部增加 x $(x \geq 0)$。
2. 將某一行土地的高度全部減少 x $(x \geq 0)$。

深淵不會受到你的操作影響。你希望最後所有土地都在 $[L, U]$ 之間，並且所有土地的高度總和越大越好，請給出一組最佳解。$(1 \leq m, n \leq 75)$

線性規劃 – 例題

$$
\begin{array}{llcll}
\text{maximize} & \sum R_i x_i + \sum C_j x_j & & & \\
\text{subject to} & x_i - x'_j & \leq & U - h_{i,j}, & \text{對所有不是深淵的 } (i,j) \\
 & -x_i + x'_j & \leq & h_{i,j} - L, & \text{對所有不是深淵的 } (i,j) \\
 & x_i, x'_j & \geq & 0, & \forall\, i,j
\end{array}
$$

線性規劃 – 例題

$$
\begin{array}{rrcll}
\text{maximize} & \sum R_i x_i + \sum C_j x_j & & & \\
\text{subject to} & x_i - x'_j & \leq & U - h_{i,j}, & \text{對所有不是深淵的 } (i,j) \\
& -x_i + x'_j & \leq & h_{i,j} - L, & \text{對所有不是深淵的 } (i,j) \\
& x_i, x'_j & \geq & 0, & \forall\, i, j
\end{array}
$$

$$
\begin{array}{rll}
\text{minimize} & \sum_{i,j} (U - h_{i,j}) y_{i,j} + (h_{i,j} - L) y'_{i,j} & \\
\text{subject to} & \sum_j y_{i,j} - y'_{i,j} \geq R_i, & \forall i \in [1, m] \\
& \sum_i -y_{i,j} + y'_{i,j} \geq -C_j, & \forall i \in [1, n]
\end{array}
$$

Epilogue