

Extended Application of Burnside's Lemma to Higher Dimension using Matrix and Linear Transformation

楊記綱

2022 年 9 月 11 日

概 要

1 `module Burnside where`

針對清大課程 Burnside's Lemma 去從線性轉換與矩陣、向量的角度去做延伸。令嘗試將此想法推廣到更高維度、非正多面體與較複雜之正多面體。

目錄

1	Brief Introduction to Matrix and Linear Transformation	2
2	Properties	2
2.1	Calculate $\#T$	2
2.2	Calculate T	3
3	Extend Research Topics (未寫)	3
3.1	非正多面體	3
3.2	高維物體	3
3.3	魔術方塊	3
A	Matrix	4

1 Brief Introduction to Matrix and Linear Transformation

So what does matrix in linear transformation means? Well, for any $n \times n$ matrix, you could think of each column as representing each axis' unit vector's position after the transformation. So for example:

$$\text{Let } T = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad (1)$$

$$\text{means, shifting } \hat{i} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \hat{j} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \hat{k} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{ to} \quad (2)$$

$$\hat{i}' = \begin{bmatrix} a \\ d \\ g \end{bmatrix}, \hat{j}' = \begin{bmatrix} b \\ e \\ h \end{bmatrix}, \hat{k}' = \begin{bmatrix} c \\ f \\ i \end{bmatrix} \quad (3)$$

Hence by these definitions we could conclude that:

{data}

定理 1 (最少資料量)

For any object within n -Dimensional space, by specifying n -Surfaces' location could unique identify an object's orientation.

2 Properties

當我們在談到旋轉一物體的時候我們可以將其總結成：

$$\mathbb{T} = \{T \mid T \cdot x \in \mathbb{X}, x \in \mathbb{X}\} \quad (4)$$

其中 \mathbb{T} 代表所有合法的 $n \times n$ 矩陣使得作用於 \mathbb{X} 中任意元件會得結果 $T \cdot x = x' \ni x, x' \in \mathbb{X}$ 。那也就是換成講義裡的記號 $|G| = \#\mathbb{T}$ 。

2.1 Calculate $\#\mathbb{T}$

還記得在定理 1 中所說的最少資料量嗎？現在對我們的 n 維物體來標示各種 orientation 的話，舉 \mathbb{R}^3 物體為例，可靠標記其中三個面的方位即可。所以（為了後續計算方便）我們就永遠選相鄰的三個邊，以數對 (A, B, C) 表示¹。那現在令所有可能的數對之集合：

$$\mathbb{P} = \{(D, E, F) \mid \angle DOE = \angle AOB, \angle DOF = \angle AOC, \angle EOF = \angle BOC\} \quad (5)$$

其中我們叫這保角性質我們所選之數對 (A, B, C) 的 Structure，而單純允許旋轉而不允許映射的情況下，這是一個該被確保的 Structure。另外須注意這 Structure 是有方向之分的，通常 A, B, C 採逆時鐘排列（從原點向外指，右手方向）。

那現在定義完 \mathbb{P} ，要計算旋轉方式就簡單多了。首先我們知道 $\forall P \in \mathbb{P}, \exists T \in \mathbb{T} \ni T \cdot (A, B, C) = P$ （其中 $T \cdot (A, B, C) = (TA, TB, TC)$ ）所以 $\#\mathbb{P} = \#\mathbb{T} = |G|$ 。而至於 \mathbb{P} 則可用排列組合推出，對立方體舉例：

先任意選一面 A

再選一面相鄰 A 的面 B

最後再選唯一一個在這兩面逆時鐘方向的面 C

$$\text{得 } \#\mathbb{P} = C_1^6 \times C_1^4 \times C_1^1 = 24$$

¹本文中所有表示面之數，皆為向量，且皆由原點指向該面之重心

對正四面體亦同：

先任意選一面 A

再選一面相鄰 A 的面 B

最後再選唯一一個在這兩面逆時鐘方向的面 C

$$\text{得 } \#P = C_1^4 \times C_1^3 \times C_1^1 = 12$$

或者嘗試將它 Generalized 對任意正多面體：

$$|G| = (\text{面數}) \times (\text{一面的邊數} - \text{一面相鄰的面數}) \quad (6)$$

2.2 Calculate \mathbb{T}

那算完他的數量，我們有沒有從它回推 \mathbb{T} 裡面的內容呢？有的（至少用電腦算式簡單的），我們甚至能算出他的組數（那個 k^n 的 n ）一樣拿立方體舉例：

$$\text{令所有面之向量之列表 } L = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \quad (7a)$$

$$\text{舉例取轉換 } T = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7b)$$

$$\text{得 } TL = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \quad (7c)$$

其中我們可看到，縱列 1~4 向右位移了一格，而 5,6 不變。因此可得對變換 T （ z 軸左手旋轉 90 度）可分成三組，也就是對 L, TL 做 disjoint set 後數他的數量。

就此我們就可以算出他的塗色可能性了。

```
2 disjointSetNum :: Matrix -> Matrix -> Int
3 disjointSetNum (Matrix a) (Matrix b) = sum $ map fromEnum (map eqPair (zip a b))
```

3 Extend Research Topics (未寫)

3.1 非正多面體

3.2 高維物體

3.3 魔術方塊

A Matrix

```
4 newtype Matrix = Matrix [[Int]]
5 vector :: [Int] -> Matrix
6 vector a = Matrix [a]
```

```
7 instance Show Matrix where
8     show (Matrix [x:xs])
9         | null xs = "[" ++ show x ++ "]"
10        | otherwise = "[" ++ show x ++ "]\n" ++ show (Matrix [xs])
11     show (Matrix rows)
12         | length (head rows) > 1 = "[" ++ (intercalate ", " (map (show.head) rows)) ++ "]\n"
13                                     ++ show (Matrix (map tail rows))
14        | otherwise = "[" ++ (intercalate ", " (map (show.head) rows)) ++ "]"
```

```
15 a = Matrix [[1,2],[2,0]]
16 b = Matrix [[2,4],[2,0]]
```

```
17 concatM :: Matrix -> Matrix -> Matrix
18 concatM (Matrix x) (Matrix y) = Matrix $ x ++ y
19 diagFlip :: Matrix -> Matrix
20 diagFlip (Matrix xs)
21     | length (head xs) > 1 = concatM (Matrix [map head xs]) (diagFlip (Matrix (map tail xs)))
22     | otherwise = Matrix [map head xs]
```

```
23 instance Num Matrix where
24     (Matrix [xs]) + (Matrix [ys]) = Matrix [zipWith (+) xs ys]
25     (Matrix (x:xs)) + (Matrix (y:ys)) = concatM (Matrix [zipWith (+) x y]) ((Matrix xs)+(Matrix ys))
26     (Matrix []) + a@(Matrix _) = a
27     a@(Matrix _) + (Matrix []) = a
28     x * (Matrix ys) = Matrix $ map f ys
29                     where f y = map (sum.zipWith (*)) y xs'
30                     (Matrix xs') = diagFlip x
```