期末報告週報

第十五組 - B11901018 朱冠融

本周主要負責項目

- 演算法與模擬

演算法與模擬：方法一

- 將INPUT拆解為眾多BLOCK
- 將每個BLOCK獨立進行模擬
- 儲存每次模擬後的形狀與位置

演算法與模擬：方法一

- INPUT：輸入（type = np.array）

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 0, 0, 0, 0, 0, 1, 0],
[0, 1, 0, 0, 0, 0, 0, 0, 1, 1],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

- BLOCK：區塊（type = np.array）

```
[0, 0, 0],
[0, 1, 0],
[0, 0, 0],
```

```
[0, 0, 0, 0, 0],
[0, 1, 1, 0, 0],
[0, 0, 1, 1, 0],
[0, 1, 0, 0, 0],
[0, 0, 0, 0, 0],
```

```
[0, 0, 0, 0],
[0, 1, 0, 0],
[0, 0, 1, 0],
[0, 0, 0, 0],
```

```
[0, 0, 0, 0, 0],
[0, 0, 1, 0, 0],
[0, 0, 1, 1, 0],
[0, 1, 1, 0, 0],
[0, 0, 0, 0, 0],
```

演算法與模擬：方法一

- 優點：對於龐大輸入資料時可以加速運算，尤其是當輸入資料分散且不互相影響
- 缺點：若每個BLOCK間會干擾彼此，運算速度會很慢；輸入資料小且多時（如分析4*4的所有可能），無法完全發揮BLOCK的功用

演算法與模擬：方法二

- 將INPUT整個進行模擬，存成TEXT
- 針對特定條件（如維持穩定）篩選TEXT的內容
- 儲存篩選後的結果

演算法與模擬：方法二

- INPUT：輸入（type = np.array）

```
[0, 0, 0, 0],
[1, 1, 1, 1],
[0, 0, 0, 0],
[0, 0, 1, 1],
```

```
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 1, 1, 1, 1, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 0],
[0, 0, 0, 0, 0, 0],
```

- TEXT：外觀（type = str）

```
000000000000011110000000000110000000
```

目前程式實際運行結果（擷取）

- 6*6 的穩定狀態

  （輸出為 .txt）

```
Algorithm test
Input cell size = 6 * 6
[[0 0 0 0 0 0]                [[0 0 0 0 0 0]                [[0 0 0 0 0 0]
 [0 0 0 0 0 0]                 [0 0 0 0 0 0]                 [0 0 0 0 0 0]
 [0 0 0 0 0 0]                 [0 0 0 0 0 0]                 [0 0 0 1 1 0]
 [0 0 0 0 0 0]                 [0 1 0 1 1 0]                 [0 0 1 0 1 0]
 [0 0 1 1 1 0]                 [0 1 1 0 1 0]                 [0 0 1 1 0 0]
 [0 0 0 0 0 0]]                [0 0 0 0 0 0]]                [0 0 0 0 0 0]]
[[0 0 0 0 0 0]                [[0 0 0 0 0 0]                [[0 0 0 0 0 0]
 [0 0 0 0 0 0]                 [0 0 0 0 0 0]                 [0 0 0 0 0 0]
 [0 0 0 0 0 0]                 [0 0 0 1 0 0]                 [0 0 0 1 1 0]
 [0 0 0 1 1 0]                 [0 0 1 0 1 0]                 [0 1 0 0 1 0]
 [0 0 0 1 1 0]                 [0 0 0 1 0 0]                 [0 1 1 0 0 0]
 [0 0 0 0 0 0]]                [0 0 0 0 0 0]]                [0 0 0 0 0 0]]
[[0 0 0 0 0 0]                [[0 0 0 0 0 0]                [[0 0 0 0 0 0]
 [0 0 0 0 0 0]                 [0 0 0 0 0 0]                 [0 0 0 0 0 0]
 [0 0 0 0 0 0]                 [0 0 0 1 0 0]                 [0 0 1 1 0 0]
 [0 0 1 1 1 0]                 [0 0 1 0 1 0]                 [0 1 0 0 1 0]
 [0 1 1 1 0 0]                 [0 0 0 1 1 0]                 [0 0 1 1 0 0]
 [0 0 0 0 0 0]]                [0 0 0 0 0 0]]                [0 0 0 0 0 0]]
```

目前困難點一

- 模擬與運算量隨輸入資料成指數型成長，

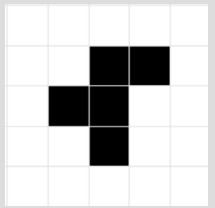
  如：4*4有2^{16} (65536)種組合，6*6有2^{36} (~6.87E10)種組合

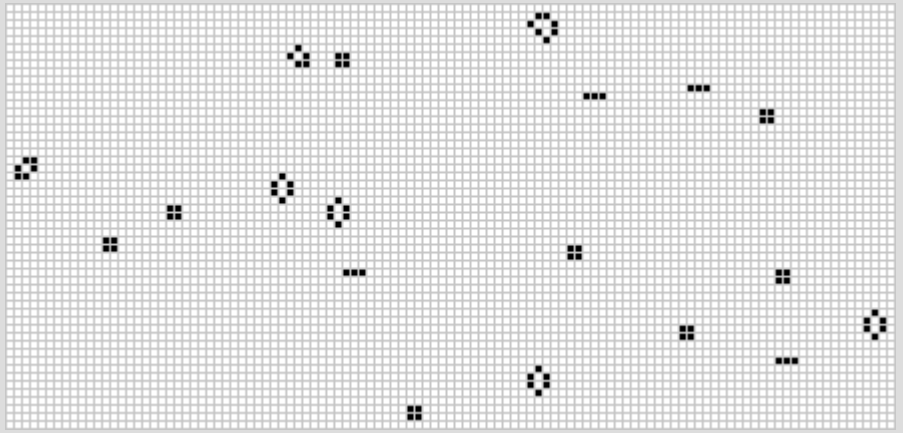
- 須尋找更佳的演算法，如：進行事前篩選、優化運算

目前困難點二

- 小範圍的細胞演化到後期可能變極大範圍，

  如：





T=1103

可能的解決方法：困難點一

- 困難點一：Complexity = $O(2^n)$
- 拆解成小塊模擬（如：同時運用方法一與方法二）：

  若有一整列或一整行為 0（使用 all( ), any( )），

  可以分成兩塊探討

可能的解決方法：困難點二

- 困難點二：size increment
- 1. 限制模擬範圍，超出範圍的跳過

  （仍可正常模擬穩定狀態）

  2. 針對超出範圍的改變模擬範圍

程式運行方式

- 建立多個函數
- 將二進位數列轉為INPUT（即0001, 0010, 0011, 0100, …）
- 將結果（如：哪些初始狀況會「收斂」）寫於 output.txt

補充：目前無法證明的假設

- 所有初始狀態都會收斂，

  變成穩定（含「空」）、飛船或震盪？

  （目前仍無法論證，專題也沒有要討論，但可以提供想法）

限制條件

- 在「所有初始狀況都會收斂」的前提下，須限制在一定周期內會不會有收斂的情況
- 因此我們這裡的「收斂」定義成「會回到初始狀態」

参考資料

- https://conwaylife.com/wiki/Main_Page