

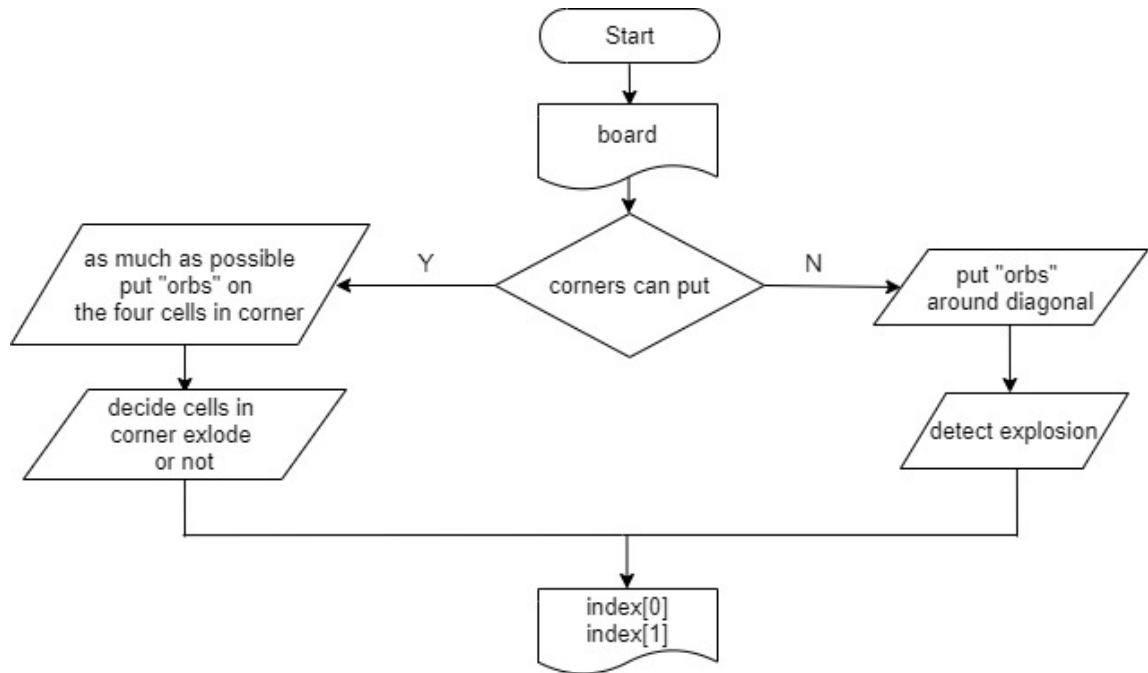
# Project #3: Chain Reaction

姓名：陳泳晗

學號：107062162

## 1) Project Description

### 1-1) Program Flow Chart



## 1-2) Detailed Description

我設計的 algorithm\_A 主要分成兩個部分。

第一個部分就是四個角落的情況，因為四個角落只要放置兩個 “orbs” 就會引發爆炸，而且角落的 orthogonal adjacent cells 只有兩個，只要控管好旁邊的兩個 cells（如果有敵方 color 的棋子 == 2，就將角落引爆，根據連續爆炸的規則，旁邊的 cells 爆炸之後又會在 corner 放上一個己方 color 的 “orbs”），就可以確保角落裡一直都是己方 color，所以盡量先佔據角落。

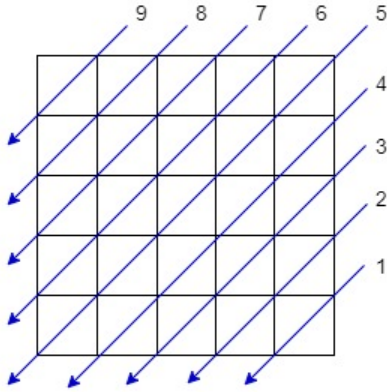
如果角落是空的，就將 “orbs” 放置在上面。

```
//put "orbs" on four cells in corner
if(board.get_cell_color(0, 0) == 'w'){
    index[0] = 0;
    index[1] = 0;
}
else if(board.get_cell_color(4, 5) == 'w'){
    index[0] = 4;
    index[1] = 5;
}
```

如果角落的旁邊有可以引發爆炸的敵方 cells，就將 corner 引爆。

```
//four corner explode or not
else if( board.get_cell_color(0,0)==color &&
        ((board.get_cell_color(0,1)!=color && board.get_orbs_num(0,1)==2) ||
         (board.get_cell_color(1,0)!=color && board.get_orbs_num(1,0)==2))){
    index[0] = 0;
    index[1] = 0;
}
```

第二個部分就是當角落已經被佔據了之後其餘 cells 的處理方式，就是依照下圖的順序，依序從右下到左上將每一個 cells 的 “orbs” 放到差一個會爆炸的程度，然後再以類似角落的處理方式來判斷要不要引爆 cells（周圍有敵方 “orbs” 並且已經到差一個 “orbs” 的程度）



從右下往左上掃，如果還沒滿 ( $\text{capacity} - \text{orbs\_num} > 1$ ) 就將 “orbs” 放上去

```
//put "orbs" around diagonal
for(int k=0;k<ROW;k++){
    for(int i=0, j=k;i<=k &&j>=0;i++, j--){
        if(board.get_cell_color(i,j)==color || board.get_cell_color(i,j)=='w'){
            if( (board.get_capacity(i,j)-board.get_orbs_num(i,j))>1 ){
                row=i;
                row_tmp=i;
                col=j;
                col_tmp=j;
            }
            else{
                row = row_tmp;
                col = col_tmp;
            }
        }
    }
}
```

如果己方可引爆的 cells 旁邊有可以引發爆炸的敵方 cells，就將這個 cells 引爆。

```
//detect explosion
bool explode=false;
for(int i=0;i<ROW;i++){
    for(int j=0;j<COL;j++){
        if(board.get_cell_color(i,j)==color || board.get_cell_color(i,j)=='w'){
            if( (board.get_capacity(i,j)-board.get_orbs_num(i,j))==1 ){
                if( (i-1)>=0 && board.get_cell_color(i-1, j)!=color &&
                    (board.get_capacity(i-1,j)-board.get_orbs_num(i-1,j))==1){
                    row=i;
                    col=j;
                    explode=true;
                    break;
                }
            }
        }
    }
}
```

## 2) Screen Shots

### 2-1) Partial Implemented Code

```
1. void algorithm_A(Board board, Player player, int index[]){
2.
3.     srand(time(NULL));
4.     int row, col;
5.     int row_tmp, col_tmp;
6.     int color = player.get_color();
7.
8.     //put "orbs" on four cells in corner
9.     if(board.get_cell_color(0, 0) == 'w'){
10.         index[0] = 0;
11.         index[1] = 0;
12.     }
13.     else if(board.get_cell_color(4, 5) == 'w'){
14.         index[0] = 4;
15.         index[1] = 5;
16.     }
17.     else if(board.get_cell_color(4, 0) == 'w'){
18.         index[0] = 4;
19.         index[1] = 0;
20.     }
21.     else if(board.get_cell_color(0, 5) == 'w'){
22.         index[0] = 0;
23.         index[1] = 5;
24.     }
25.     //four corner explode or not
26.     else if(board.get_cell_color(0,0)==color && ((board.get_cell_color(0,1)!=color
&& board.get_orbs_num(0,1)==2) || (board.get_cell_color(1,0)!=color && board.get_orbs_num(1,0)==2))){
27.         index[0] = 0;
28.         index[1] = 0;
29.     }
30.     else if(board.get_cell_color(4,5)==color && ((board.get_cell_color(4,4)!=color
&& board.get_orbs_num(4,4)==2) || (board.get_cell_color(3,5)!=color && board.get_orbs_num(3,5)==2))){
31.         index[0] = 4;
32.         index[1] = 5;
33.     }
34.     else if(board.get_cell_color(4,0)==color && ((board.get_cell_color(3,0)!=color
&& board.get_orbs_num(3,0)==2) || (board.get_cell_color(4,1)!=color && board.get_orbs_num(4,1)==2))){
35.         index[0] = 4;
36.         index[1] = 0;
37.     }
38.     else if(board.get_cell_color(0,5)==color && ((board.get_cell_color(0,4)!=color
&& board.get_orbs_num(0,4)==2) || (board.get_cell_color(1,5)!=color && board.get_orbs_num(1,5)==2))){
39.         index[0] = 0;
40.         index[1] = 5;
41.     }
42.     else{
43.         //put "orbs" around diagonal
44.         for(int k=0;k<ROW;k++){
45.             for(int i=0, j=k;i<=k && j>=0;i++, j--){
46.                 if(board.get_cell_color(i,j)==color || board.get_cell_color(i,j)=='w'){
47.                     if( (board.get_capacity(i,j)-board.get_orbs_num(i,j))>1 ){
48.                         row=i;
49.                         row_tmp=i;
50.                         col=j;
51.                         col_tmp=j;
52.                     }
53.                     else{
54.                         row = row_tmp;
55.                         col = col_tmp;
56.                     }
57.                 }
            }
        }
    }
}
```

## 2-2) GitHub Control History

[https://github.com/YHanTan/DS\\_project3](https://github.com/YHanTan/DS_project3)

YHanTan / DS\_project3

Unwatch

1

Star

0

Fork

0

<> Code

Issues 0

Pull requests 0

Actions

Projects 0

Wiki

Security

Insights

Settings

Branch: master

Commits on Jan 15, 2020

last_version	Verified	6554244	
YHanTan committed 38 minutes ago			
without avoid "null"	Verified	8abb359	
YHanTan committed 1 hour ago			
avoid "null"_v2	Verified	bce51f9	
YHanTan committed 1 hour ago			
to avoid "null"	Verified	390a3bc	
YHanTan committed 3 hours ago			
adjust order of loop	Verified	9c0e7d1	
YHanTan committed 3 hours ago			
put the orbs by diagonal	Verified	2e514ea	
YHanTan committed 3 hours ago			
templates & README	Verified	1b26450	
YHanTan committed 6 hours ago			
templates ...	Verified	e8423b0	
YHanTan committed 6 hours ago			
6/10 ...	Verified	1622479	
YHanTan committed 6 hours ago			

## 2-3) Compare with TA' s AI Code with Student Id

algorithm\_B

	Red	Blue	Result
1	algorithm_B	algorithm_A	<pre> Round: 46 Place orb on (0, 3)  =====  X   XX   X   XX   XX   X   X   XXX       X   XXX   XX         XX   X   XXX   X   XX   XX   XXX   XXX   X   XX   X   X       XX   XX   X  =====  Blue Player won the game !!! </pre>
2	algorithm_A	algorithm_B	<pre> Round: 45 Place orb on (1, 0)  =====      00   00   0   0   0   0   000   0   00   000   0   00   0       000     00   00   0   0   00   000   00   0       0   00   00   0  =====  Red Player won the game !!! </pre>
3	algorithm_B	algorithm_A	<pre> Round: 58 Place orb on (0, 5)  =====  X   XXX   XX   XX     XX   XX       X   XX   XXXX     X   XXXX   XX       XXX   XX   XX   XXX   XXX   XX   XX     X   XX   X   XX     XXX  =====  Blue Player won the game !!! </pre>
4	algorithm_A	algorithm_B	<pre> Round: 55 Place orb on (1, 3)  =====  00       00   00   00   0   0   00   00   00   0   00   00     0   0   00   00   0   0000   000   000   000   00   0   0   00   0   0   0  =====  Red Player won the game !!! </pre>
5	algorithm_B	algorithm_A	<pre> Round: 16 Place orb on (4, 0)  =====  X             X   XX                   XXXX         X   XX     XX       XX   X  =====  Blue Player won the game !!! </pre>
	algorithm_A vs algorithm_B		5:0

Result after 100 rounds: 87:13

# algorithm\_C

	Red	Blue	Result
1	algorithm_C	algorithm_A	<p>Round: 18 Place orb on (2, 5)</p> <pre> =====  X                  XX   X                       XXX  X                 x           X                       XX   XX    X                  XX   X    ===== Blue Player won the game !!! </pre>
2	algorithm_A	algorithm_C	<p>Round: 17 Place orb on (2, 0)</p> <pre> =====  0   00              0     0   000                  0      0                 0   00              0     0   0            0   0    ===== Red Player won the game !!! </pre>
3	algorithm_C	algorithm_A	<p>Round: 18 Place orb on (2, 5)</p> <pre> =====  X                  XX   X                       XXX  X                 x           X                       XX   XX    X                  XX   X    ===== Blue Player won the game !!! </pre>
4	algorithm_A	algorithm_C	<p>Round: 35 Place orb on (0, 5)</p> <pre> =====  0      00  000 00       0   000 0      0   00    0   0   00  00  0   0                 00  00  00    0         00  00  0    ===== Red Player won the game !!! </pre>
5	algorithm_C	algorithm_A	<p>Round: 18 Place orb on (2, 5)</p> <pre> =====  X                  XX   X                       XXX  X                 x           X                       XX   XX    X                  XX   X    ===== Blue Player won the game !!! </pre>
	algorithm_A vs algorithm_C		5:0

Result after 100 rounds: 72:28

# algorithm\_D

	Red	Blue	Result
1	algorithm_D	algorithm_A	<p>Round: 34 Place orb on (0, 5)</p> <pre> =====  X         X   XX      XX              XX      XXX                    XXXX XXX   XX              XX   X   XXX        X         XX   X   XX   XX    ===== Blue Player won the game !!! </pre>
2	algorithm_A	algorithm_D	<p>Round: 58 Place orb on (3, 3)</p> <pre> =====  X   XX   X   XX   XX           XXX      XXX   XX   X     XXX X   XX   XXX      XX     X   XXXX XXX   XXX   XXX   XX     XX      X   X   X   X    ===== Blue Player won the game !!! </pre>
3	algorithm_D	algorithm_A	<p>Round: 12 Place orb on (0, 5)</p> <pre> =====  X            X      X                    XXXX XX                            X                  X   X    ===== Blue Player won the game !!! </pre>
4	algorithm_A	algorithm_D	<p>Round: 13 Place orb on (4, 0)</p> <pre> =====  O                  O     O                          0000                  OO      000            O    ===== Red Player won the game !!! </pre>
5	algorithm_D	algorithm_A	<p>Round: 60 Place orb on (1, 1)</p> <pre> =====  XX   X   XX   XX   X   X        XX   X   XXX   XX   XX     XXX      XXXX XX   XX        X   XXXX XX   XX      XXXX   X      XX         XXXX X    ===== Blue Player won the game !!! </pre>
	algorithm_A vs algorithm_D		4:1

Result after 100 rounds: 92:8



# algorithm\_E

	Red	Blue	Result
1	algorithm_E	algorithm_A	<p>Round: 62 Place orb on (0, 0)</p> <pre> =====        XX    XX    XX    XX    XX   XX    XX    XXX          XX         X     XX    XXX    X     XXX    X    X     XX    XXX    XXX    XXX    XX         X     XX    X     X         ===== </pre> <p>Blue Player won the game !!!</p>
2	algorithm_A	algorithm_E	<p>Round: 55 Place orb on (3, 4)</p> <pre> =====  0     0     000          00    0    00    000          00    00    000   00          000    000    00               00    000    00          000   0     00    000          000        ===== </pre> <p>Red Player won the game !!!</p>
3	algorithm_E	algorithm_A	<p>Round: 57 Place orb on (0, 4)</p> <pre> =====  0     000          00       0    0     000    0     00    00    00   00    00    00    00    000    000   00    00    000    00    00         0     0     0     0        000  ===== </pre> <p>Red Player won the game !!!</p>
4	algorithm_A	algorithm_E	<p>Round: 55 Place orb on (0, 2)</p> <pre> =====        0     00    00    00    00   0     000          00    00         00    000    000    0000          00   0     00    000    00    000    00         0     00    0     0         ===== </pre> <p>Red Player won the game !!!</p>
5	algorithm_E	algorithm_A	<p>Round: 50 Place orb on (0, 5)</p> <pre> =====  X     XX    X     XX    XX         X           XXX    XX          XXX   X     X     XX          XXXX         XX    XX    XXX    XXX          XXX   X     X     X           XXXX        ===== </pre> <p>Blue Player won the game !!!</p>
	algorithm_A vs algorithm_E		4:1

Result after 100 rounds: 90:10

可以打敗各個 algorithmn 的理由我覺得是因為我的 algorithm 是以比較保守的方式去處理 “orbs” 的放置，我判斷的方式是保證引爆過後敵方的至少一個 cells 會變成己方的 cells，而且引爆的 cells 也會被放置回己方的 cells，不會為 empty，就是保證起碼會有一次連環爆，讓己方的 cells 在每次決定引爆的時候都會+1，這樣在對上助教的 algorithm 時，因為助教的 algorithm\_D&E 單單只是從 “orbs” 的多寡來判斷要不要放置和引爆，所以我這種保守的策略就挺有效的。但是在對付 algorithm\_B&C 這兩個隨機和半隨機的 algorithm 的時候，我的 algorithm 的效果就沒那麼好了，但是因為我的第一個部分是要盡量佔據角落，那可以保證不會一開始就落於下風，至於後面的放置和引爆，如果不要形成拉鋸戰（“orbs” 的位置過於零散），我的 algorithm 的勝率還是偏高的。