課題：I235 4th Report　　言語：C++(Console Application)

氏名：GAO, Yuwei

学生番号：s1910092　　提出日：2019/05/08

コンテンツ

課題１　●○並べ方問題

問題を逆に考える：
○○○○○●●●●●の状態から何回操作したら○●○●○●○●○●になれる？

最初の状態をS_0とする（○○○○○●●●●●）。
ゴールの状態をS_Gとする（○●○●○●○●○●）。
n 回操作後の状態をS_nとする。

状態Sがゴールであるかとの判断関数f(S)：

$$\mathrm{f(S)} = \begin{cases} 1, S = S_G \\ 0, \mathrm{S} \neq S_G \end{cases}$$

白丸と黒丸の下に無限長い白黒反復四角を放置すると：
最初の状態：　○○○○○●●●●●
□■□■□■□■□■□■□■□■□■□■□■□■
ゴールの状態：　○●○●○●○●○●
□■□■□■□■□■□■□■□■□■□■□■□■

関数$\mathrm{e}(S)$＝下の四角の色と異なる白丸と黒丸の個数：

$$\mathrm{e}(S_0) = 4$$
$$\mathrm{e}(S_G) = 0$$

状態Sに、互いに隣となる同じ色な丸があるか　との判断関数 $g(S)$：

$$g(S)=\begin{cases}1, 互いに隣となる同じ色な丸がない\\0, 互いに隣となる同じ色な丸がある\end{cases}$$

状態Sに、すべての丸は互いに隣であるか　との判断関数$h(S)$：

$$h(S)=\begin{cases}1, 全ての丸は互いに隣である\\0, 全ての丸は互いに隣ではない\end{cases}$$

状態S_nに操作を実行し、$S_n \xrightarrow{Action} S_{n+1}$

$e(S_{n+1})-e(S_n)$の値は三つの可能性しかいない：

$$e(S_{n+1})-e(S_n)=2$$
$$e(S_{n+1})-e(S_n)=0$$
$$e(S_{n+1})-e(S_n)=-2$$

$e(S_{n+1})-e(S_n)=2$の場合：

S_n ○● → S_{n+1} ○● か　S_n ●○ → S_{n+1} ●○

□■□■□■□　■□■□■□■　■□■□■□□　■□■□■□■

$e(S_{n+1})-e(S_n)=0$の場合：

S_n ○● → S_{n+1} ○● か　S_n ●○ → S_{n+1} ●○ か

□■□■□■□　□■□■□■□　■□■□■□□　□■□■□■□

S_n ○○ → S_{n+1} ○○ か　S_n ●● → S_{n+1} ●● など

□■□■□■□　□■□■□■□　■□■□■□□　□■□■□■□

$e(S_{n+1})-e(S_n)=-2$の場合：

S_n ●○ → S_{n+1} ●○ か　S_n ○● → S_{n+1} ○●

□■□■□■□　■□■□■□■　■□■□■□□　■□■□■□■

下記の 4 つの条件がある場合には：

$$p(S)：\mathrm{f}(S)=1$$

$$\mathrm{q}(\mathrm{S})：\mathrm{e}(S)=0$$

$$\mathrm{r}(\mathrm{S})：\mathrm{g}(S)=1$$

$$\mathrm{s}(\mathrm{S})：h(S)=1$$

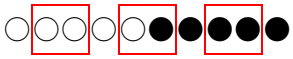
下記の公式が成立する：

$$\mathrm{q}(\mathrm{S})\wedge\mathrm{s}(\mathrm{S})\Leftrightarrow p(\mathrm{S})$$
$$\mathrm{r}(\mathrm{S})\wedge\mathrm{s}(\mathrm{S})\Leftrightarrow p(\mathrm{S})$$
$$\mathrm{q}(\mathrm{S})\Rightarrow\mathrm{r}(\mathrm{S})$$
$p(\mathrm{S})$は，$\mathrm{q}(\mathrm{S})$であるための十分条件
$p(\mathrm{S})$は，$\mathrm{r}(\mathrm{S})$であるための十分条件
$p(\mathrm{S})$は，$\mathrm{s}(\mathrm{S})$であるための十分条件
$p(\mathrm{S})$は，$\mathrm{q}(\mathrm{S})\wedge\mathrm{s}(\mathrm{S})$であるための十分条件
$p(\mathrm{S})$は，$\mathrm{r}(\mathrm{S})\wedge\mathrm{s}(\mathrm{S})$であるための十分条件
$\mathrm{q}(\mathrm{S})$は，$\mathrm{r}(\mathrm{S})$であるための十分条件

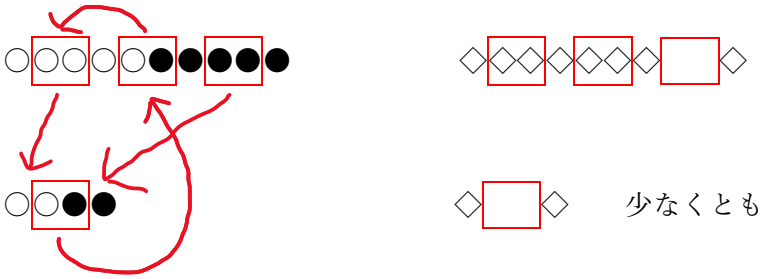
$\mathrm{p}(\mathrm{S})$に満たす最短手順$\geq \mathrm{q}(\mathrm{S})$か$\mathrm{r}(\mathrm{S})$か$\mathrm{s}(\mathrm{S})$…か$\mathrm{q}(\mathrm{S})\wedge\mathrm{s}(\mathrm{S})$に満たす最短手順

最短手順の下界を得るアルゴリズム：

まず最短r(S)に着目：



そして最短r(S)の上に s(S)を考えよう：

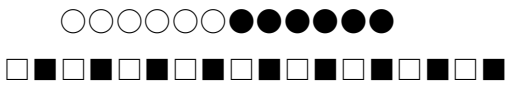


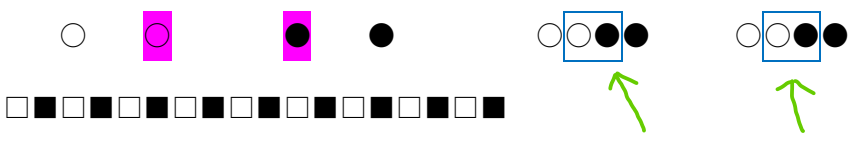
少なくとも 2Step が必要

左の矢印の数量+右の“最短 2Step”＝4+2＝6

つまり最短手順は≥ 6　（具体的な操作を得られないが、範囲を得られる）

黒丸と白丸が 6 ずつの場合：





$$e(S_0) = 6$$

$$e(S_G) = e(S_0) - 0 - 0 - 0 - 0 - 2 - 2 - 2 - 0 - 0 - 0 = 0$$

黒丸と白丸が 6 ずつの場合、最短手順は≥ 10

このような感じで：

黒丸と白丸が 5 ずつの場合、最短手順は≥ 6

黒丸と白丸が 6 ずつの場合、最短手順は≥ 10

黒丸と白丸が 7 ずつの場合、最短手順は≥ 10

黒丸と白丸が 8 ずつの場合、最短手順は≥ 11

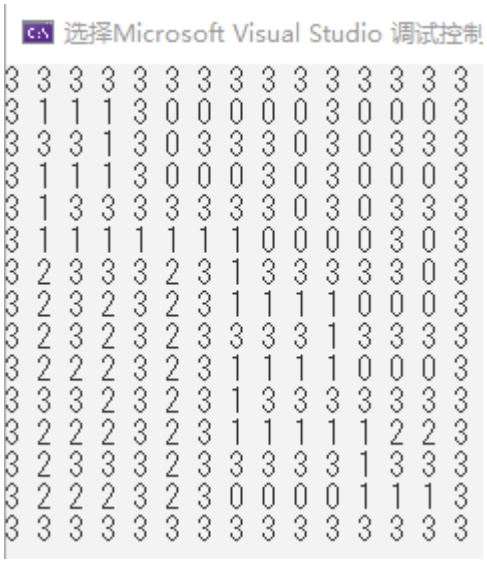
黒丸と白丸が 9 ずつの場合、最短手順は≥ 14

黒丸と白丸が 10 ずつの場合、最短手順は≥ 15

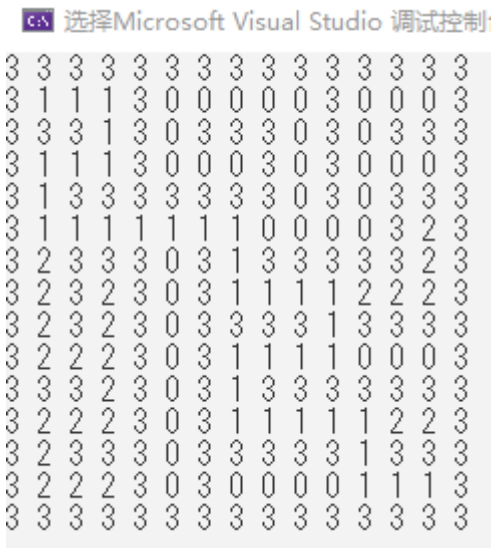
課題２　迷路　縦型

（ランダム要素があるので、二回の結果を貼った。）

実行結果 1:



実行結果 2:



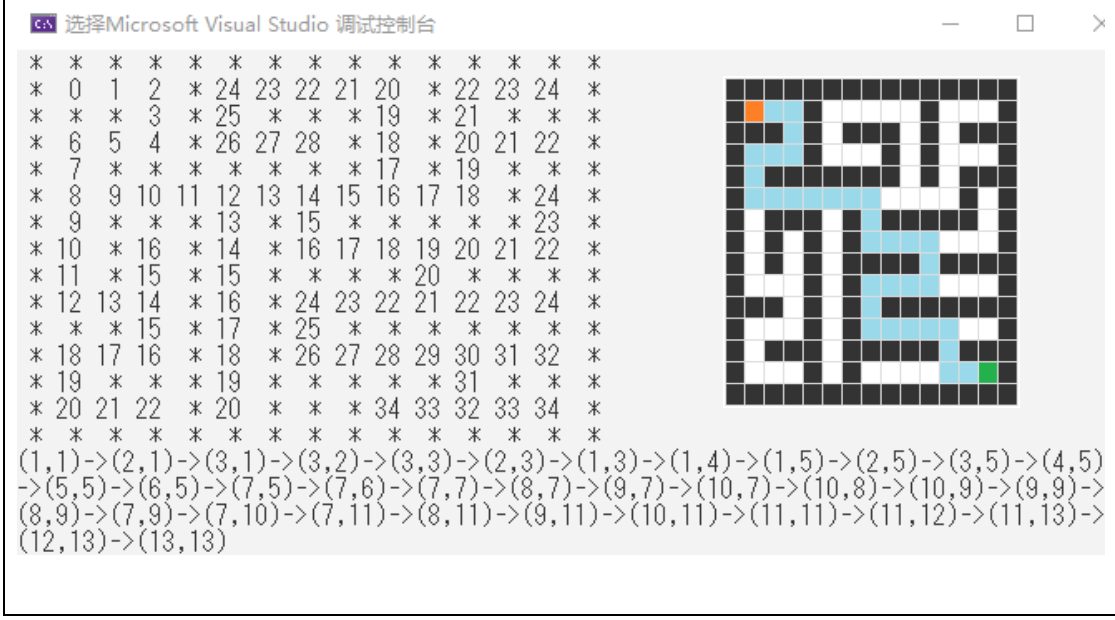
```cpp
#include <iostream>
#include <stdio.h>
#include <time.h>
#include <vector>
#define WIDTH 15
enum NEXT
{
    UP = 0,
    DOWN = 1,
    LEFT = 2,
    RIGHT = 3
};
std::vector <int> maze{//15×15の配列にループのない迷路
    'X','X','X','X','X','X','X','X','X','X','X','X','X','X','X',
    'X',' ',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ','X',
    'X','X','X',' ','X',' ',' ','X','X','X',' ','X',' ','X','X',
    'X',' ',' ',' ','X',' ',' ',' ',' ','X',' ','X',' ',' ','X',
    'X',' ','X','X','X','X','X','X','X',' ','X',' ','X','X','X',
    'X',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','X',' ','X',
    'X',' ','X','X','X',' ','X',' ','X','X','X','X','X',' ','X',
    'X',' ','X',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ','X',
    'X',' ','X',' ','X','X','X','X',' ','X','X','X','X',' ','X',
    'X',' ',' ',' ',' ',' ','X',' ','X',' ',' ',' ',' ',' ','X',
    'X','X','X',' ','X',' ','X',' ','X','X','X','X','X','X','X',
    'X',' ',' ',' ',' ','X',' ','X',' ',' ',' ',' ',' ',' ','X',
    'X',' ','X','X','X',' ','X','X','X','X','X',' ','X','X','X',
    'X',' ',' ',' ',' ','X',' ','X',' ',' ',' ',' ',' ',' ','X',
    'X','X','X','X','X','X','X','X','X','X','X','X','X','X','X'
};
static class MATH {
public:
    static int getI(int x, int y) { return y * WIDTH + x; }
    static int getNext(int next, int i)
    {
        switch (next)
        {
        case UP:return i - WIDTH; break;
        case DOWN:return i + WIDTH; break;
        case LEFT:return i - 1; break;
        case RIGHT:return i + 1; break;
        }
    }
};
int main()
{
    for (auto& i : maze)
    {
        i == 'X' ? i = 3 : i = 0;//0が空所，3が壁を表すことにする．
    }
    srand((unsigned int)time(NULL));
    std::vector<int> randomNumers;
    int now = MATH::getI(1, 1);//(1,1)からスタート
    int goal = MATH::getI(WIDTH - 2, WIDTH - 2);//(13,13)をゴール点と設定
    maze[now] = 1;//(1,1)は 1（進入済）にしておく
    while (now != goal)//(13, 13)に到達すればゴール
    {
```

```
57          randomNumers.clear();
58          while (randomNumers.size() <= RIGHT)//今いる場所から上下左右に0があれ
              ばランダムにそこに進み
59          {
60              int i;
61              std::vector<int>::iterator ret;
62              do
63              {
64                  i = rand() % (RIGHT + 1);
65                  ret = std::find(randomNumers.begin(), randomNumers.end(), i);
66
67              } while (ret != randomNumers.end());
68              randomNumers.push_back(i);
69          }
70          bool found = false;
71          for (int i = UP; i <= RIGHT; i++)
72          {
73              if (maze[MATH::getNext(randomNumers[i], now)] == 0)
74              {
75                  maze[MATH::getNext(randomNumers[i], now)] = 1; //進んだ先を1
                    （進入済）にする
76                  now = MATH::getNext(randomNumers[i], now);
77                  found = true;
78                  break;
79              }
80          }
81          if (!found)//上下左右に0がなければ
82          {
83              for (int i = UP; i <= RIGHT; i++)
84              {
85                  if (maze[MATH::getNext(randomNumers[i], now)] == 1)
86                  {
87                      maze[now] = 2;//今の場所を2（希望なし）にして
88                      now = MATH::getNext(randomNumers[i], now); //上下左右の1
                        のうちどれかをランダムに選んで進む
89                      break;
90                  }
91              }
92          }
93      }
94      for (int i = 0; i < WIDTH; i++)
95      {
96          for (int j = 0; j < WIDTH; j++)
97          {
98              std::cout << maze[i * WIDTH + j] << " ";
99          }
100         std::cout << "\n";
101     }
102     return 0;
103 }
104 /*////////////実行結果*2///////////////
105 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
106 3 1 1 1 3 0 0 0 0 0 3 0 0 0 3
107 3 3 3 1 3 0 3 3 3 0 3 0 3 3 3
108 3 1 1 1 3 0 0 0 3 0 3 0 0 0 3
109 3 1 3 3 3 3 3 3 3 0 3 0 3 3 3
```

```
110  3 1 1 1 1 1 1 1 0 0 0 0 3 2 3
111  3 0 3 3 3 0 3 1 3 3 3 3 3 2 3
112  3 0 3 0 3 0 3 1 1 1 1 2 2 2 3
113  3 0 3 0 3 0 3 3 3 3 1 3 3 3 3
114  3 0 0 0 3 0 3 1 1 1 1 0 0 0 3
115  3 3 3 0 3 0 3 1 3 3 3 3 3 3 3
116  3 0 0 0 3 0 3 1 1 1 1 1 0 0 3
117  3 0 3 3 3 0 3 3 3 3 3 1 3 3 3
118  3 0 0 0 3 0 3 0 0 0 0 1 1 1 3
119  3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
120
121  3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
122  3 1 1 1 3 0 0 0 0 0 3 0 0 0 3
123  3 3 3 1 3 0 3 3 3 0 3 0 3 3 3
124  3 1 1 1 3 0 0 0 3 0 3 0 0 0 3
125  3 1 3 3 3 3 3 3 3 0 3 0 3 3 3
126  3 1 1 1 1 1 1 1 0 0 0 0 3 2 3
127  3 2 3 3 3 0 3 1 3 3 3 3 3 2 3
128  3 2 3 2 3 0 3 1 1 1 1 2 2 2 3
129  3 2 3 2 3 0 3 3 3 3 1 3 3 3 3
130  3 2 2 2 3 0 3 1 1 1 1 0 0 0 3
131  3 3 3 2 3 0 3 1 3 3 3 3 3 3 3
132  3 2 2 2 3 0 3 1 1 1 1 1 2 2 3
133  3 2 3 3 3 0 3 3 3 3 3 1 3 3 3
134  3 2 2 2 3 0 3 2 2 2 2 1 1 1 3
135  3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
136  *////////////////////////////////////
```

課題３　迷路　横型

実行結果:



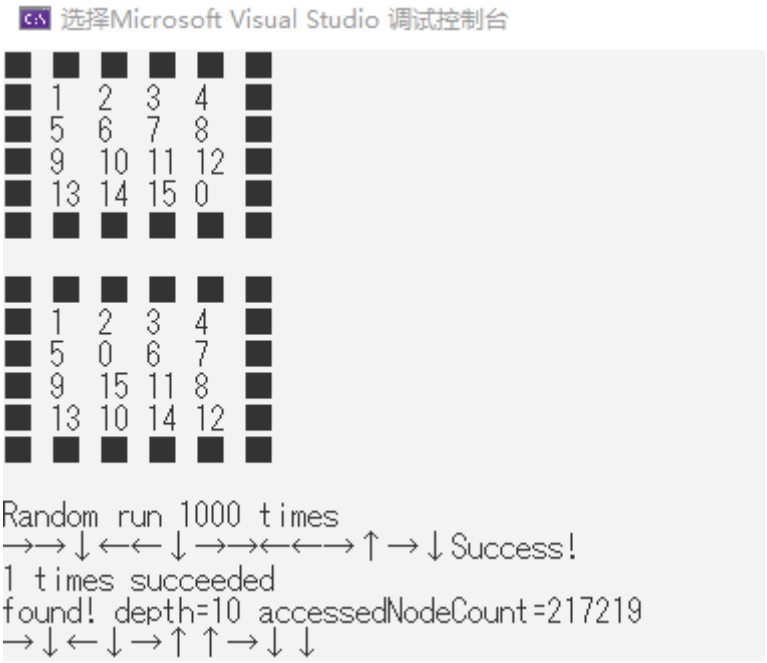
```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#define WIDTH 15
enum NEXT
{
    UP = 0,
    DOWN = 1,
    LEFT = 2,
    RIGHT = 3
};
std::vector <int> maze{//15×15の配列に迷路を適当に作る
    'X','X','X','X','X','X','X','X','X','X','X','X','X','X','X',
    'X',' ',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ','X',
    'X','X','X',' ',' ','X',' ',' ','X','X','X',' ',' ','X',' ',
    'X',' ',' ',' ',' ',' ','X',' ',' ',' ',' ',' ','X',' ',' ',
    'X',' ','X','X','X','X','X','X','X',' ','X',' ','X','X','X',
    'X',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',
    'X',' ','X','X','X',' ',' ','X',' ','X','X','X','X','X',' ',
    'X',' ','X',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',' ',
    'X',' ','X',' ',' ','X',' ',' ','X','X','X','X',' ','X','X',
    'X',' ',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',
    'X','X','X',' ',' ','X',' ',' ','X',' ','X','X','X','X','X',
    'X',' ',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',
    'X',' ','X','X','X',' ',' ','X','X','X','X','X',' ','X','X',
    'X',' ',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',
    'X','X','X','X','X','X','X','X','X','X','X','X','X','X','X'
};
static class MATH {
public:
    static int getX(int i) { return i % WIDTH; }
    static int getY(int i) { return i / WIDTH; }
    static int getI(int x, int y) { return y * WIDTH + x; }
    static int getNext(int next, int i)
    {
        switch (next)
        {
        case UP:return i - WIDTH; break;
        case DOWN:return i + WIDTH; break;
        case LEFT:return i - 1; break;
        case RIGHT:return i + 1; break;
        }
    }
};
int main()
{
    for (auto& i : maze)
    {
        i == 'X' ? i = -2 : i = -1;
    }
    int now = MATH::getI(1, 1);//(1,1)からスタート
    int goal = MATH::getI(WIDTH - 2, WIDTH - 2);//(13,13)をゴール点と設定
    maze[now] = 0;//(1,1)は 0にしておく
    std::vector<int>* searchList = new std::vector<int>(); //探索する予定List
    searchList->push_back(now);
    while (maze[goal] == -1)//(13, 13)に到達すればゴール
```

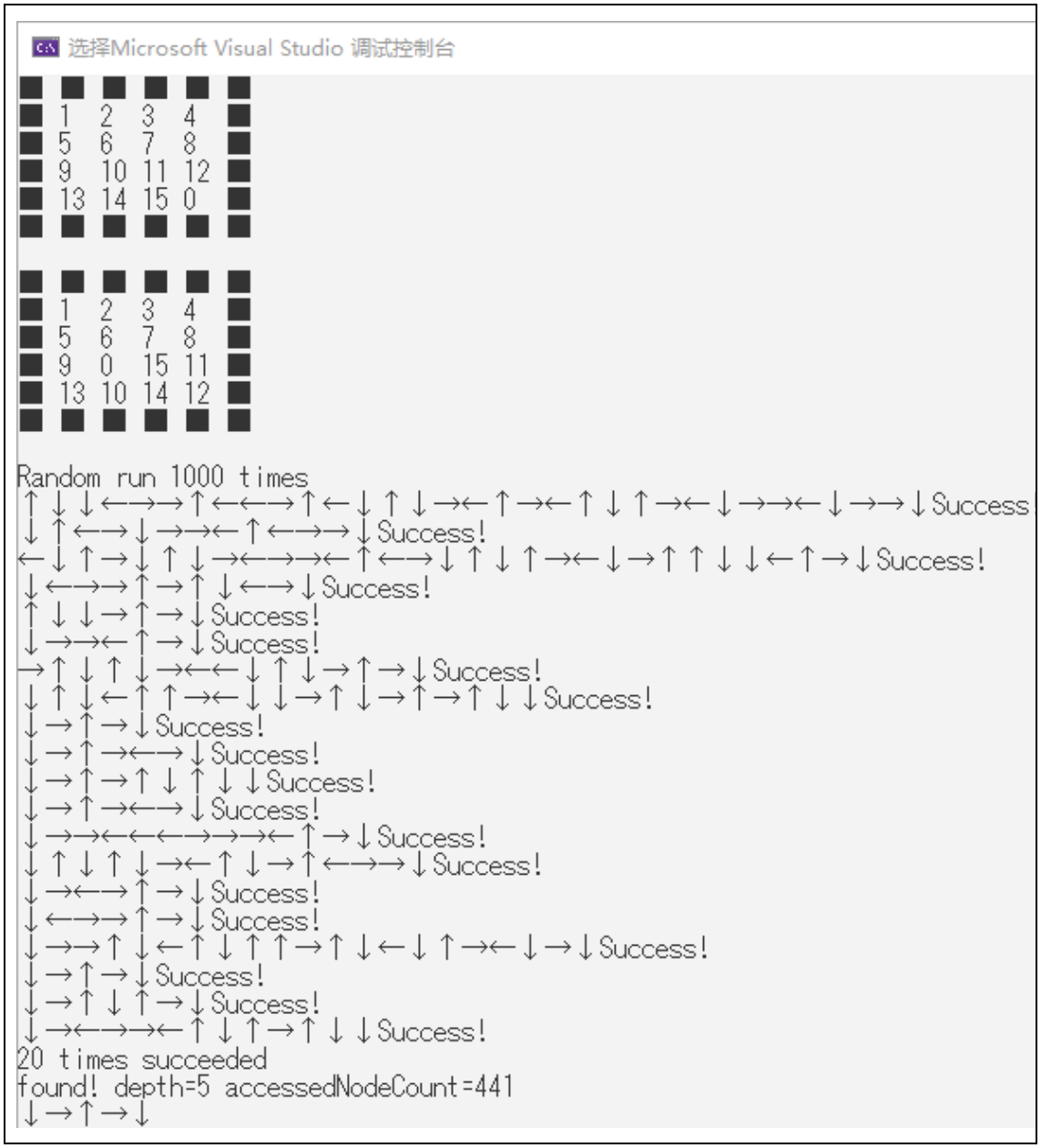
```
57      {
58          std::vector<int>* lin = new std::vector<int>();
59          for (auto& i : *searchList)
60          {
61              for (int j = UP; j <= RIGHT; j++)
62              {
63                  if (maze[MATH::getNext(j, i)] == -1)
64                  {
65                      maze[MATH::getNext(j, i)] = maze[i] + 1; //進んだ先を1（進入済）にする
66                      lin->push_back(MATH::getNext(j, i));//次の探索する予定List
67                  }
68              }
69          }
70          delete(searchList);
71          searchList = lin;//探索する予定Listを更新
72      }
73      for (int i = 0; i < WIDTH; i++)
74      {
75          for (int j = 0; j < WIDTH; j++)
76          {
77              if (maze[i * WIDTH + j] < 0)std::cout << " * ";
78              else
79              {
80                  if (maze[i * WIDTH + j] > 9)std::cout << maze[i * WIDTH + j] << " ";
81                  else std::cout << " " << maze[i * WIDTH + j] << " ";
82              }
83
84          }
85          std::cout << "\n";
86      }
87      now = goal;
88      std::vector<int> output;
89      while (maze[now] != 0)
90      {
91          for (int i = UP; i <= RIGHT; i++)
92          {
93              if (maze[MATH::getNext(i, now)] == maze[now] - 1)//逆向きに辿って経路を求める
94              {
95                  now = MATH::getNext(i, now);
96                  output.push_back(now);
97                  break;
98              }
99          }
100     }
101     std::for_each(output.rbegin(), output.rend(), [](auto&& s) {// (1,1) -> (1,2) -> (2,2)… (13,13)などと示す
102         std::cout << "(" << MATH::getX(s) << "," << MATH::getY(s) << ")->";
103         });
104     std::cout << "(" << MATH::getX(goal) << "," << MATH::getY(goal) << ")";
105     return 0;
106 }
107 /*////////////実行結果////////////////
108  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
```

```
109   *  0  1  2  * 24 23 22 21 20  * 22 23 24  *
110   *  *  *  3  * 25  *  *  * 19  * 21  *  *  *
111   *  6  5  4  * 26 27 28  * 18  * 20 21 22  *
112   *  7  *  *  *  *  *  *  * 17  * 19  *  *  *
113   *  8  9 10 11 12 13 14 15 16 17 18  * 24  *
114   *  9  *  *  * 13  * 15  *  *  *  *  * 23  *
115   * 10  * 16  * 14  * 16 17 18 19 20 21 22  *
116   * 11  * 15  * 15  *  *  *  * 20  *  *  *  *
117   * 12 13 14  * 16  * 24 23 22 21 22 23 24  *
118   *  *  * 15  * 17  * 25  *  *  *  *  *  *  *
119   * 18 17 16  * 18  * 26 27 28 29 30 31 32  *
120   * 19  *  *  * 19  *  *  *  *  * 31  *  *  *
121   * 20 21 22  * 20  *  *  * 34 33 32 33 34  *
122   *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
123  (1,1)->(2,1)->(3,1)->(3,2)->(3,3)->(2,3)->(1,3)->(1,4)->(1,5)->(2,5)->(3,5)->
       (4,5)->(5,5)->(6,5)->(7,5)->(7,6)->(7,7)->(8,7)->(9,7)->(10,7)->(10,8)->
       (10,9)->(9,9)->(8,9)->(7,9)->(7,10)->(7,11)->(8,11)->(9,11)->(10,11)->
       (11,11)->(11,12)->(11,13)->(12,13)->(13,13)
124  *//////////////////////////////////////
```

課題４　16 パズル

・正しく並んだ状態 A から 20 回適当に行動をとることで，初期盤面 B を作れ.
（簡単すぎないように）
20 回行動したら、なんか 100 回ランダム*1000 に行動を繰り返しても、全然見つからないので、とりあえず 10 回くらいにした。（複雑すぎるかなあ）
・「B から 100 回ランダムに行動」を繰り返すことで，（途中で）A に到達する解を何個か見つけてみよ．またその手数を示せ.
10~12 回行動したら、100 回ランダム*1000 に行動を繰り返して、0-2 個解を見つかる場合が多い。（図 1）
5~7 回行動したら、100 回ランダム*1000 に行動を繰り返して、15-25 個解を見つかる場合が多い。（図 2）
B から A に到達する最短手順を，反復深化法で求めよ．またその手数，探索したノード数を示せ.
10~12 回行動したら、Depth=10、探索したノード数=217219。（図 1）
5~7 回行動したら、Depth=5、探索したノード数=441。（図 2）
探索ノード数を減らすような工夫を考えよ.
探索完成されたノードを記録し、そして同じな状態になったら、繰り返し探索しないようにすれば探索ノード数を減らすよう。




选择Microsoft Visual Studio 调试控制台
■ ■ ■ ■ ■ ■
■ 1 2 3 4 ■
■ 5 6 7 8 ■
■ 9 10 11 12 ■
■ 13 14 15 0 ■
■ ■ ■ ■ ■ ■
■ ■ ■ ■ ■ ■
■ 1 2 3 4 ■
■ 5 6 7 8 ■
■ 9 0 15 11 ■
■ 13 10 14 12 ■
■ ■ ■ ■ ■ ■
Random run 1000 times
↑↓↓←→→↑←←→↑←↓↑↓→←↑→←↑↓↑→←↓→→←↓→→↓Success
↓↑←→↓→→←↑←→→↓Success!
←↓↑→↓↑↓→←→→←↑←→↓↑↓↑→←↓→↑↑↓↓←↑→↓Success!
↓←→→↑→↑↓←→↓Success!
↑↓↓→↑→↓Success!
↓→→←↑→↓Success!
→↑↓↑↓→←←↓↑↓→↑→↓Success!
↓↑↓←↑↑→←↓↓→↑↓→↑→↑↓↓Success!
↓→↑→↓Success!
↓→↑→←→↓Success!
↓→↑→↑↓↑↓↓Success!
↓→↑→←→↓Success!
↓→→←←←→→→←↑→↓Success!
↓↑↓↑↓→←↑↓→↑←→→↓Success!
↓→←→↑→↓Success!
↓←→→↑→↓Success!
↓→→↑↓←↑↓↑↑→↑↓←↓↑→←↓→↓Success!
↓→↑→↓Success!
↓→↑↓↑→↓Success!
↓→←→→←↑↓↑→↑↓↓Success!
20 times succeeded
found! depth=5 accessedNodeCount=441
↓→↑→↓

```cpp
#include <iostream>
#include <vector>
#include <stdio.h>
#include <time.h>
#include <algorithm>
#define WIDTH (4+2)
#define LOOPCOUNT 1000
#define LOOP 100
#define DEPTH 20
enum
{
    UP = 0,
    DOWN = 1,
    LEFT = 2,
    RIGHT = 3
};
static class MATH {
public:
    static int getX(int i) { return i % WIDTH; }
    static int getY(int i) { return i / WIDTH; }
    static int getI(int x, int y) { return y * WIDTH + x; }
    static int getNext(int next, int i)
    {
        switch (next)
        {
        case UP:return i - WIDTH; break;
        case DOWN:return i + WIDTH; break;
        case LEFT:return i - 1; break;
        case RIGHT:return i + 1; break;
        }
    }
    static std::vector<int>* getMoveableList(std::vector<int>& v)
    {
        auto vv = new std::vector<int>();
        for (int i = UP; i <= RIGHT; i++)
        {
            if (isMoveable(i, v))vv->push_back(i);
        }
        return vv;
    }
    static bool isMoveable(int direction, std::vector<int>& v)
    {
        if (v[getNext(direction, -v[0])] < 0)return false;
        return true;
    }
    static std::vector<int>* move(int direction, std::vector<int> & v)
    {
        int lin = v[getNext(direction, -v[0])];
        v[getNext(direction, -v[0])] = 0;
        v[-v[0]] = lin;
        v[0] = -getNext(direction, -v[0]);
        return &v;
    }
};
void print(std::vector<int> pazuru)
{
```

```
57      for (int i = 0; i < WIDTH; i++)
58      {
59          for (int j = 0; j < WIDTH; j++)
60          {
61              if (pazuru[i * WIDTH + j] > 9) { std::cout << pazuru[i * WIDTH +
                  j] << " "; }
62              else if (pazuru[i * WIDTH + j] < 0) {
63                  std::cout << "■ ";
64              }
65              else
66              {
67                  std::cout << pazuru[i * WIDTH + j] << "  ";
68              }
69          }
70          std::cout << "\n";
71      }
72      std::cout << "\n";
73  }
74  int main()
75  {
76      std::vector<int> pazuru;
77      for (int i = 0, wallCount = 0; i < WIDTH * WIDTH; i++)
78      {
79          if (MATH::getX(i) == 0 || MATH::getX(i) == WIDTH - 1 ||
80              MATH::getY(i) == 0 || MATH::getY(i) == WIDTH - 1)
81          {
82              pazuru.push_back(-1);
83              wallCount++;
84          }
85          else if (MATH::getX(i) == WIDTH - 2 && MATH::getY(i) == WIDTH - 2) {
86              pazuru.push_back(0);
87              pazuru[0] = -i;
88          }
89          else
90          {
91              pazuru.push_back(i - wallCount + 1);
92          }
93      }
94      print(pazuru);//正しく並んだ状態
95      auto s_a = pazuru;//s_a=正しく並んだ状態
96      MATH::move(UP, pazuru);//20回適当に行動をとること(ここは10回)
97      MATH::move(LEFT, pazuru);
98      MATH::move(DOWN, pazuru);
99      MATH::move(LEFT, pazuru);
100     MATH::move(UP, pazuru);
101     MATH::move(LEFT, pazuru);
102     MATH::move(RIGHT, pazuru);
103     MATH::move(RIGHT, pazuru);
104     MATH::move(RIGHT, pazuru);
105     MATH::move(UP, pazuru);
106     //MATH::move(LEFT, pazuru);
107     //MATH::move(LEFT, pazuru);
108     //MATH::move(LEFT, pazuru);
109     //MATH::move(DOWN, pazuru);
110     //MATH::move(DOWN, pazuru);
111     //MATH::move(RIGHT, pazuru);
```

```
112     //MATH::move(UP, pazuru);
113     //MATH::move(UP, pazuru);
114     //MATH::move(UP, pazuru);
115     //MATH::move(RIGHT, pazuru);
116     auto s_b = pazuru; //初期盤面Bを作れ
117     print(pazuru);
118     srand((unsigned int)time(NULL));
119     int successCount = 0;
120     std::cout << "Random run " << LOOPCOUNT << " times\n";
121     for (int i = 0; i < LOOPCOUNT; i++)//「Bから100回ランダムに行動」
          LOOPCOUNT=1000回を繰り返すこと
122     {
123         pazuru = s_b;
124         auto output = new std::vector<int>();
125         for (int j = 0; j < LOOP; j++)//BからLOOP=100回ランダムに行動
126         {
127             auto ml = MATH::getMoveableList(pazuru);
128             int direction = rand() % (ml->size());
129             MATH::move((*ml)[direction], pazuru);
130             output->push_back((*ml)[direction]);//手数を示せ
131             if (pazuru == s_a)//Aに到達する解を何個か見つけて
132             {
133                 successCount++;//個数を表示
134                 for (auto k : *output)//手数を示せ
135                 {
136                     switch (k)
137                     {
138                     case UP:std::cout << "↑"; break;
139                     case DOWN:std::cout << "↓"; break;
140                     case LEFT:std::cout << "←"; break;
141                     case RIGHT:std::cout << "→"; break;
142                     default:break;
143                     }
144                 }
145                 std::cout << "Success!\n";
146                 break;
147             }
148             delete(ml);
149         }
150         delete(output);
151     }
152     std::cout << successCount << " times succeeded\n";
153     //BからAに到達する最短手順を，反復深化法で求め
154     std::vector<std::pair<std::vector<int>, std::vector<int>>> search;
155     std::vector<std::pair<std::vector<int>, std::vector<int>>> nextSearch;
156     search.push_back({ s_b,*(new std::vector<int>) });
157     unsigned long accessedNodeCount = 0;//探索したノード数
158     for (int i = 0; i < DEPTH; i++)
159     {
160         for (auto& j : search)
161         {
162             auto k = MATH::getMoveableList(j.first);
163             for (auto& l : *k)
164             {
165                 auto jClong = *(new std::pair<std::vector<int>,
                      std::vector<int>>());
```

```
166                     jClong = j;
167                     MATH::move(1, jClong.first);
168                     jClong.second.push_back(1);
169                     nextSearch.push_back(jClong);
170                 }
171                 delete(k);
172             }
173             for (auto& j : nextSearch)
174             {
175                 accessedNodeCount++;
176                 if (j.first == s_a)
177                 {
178                     std::cout << "found! depth=" << i + 1 << " accessedNodeCount=" <
                        << accessedNodeCount << "\n";
179                     for (auto& k : j.second)//手数
180                     {
181                         switch (k)
182                         {
183                         case UP:std::cout << "↑"; break;
184                         case DOWN:std::cout << "↓"; break;
185                         case LEFT:std::cout << "←"; break;
186                         case RIGHT:std::cout << "→"; break;
187                         default:break;
188                         }
189                     }
190                     return 0;
191                 }
192             }
193             search = nextSearch;
194             nextSearch.clear();
195         }
196         return 0;
197 }
198 /*////////////実行結果////////////////
199 ■ ■ ■ ■ ■ ■
200 ■ 1  2  3  4  ■
201 ■ 5  6  7  8  ■
202 ■ 9  10 11 12 ■
203 ■ 13 14 15 0  ■
204 ■ ■ ■ ■ ■ ■
205
206 ■ ■ ■ ■ ■ ■
207 ■ 1  2  3  4  ■
208 ■ 5  0  6  7  ■
209 ■ 9  15 11 8  ■
210 ■ 13 10 14 12 ■
211 ■ ■ ■ ■ ■ ■
212
213 Random run 1000 times
214 ↑↓←→→→↑←↓↑→↓↑←↓↑↓↓←↓→↑↑↑←→→←→↓↓↑↓↓Success!
215 1 times succeeded
216 found!depth = 10 accessedNodeCount = 217219
217 →↓←↓→↑↑→↓↓
218 *////////////////////////////////////
```