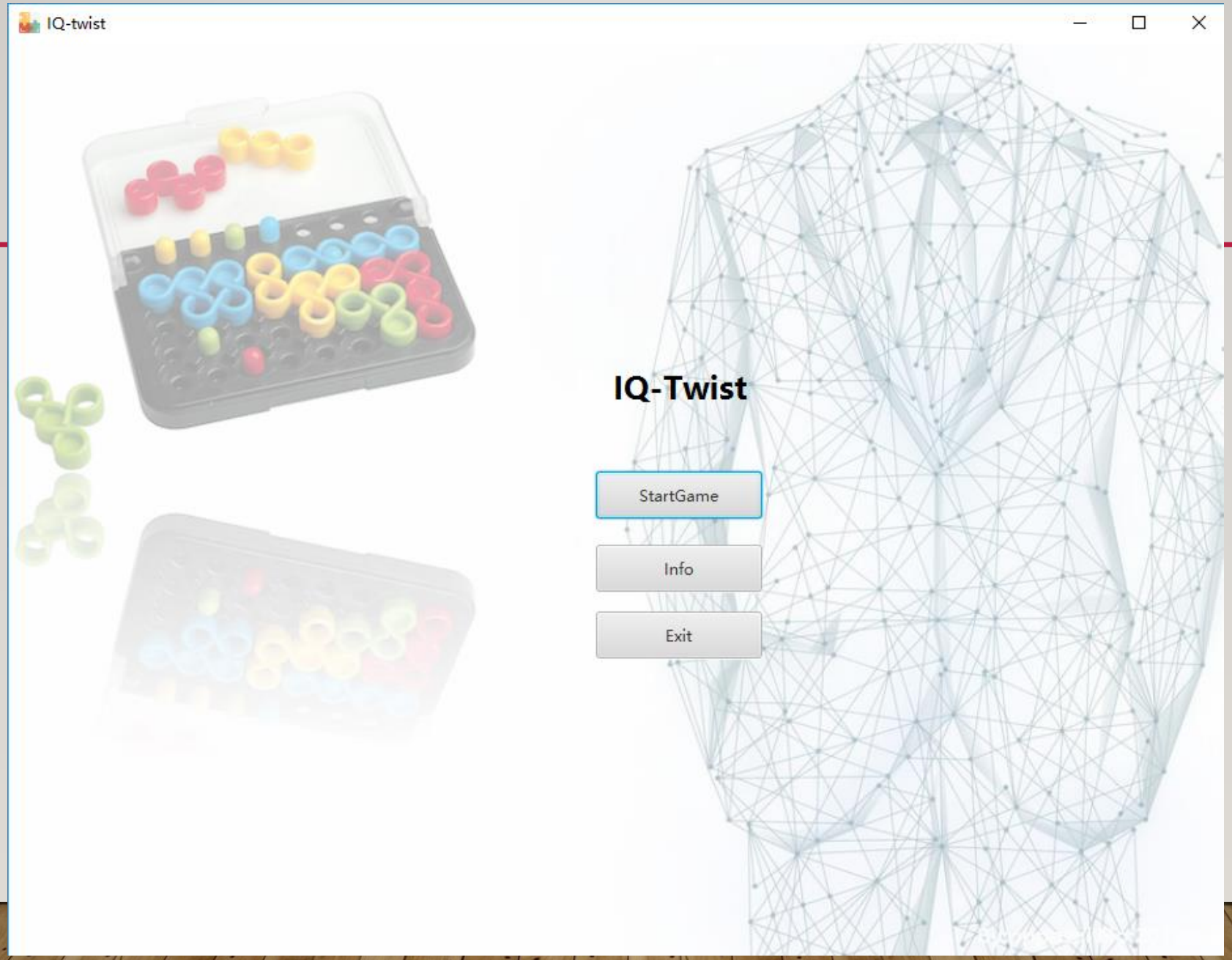
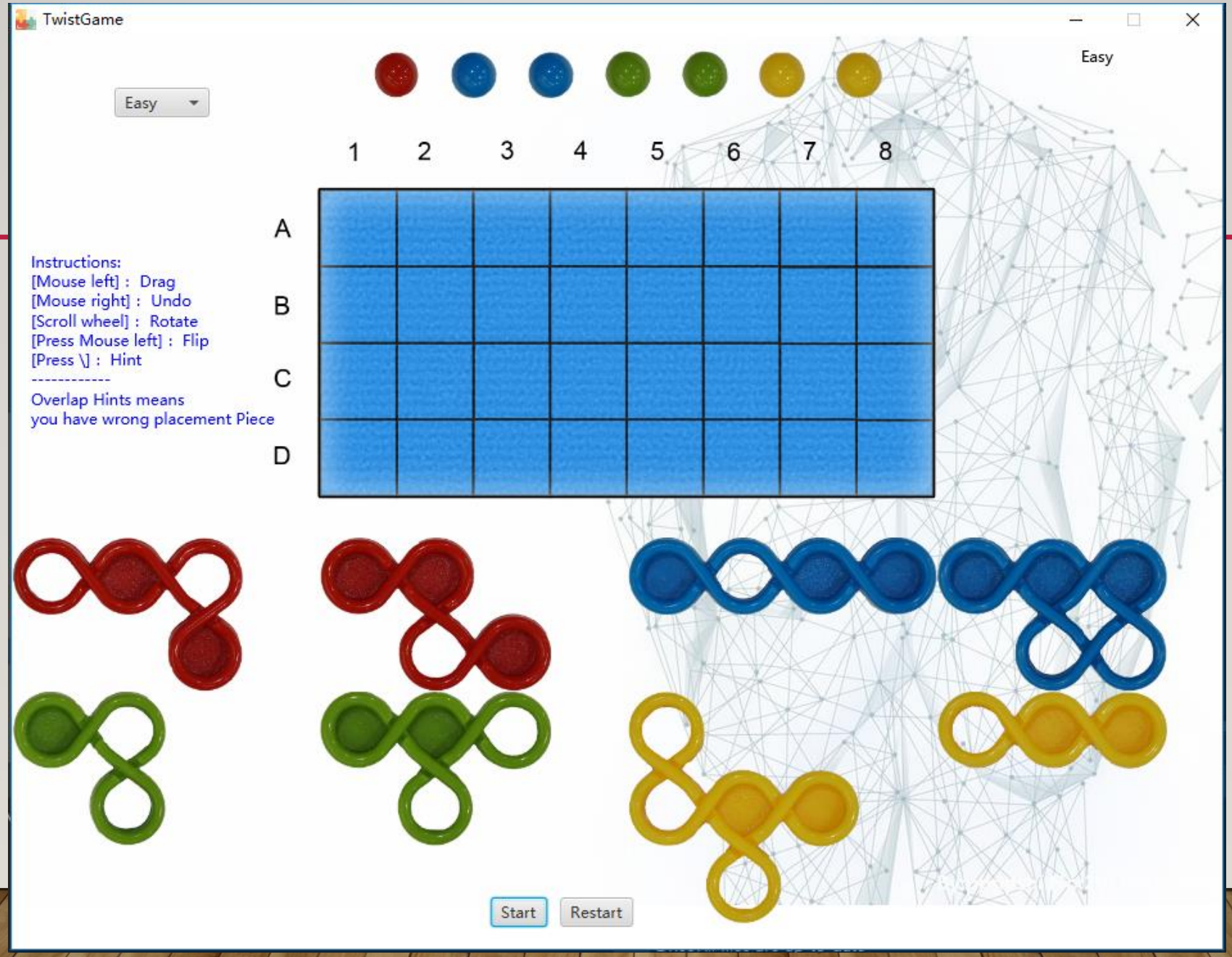
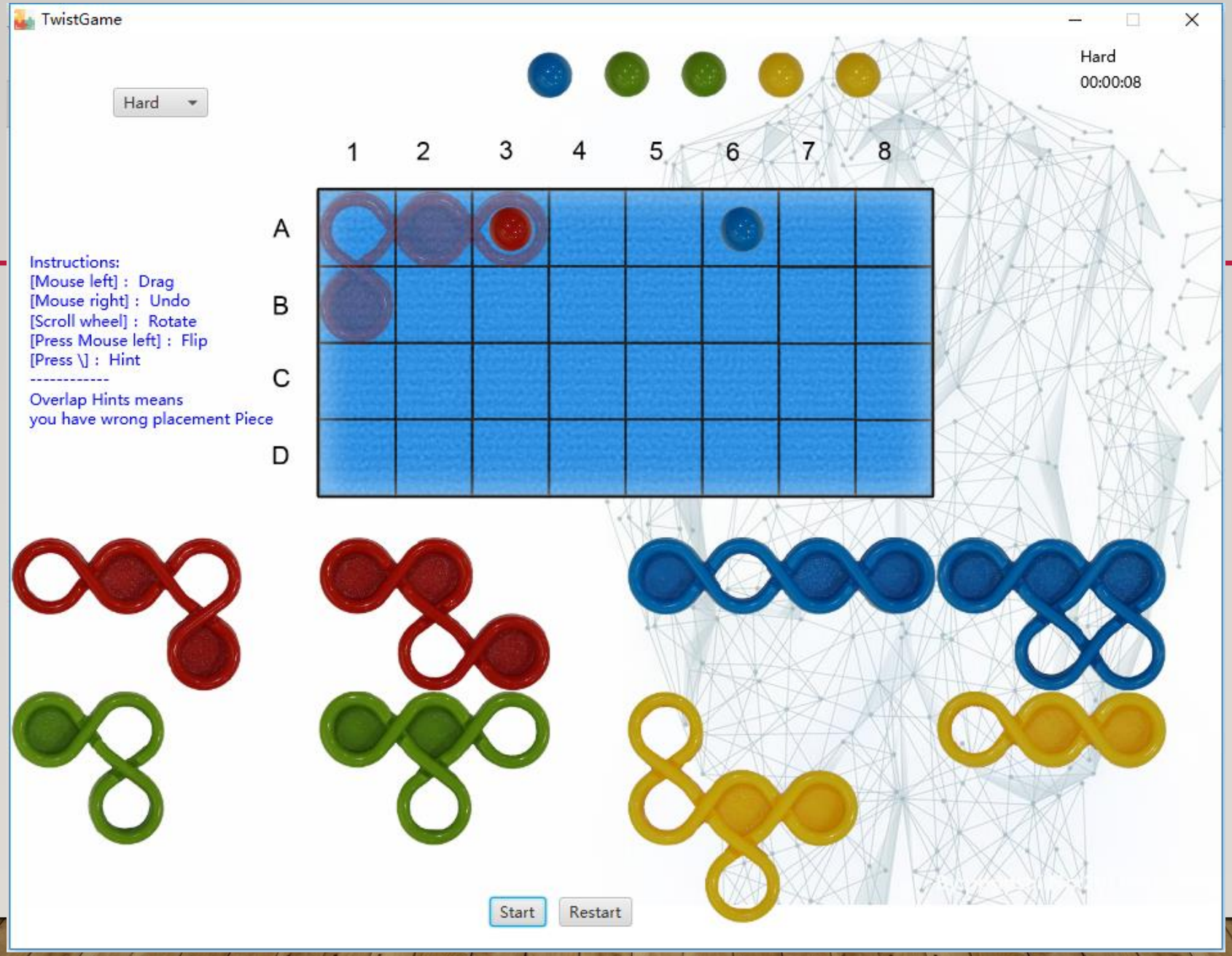
# COMP1110 ASSIGNMENT2


GROUP MEMBERS:

| NAMES: | STUDENT ID: |
| --- | --- |
| XIANG LI | U6716878 |
| NING CAI | U6456964 |
| ZHI WANG | U6171870 |


IQ-twist
IQ-Twist
StartGame
Info
Exit


TwistGame
Easy
Easy
1 2 3 4 5 6 7 8
A
B
C
D
Instructions:
[Mouse left] : Drag
[Mouse right] : Undo
[Scroll wheel] : Rotate
[Press Mouse left] : Flip
[Press \] : Hint
---
Overlap Hints means
you have wrong placement Piece
Start
Restart


TwistGame
Hard
Hard
00:00:08
1 2 3 4 5 6 7 8
A
B
C
D
Instructions:
[Mouse left] : Drag
[Mouse right] : Undo
[Scroll wheel] : Rotate
[Press Mouse left] : Flip
[Press \] : Hint
---
Overlap Hints means
you have wrong placement Piece
Start
Restart

# WHAT WE GOING TO TALK TODAY

INTRODUCE ALL COMPONENTS OF THE GAME

SHOW YOU HOW TO PLAY THE GAME

SOLUTION ALGORITHM

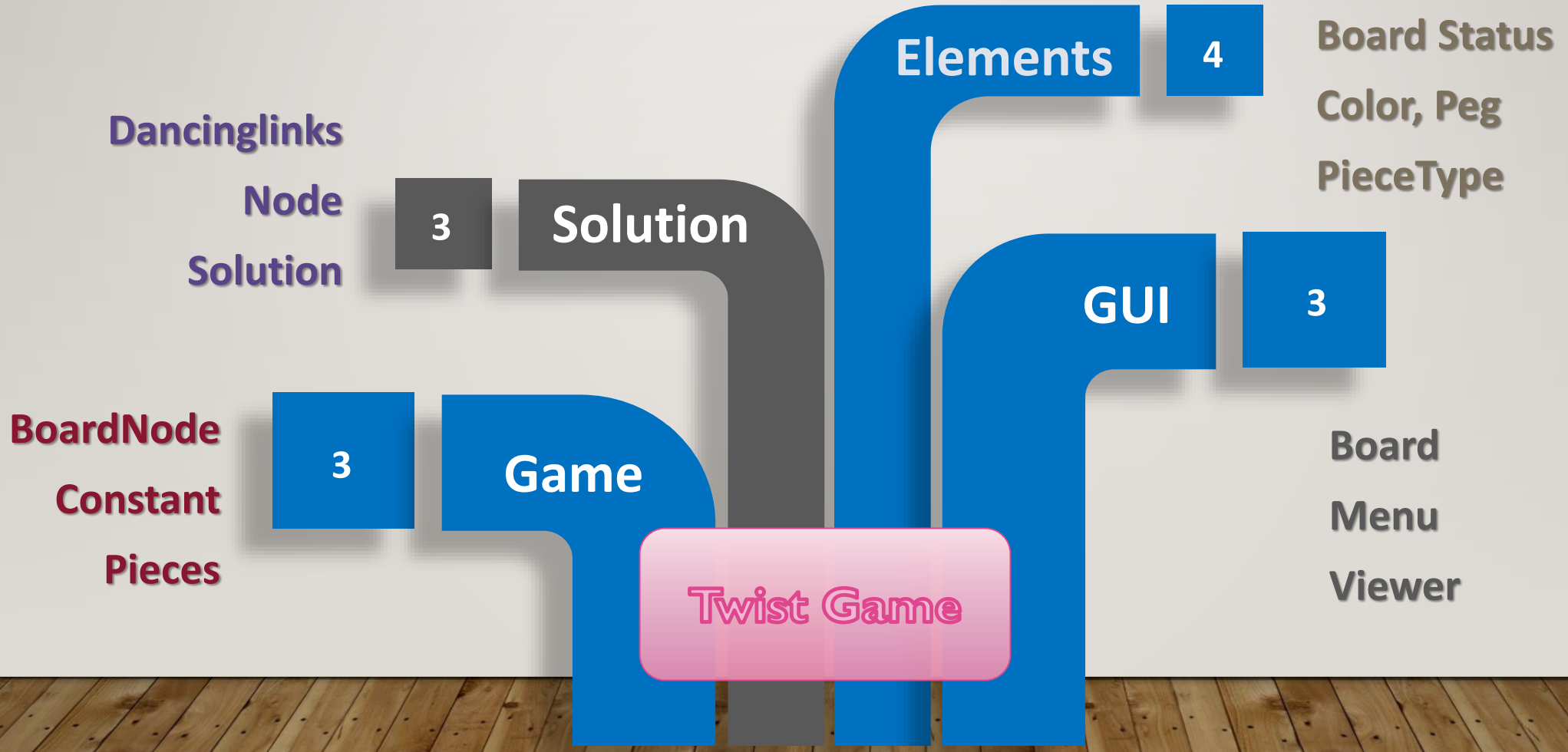
OUR DESIGN PROCESS

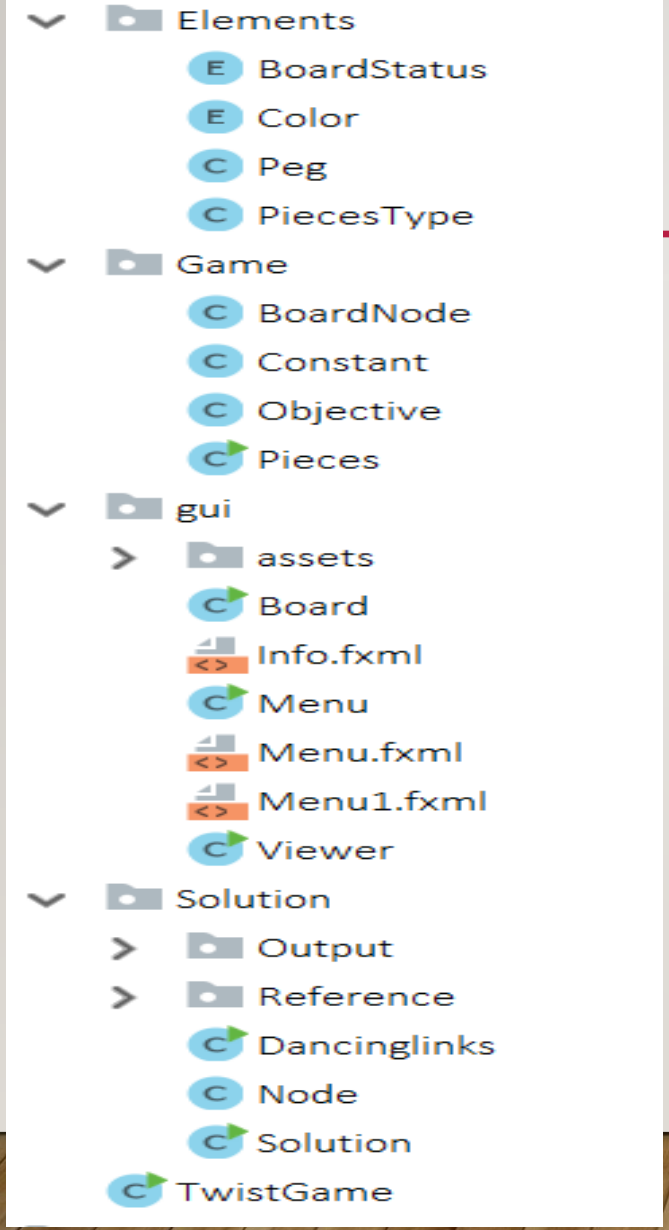
01

PART

# INTRODUCE ALL COMPONENTS OF THE GAME

# BASIC STRUCTURE






# Packages and classes

Elements: This package creates every type of the components and property of this game, which are waiting to be used in the main parts.

Solution: This package tries to find all the possible solution of the game and give a solver of the game (support task 9 & task 11)

Game: This package decodes components into codes we need.

GUI: This package claims the visual windows of the game and achieves the play methods we expected.

Twist Game : This class includes the main logic of the game.

# INSTRUCTIONS TO PLAY THE GAME

- How to move the pieces:

  Just hold the left click and move around the pieces

- How to rotate pieces:

  Just roll the mouse wheel

- How to flip pieces:

  Just click the mouse wheel

- When you make the wrong choices:

  It will be a warning sounds like“Ha-ow~”

- How to select the challenge levels:

  Just in put the number you prefer from 0 to 100

- How to find the hint of next step:

  Just press “\” to get help

# Elements

Color: This class determined the 4 color of the pegs and pieces which would be **Red**, **Green**, **Blue**, **Yellow**.

Pegs: This class determined all 7 pegs with 4 color, which will played at the beginning of the game.

PiecesType: This class determined pieces in all shapes and how they can be controlled by rotating or flipping.

## Pieces

This class will create all the 64 pieces in the list with all the properties of them, which will be used in the process of the game.

## BoardNode

This class represents a map we need in 8 ×4 node, and it will apply the node with **1**(Full), **-1**(Hole), **0**(Empty).

Board + Viewer

This two classes construct the game interface, and makes the game operations by using the mouse to move, rotate or flip the pieces, and at the same time absorb the pieces which can be attached to the proper positions when pieces are nearby.

# Menu

This class created a new window which show the menu of the game, there are three options:

**Start** (Which will bring you to the main window of the game, and start your journey with our IQ puzzle game)

**Information** (Which introduces our team to you)

**Exit** (When you don't want to play our game anymore, then click it)

# TwistGame

- In TwistGame class, after decoding, we first check whether a piece or peg placement and placement are well-formed. Then try to check whether a placement string is valid. After that, given a particular starting placement, we then check all solutions to the game.

02

PART

# HOW TO PLAY THE GAME

# SHOW TIME!

We will show you how you can play our games!

http://www.nipic.com/show/18504023.html

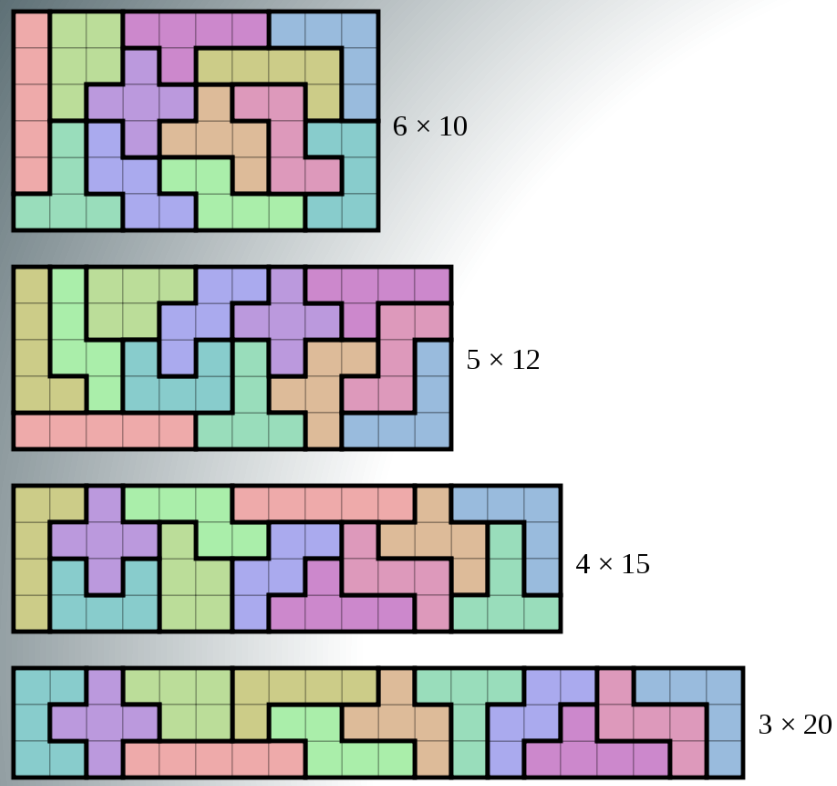
03

PART

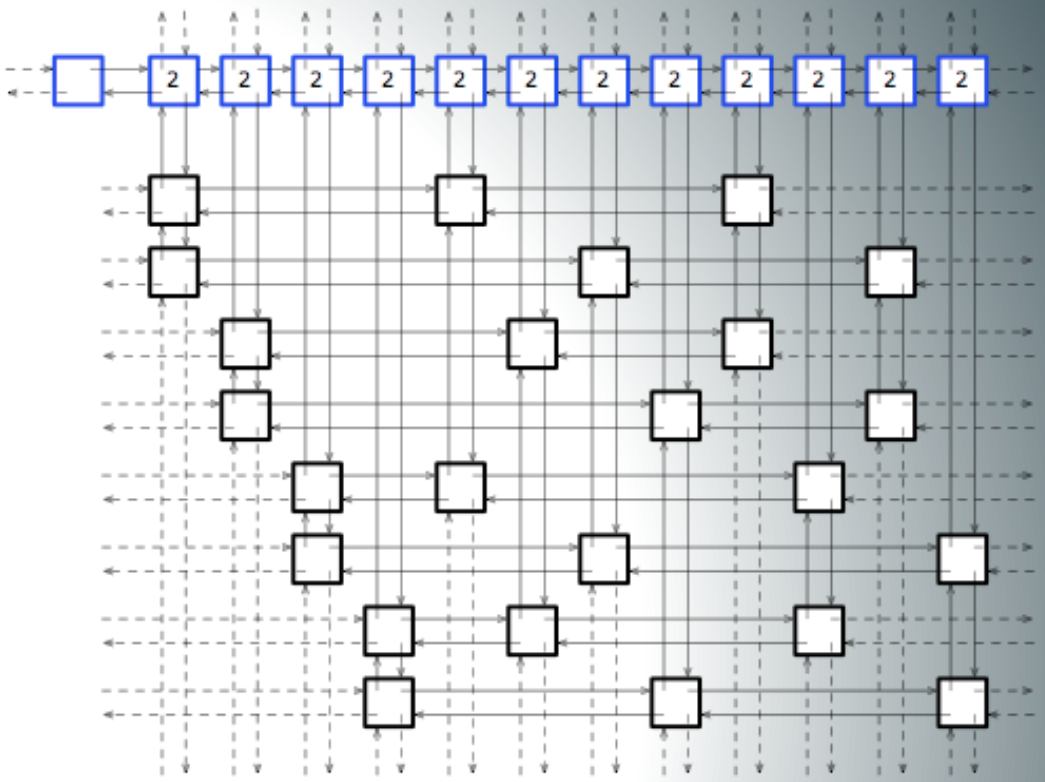
# Solution

# Implement Algorithm X

# (Dancing Links)




A pentomino tiling problem(Exact cover)

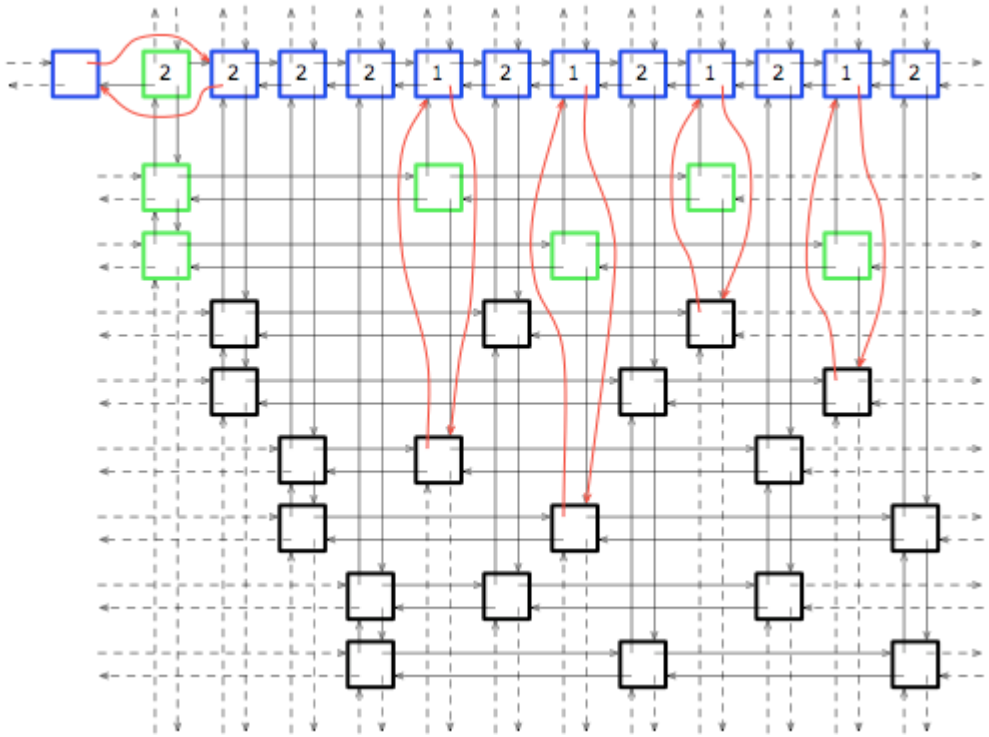



Four-way-linked List

# Implement Algorithm X (Dancing Links)

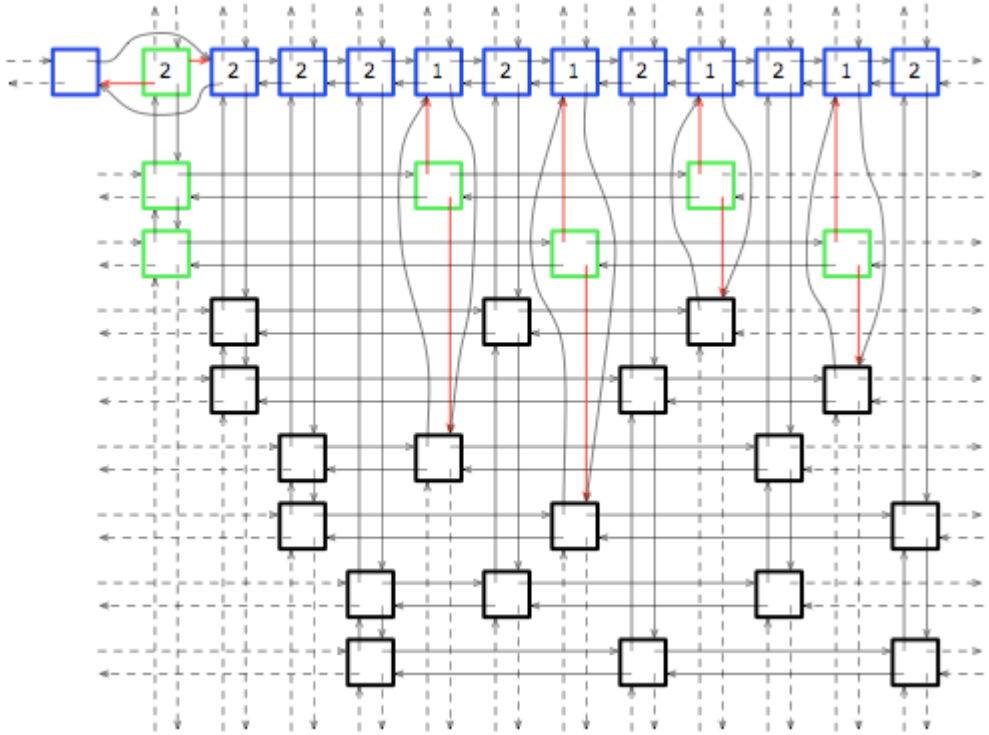
Retrieved From https://en.wikipedia.org/wiki/Exact_cover

# Implement Algorithm X (Dancing Links)




## Remove

```
private void remove(Node removeNode) {
    removeNode.left.right = removeNode.right;
    removeNode.right.left = removeNode.left;
    for (Node i = removeNode.down; i != removeNode; i = i.down) {
        for (Node j = i.right; j != i; j = j.right) {
            j.up.down = j.down;
            j.down.up = j.up;
            j.header.size--;
        }
    }
}
```

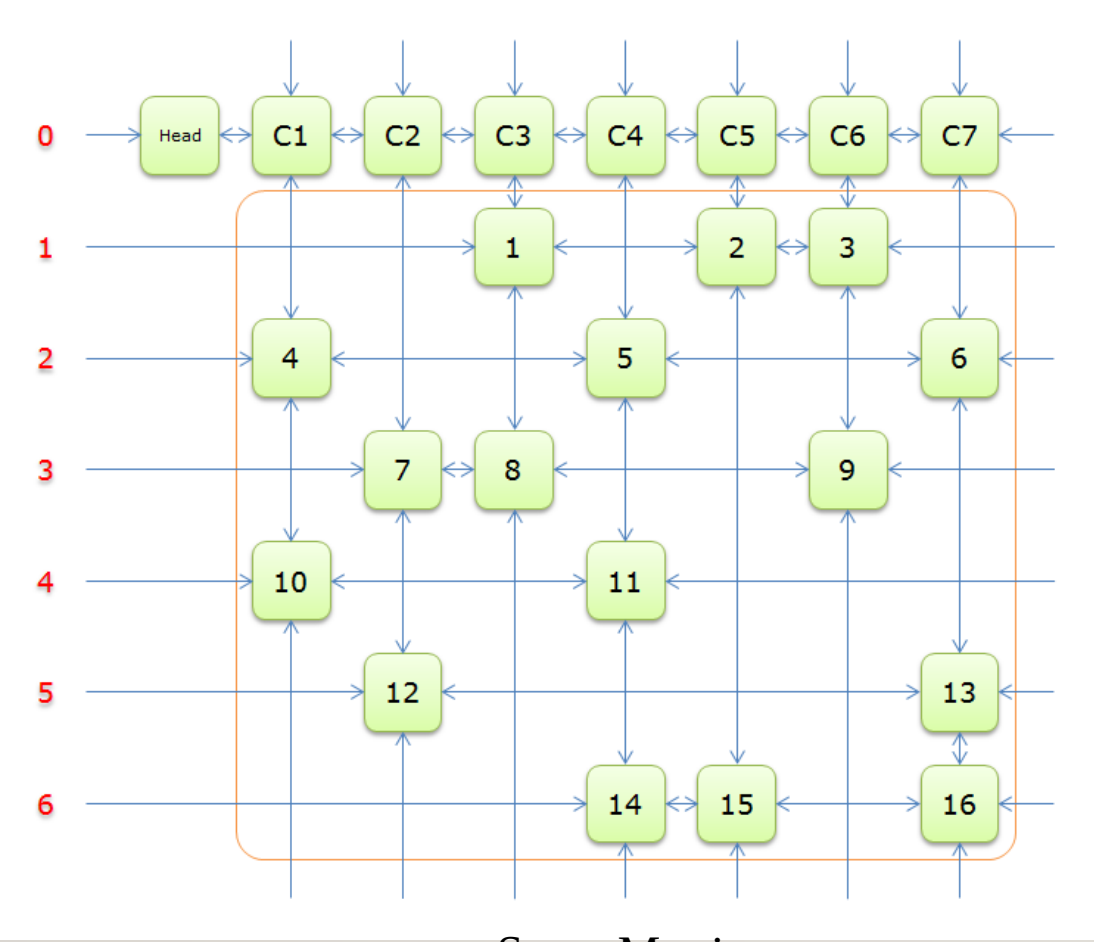



## Reverse

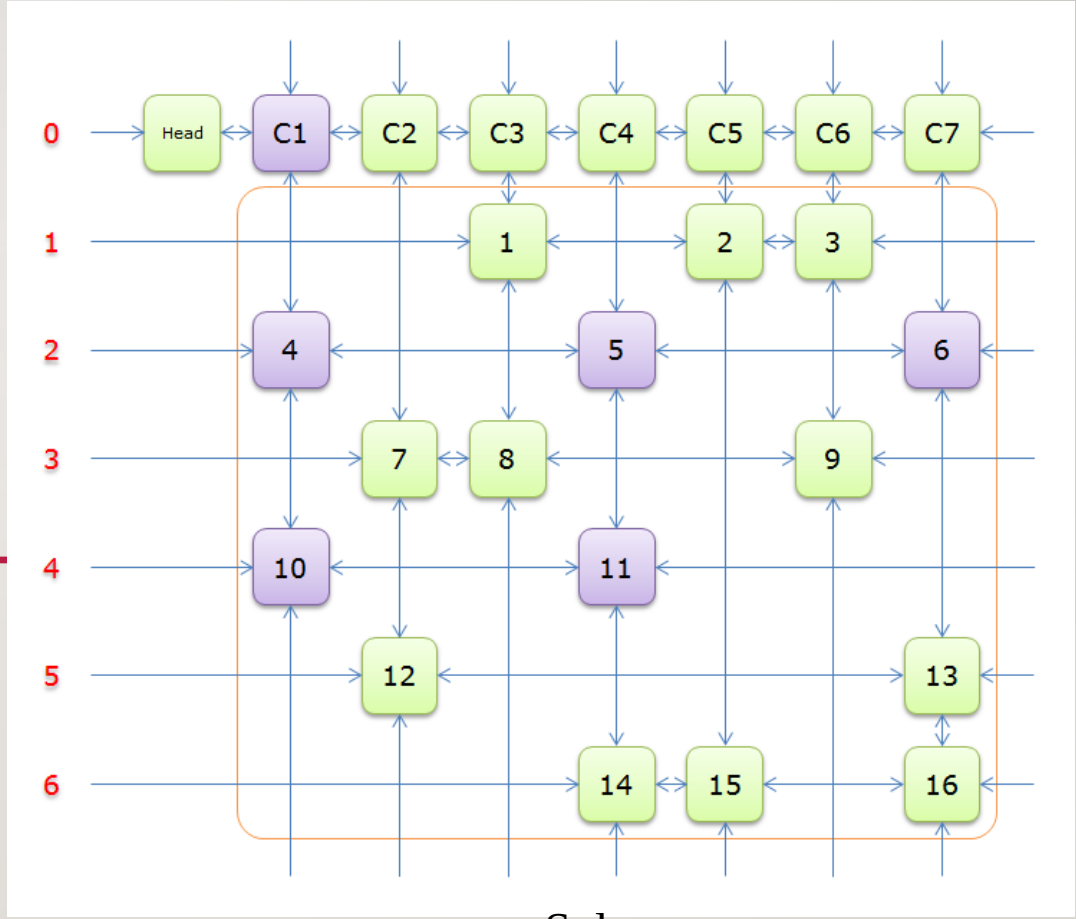
```
private void reverse(Node reverseNode) {
    reverseNode.left.right = reverseNode;
    reverseNode.right.left = reverseNode;
    for (Node i = reverseNode.up; i != reverseNode; i = i.up) {
        for (Node j = i.left; j != i; j = j.left) {
            j.up.down = j;
            j.down.up = j;
            j.header.size++;
        }
    }
}
```

Retrieved From http://garethrees.org/2007/06/10/zendoku-generation

# Implement Algorithm X (Dancing Links)




Spare Matrix




Solve

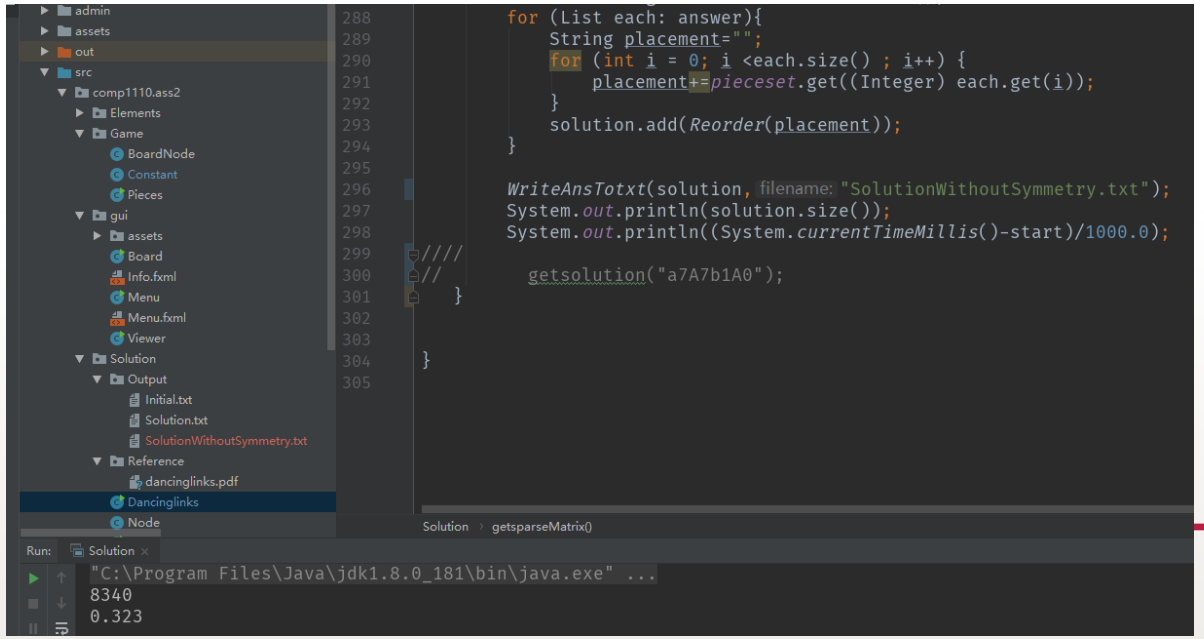
Retrieved From https://www.cnblogs.com/grenet/p/3163550.html

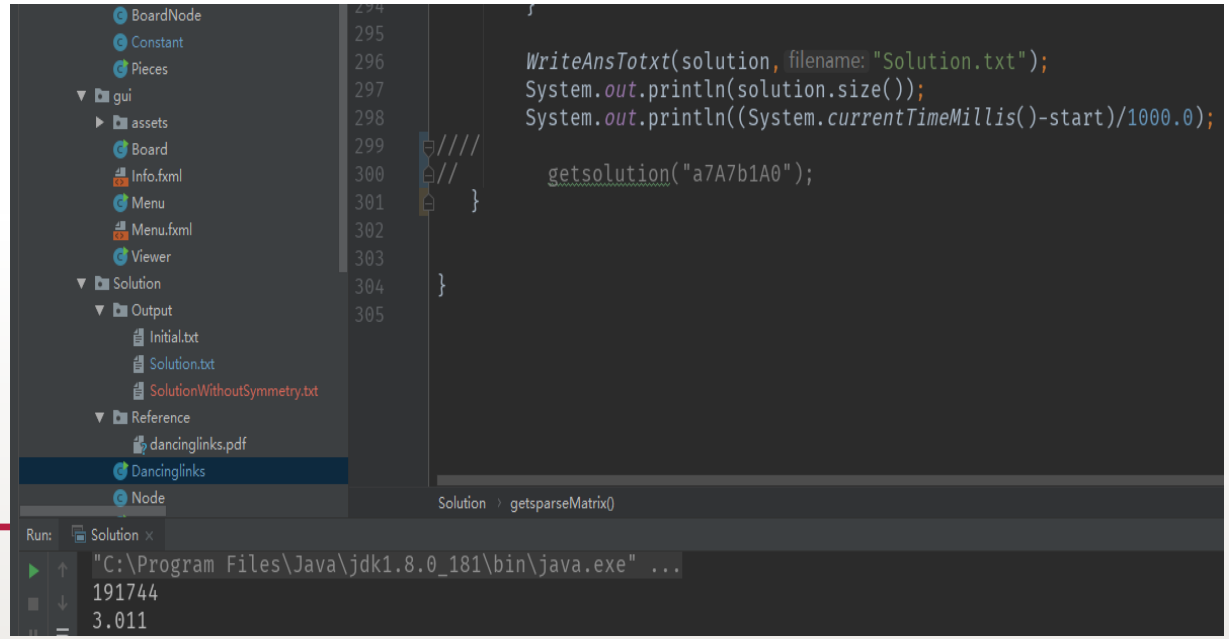
Encoding

| | Position Cconstrain | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Piece Cconstrain | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| a1A0 | 1 | 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 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a2A0 | 0 | 1 | 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 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a3A0 | 0 | 0 | 1 | 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 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| b1A0 | 1 | 1 | 0 | 0 | 0 | 0 | 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 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| b2A0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 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 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| b3A0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 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 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Matrix Size：912*40

# Solution




Without Symmetry： 8340(0.323 S)

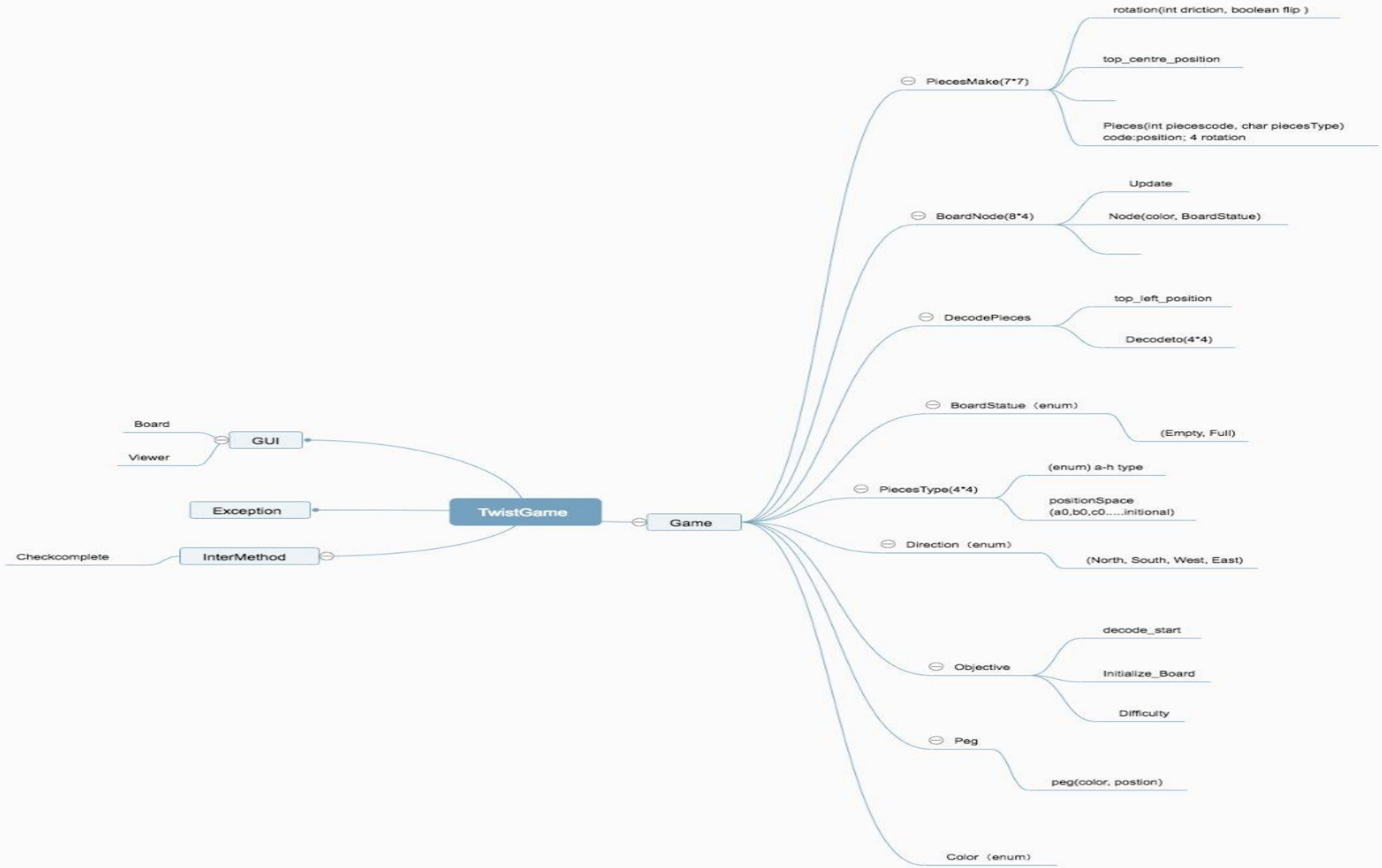
Without Strict Symmetry： 191744(3.011 S)

Not sure if program has traversed all possibilities.

04

PART

# OUR DESIGN PROCESS


TwistGame
GUI
Board
Viewer
Exception
InterMethod
Checkcomplete
Game
PiecesMake(7*7)
rotation(int driction, boolean flip )
top_centre_position
Pieces(int piecescode, char piecesType)
code:position; 4 rotation
BoardNode(8*4)
Update
Node(color, BoardStatue)
DecodePieces
top_left_position
Decodeto(4*4)
BoardStatue（enum）
(Empty, Full)
PiecesType(4*4)
(enum) a-h type
positionSpace
(a0,b0,c0.....initional)
Direction（enum）
(North, South, West, East)
Objective
decode_start
Initialize_Board
Difficulty
Peg
peg(color, postion)
Color（enum）

# THANK YOU FOR LISTENING!


http://www.quanjing.com/imgbuy/ul0945-5360.html

**Welcome**

**any**

**questions!**