Artificial Intelligence (CS303)
Lab Courses

Lab 1: Project 1-Gomoku

Outline of this Lab

- Project Description

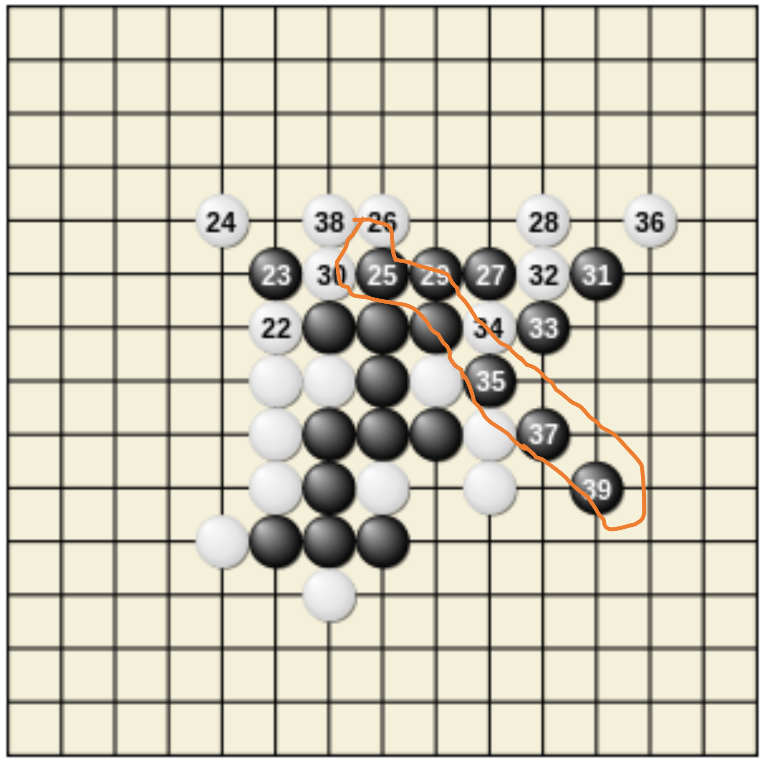
- Basic Search Algorithm for Gomoku

Project Description

Description

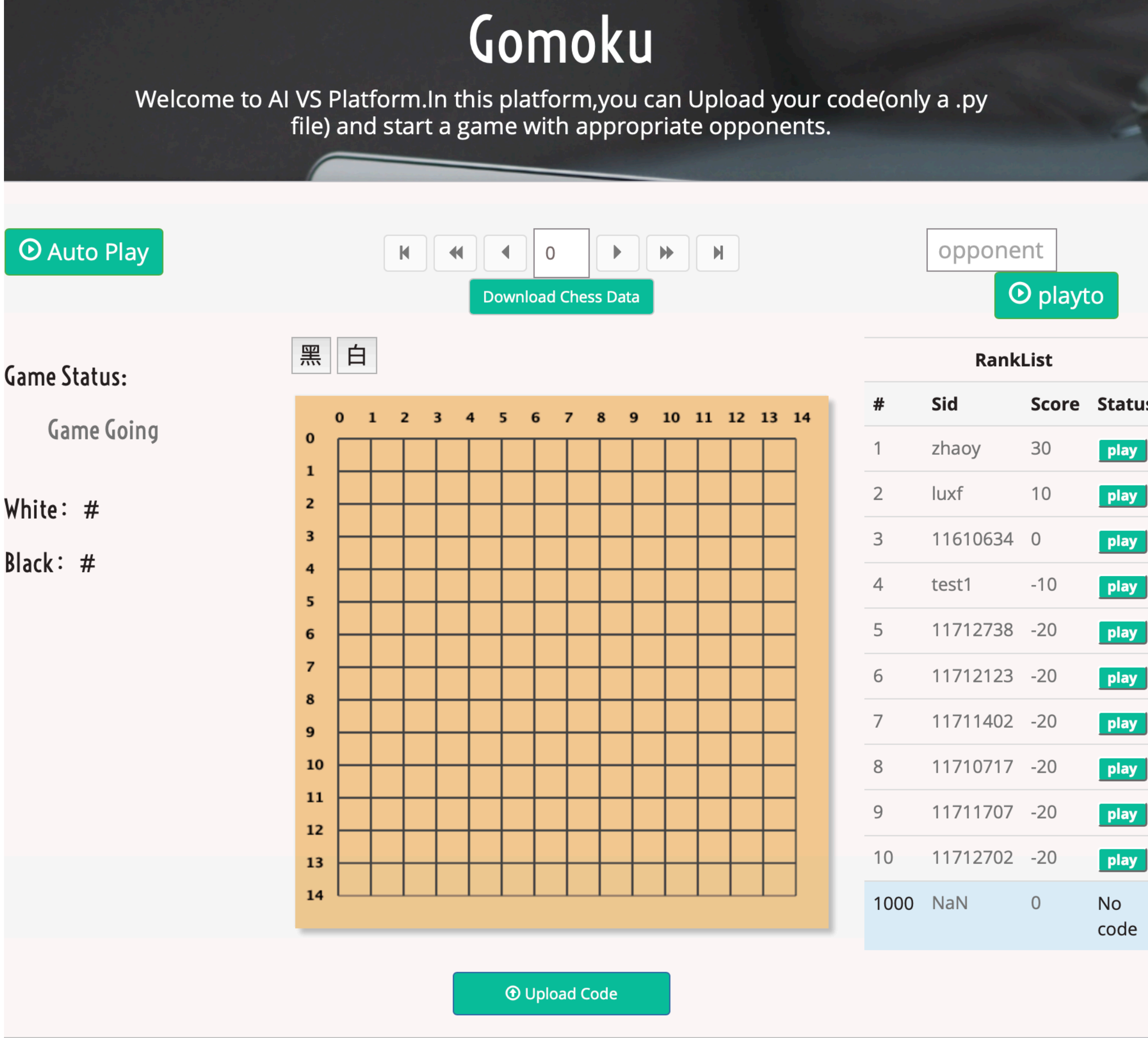
- Gomoku is a relatively simple board game. Its instrument is universal with Go, and it originated from one of the ancient Chinese black and white chess. Usually, the two players place black and white chess pieces at the intersection of the straight line and the horizontal line of the board in turns. The winner is the first player to form an unbroken line of five chess pieces horizontally, vertically, or diagonally.

- We use the default board of size 15*15 board. Students need to implement the AI algorithm of Gomoku according to the interface requirements and submit it to the platform as required.



The Gomoku Battle Platform

http://10.20.26.45:8080/login



Gomoku

Welcome to AI VS Platform.In this platform,you can Upload your code(only a .py file) and start a game with appropriate opponents.

Game Status:

Game Going

White： #

Black： #

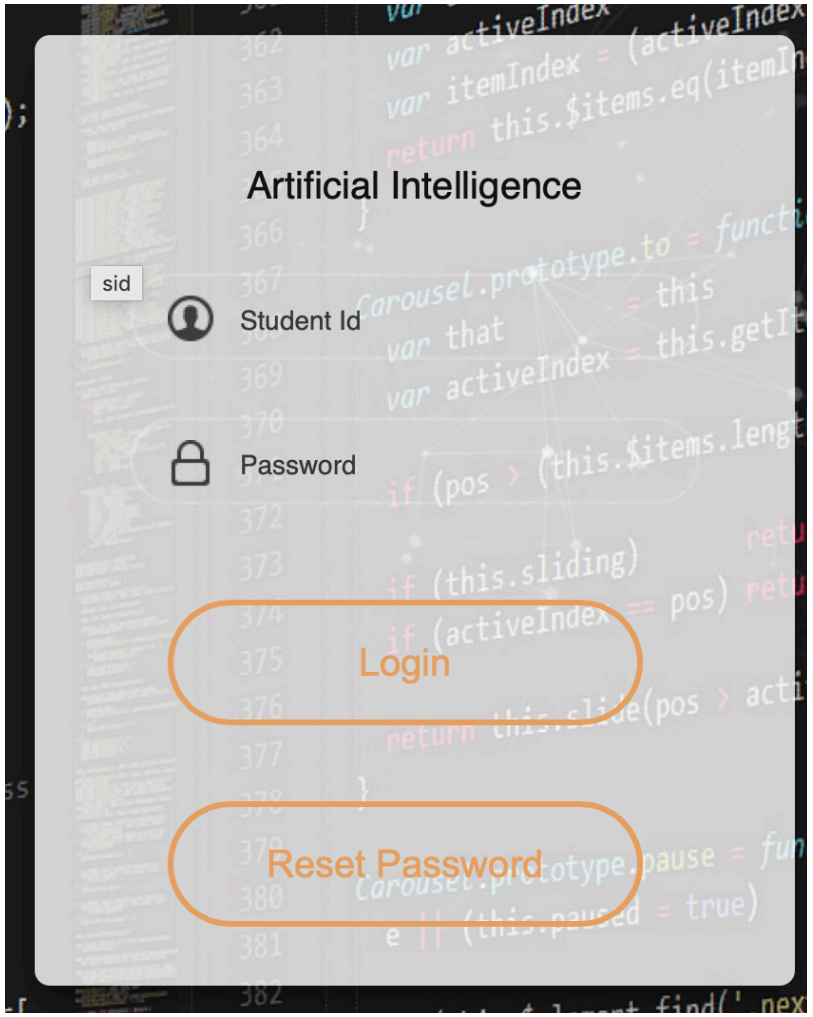
RankList

#	Sid	Score	Statu
1	zhaoy	30	play
2	luxf	10	play
3	11610634	0	play
4	test1	-10	play
5	11712738	-20	play
6	11712123	-20	play
7	11711402	-20	play
8	11710717	-20	play
9	11711707	-20	play
10	11712702	-20	play
1000	NaN	0	No code

The Gomoku Battle Platform

http://10.20.26.45:8080/login

The Gomoku platform is logged in with the student ID and password. The default password is student ID. At the first login, the system will remind you to change password. After logging in successfully, you can see the UI as the picture before shows.



Evaluation Rule

- Three Phases
  - Lab 1: Usability Testing (可行性测试)
  - Lab 2: Points Race（积分赛）
  - Lab 3: Round Robin （循环赛）

	Evaluation Rule	DeadLine
Lab1	Score according to the number of test cases passed	15th September
Lab2	Score according to the ranking in the points race	22nd September
Lab3	Score according to the ranking in the round robin	29th September

Phase 1: Usability Testing

- **Use some simple board test cases where students need to find the best place to drop.**

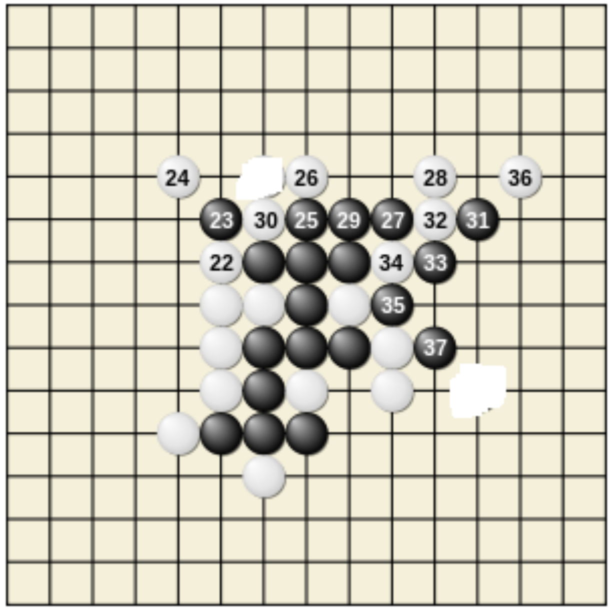
- **Only jobs that passed the usability test can pass this phase.**

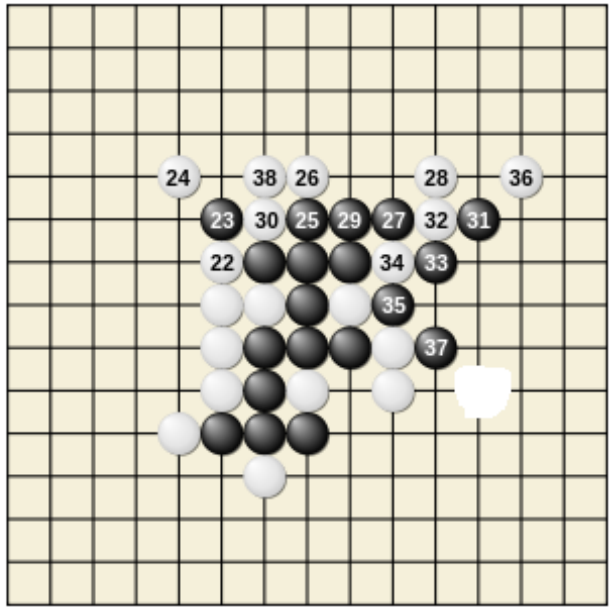
- **The goal of usability testing:**
  - No illegal operation, infinite loop or random drop
  - The OS package cannot be imported in the code file
  - The empty board test make students deal with first hand
  - There are other tests for situations that will finish the game such as five-connected, four connected, double three, flush four live three (actually to help students to have the basic intelligent judgment before entering PK).

- **A total of 25 test cases were prepared, each with 4 points, and the scores were determined by the number of test cases you passed.**

An Example

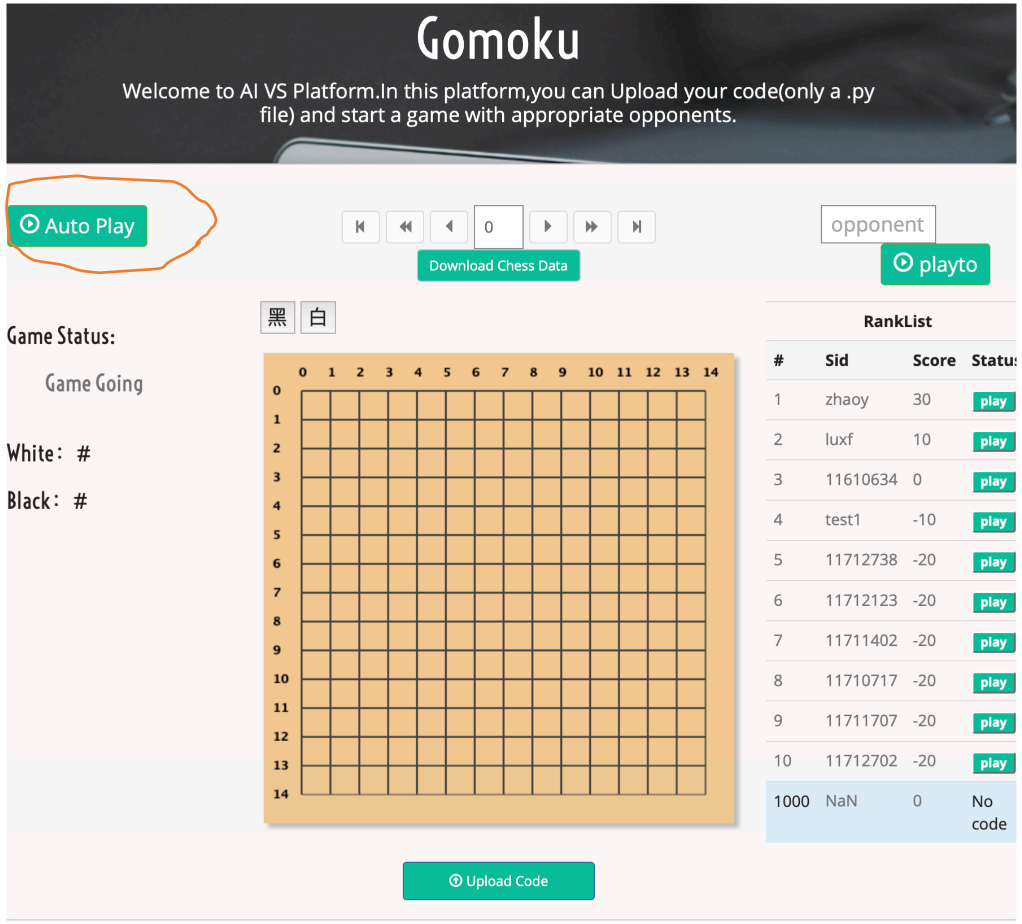
- Your search algorithm should be reasonable. For example, if you have four chess pieces horizontally, vertically, or diagonally, then your search algorithm should at least find the position with with you will win.





Phase 2: Points Race

- After submitting successfully, click on "Auto Play" to challenge the player who ranks ahead of you for a PK.
- To avoid the first-hand advantage, each PK is played for 2 rounds with two players starting the game first in turns.
- Scoring rules:
  - If one wins, then his/her score is increased by 5.
  - If one loses, then his/her score is deducted by 5.
  - If it is a draw, then the scores remain unchanged.



Gomoku

Welcome to AI VS Platform.In this platform,you can Upload your code(only a .py file) and start a game with appropriate opponents.

Game Status:

Game Going

White: #

Black: #

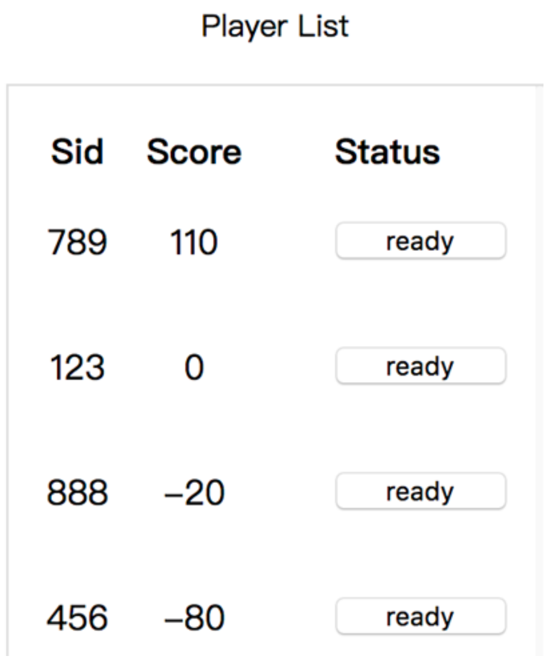
RankList

#	Sid	Score	Statu
1	zhaoy	30	play
2	luxf	10	play
3	11610634	0	play
4	test1	-10	play
5	11712738	-20	play
6	11712123	-20	play
7	11711402	-20	play
8	11710717	-20	play
9	11711707	-20	play
10	11712702	-20	play
1000	NaN	0	No code

Phase 2: Points Race

- Algorithm can be updated and submitted to the Gomoku battle platform before Lab 2 DDL.

- After Lab 2 DDL, the scores are given according to the ranking.


Player List

Sid	Score	Status
789	110	ready
123	0	ready
888	–20	ready
456	–80	ready

Phase 3: Round Robin

- After the Lab2 DDL, the round robin is started.
- Every student will play a PK with each other student in the class.
- The ranking for each student is given according to the accumulated score.

Evaluation Rule

- After Lab3, a carefully written report need to be submitted.

Coding_Score = （Lab1+Lab2+Lab3） /3

Project1_Score = Coding_Score *0.7+ Report_Score *0.3

Code Requirement

Python version 3.6

Code Template

```
import numpy as np
import random
import time

COLOR_BLACK=-1
COLOR_WHITE=1
COLOR_NONE=0
random.seed(0)
```

Code Requirement

```
#don't change the class name
class AI(object):
    #chessboard_size, color, time_out passed from agent
    def __init__(self, chessboard_size, color, time_out):
        self.chessboard_size = chessboard_size
        #You are white or black
        self.color = color
        #the max time you should use, your algorithm's run time must not exceed the time limit.
        self.time_out = time_out
        # You need add your decision into your candidate_list. System will get the end of your candidate_list as your decision .
        self.candidate_list = []
```

```
# The input is current chessboard.
    def go(self, chessboard):
        # Clear candidate_list
        self.candidate_list.clear()
        
#======================================================================
        #Write your algorithm here
        #Here is the simplest sample:Random decision
        idx = np.where(chessboard == COLOR_NONE)
        idx = list(zip(idx[0], idx[1]))
        pos_idx = random.randint(0, len(idx)-1)
        new_pos = idx[pos_idx]
        #===============Find new pos===========================================
        # Make sure that the position of your decision in chess board is empty.
        #If not, return error.
        assert chessboard[new_pos[0],new_pos[1]]== COLOR_NONE
        #Add your decision into candidate_list, Records the chess board
        self.candidate_list.append(new_pos)
```

Code Requirement

- The time to make a decision can not be longer than 5s

- Time Measurement

```
start = time.time()
... algorithm ...
run_time = (time.time() - start)
```

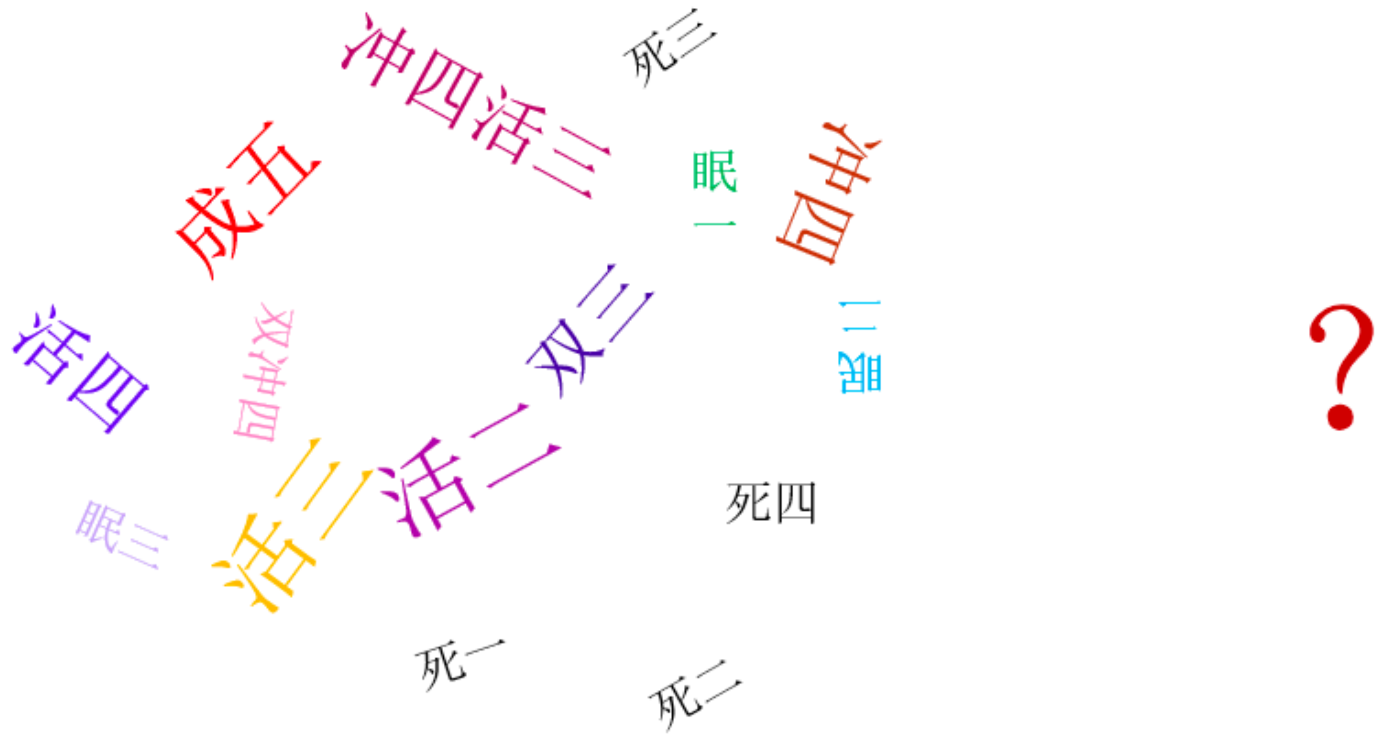
- The use of memory cannot go beyond 100M, the whole battle cannot take longer than 180s.

Basic Search Algorithm for Gomoku

Important Questions in Search

- How to evaluate an action?
- How to search efficiently?

Chess Type



Into Five (成五)

- Five-connected: There are five consecutive chess pieces of the same color. Once this type of chess appears, it means that the chess of this color wins.

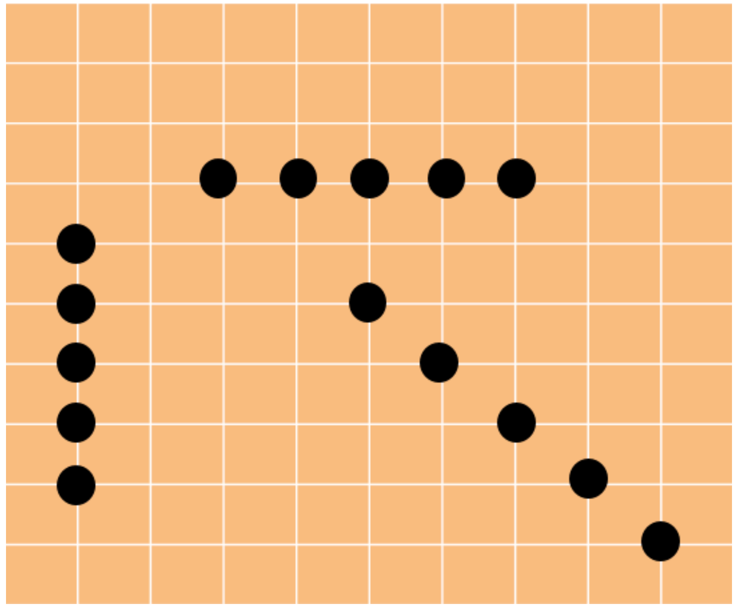
  五连：连续的五个同色棋子的棋型，一旦出现此种棋型, 意味着该色棋取胜。

- Long-connected: More than five consecutive chess pieces of the same color.

  长连：连续五个以上同色棋子的棋型。

- Into five: includes both five-connected and long-connected.

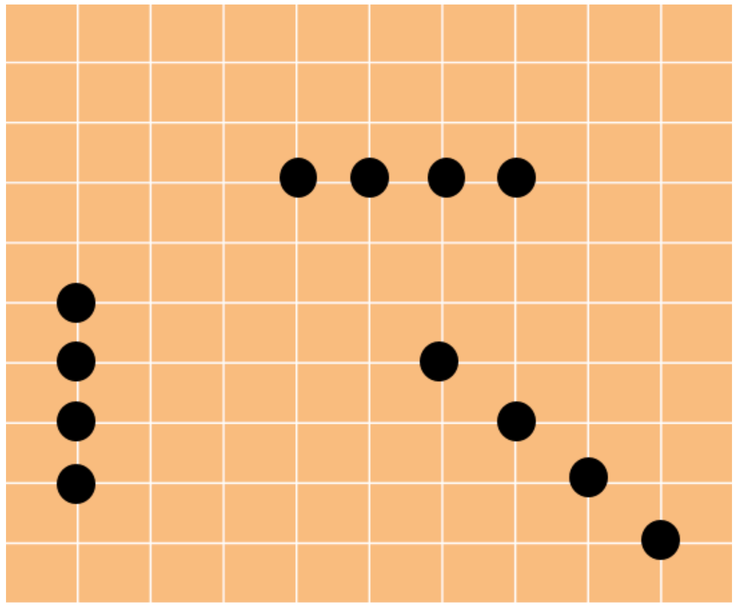
  成五：五连和长连的统称。



Live Four (活四)

- Live four: There are 4 consecutive pieces of the same color that are not blocked by the opponent’ s pieces at both ends. It means that there are two positions to drop the same color to get five pieces in a row/column/diagonal line. Once a certain color has such a type, it means the color will win.

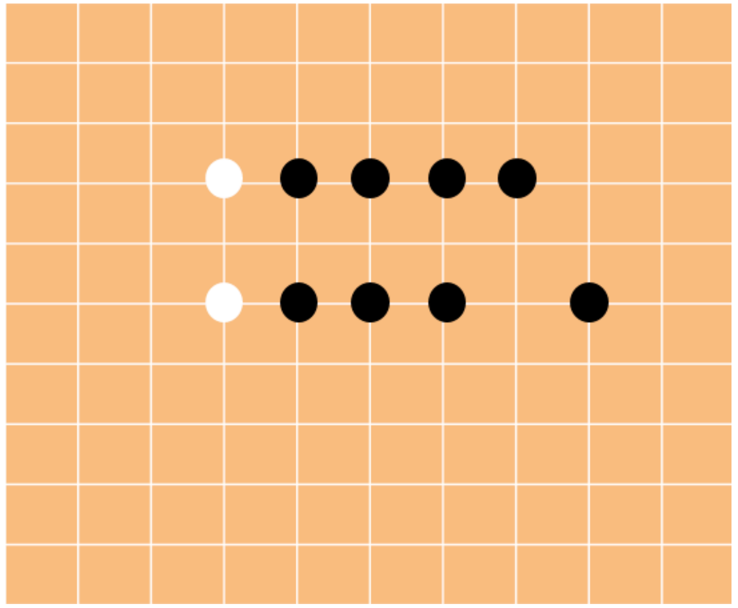
- 活四：两头没有对方棋子阻挡的连续4颗同色棋子，意味着有两个点落同样颜色子均能五连的的棋型，一旦某色棋出现此种棋型，也意味着该色棋必将取胜。



Flush Four (冲四)

- Flush four: Only one position can lead to five-connected type after a drop with the same color. Flush four is not necessarily connected. Sometimes you can use a space to get flush-four according to your needs.

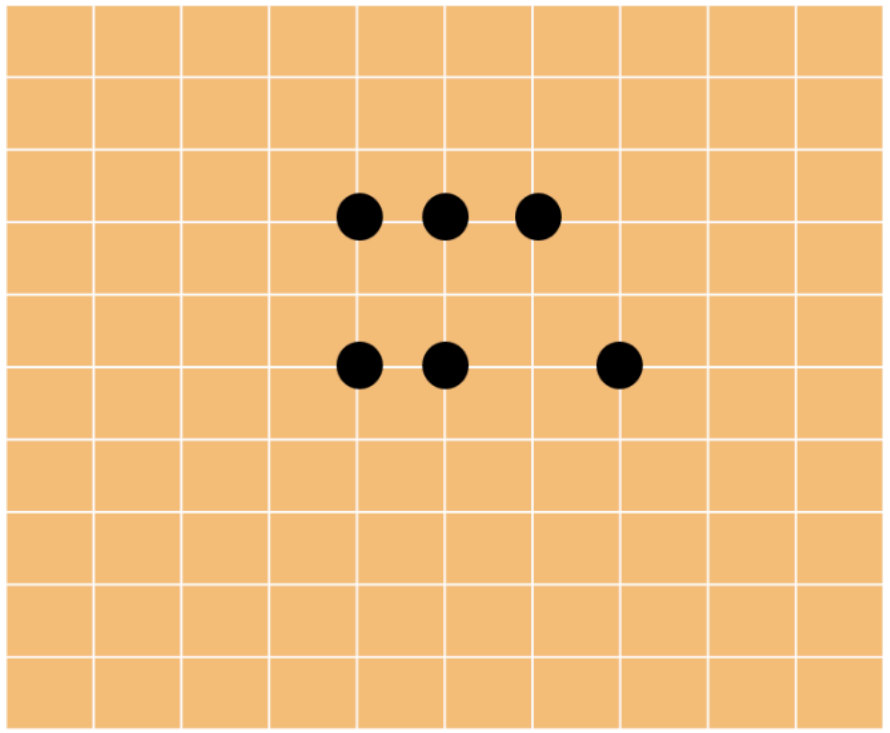
- 冲四：只有一个点落同样颜色子后能成五连的棋型。冲四不一定是挨着摆的，有时候可以根据需要隔上一个空格来冲四，称为跳冲四。



Live Three (活三)

- Live three: lead to live four after a drop, including connected-three and jump-three

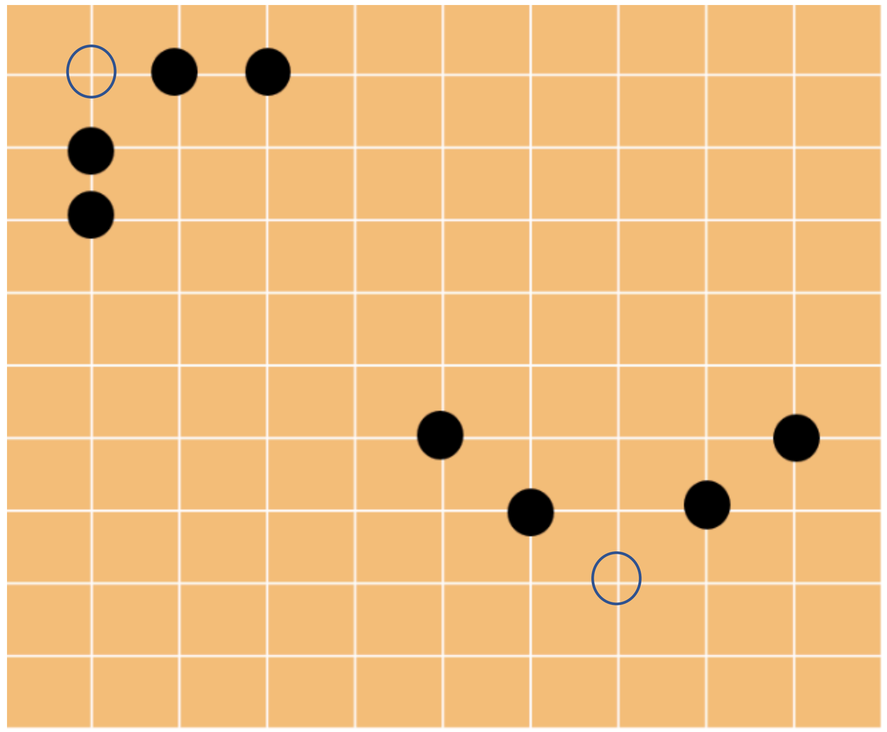
- 活三：有落子后能成活四的点的棋型，活三分为连三和跳三。



Double Three (双三)

- Double three: Lead to two live-three after a drop. It is a chess type that can win.

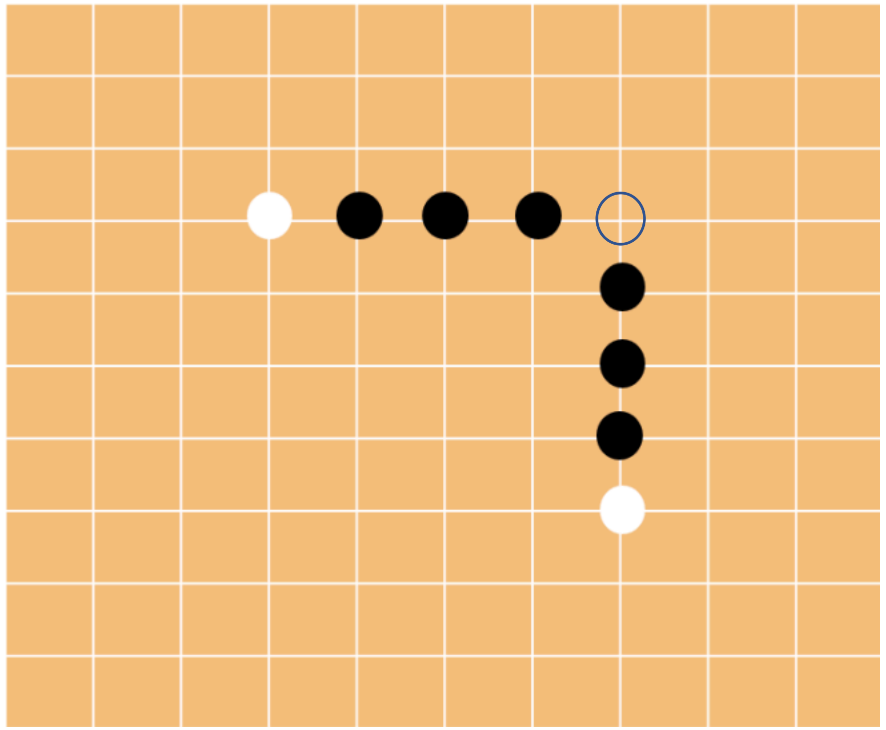
- 双三：落一颗子同时形成2个活三。是一种能必胜的棋型。



Double Flush Four (双冲四)

- Double flush four: Lead to two flush-four after a drop. It is also a winning type.

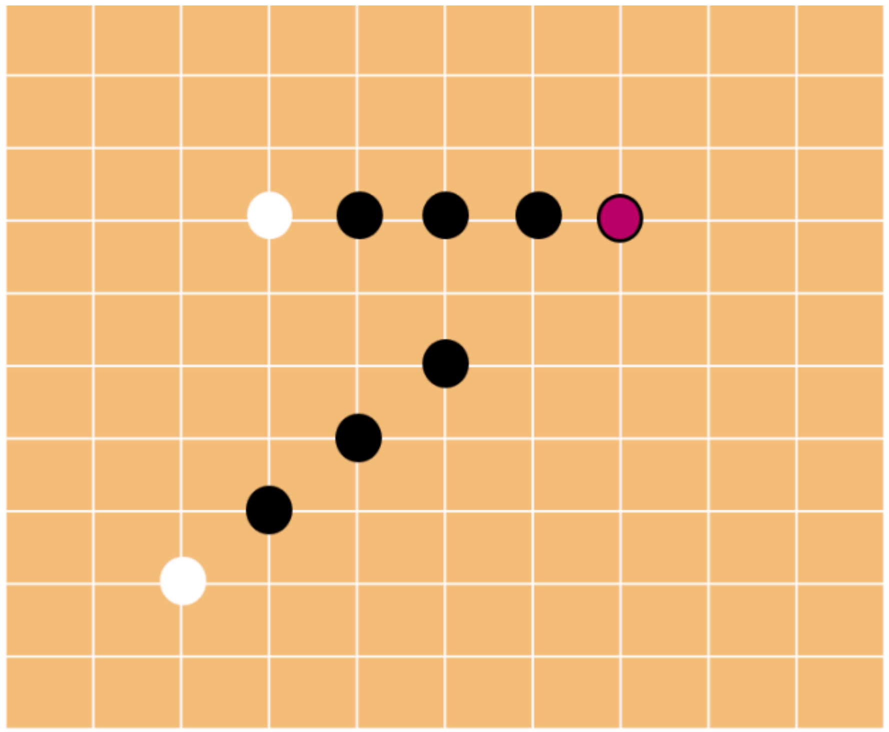
- 双冲四：落下一手棋同时形成两个冲四。也是一种必胜的棋型。



Double Flush Four (双冲四)

- Double flush four: Lead to two flush-four after a drop. It is also a winning type.

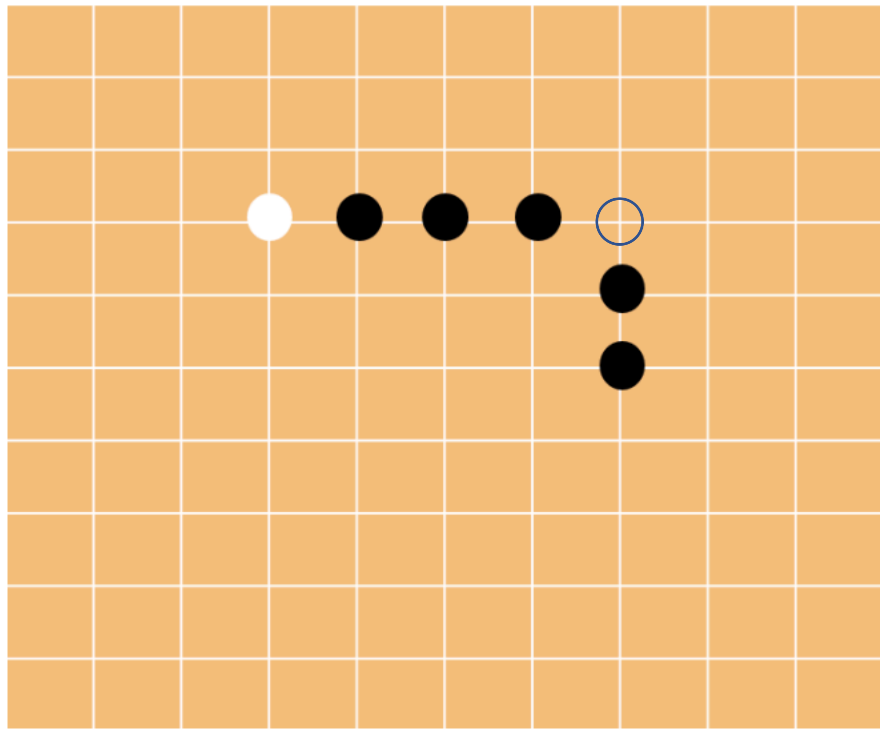
- 双冲四：落下一手棋同时形成两个冲四。也是一种必胜的棋型。



Flush Four Live Three (冲四活三)

- Flush four live three: Lead to a flush-four and live-three after a drop. It is also a winning type.

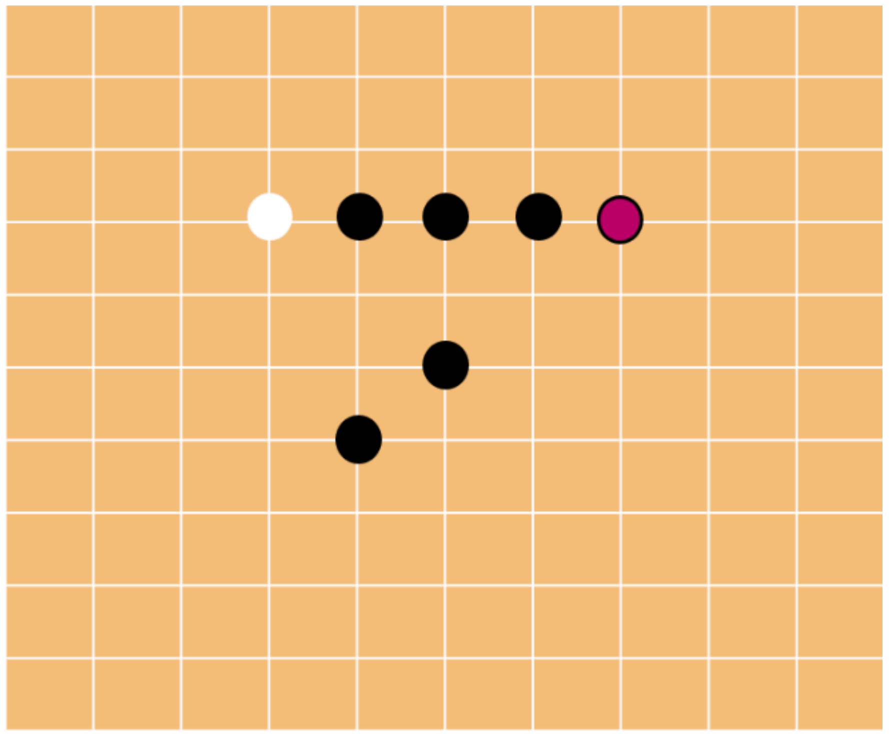
- 冲四活三：落下一手棋，同时形成冲四和活三。也是一种必胜的棋型。



Flush Four Live Three (冲四活三)

- Flush four live three: Lead to a flush-four and live-three after a drop. It is also a winning type.

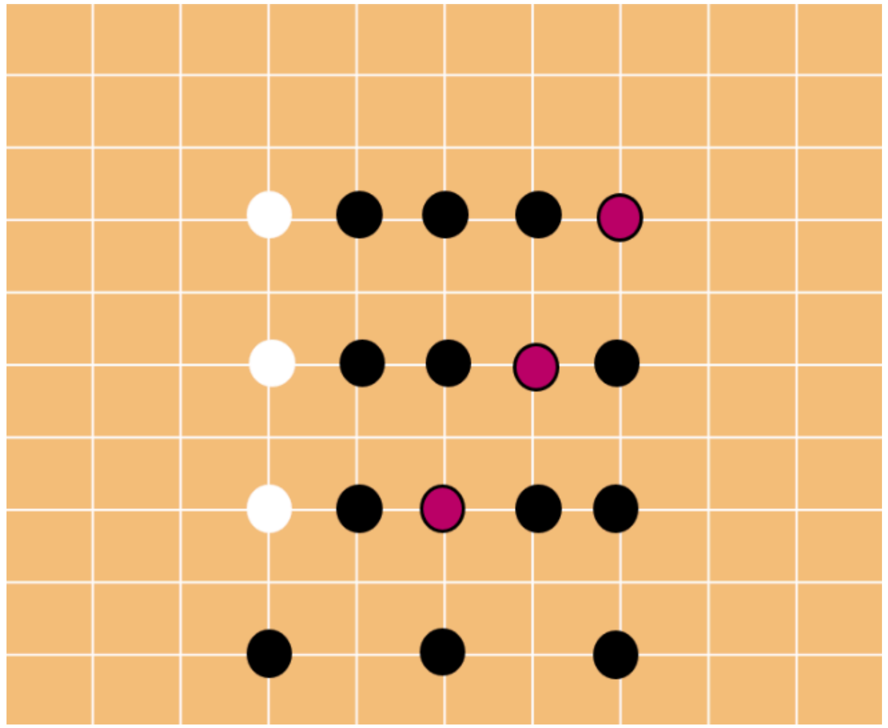
- 冲四活三：落下一手棋，同时形成冲四和活三。也是一种必胜的棋型。



Sleep Three (眠三)

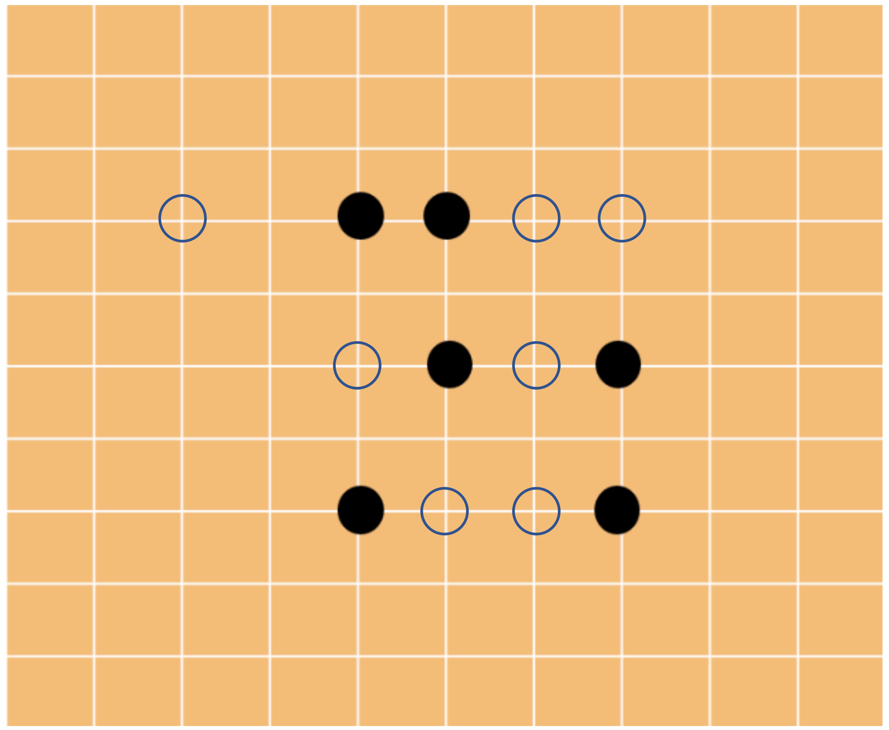
- Sleep three: Lead to flush four after a drop. There are many such chess types, which can be divided into several cases.

- 眠三：有落子后能成冲四的点的棋型。这种棋型很多，分好几种情况



Live Two (活二)

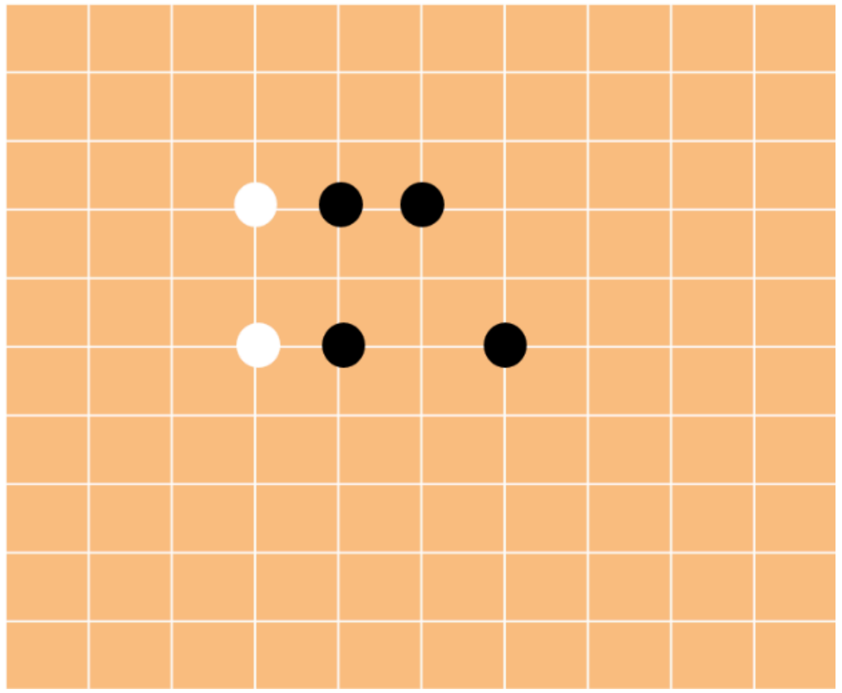
- Live two: Lead to live-three after a drop.
- 活二：有落子后能成活三的点的棋型。



Sleep Two (眠二)

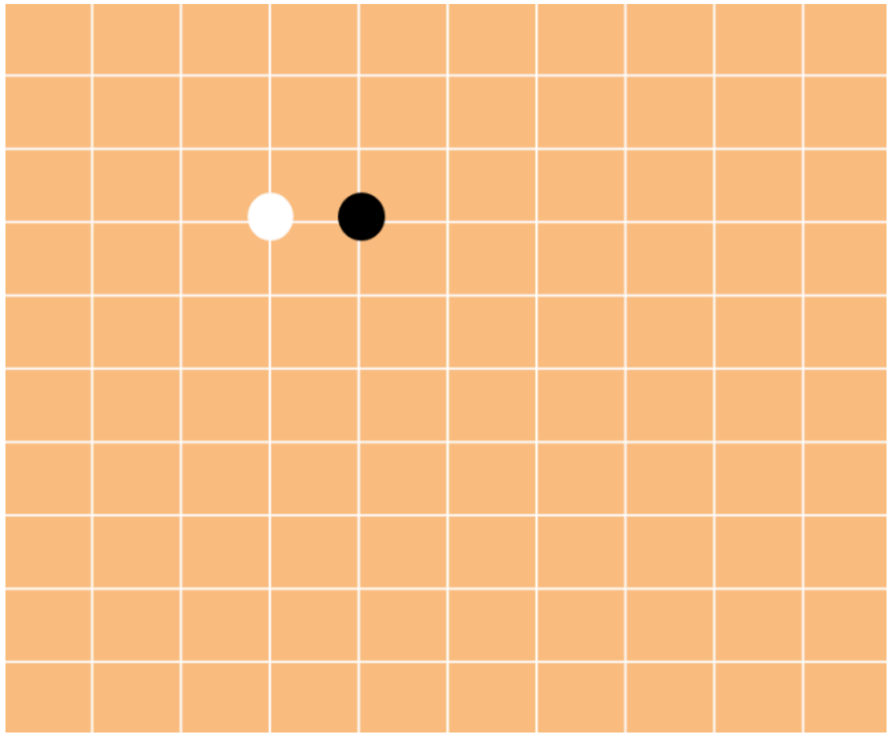
- Sleep two: Lead to sleep-three after a drop.

- 眠二：有落子后能成眠三的点的棋型。



Sleep One (眠一)

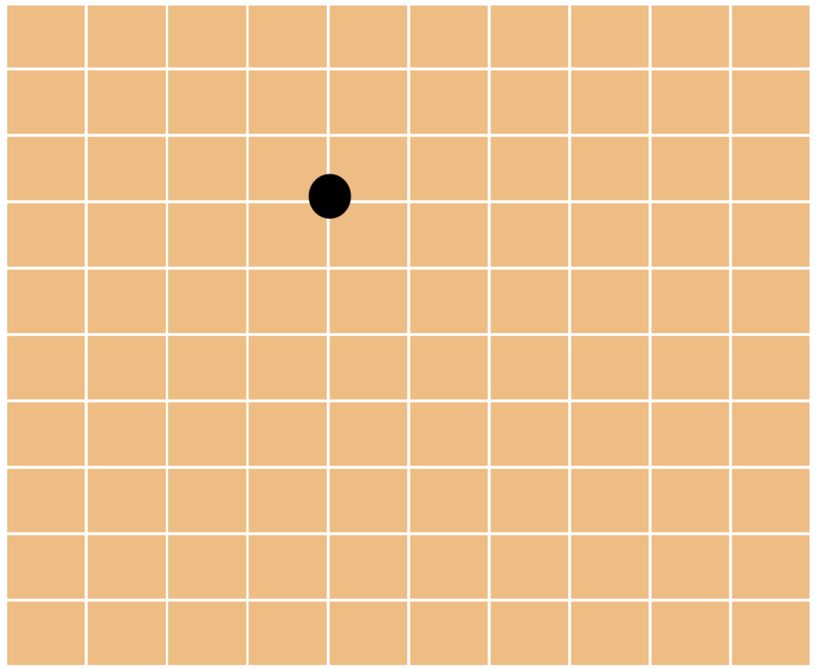
- Sleep one: Lead to sleep-two after a drop.
- 眠一：有落子后能成眠二的点的棋型。



Live One (活一)

- Live one: Lead to live-one after a fall of drop.

- 活一：有落子后能成活二的点的棋型。



Important Questions in Search

- How to evaluate an action?
- How to search efficiently?

Chess Type to Position Evaluation

- The more favorable the chess type, the more scores it gets. If we form a situation such as five-connected, long-connected, live-four, double-three, four-connected, flush four live three, which corresponds to a must-win situation, we should assign them maximum score.
- 越有利的棋型对应越大的价值，比如五连、长连、活四、双三、四三等，一旦形成就是必胜的局面，应该给予最大的分值

- Secondly, we should assign high scores to the chess type able to create a favorable situation, such as live-three, sleep three, double two.
- 其次是用于制造有利局面棋型，比如活三、眠三、两二等等

Chess Type to Position Evaluation

- In the vacancy, we should not only evaluate the benefit of white chess but also evaluate the benefit of black chess. If the opponent can get high benefit at a point. We should also try to seize this point to block the maximum benefit of the opponent.

- 在空位，不仅要估值如果下白棋的收益，也要估值下黑棋的收益，当敌方在某个位置收益最大，那么也是本方需要抢占的点，即阻断对方的最大收益.

Chess Type Representation

- Directly use representation of Go in code template
  - 0 means empty, -1 means black, 1 means white
  - 11111: five white   011110: four white   -1-1-1-1-1: five black   0-1-1-1-10 four black

- 可以直接用go的入参数组的表示法，
  - 0表示空，-1表示黑棋，1表示白棋
  - 11111: 五个白棋 011110: 4个白棋 -1-1-1-1-1: 5个黑棋 0-1-1-1-10 4个黑棋

Search Range

- Search over the whole chessboard
  - Simple to process, but low efficiency
- A Faster search method
  - Narrow the search range
  - Sub-area within a distance of 4 from the center of the currently drops.

- 最简单的方式为全棋盘搜索，优点是处理简单，缺点是效率低
- 通常为了缩小搜索范围，会考虑以当前已落子为中心的不大于4的距离。

Now, You can start Lab 1

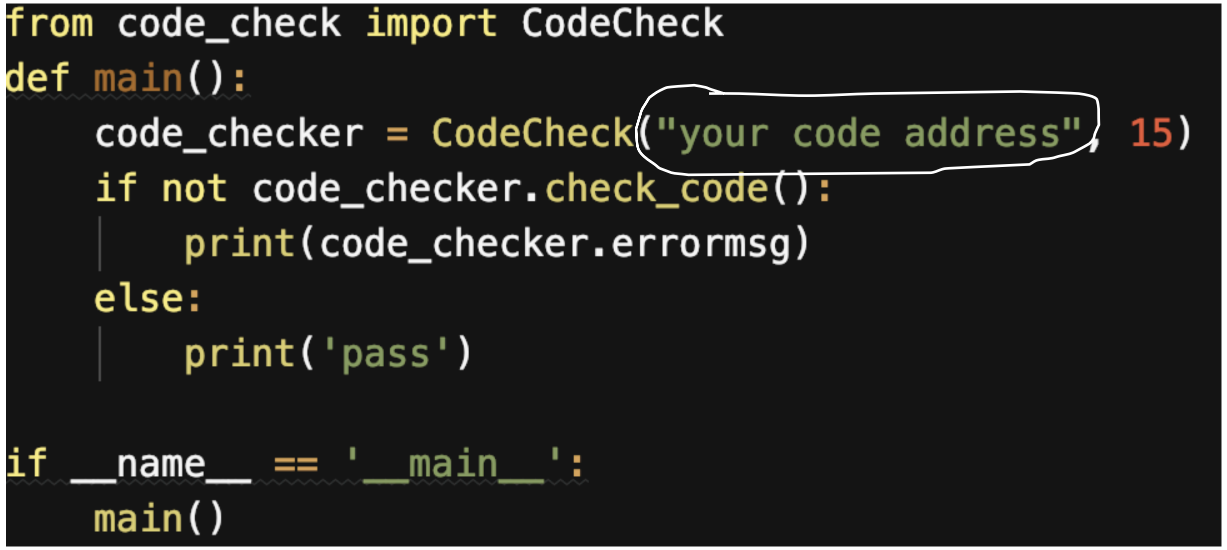
	Evaluation Rule	DeadLine
Lab1	Score according to the number of test cases passed	15th September
Lab2	Score according to the ranking in the points race	22nd September
Lab3	Score according to the ranking in the round robin	29th September

```
# The input is current chessboard.
    def go(self, chessboard):
        # Clear candidate_list
        self.candidate_list.clear()

#======================================================================
        #Write your algorithm here
        #Here is the simplest sample:Random decision
        idx = np.where(chessboard == COLOR_NONE)
        idx = list(zip(idx[0], idx[1]))
        pos_idx = random.randint(0, len(idx)-1)
        new_pos = idx[pos_idx]
        #===============Find new
pos===========================================
        # Make sure that the position of your decision in chess board is empty.
        #If not, return error.
        assert chessboard[new_pos[0],new_pos[1]]== COLOR_NONE
        #Add your decision into candidate_list, Records the chess board
        self.candidate_list.append(new_pos)
```

Code_Checker for Usability Testing

- Two Python Files
  - code_check.py
  - code_check_test.py



```
from code_check import CodeCheck
def main():
    code_checker = CodeCheck("your code address", 15)
    if not code_checker.check_code():
        print(code_checker.errormsg)
    else:
        print('pass')

if __name__ == '__main__':
    main()
```

Gomocup

- Website: http://gomocup.org/

Gomocup is a worldwide competition for articial intelligence in the area of Gomoku. It has been held annually since 2000. As of 2017, it is the largest and most inuential AI event for Gomoku in the world, with more than 40 authors from about 10 countries and regions.

- Gomocup是五子棋人工智能的世界性比赛。自2000年以来，每年举办一次。截至2017年，它是国际上最大且最有影响力的五子棋人工智能赛事，有来自约10个国家与地区的40余位作者参与。

Summary

- Introduction to Gomokou
- Requirements of Project 1
    - Three phases
    - Code and report requirements
- Chess Type and Basic Search for Gomoku
- Other Search Methods in Later Course
- Slides, project document, test code, etc, please download from:

https://cas.sustech.edu.cn/cas/login?service=https%3A%2F%2Fsakai.sustech.edu.cn%2Fsakai-login-tool%2Fcontainer

Enjoy your journey to Champion！