1. 随机产生1000个起始点；

import random

def main():

f = open("eightQueensTest.txt", "wb")

testCaseCount = 1000

board = ""

while testCaseCount > 0:

testCaseCount -= 1

for col in range(0,7):

board += str(random.randint(0,7)) + ' '

board += str(random.randint(0,7)) + '\n'

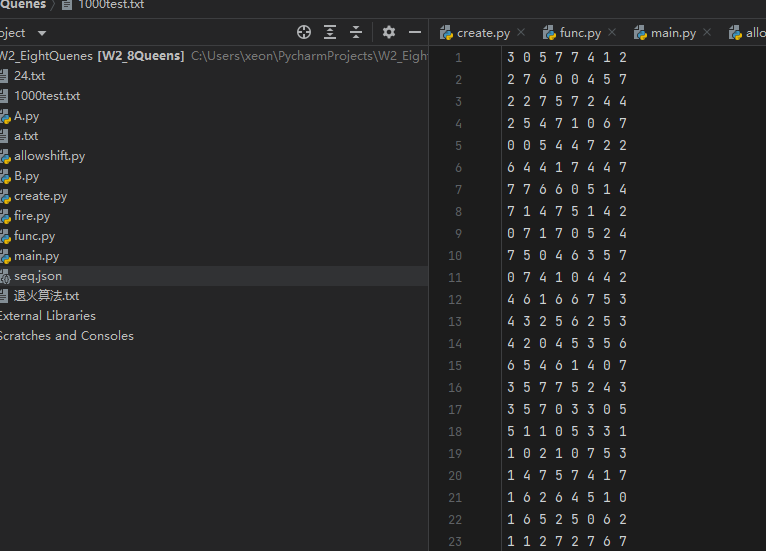
f.write(board)

f.close()

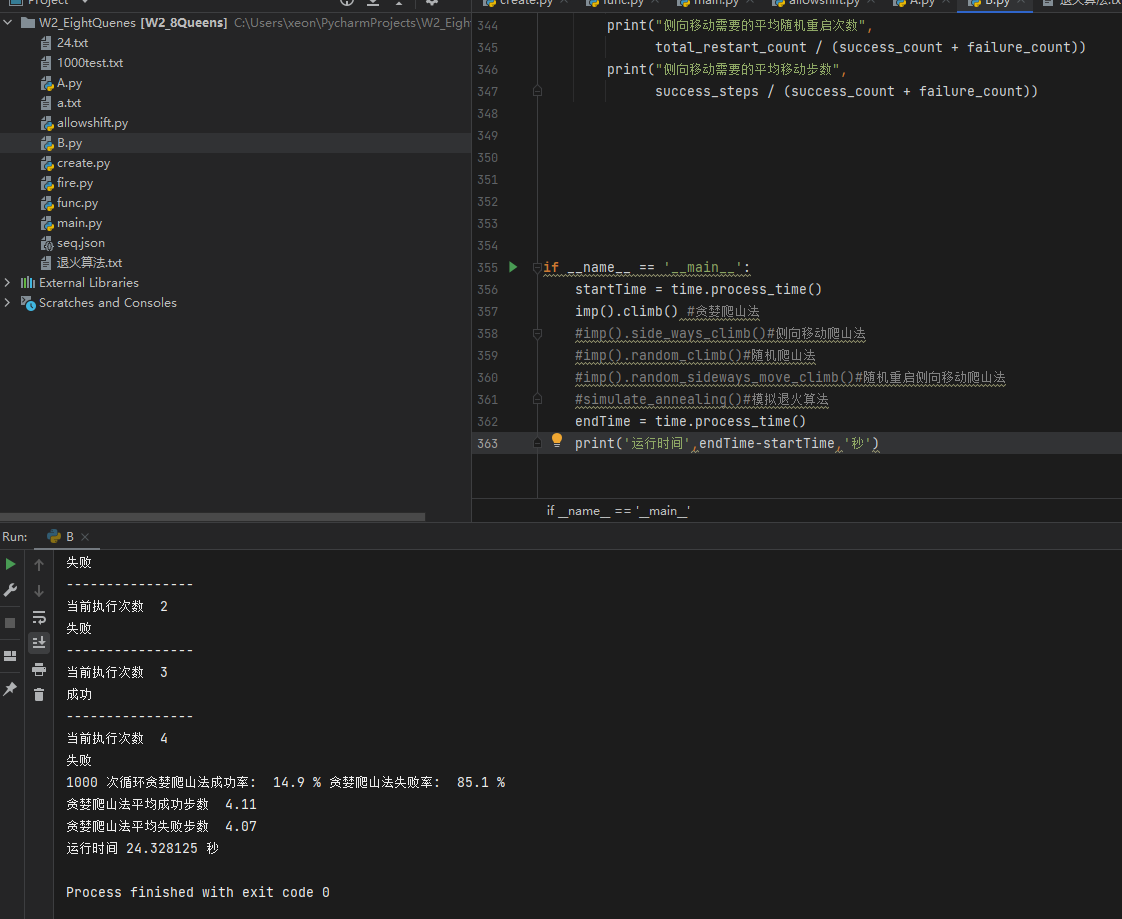
if \_\_name\_\_ == '\_\_main\_\_':

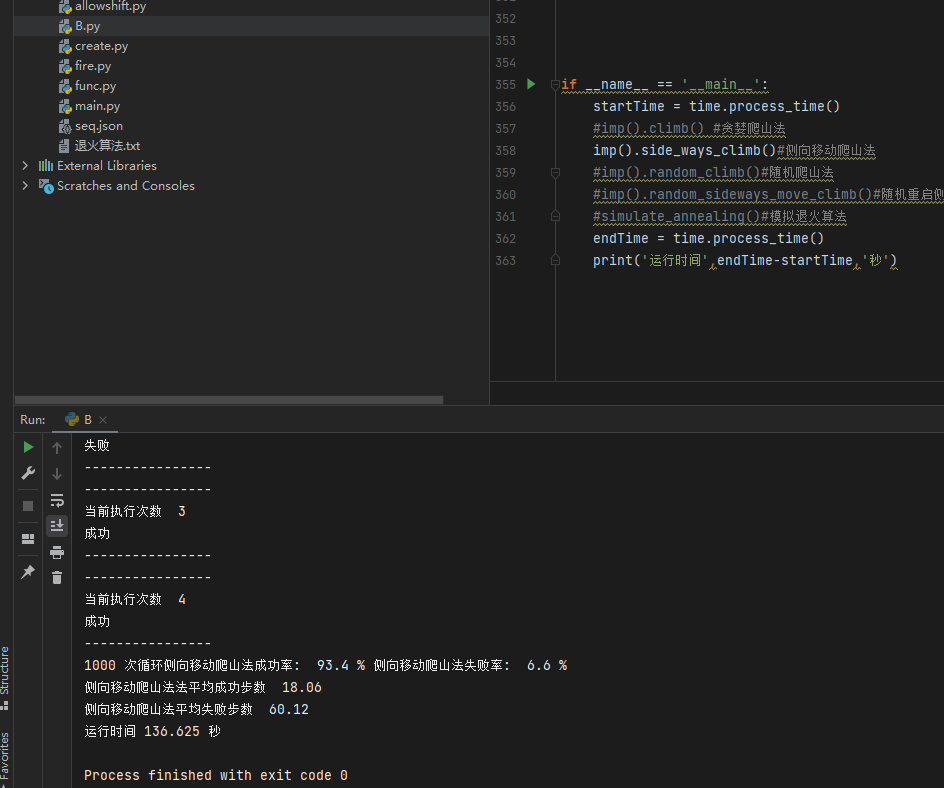
main()

生成1000个棋盘初始状态

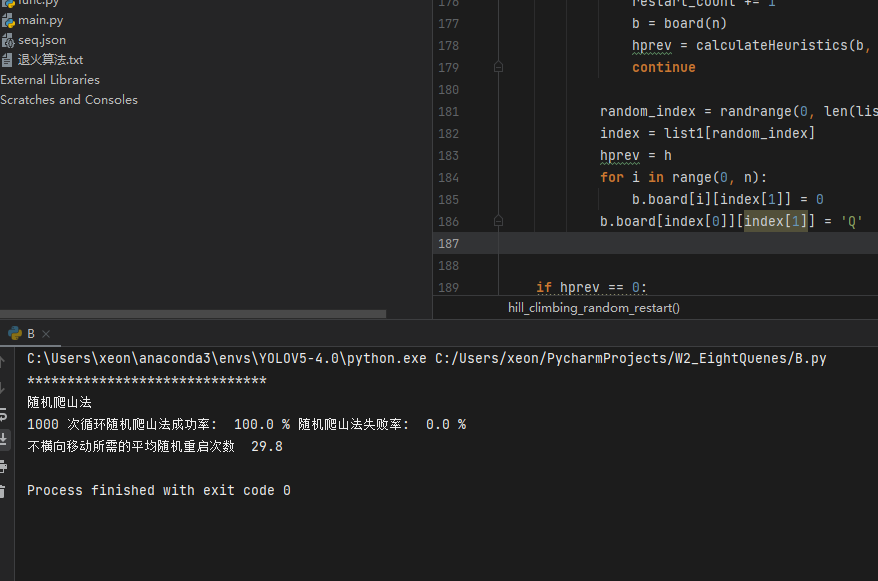


1. 分别实现贪婪爬山法、侧向移动爬山法、随机爬山法；
2. 贪婪爬山法

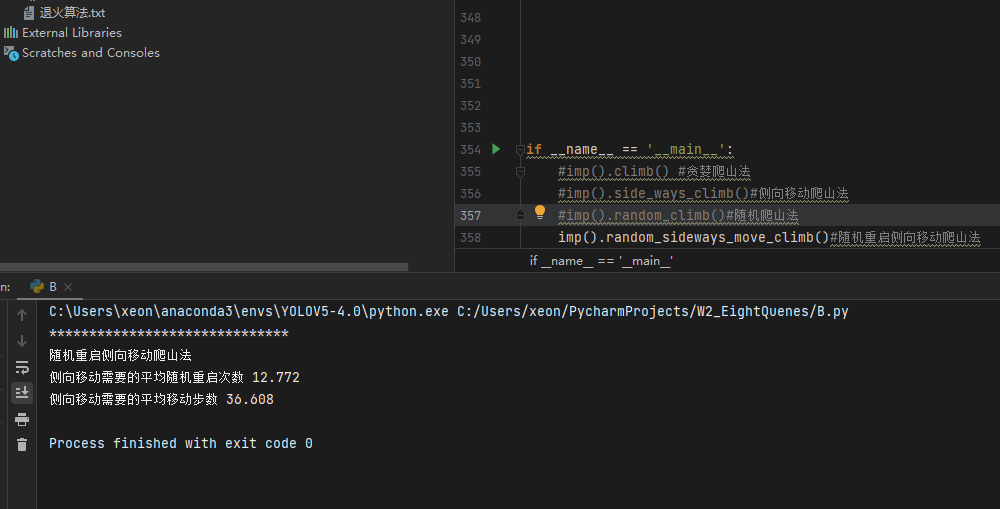


1. 侧向移动爬山法
2. 对前三个算法实现随机重启；

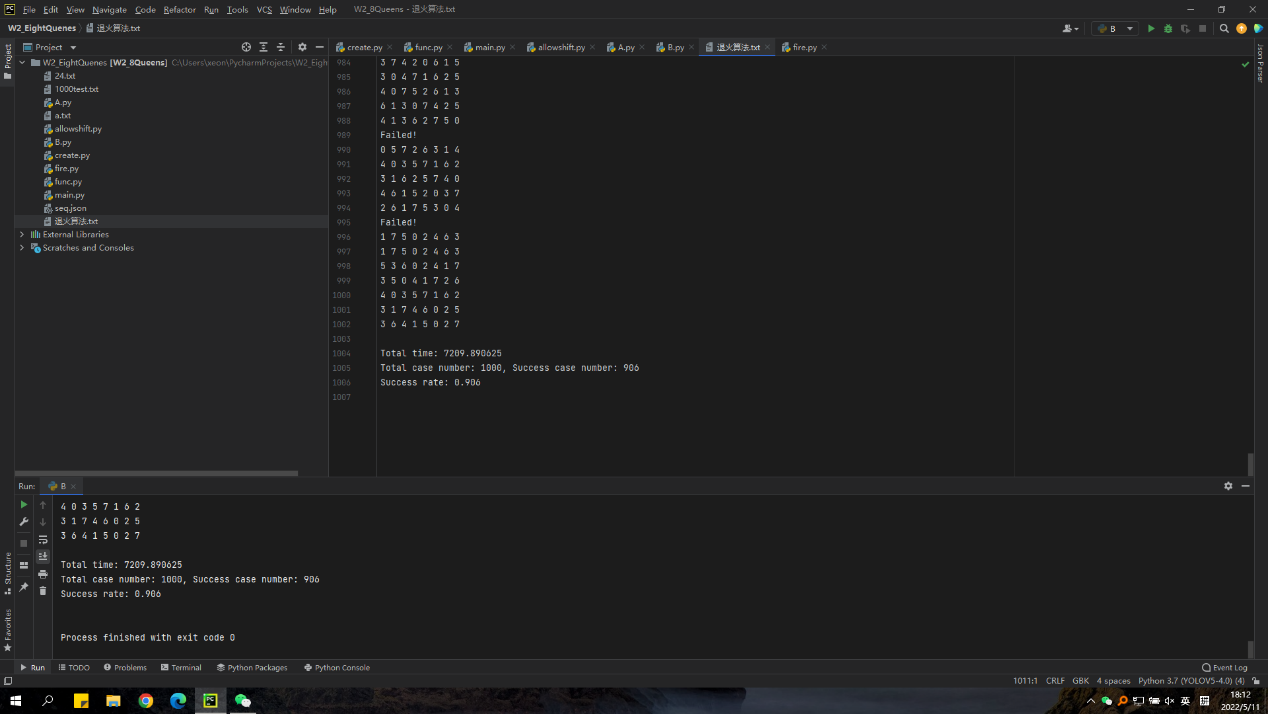
1、随机重启爬山法：



2、随机重启侧向移动爬山法



1. 实现模拟退火方法；



1. 利用1000个起始点，对前述方法分别统计成功求解的次数，成功求解

时的迭代步数，失败时的步数；

1000次

**贪婪爬山法：**成功率14.2% 共成功142次 运行时间24s

**随机重启爬山：**

1000 次循环随机爬山法成功率: 100.0 % 随机爬山法失败率: 0.0 %

所需的平均随机重启次数 31.9

运行时间 21.890625 秒

**侧向移动爬山法**：95% 共950次成功 运行时间134s

随机重启侧向移动爬山法：

侧向移动需要的平均随机重启次数 12.079

侧向移动需要的平均移动步数 34.894

运行时间 185.109375 秒

退火算法: 运行时间: 7209.890625s

成功率90.6%

详细步数在附件中

总结，贪婪爬山速度最快但是精度低，如果加上随机重启则能极大改善，耗时也是三种方法里面最短的。相比下，退火算法在这个项目中时间太长，精度也没有优势。总而言之，随机重启贪婪爬山在八皇后中最好