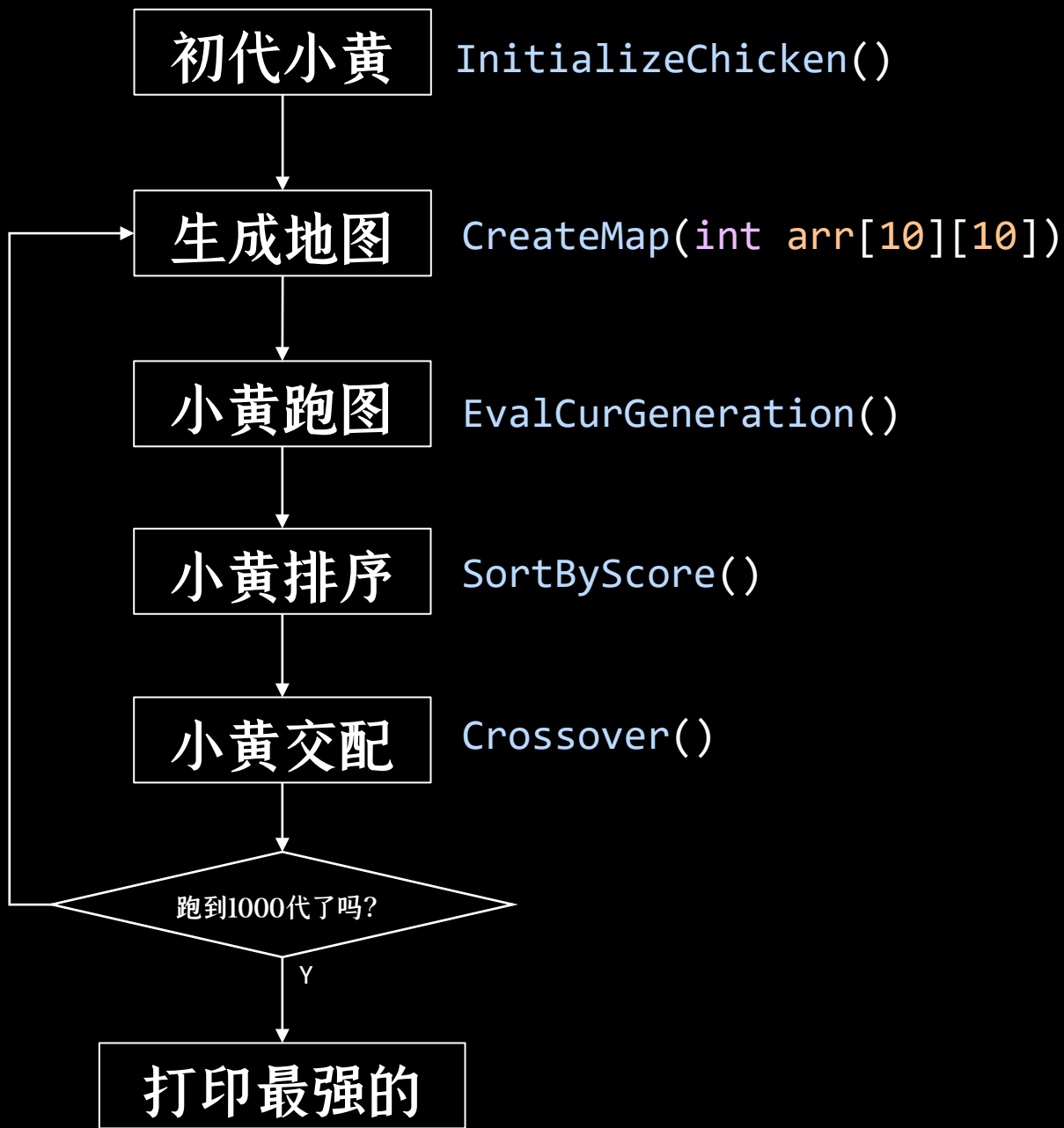


# Project2

唐傑伟

# 破题

|没那么难啦~



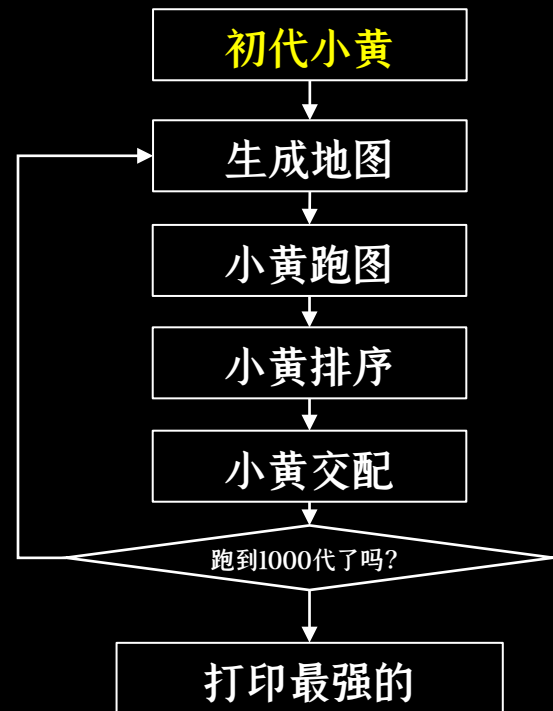
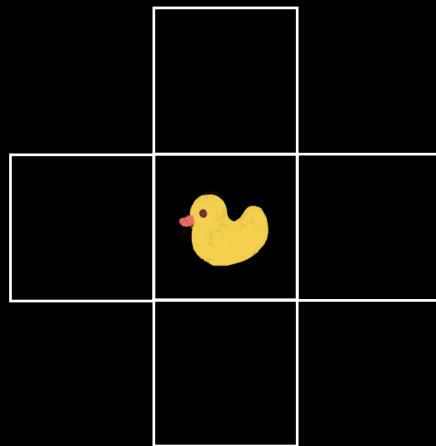
# 解题思路

|没那么难啦~

小黄的策略表  
Why 243?

$$243 = 3*3*3*3*3$$

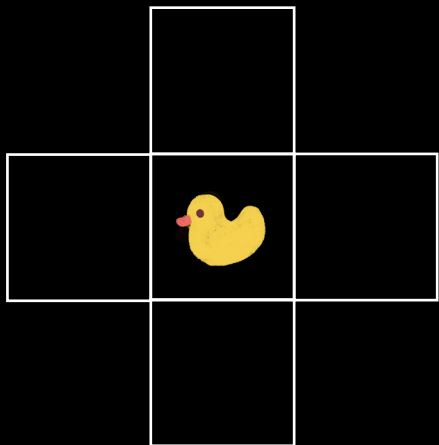
小黄和你不一样！小黄**没有记忆力**，它只能根据上下左右的情况做出判断。



# 解题思路

|没那么难啦~

小黄和你不一样！小黄没有记忆力，它只能根据上下左右的情况做出判断。

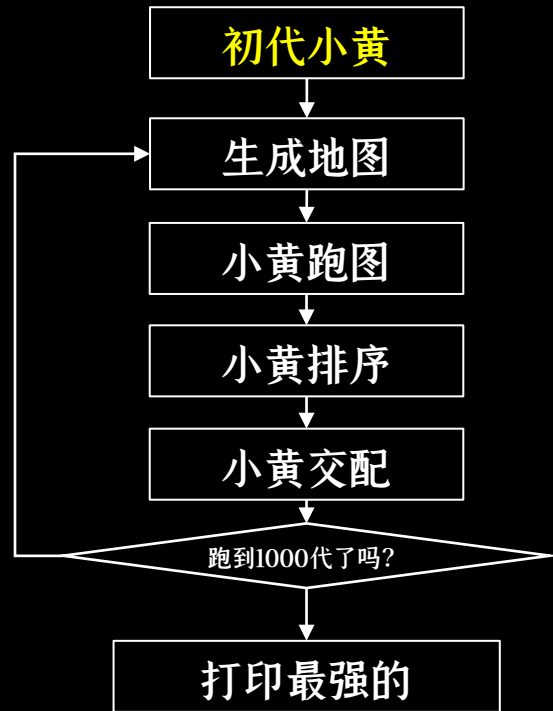


遗传算法的秘诀在于**大自然的力量**。

遗传算法通过以简化的模型模拟大自然的**自然选择**，以**较低的计算量**得到复杂问题的**较优解**。

遗传算法的劣势：

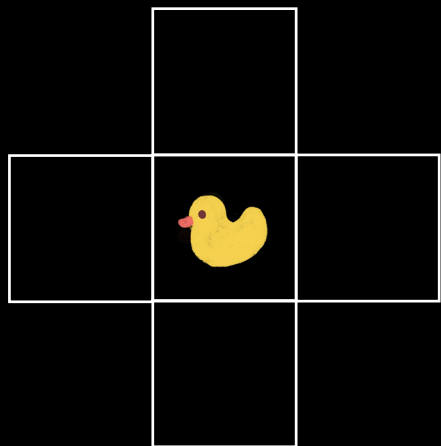
问题过于复杂的时候，容易陷入局部最优解，规避这一现象涉及到参数的调整。



# 解题思路

|没那么难啦~

小黄和你不一样！小黄没有记忆力，它只能根据上下左右的情况做出判断。

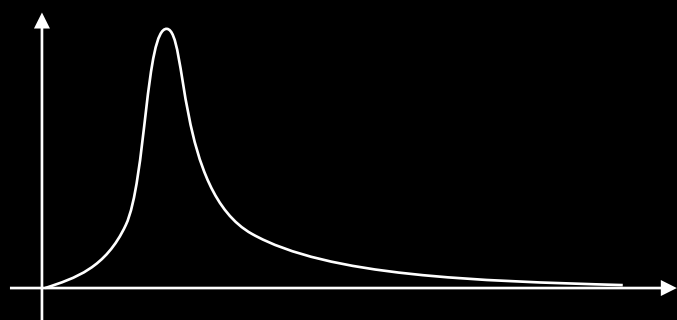


遗传算法的秘诀在于**大自然的力量**。

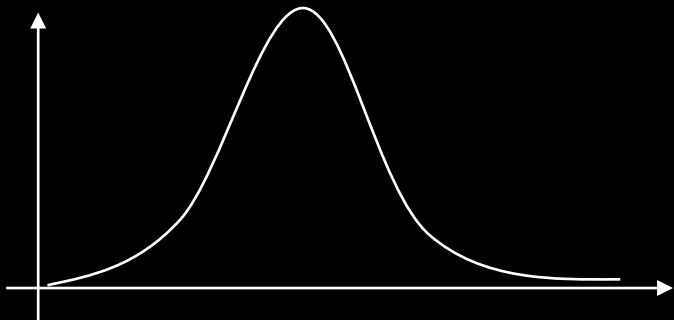
然而第一批小黄没有东西可参考，因此我们采用**随机数生成**。

数据在统计图中的形状，叫做它的**分布**。

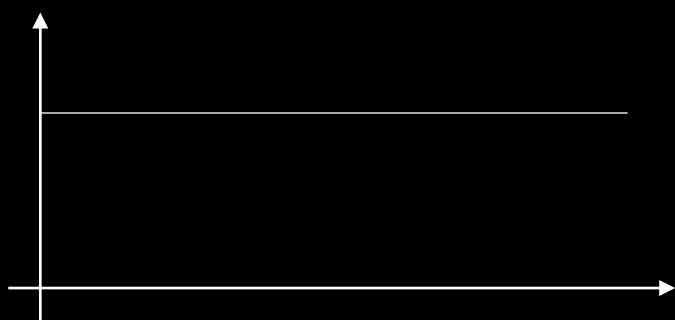
生成不同分布的随机数的算法有很多种。



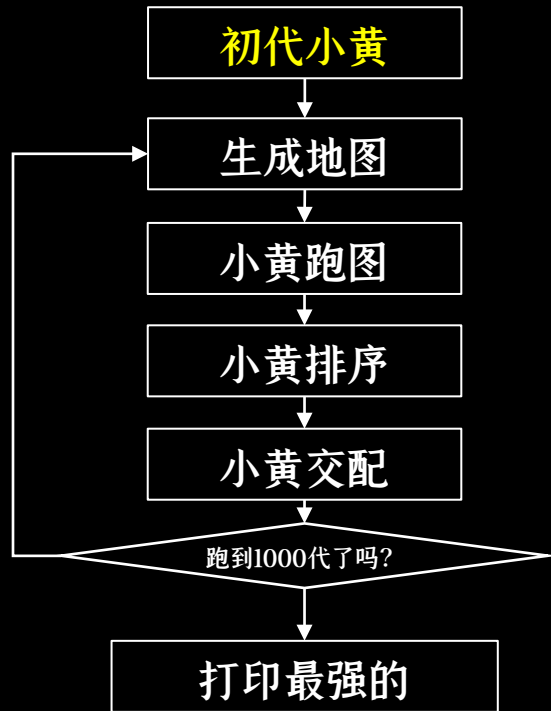
泊松分布



正态分布



均匀分布



# 解题思路

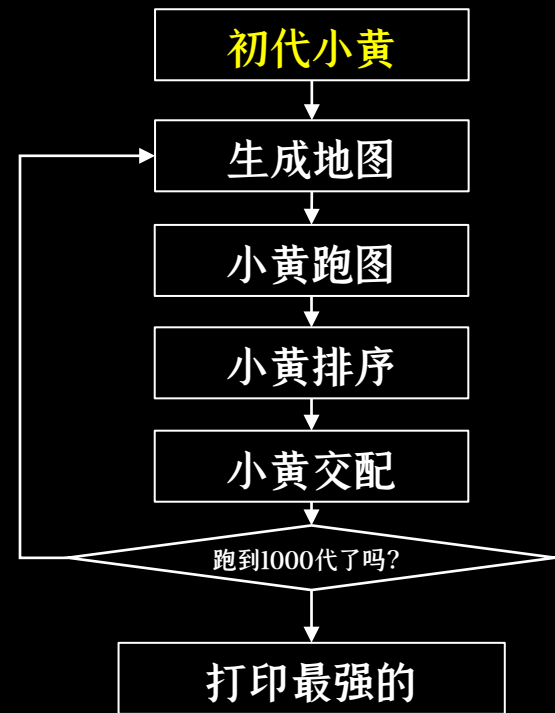
| 随机算法

C语言产生随机数的算法实质上是一种伪随机数算法。

输入



输出

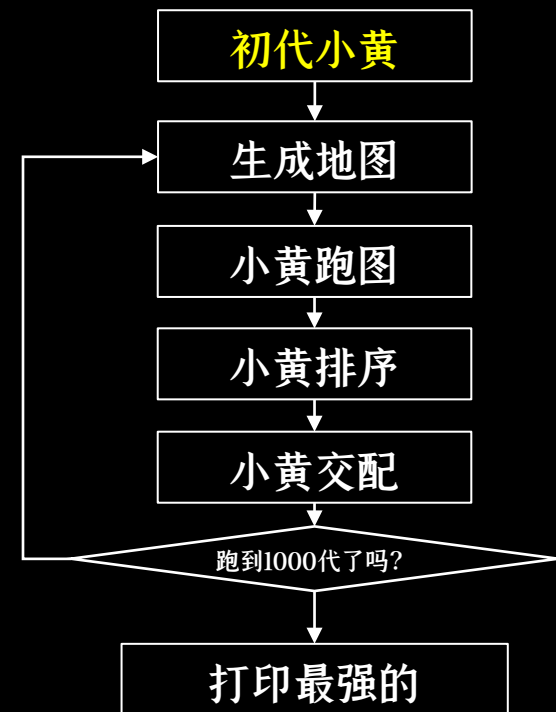


# 解题思路

| 随机算法

C语言产生随机数的算法实质上是一种伪随机数算法。

7  
1  
5  
2  
0  
4

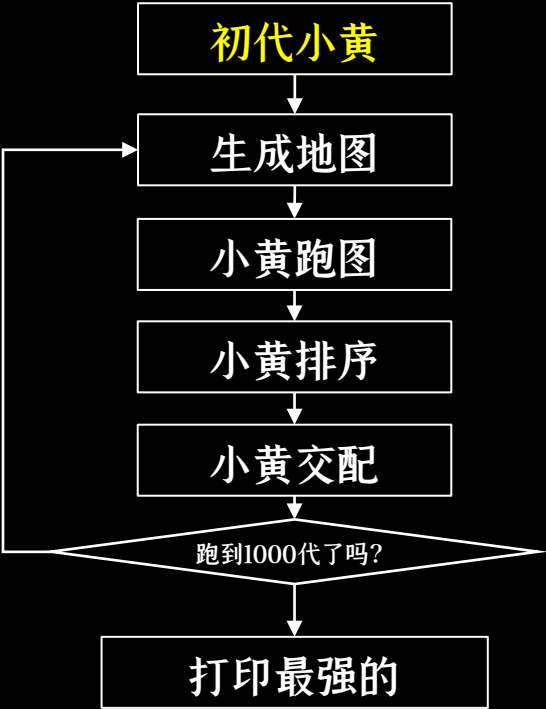


# 解题思路

| 随机算法

C语言产生随机数的算法实质上是一种伪随机数算法。

1	2	3	4	5	6	7	8	9
2	4	6	8	1	0	1	2	1
3	6	9	1	2	1	5	1	8
4	8	1	2	1	6	2	0	2
5	1	0	1	5	2	0	2	5
6	1	2	1	8	2	4	3	0

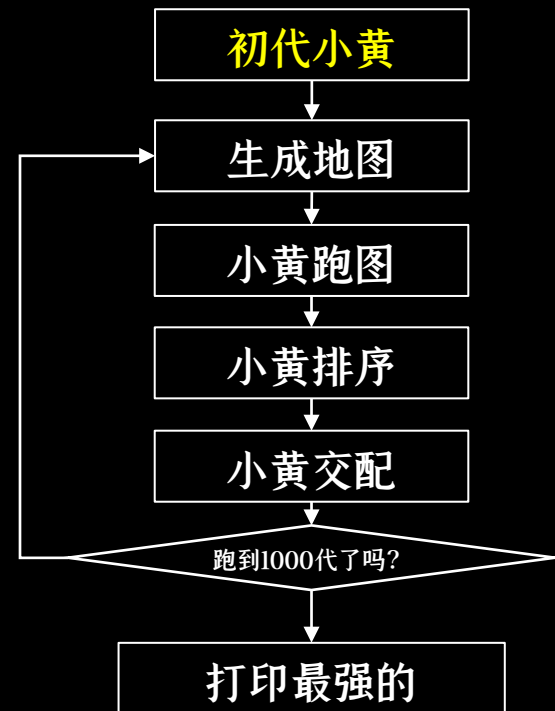




# 解题思路

| 随机算法

随机种子的作用就是固定模型训练过程中所产生的随机数。



# 解题思路

## | 随机算法

随机种子的作用就是固定模型训练过程中所产生的随机数。

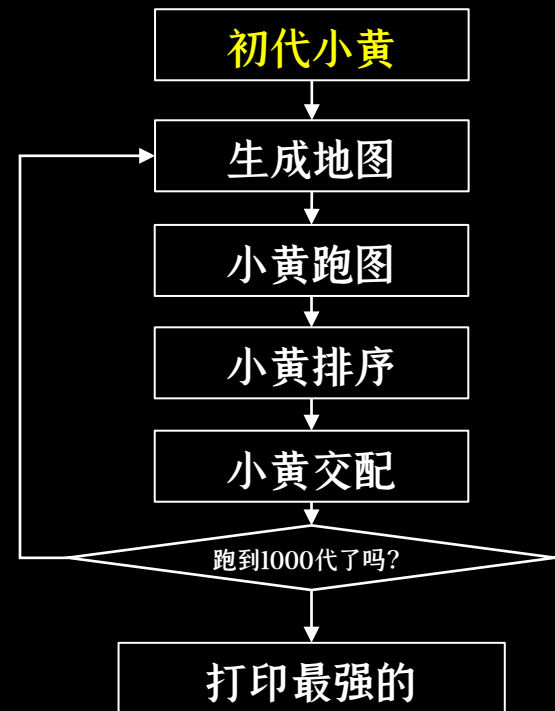
如果想要保证你的程序稳定，建议在提交的代码中**固定你的随机数种子**。

易错点：

srand(seed)的作用：重置随机数种子为当前随机数种子。

请**不要在每次生成策略表的时候都执行srand**，这样生成的所有策略表都是一致的。

time(NULL)的值以秒为间隔变化。

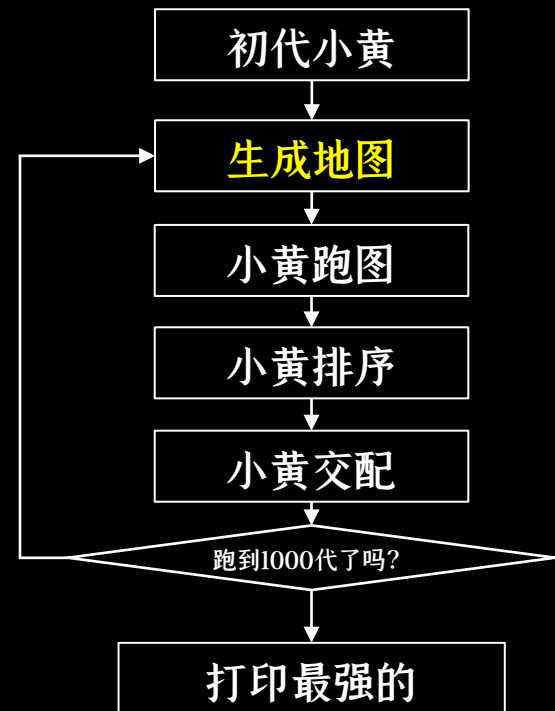


# 解题思路

| 随机算法

利用for循环和rand()函数给棋盘每个位置赋值？

生成的随机数数量不定，最好能保证随机数的数量恰好为50个。



# 解题思路

|创世主!!!

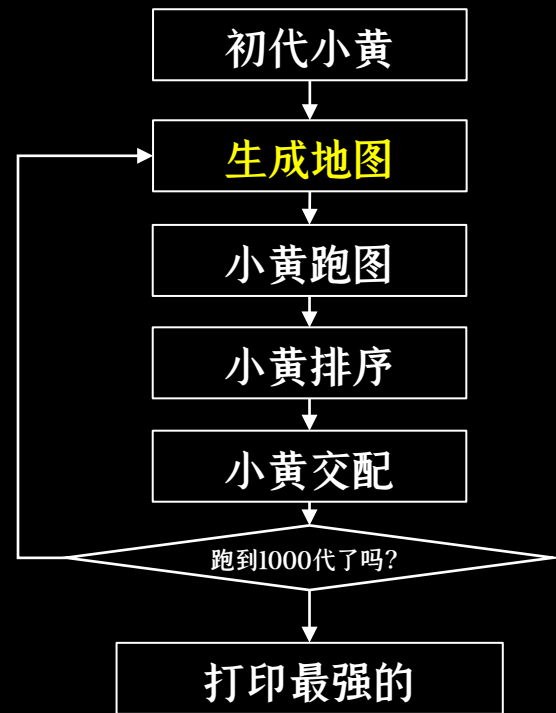
利用for循环和rand()函数给棋盘每个位置赋值？

生成的随机数数量不定，最好能保证随机数的数量恰好为50个。

shuffle()

首先制造一个前50个为1，后50个为0的数组。

利用循环遍历每一个元素为每一个元素生成一个1-100的随机整数x，使之与第x个数交换。



# 解题思路

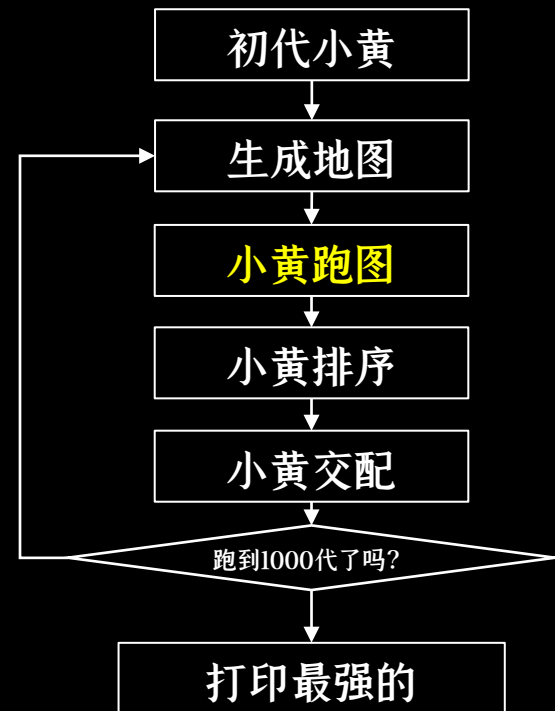
| 小黄快跑

构造一个函数，代表每一个小黄的每一步。

```
void ProcessSingleStep(int *score, int arr[10][10],  
int *x, int *y, int StepName, int *StepCounter)
```

```
int EvalSingleChick(int arr[][243], int ChickIndex)
```

```
float EvalSingleBatch(int ChickIndex)
```

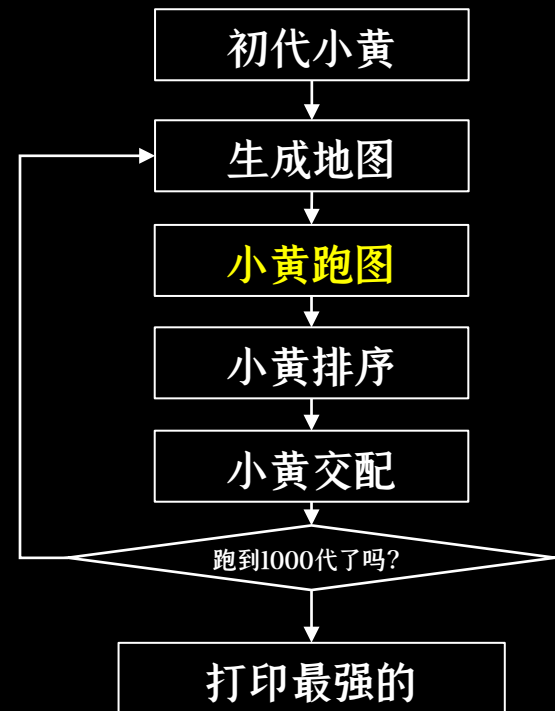


# 解题思路

|小黄快跑

完成函数的嵌套。

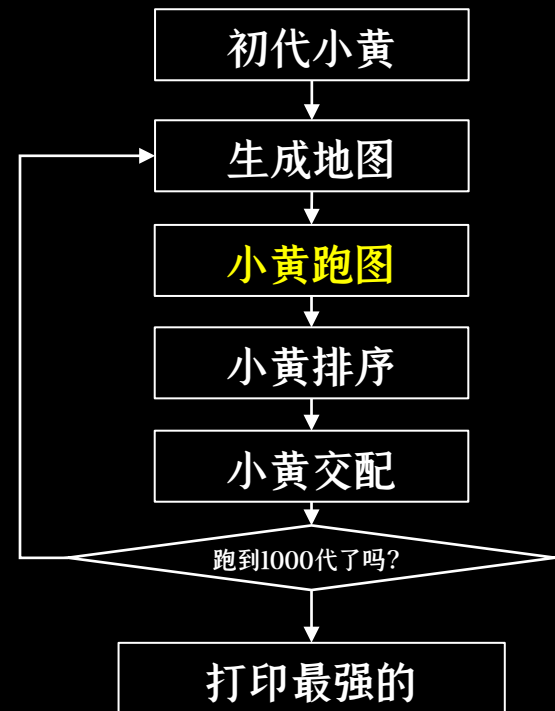
```
float EvalSingleBatch(int ChickIndex)
{
    int EvalSingleChick(int arr[][243], int ChickIndex)
    {
        void ProcessSingleStep(int *score, int arr[10][10],
            int *x, int *y, int StepName, int *StepCounter);
    }
}
```



# 宏定义

|小黄可以采取的全部动作(键位设置)

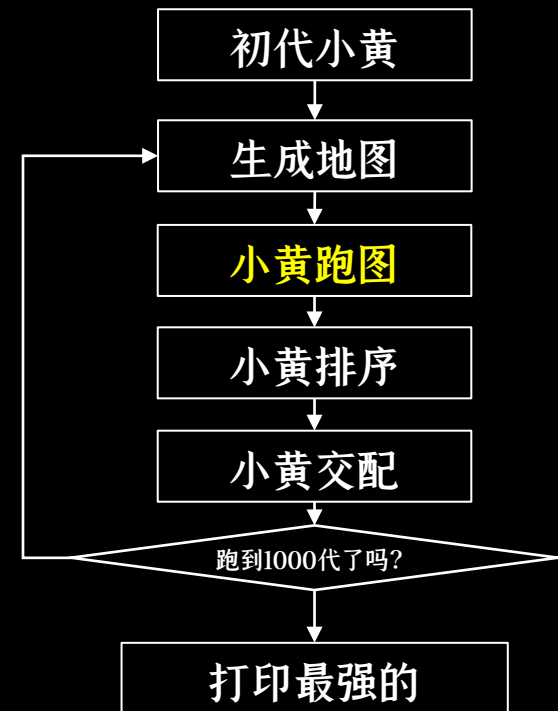
```
#define RANDOM_MOVEMENT 0
#define GO_UP 1
#define GO_DOWN 2
#define GO_LEFT 3
#define GO_RIGHT 4
#define STAY 5
#define PICK 6
```



# 宏定义

| 周围情况

```
#define WALL 2  
#define CAN 1  
#define EMPTY 0
```





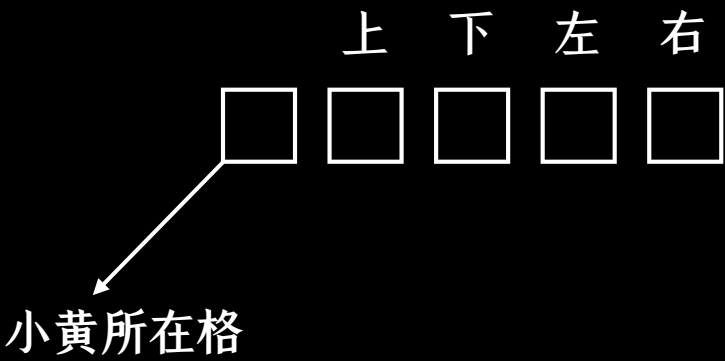
# 解题思路

| 小黄快跑

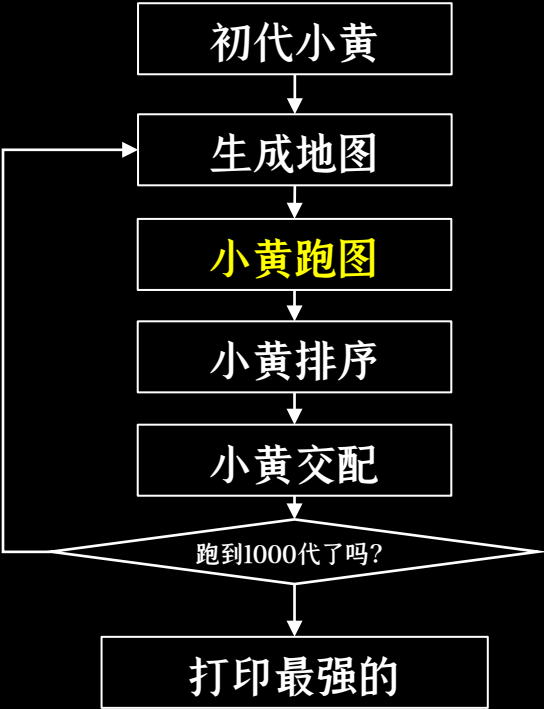
如何确定要选用的是策略表的哪一位？

约定一个映射所有情况到数组上的每一位的办法。

例如：三进制数



将三进制数转化为十进制数，恰好为0-242，可以对应下标。

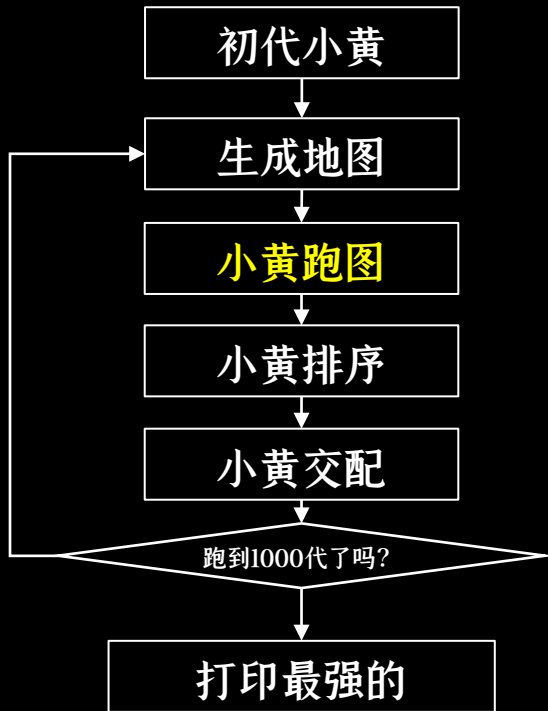


# 解题思路

| 小黄快跑

得分判定。

1. 随机移动, 即随机出现以下(No.2 ~ No.7)行为中的任何一种。
2. 向上移动, 如果小黄上方的格子为墙, 则小黄位置不变, 视为一次撞墙 (-2) 。
3. 向下移动, 如果小黄下方的格子为墙, 则小黄位置不变, 视为一次撞墙 (-2) 。
4. 向左移动, 如果小黄左边的格子为墙, 则小黄位置不变, 视为一次撞墙 (-2) 。
5. 向右移动, 如果小黄右边的格子为墙, 则小黄位置不变, 视为一次撞墙 (-2) 。
6. 不移动。
7. 捡罐子 (+10) 。



# 解题思路

| 小黄快跑

得分判定。

1. 随机移动, 即随机出现以下(No.2 ~ No.7)行为中的任何一种。

① GOTO

② 用while循环判断函数结果

③ 再执行一遍本函数 (注意在编写时, 判断是否走满了200步要放在该函数前面)

2. 向上移动, 如果小黄上方的格子为墙, 则小黄位置不变, 视为一次撞墙 (-2)。

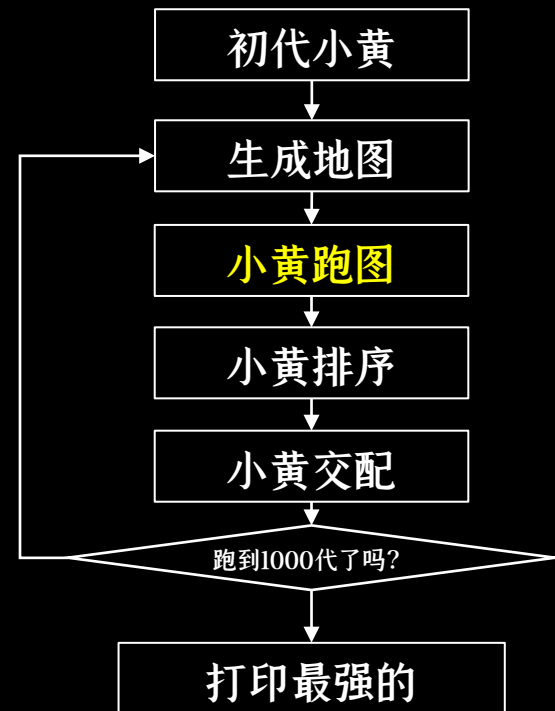
3. 向下移动, 如果小黄下方的格子为墙, 则小黄位置不变, 视为一次撞墙 (-2)。

4. 向左移动, 如果小黄左边的格子为墙, 则小黄位置不变, 视为一次撞墙 (-2)。

5. 向右移动, 如果小黄右边的格子为墙, 则小黄位置不变, 视为一次撞墙 (-2)。

6. 不移动。

7. 捡罐子 (+10)。



# 解题思路

|小黄快跑

得分判定。

1.随机移动,即随机出现以下(No.2 ~ No.7)行为中的任何一种。

- 再执行一遍本函数

```
void ProcessSingleStep(int *score, int arr[10][10], int *x, int *y,  
int StepName, int *StepCounter)
```

- 函数内调用

```
ProcessSingleStep(score, arr, x, y, StepName, StepCounter);
```

- 函数外调用

```
ProcessSingleStep(&score, plane, &coordinate_x, &coordinate_y,  
arr[ChickIndex][MapSituationToDec(plane, coordinate_x,  
coordinate_y)], &StepCounter);
```

2.向上移动,如果小黄上方的格子为墙,则小黄位置不变,视为一次撞墙 (-2) 。

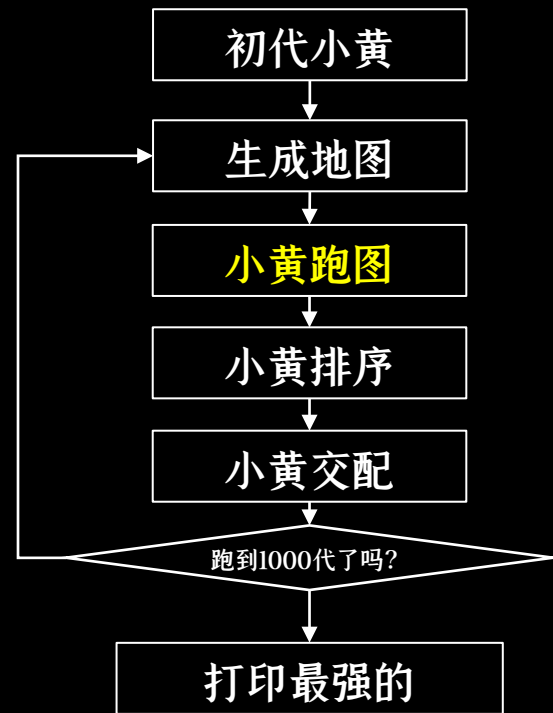
3.向下移动,如果小黄下方的格子为墙,则小黄位置不变,视为一次撞墙 (-2) 。

4.向左移动,如果小黄左边的格子为墙,则小黄位置不变,视为一次撞墙 (-2) 。

5.向右移动,如果小黄右边的格子为墙,则小黄位置不变,视为一次撞墙 (-2) 。

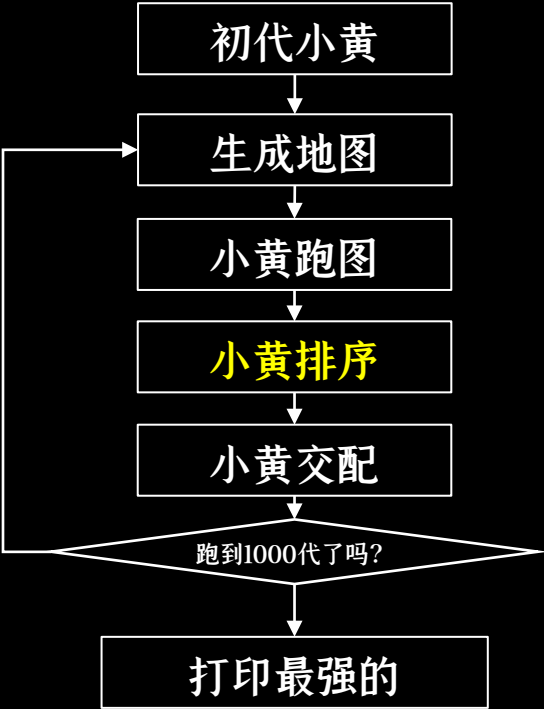
6.不移动。

7.捡罐子 (+10) 。



# 解题思路

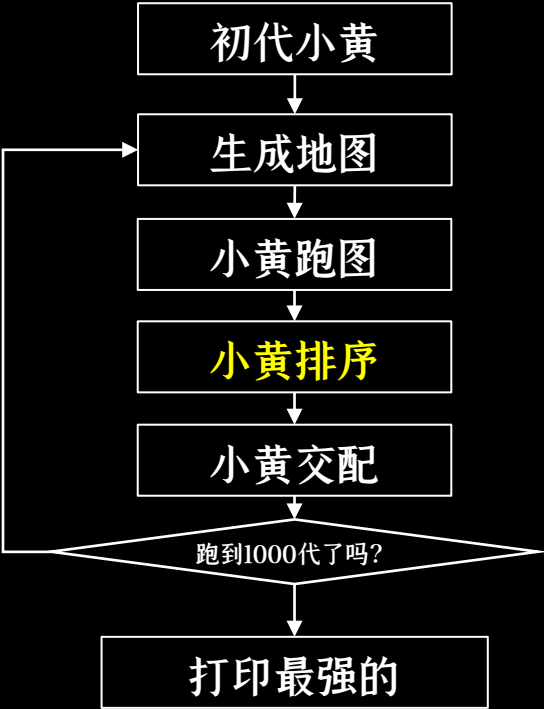
| 带权重抽取



# 解题思路

| 带权重抽取

- ①等差数列求和函数的反函数取整
- ②While循环递推



1	2	3	.....	198	199	200
---	---	---	-------	-----	-----	-----

0

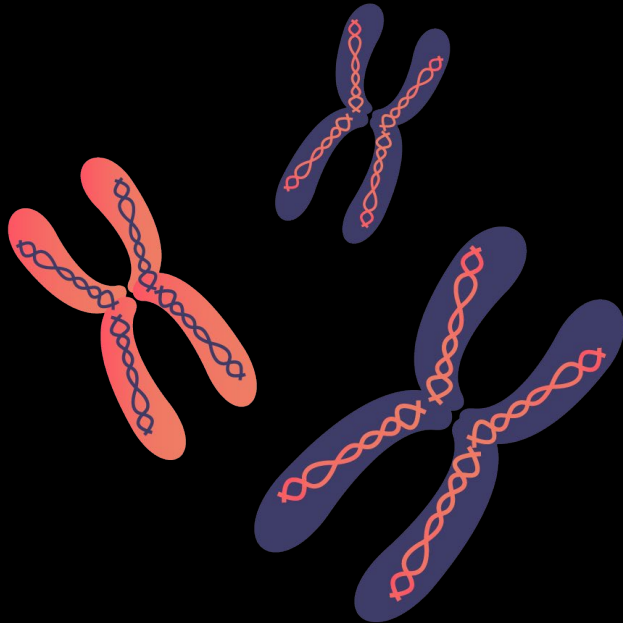
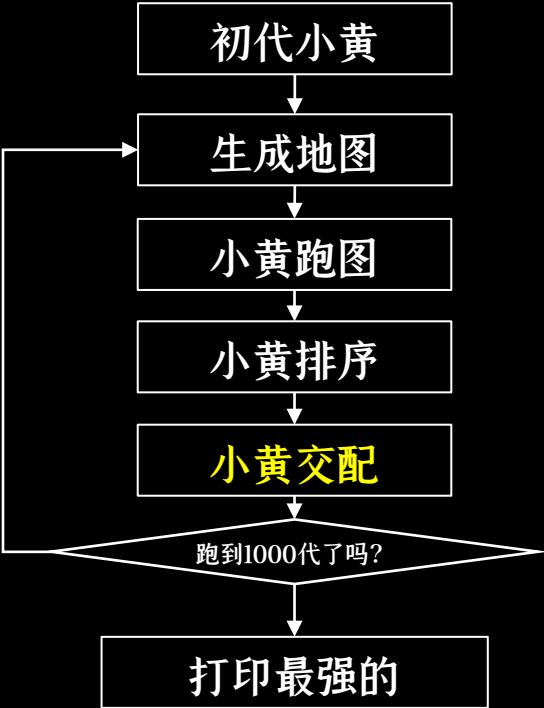
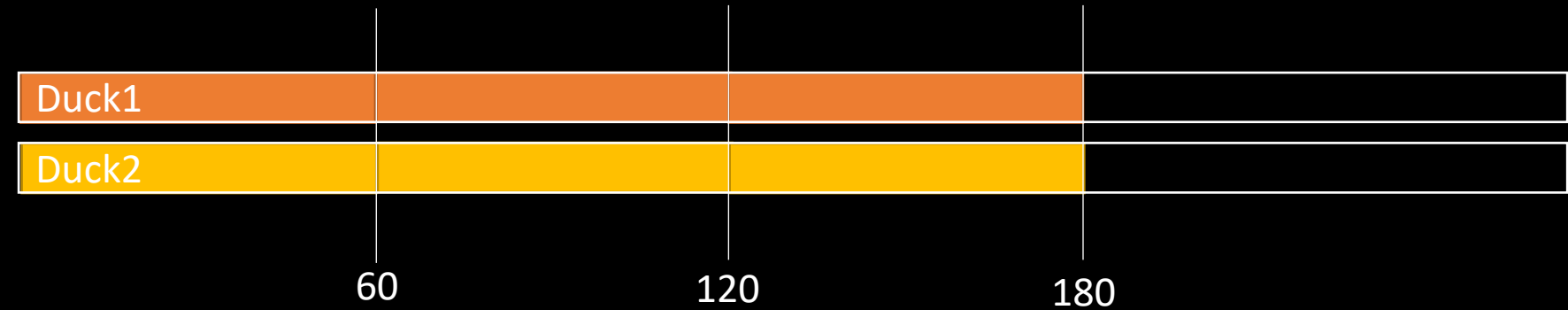
$201 \times 200 / 2$



# 解题思路

|小黄羞羞 ( ? )      #define SLICING 60

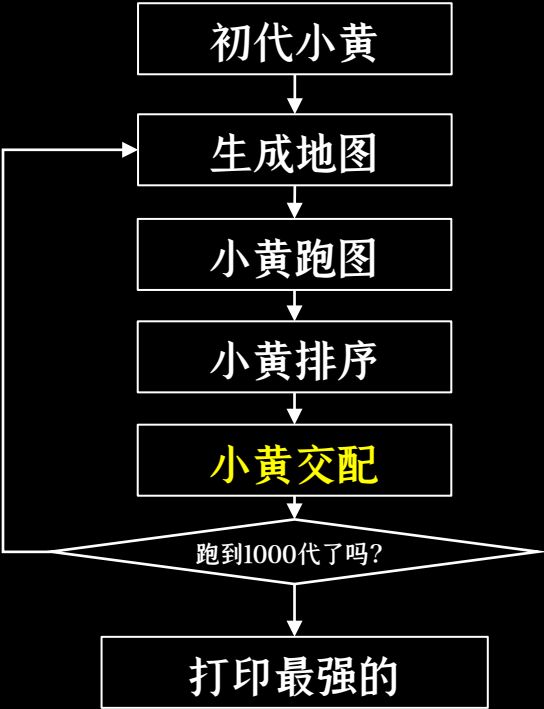
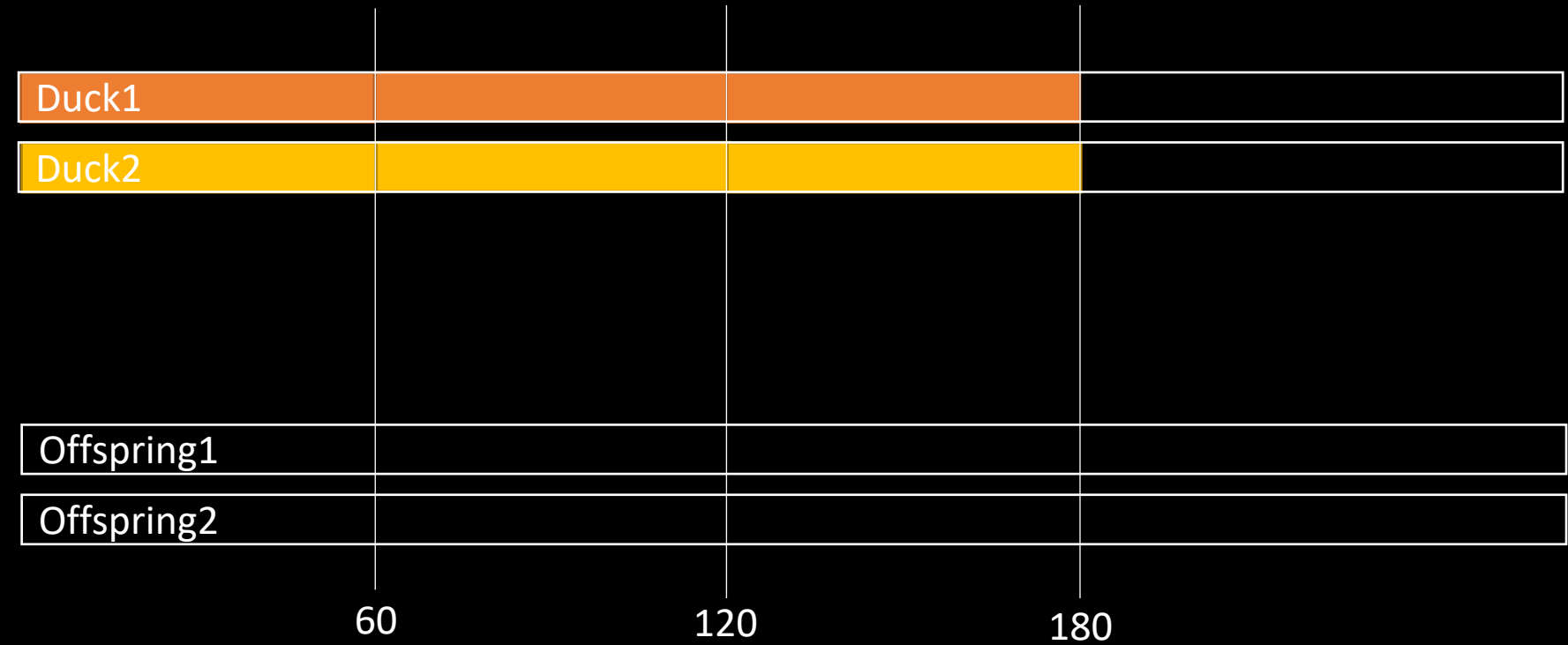
交配



# 解题思路

|小黄羞羞 ( ? )      #define SLICING 60

交配：创建新的子代对象

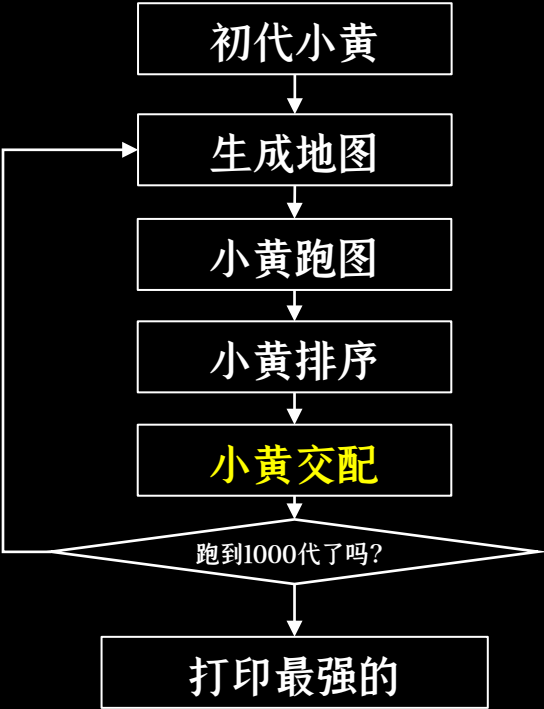
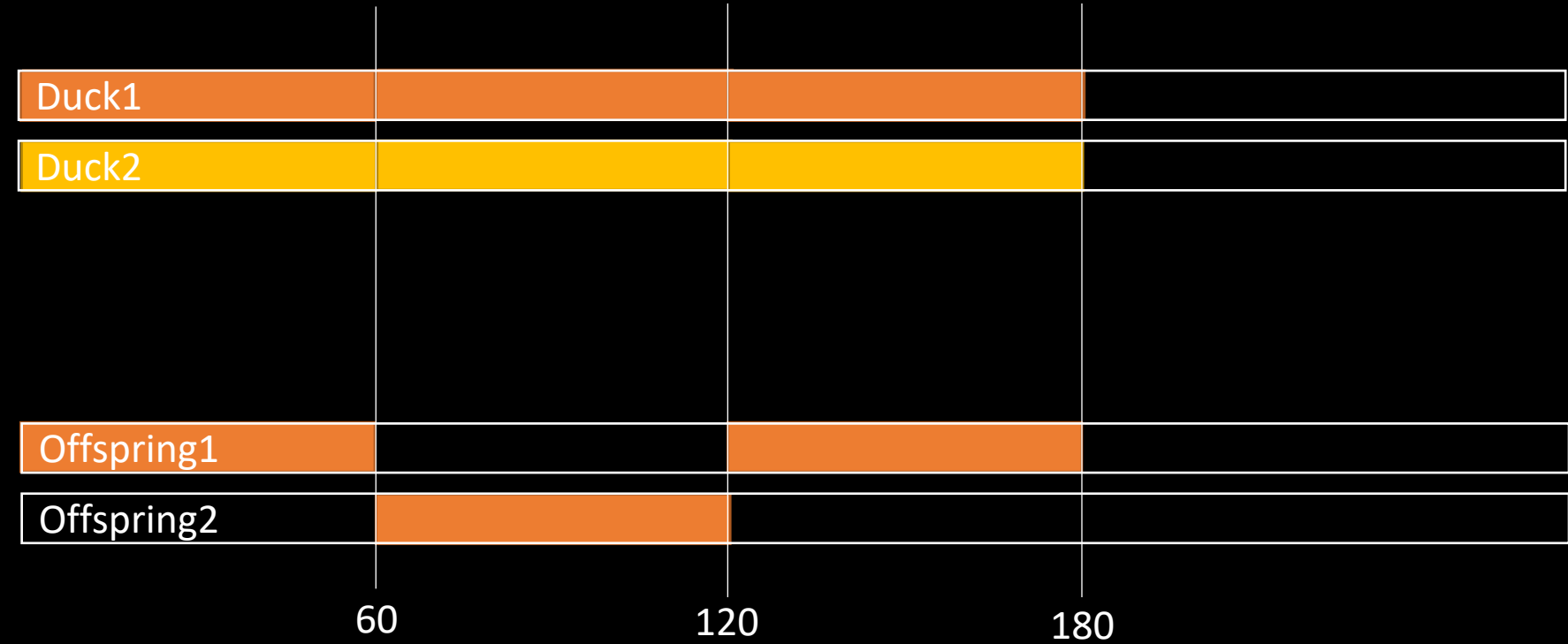




# 解题思路

|小黄羞羞 ( ? )      #define SLICING 60

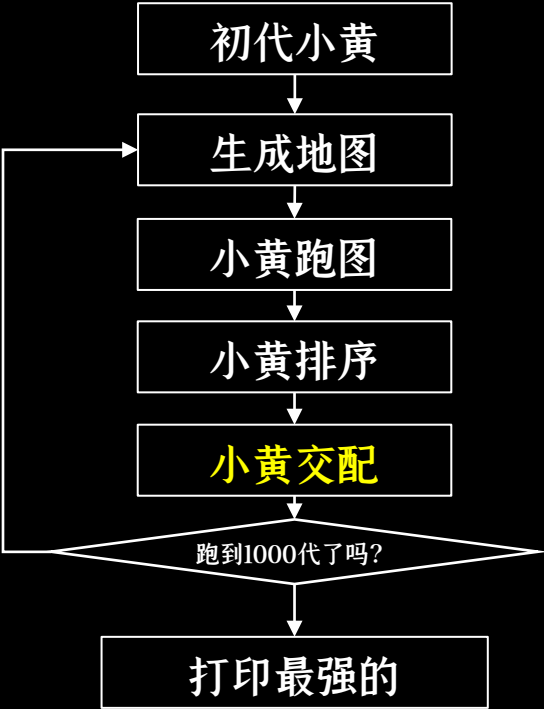
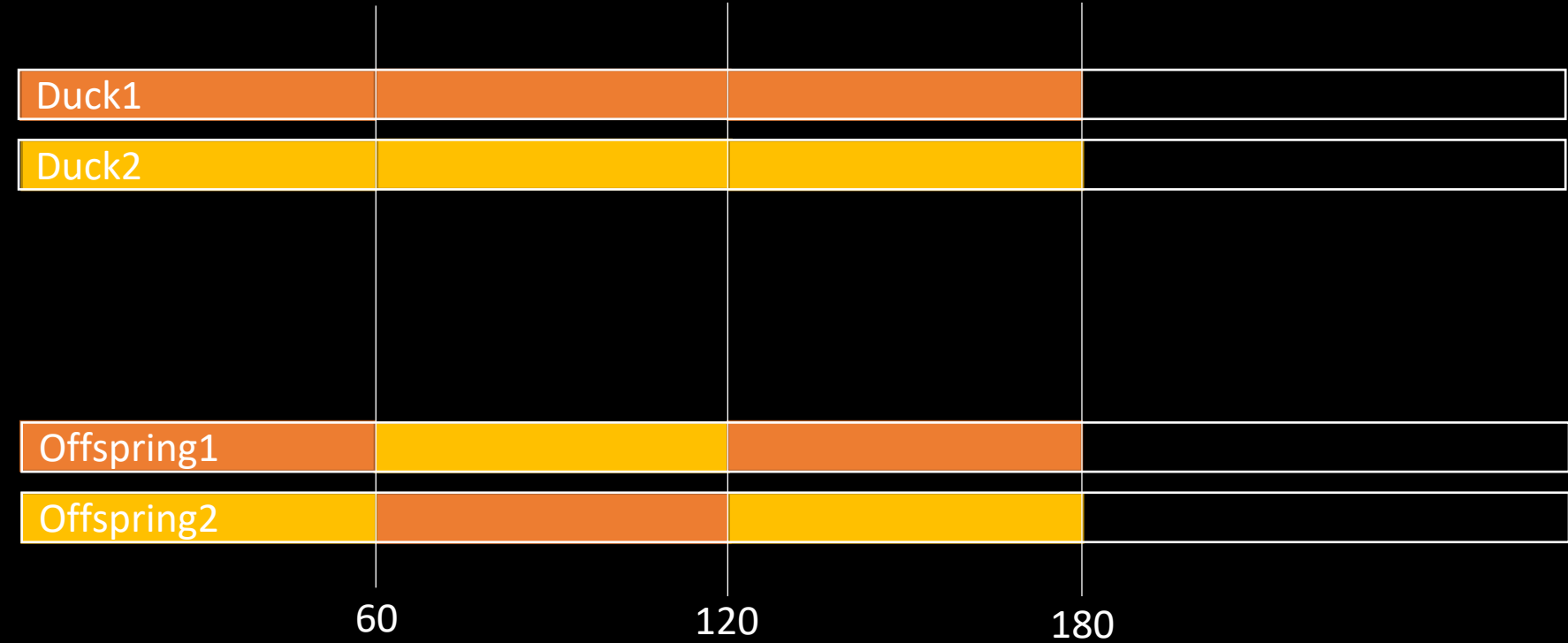
交配：利用if-else语句选定对于每一部分要选用Duck1还是2的基因片段



# 解题思路

|小黄羞羞 ( ? )      #define SLICING 60

交配：利用if-else语句选定对于每一部分要选用Duck1还是2的基因片段

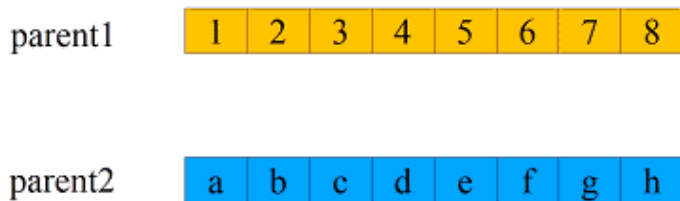


# 解题思路

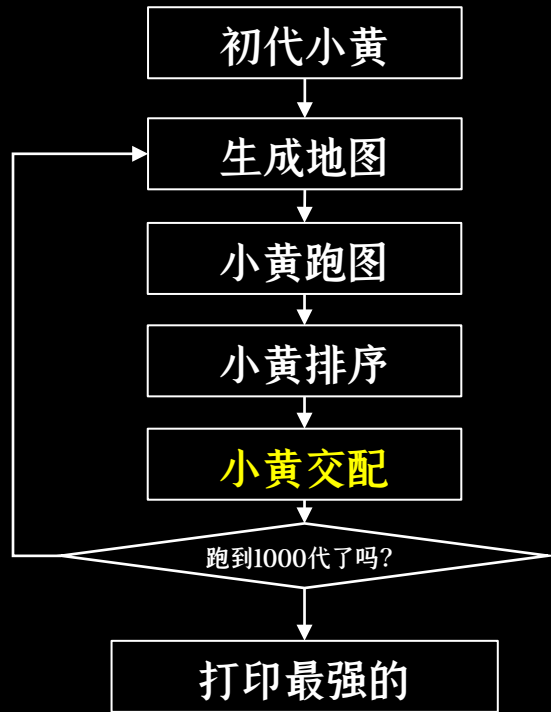
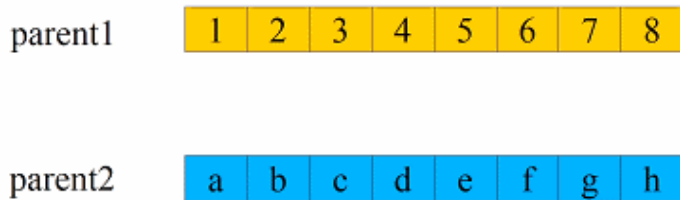
|小黄羞羞 ( ? )

不同的交配方法

均匀交叉 (Uniform crossover)



多点交叉 (Multi-point crossover)



# 解题思路

|小黄羞羞 ( ? )

#define SLICING 60

交配完毕: 使用新的数组覆盖原数组

方法:

- ①用循环依次赋值
- ②内存操作memcpy

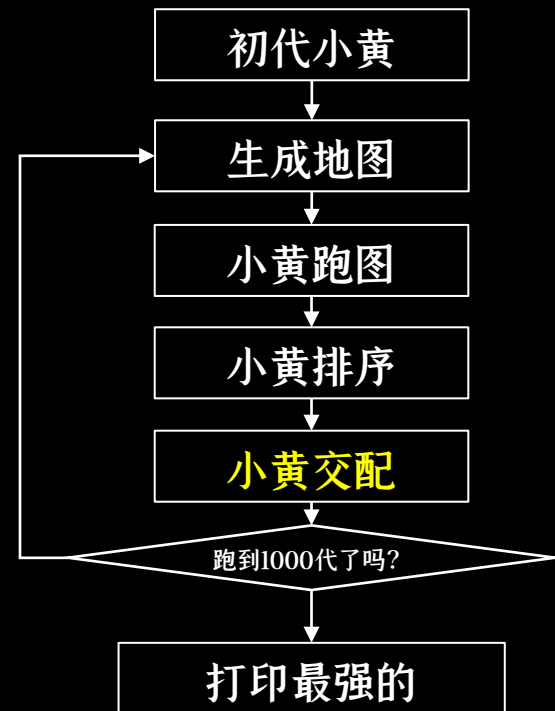
```
void *memcpy(void *str1, const void *str2, size_t n)
```

str1 -- 指向用于存储复制内容的目标数组, 类型强制转换为 void\* 指针。

str2 -- 指向要复制的数据源, 类型强制转换为 void\* 指针。

n -- 要被复制的字节数。

<https://www.runoob.com/cprogramming/c-function-memcpy.html>

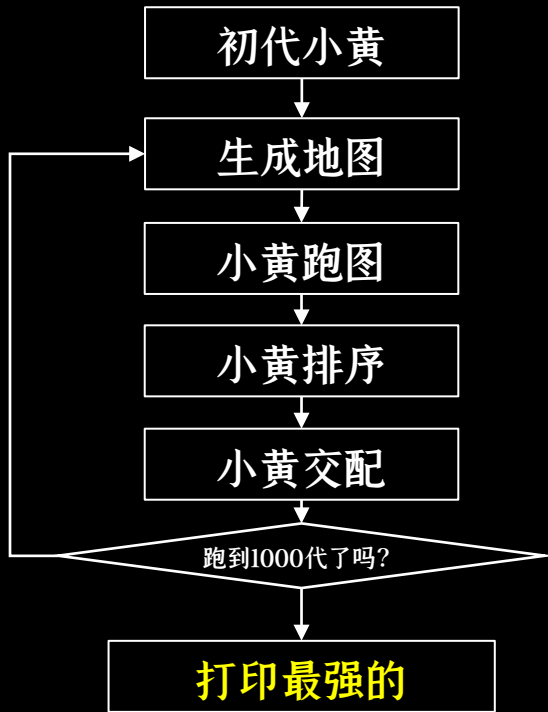


# 解题思路

| 养兵千日用兵一时

循环输出最高分数的结果。

- ①直接输出最新一代排完序的第一个。（省事）
- ②在每次EvalSingleChick的时候，都存下刷新高分记录的小黄的基因表。



# 解题思路

| 养兵千日用兵一时

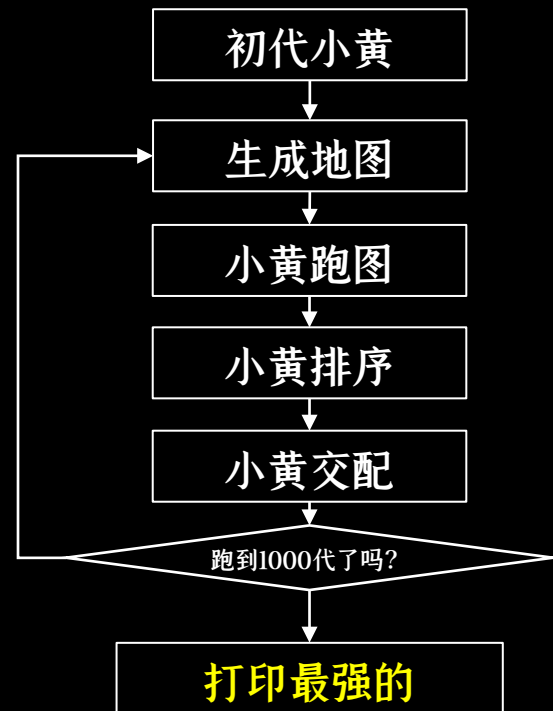
把结果保存到文件中以供其他程序读取（你可能需要）

文件指针

```
FILE *fp = NULL;
```

打开文件

```
fp = fopen("test.txt", "w+");
```



# 解题思路

| 养兵千日用兵一时

把结果保存到文件中以供其他程序读取（你可能需要）

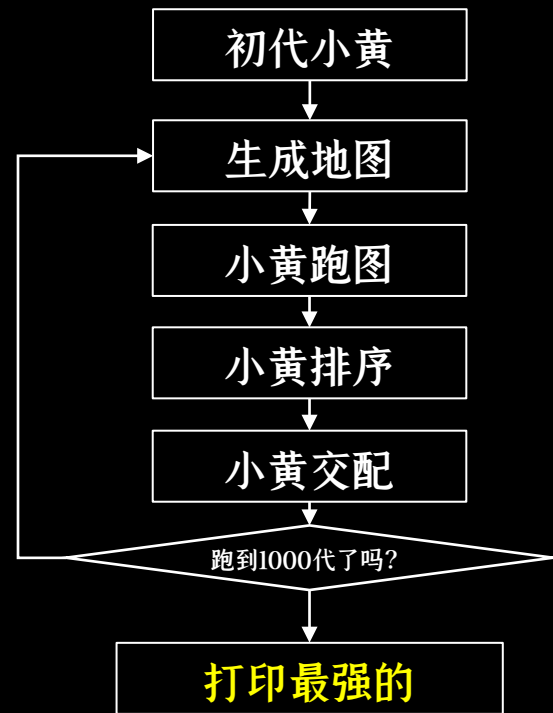
文件指针

```
FILE *fp = NULL;
```

打开文件

```
fp = fopen("test.txt", "w+");
```

文件路径，如果不是完整路径，就以程序运行目录为根目录



# 解题思路

| 养兵千日用兵一时

把结果保存到文件中以供其他程序读取（你可能需要）

文件指针

```
FILE *fp = NULL;
```

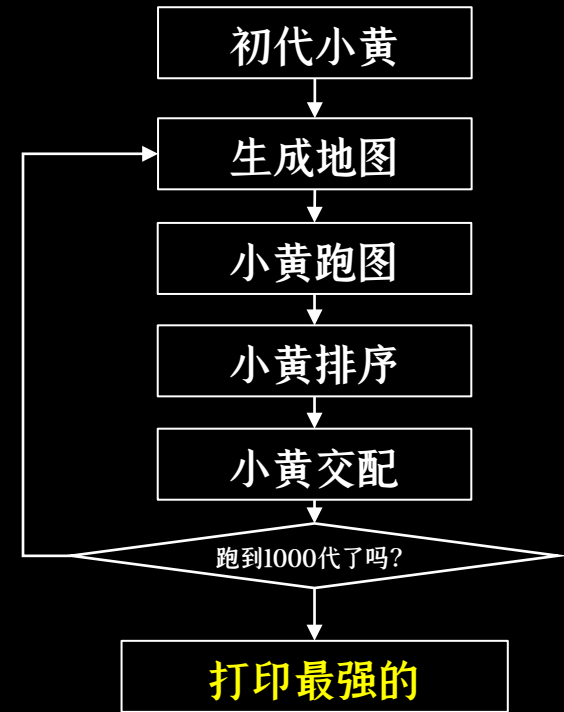
打开文件

```
fp = fopen("test.txt", "w+");
```

"r"	打开一个用于读取的文件。该文件必须存在。
"w"	创建一个用于写入的空文件。如果文件名称与已存在的文件相同，则会删除已有文件的内容，文件被视为一个新的空文件。
"a"	追加到一个文件。写操作向文件末尾追加数据。如果文件不存在，则创建文件。
"r+"	打开一个用于更新的文件，可读取也可写入。该文件必须存在。
"w+"	创建一个用于读写的空文件。
"a+"	打开一个用于读取和追加的文件。

Params

filename 文件路径，如果不是完整路径，就以程序运行目录为根目录  
mode -- 字符串，表示文件的访问模式





# 解题思路

| 养兵千日用兵一时

把结果保存到文件中以供其他程序读取（你可能需要）

文件指针

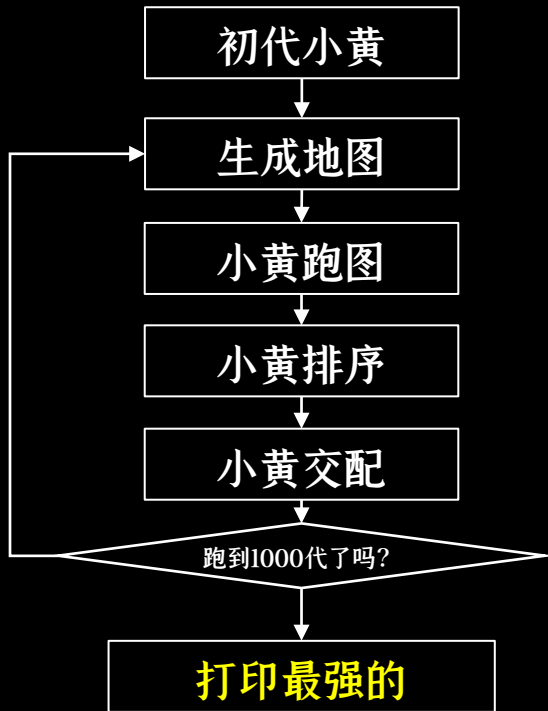
```
FILE *fp = NULL;
```

写入文件

```
fprintf(fp,.....);
```

关闭文件

```
fclose(fp);
```

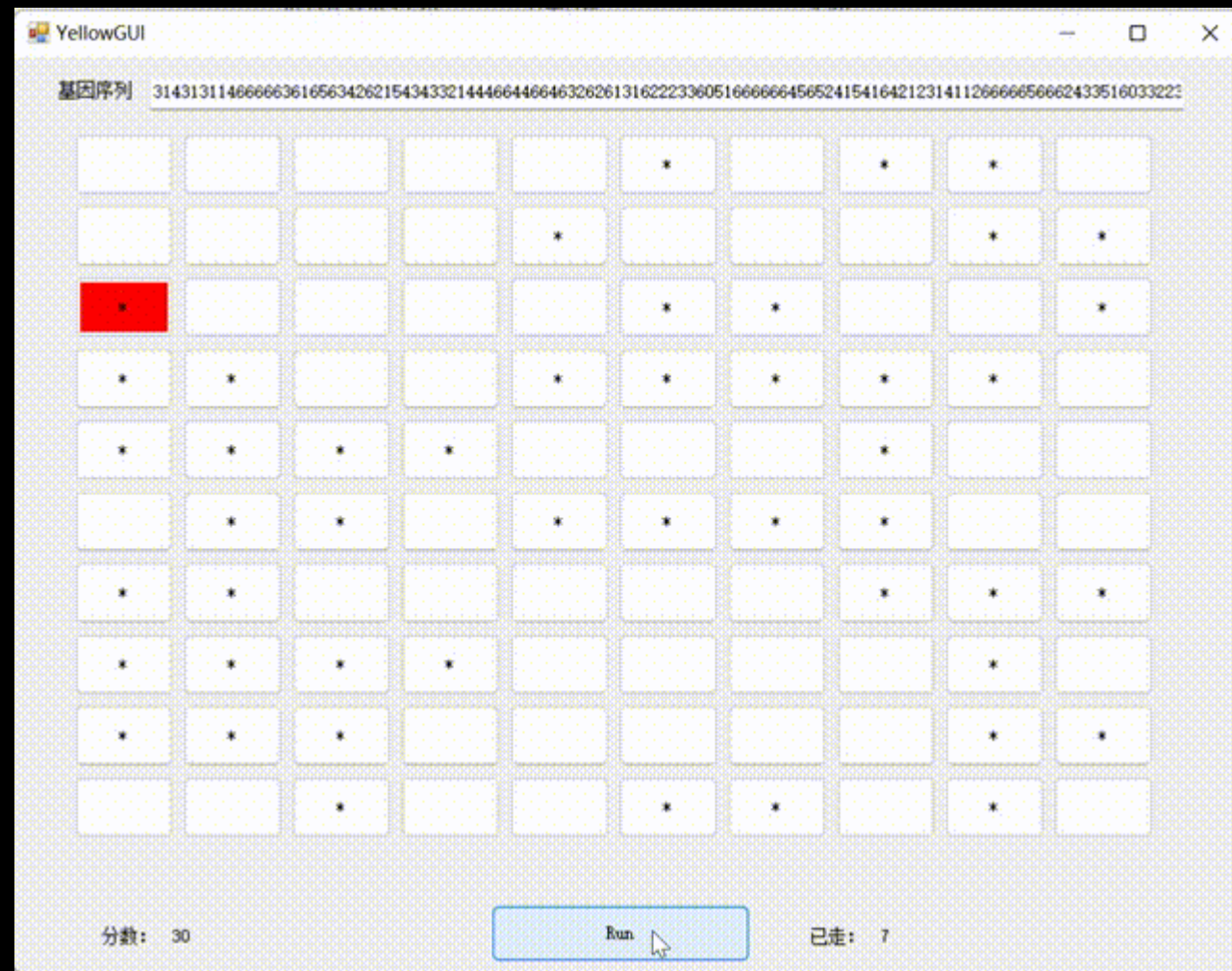


# 一些Bonus

|GUI—使用一些强有力的工具



obj-C+XIB by左老师

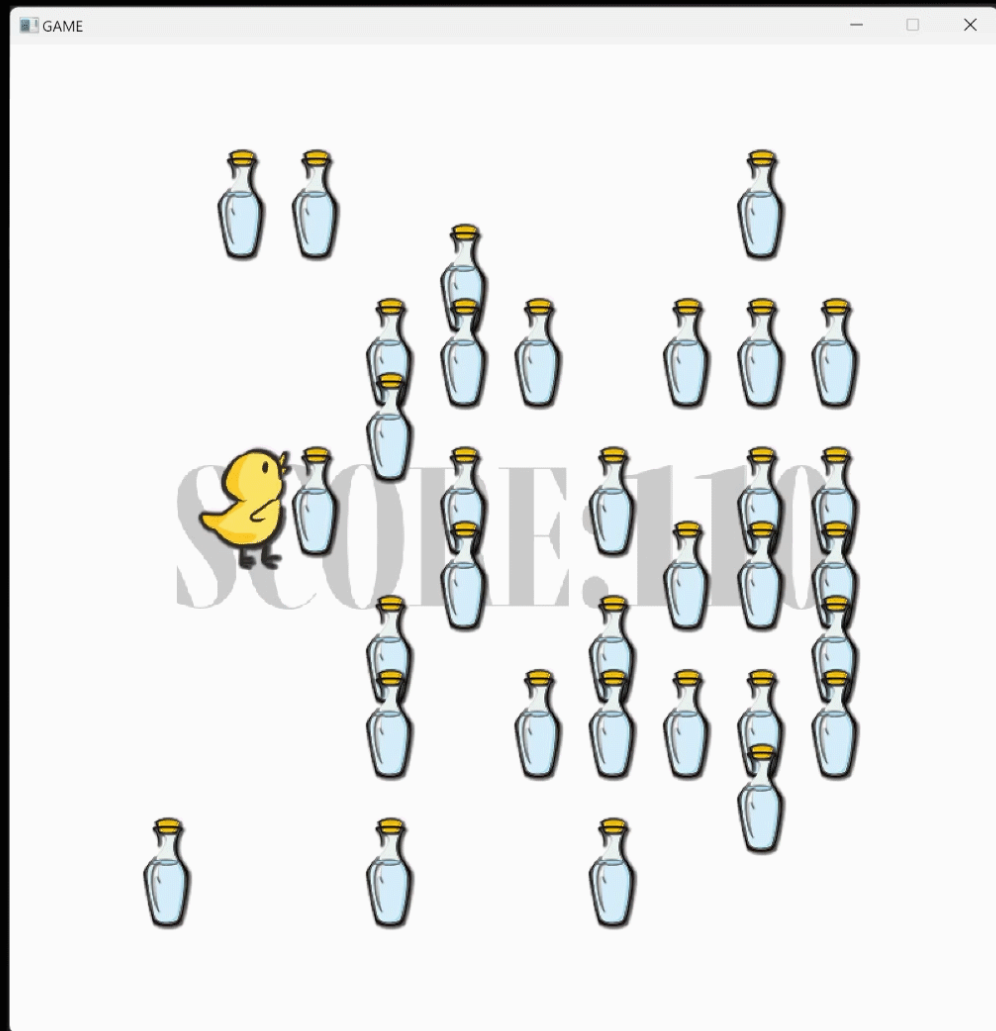


.NET Framework

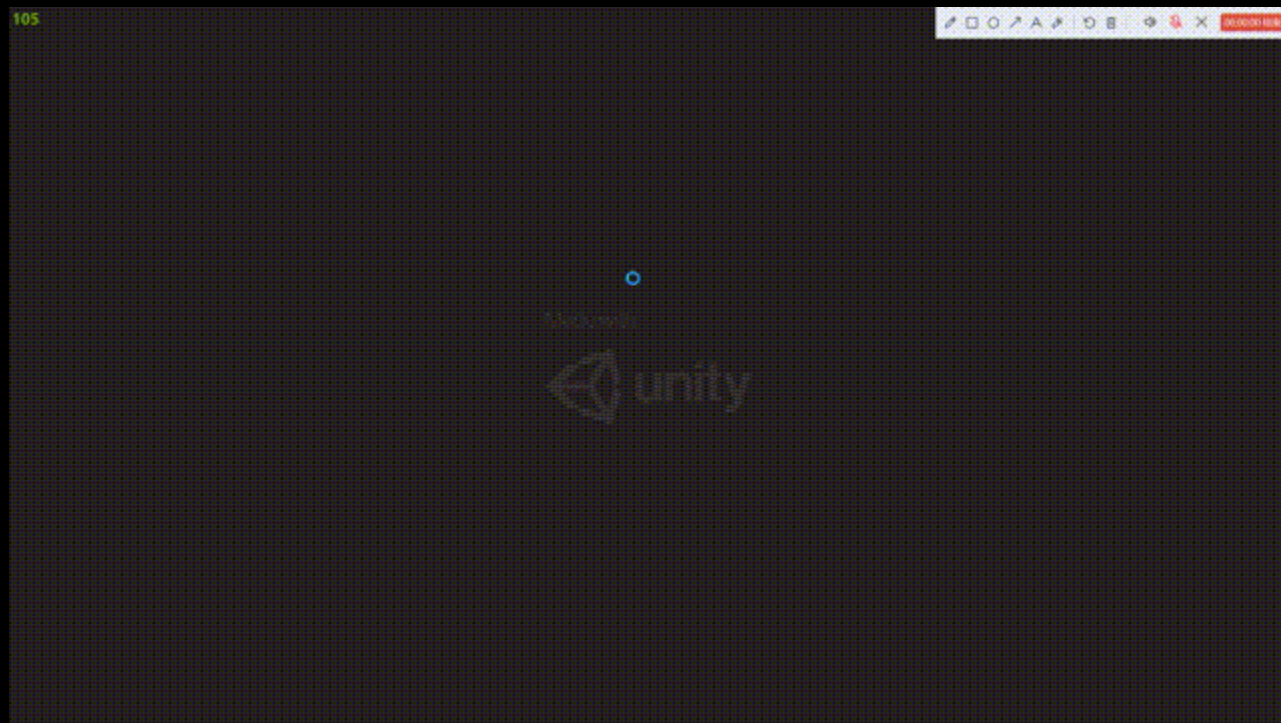
方块小黄润

# 一些Bonus

GUI



SDL GUI by黄宸一

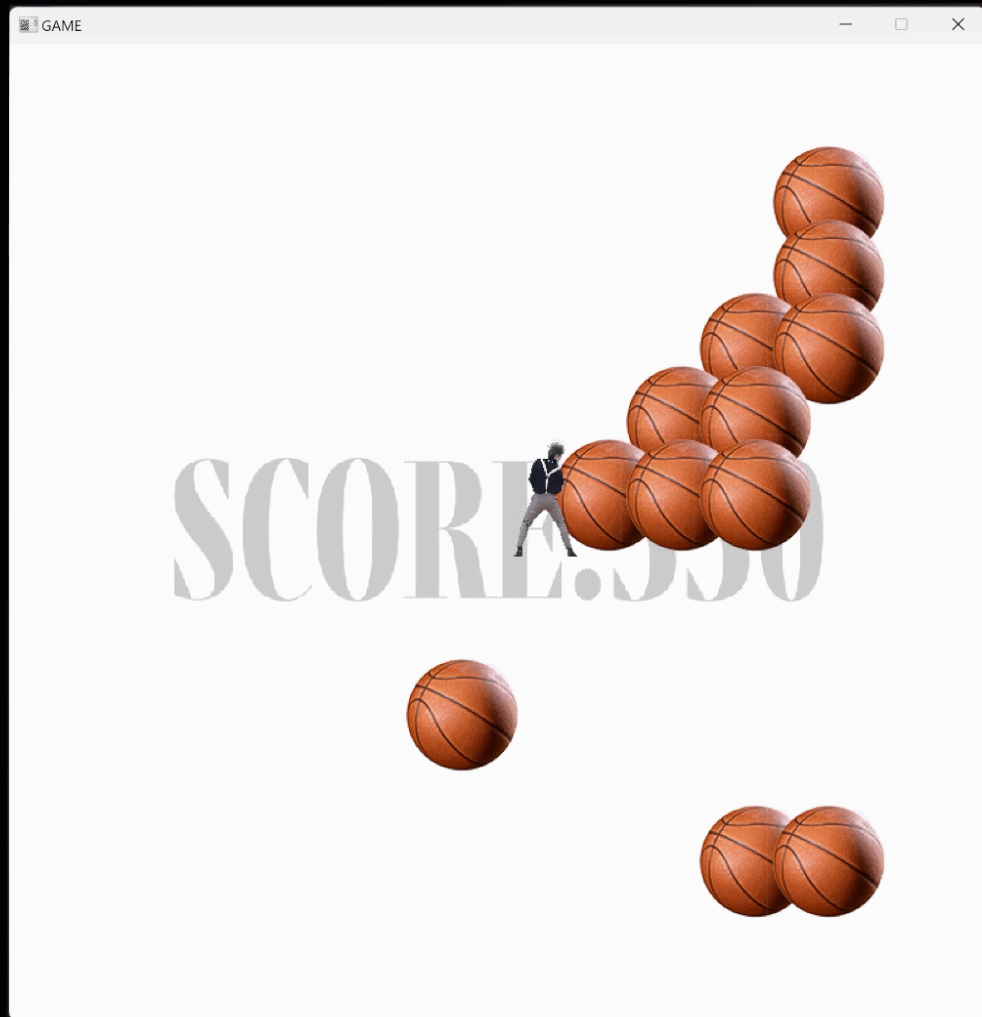


Unity 3D UGUI

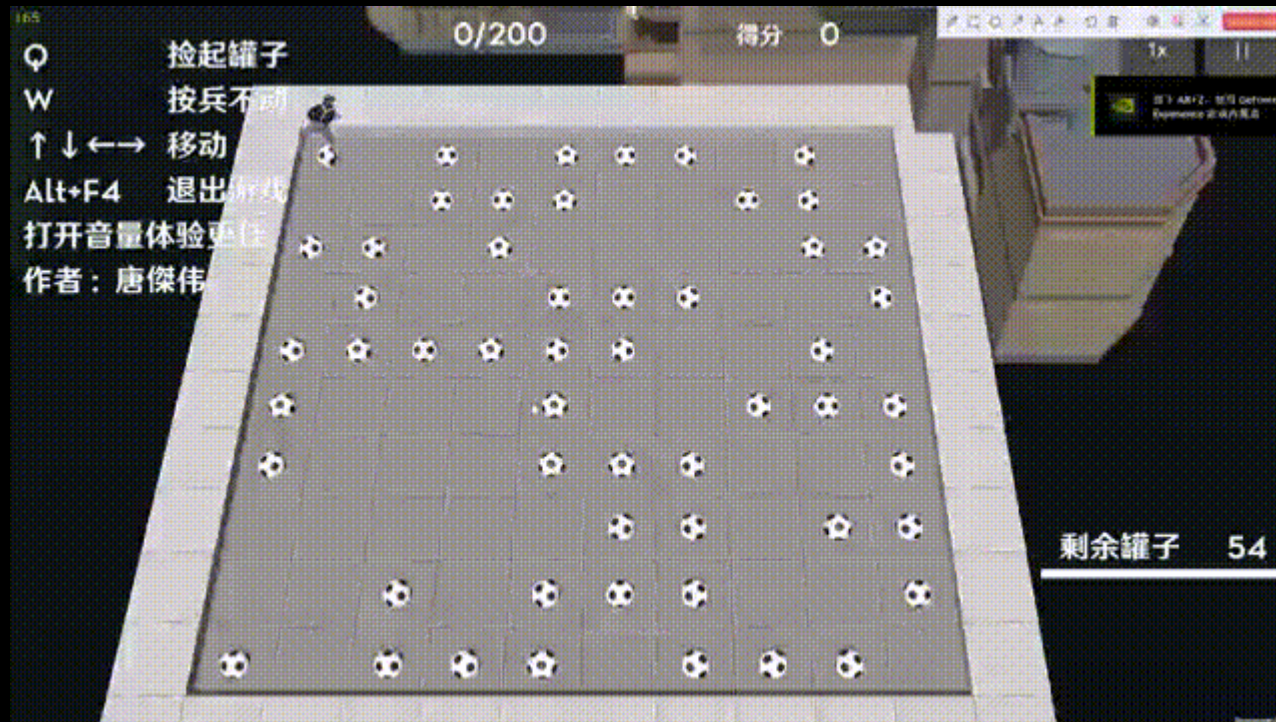
尝试一些炫酷的操作.....

# 一些Bonus

## GUI



SDL GUI by黄宸一



Unity 3D UGUI

或者.....



chatGPT





# chatGPT



## 你能用法语给我写一首情诗吗



是的，我可以尝试为您写一首情诗。

Voici un poème d'amour

Que j'ai écrit pour toi

Avec des mots simples

Et mon coeur qui bat

Tu es ma raison de vivre

Mon soleil, ma lumière

Tu es ma passion, mon amour

Et je t'aimerai toujours

Tu es mon univers

Mon tout, mon étoile

Je veux passer ma vie

À tes côtés, mon amour

Je t'aime de tout mon coeur

Et je te le prouverai

Chaque jour, chaque heure

Mon amour pour toi ne mourra jamais

这是一首我为你写的情诗，用简单的语言表达我内心深处的感受。你是我生活的意义，是我的阳光，是我的热情和爱。我会永远爱你，你是我的一切，是我的明星。我想用我的一生来陪伴你，我全心全意地爱你，并会每天每小时都向你证明这份永不消逝的爱。

MI

github链接可以吗



是的，如果你有一个 GitHub 项目，并希望我帮助解决问题，可以发送项目的 GitHub 链接给我。这样

# 杀疯了！

HX

你的代码报错了，错误信息是22:56: error: '>>' should be '> >' within a nested template argument list

```
priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>>> q;
```



这是因为 C++ 中的输入输出流运算符 `>>` 和模板参数分隔符 `>` 在某些情况下可能会产生歧义，因此需要使用空格将其区分开来。在本代码中，我们在定义优先队列时需要使用模板参数，所以需要在模板参数分隔符 `>` 的两侧各加一个空格。

为了修复这个错误，我们可以在定义优先队列时将模板参数分隔符 `>` 两侧各加一个空格：

Copy code

```
priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int> > > q;
```

修改后的代码应该能够正常编译和运行了。



# chatGPT

## •In-Context Learning之谜

为什么GPT-3会有很强的In-Context Learning能力？而其他模型有一定差距。是不是OpenAI的工程师在训练时无意中使用了某种数据排序？或者初始化种子恰好在某个特殊的取值？或者训练样本里刚好有个不经意的样本？

## •Chain-of-Thought之谜

思维链的能力来自于哪里？初代GPT-3应该是没有的，在加入code预训练之后无意中触发了思维链的能力。为什么Code预训练会使得GPT的推理能力变强？

## •Human-in-Loop之谜

一旦人在环路，这个迭代过程就不可复制了。某天某个标注员突然心情大好，写了个回答，被Reward model记住了，这个会被强化学习无限放大。

<https://www.zhihu.com/question/570782945/answer/2801550100>

# PJ2自救指南

|你一定可以的

试试每一代都挑几只小黄打印策略表!

1.是否出现了很长的平台期?

-- 分数一直保持在一个分数几乎完全不变, 还是分数在一个分数间波动?

生成策略表或者交叉算法有问题

多试试!

-- 分数超过了500分?

得分判定或结束判定可能有问题

2.程序根本跑不起来?

-- 试试在程序四处打上断点 (或printf) 看看断点能否被触发或是有无输出消息。

注意一些全局变量是无视代码顺序的。

3.读取策略表不能?

-- 检查生成的文件的编码

4.代码跑一遍要好久?

-- 检查你的算法, 是否将一些计算写错了位置? (例如, 每跑完一只小黄都计算一遍平均分或是排序)





# 调整随机数

```
#define STAY 5
```

```
314313114666663666564461245444312444666446646216613333232332051666666266264414164  
212314114666663666245150104332413554426666236653143015212313231666624532041533105  
104343551664361402310124155412412330604662162330633111312304641600605552565042604
```

# 调整随机数



# 调整随机数

55                      5 5                      5                      5                      5                      5 5 5                      5 5                      5

314313114666663666564461245444312444666446646216613333232332051666666266264414164  
212314114666663666245150104332413554426666236653143015212313231666624532041533105  
104343551664361402310124155412412330604662162330633111312304641600605552565042604

# 调整随机数

555555555555555555

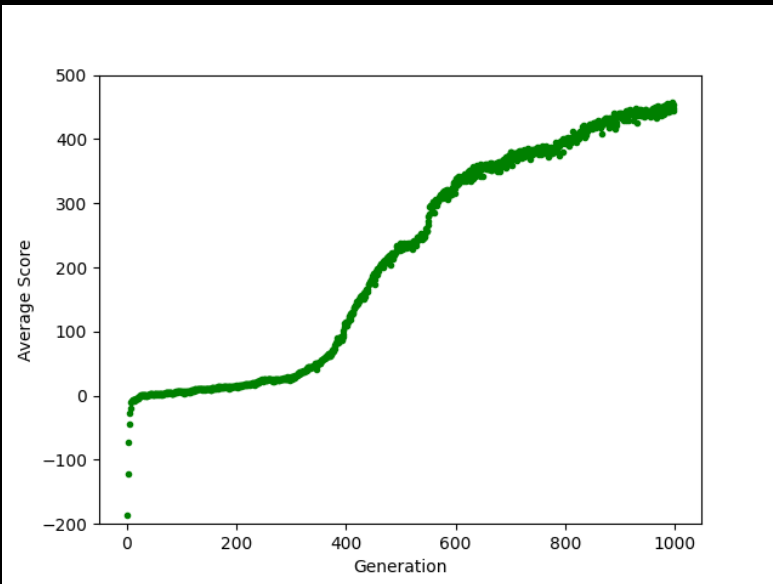
314313114666663666564461245444312444666446646216613333232332051666666266264414164  
212314114666663666245150104332413554426666236653143015212313231666624532041533105  
104343551664361402310124155412412330604662162330633111312304641600605552565042604

# 调整随机数

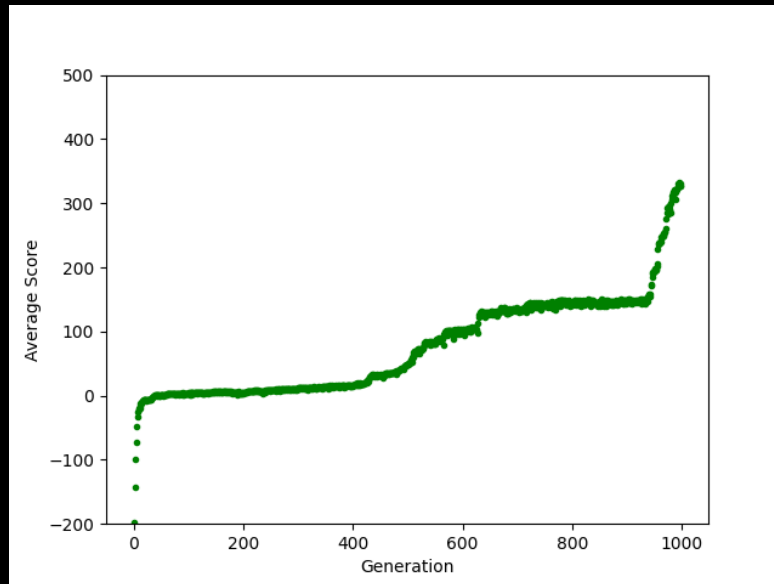
314313114666663666564461245444312444666446646216613333232332051666666266264414164  
212314114666663666245150104332413554426666236653143015212313231666624532041533105  
104343551664361402310124155412412330604662162330633111312304641600605552565042604

字符	出现次数
0	18
1	39
2	28
3	38
4	40
5	21
6	59

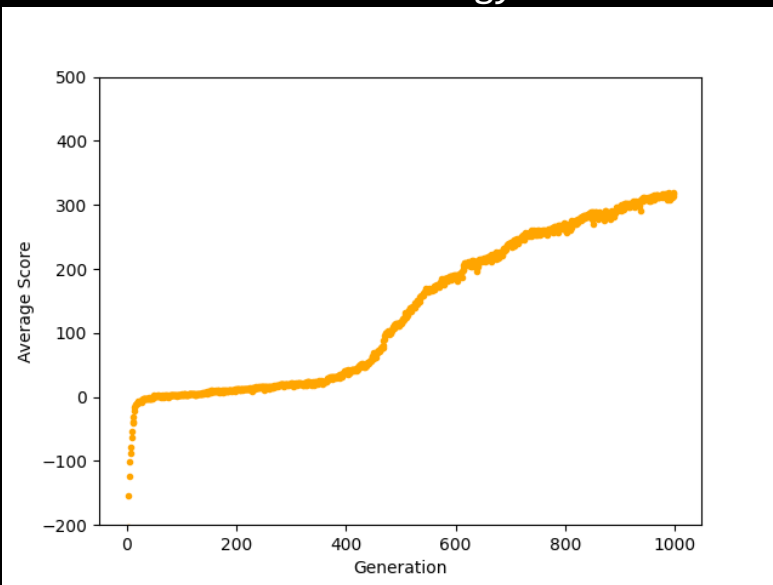
# 调整随机数 | 随机算法



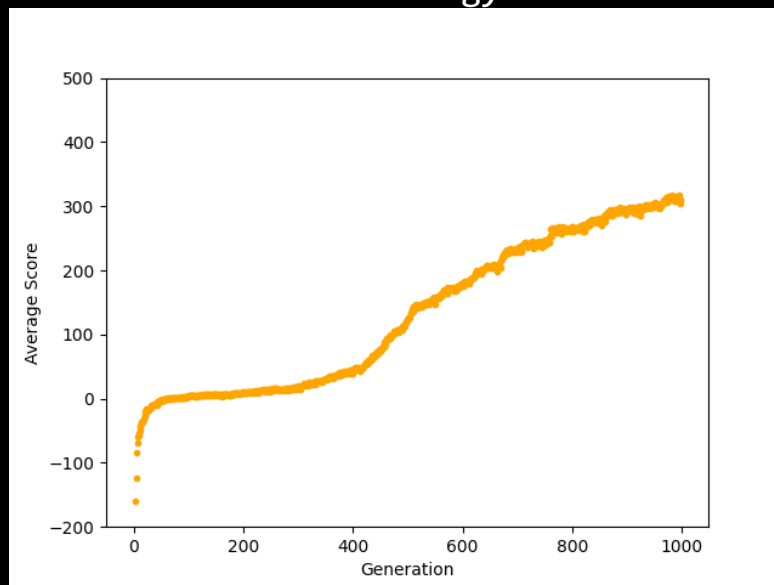
Seed1 Without Strategy Preference



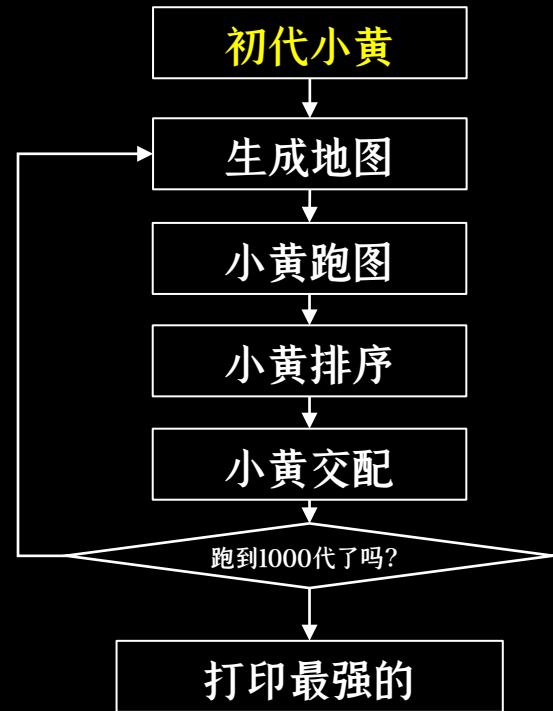
Seed2 Without Strategy Preference



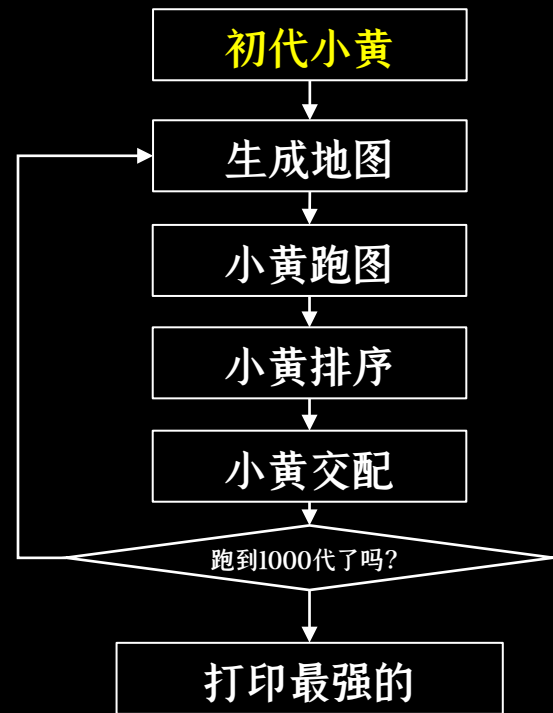
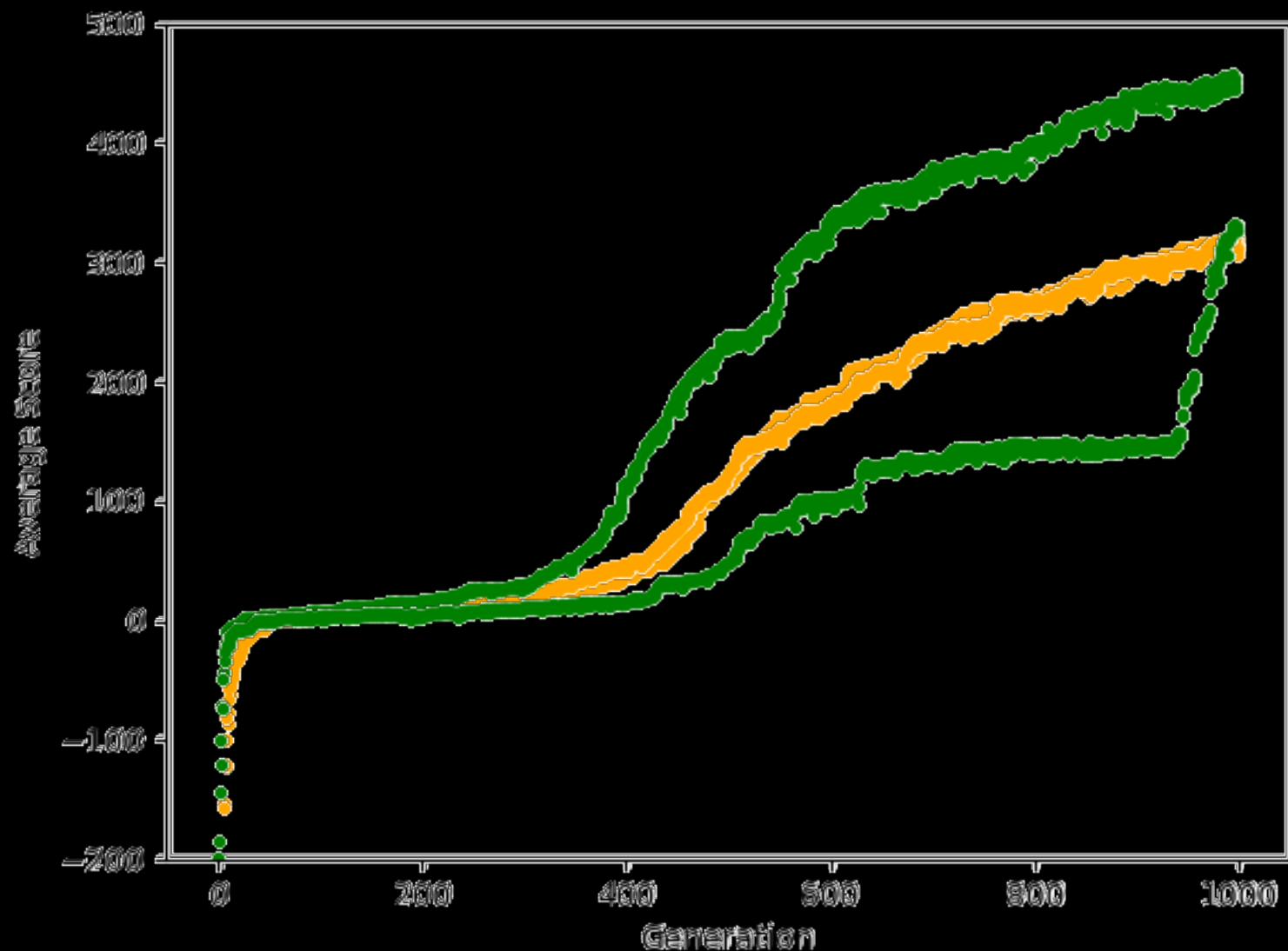
Seed1 With Strategy Preference



Seed2 With Strategy Preference

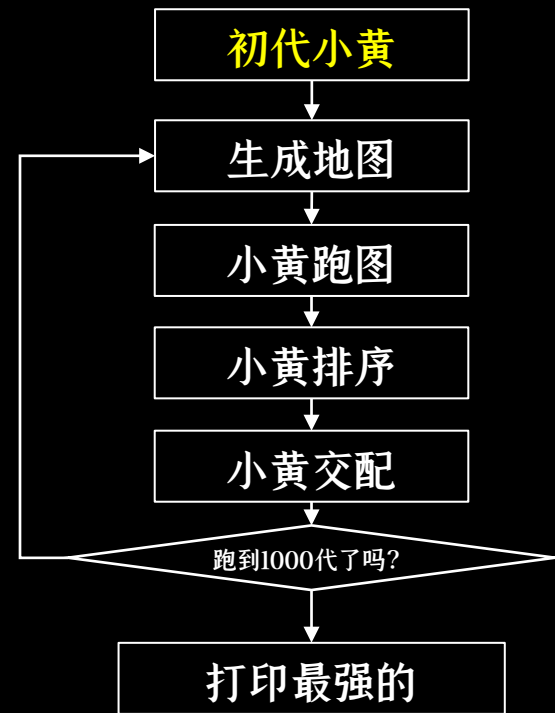
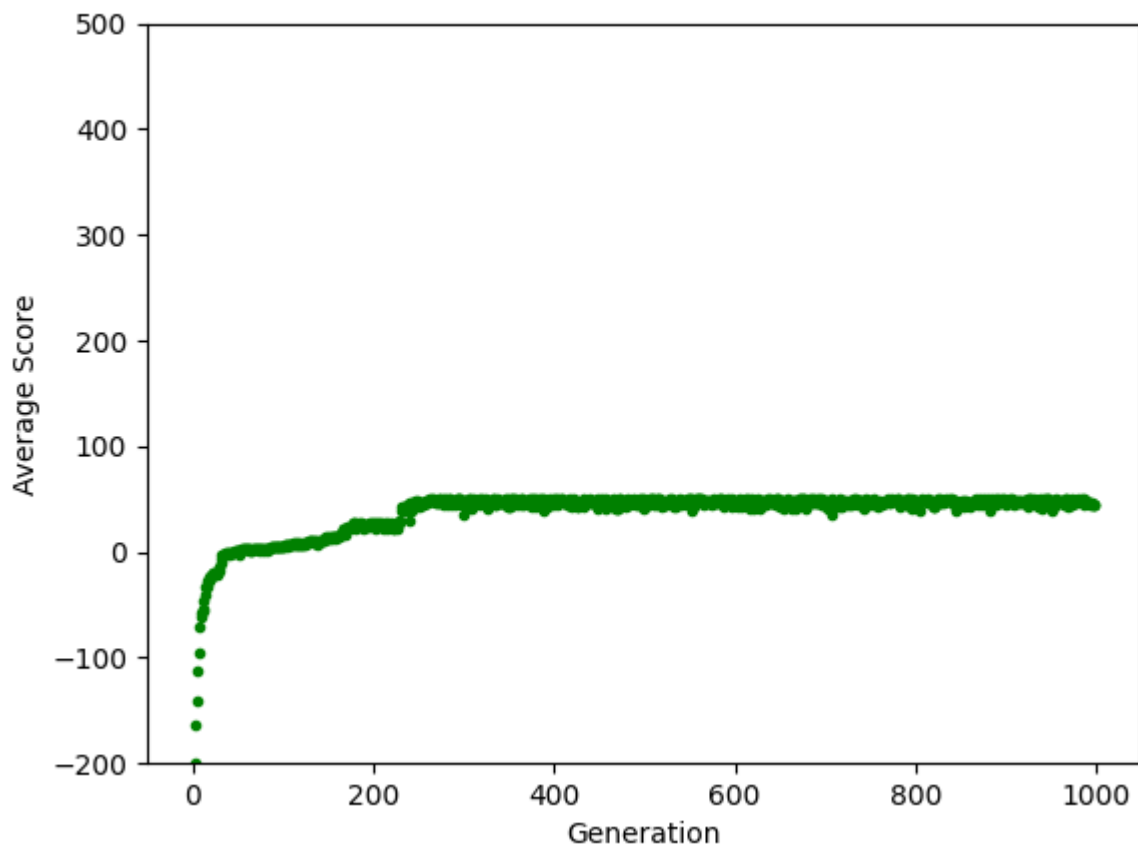


# 调整随机数 | 随机算法



# 调整随机数 | 随机算法

当然，很多时候你跑出来的很可能是这样的.....





# 调整参数

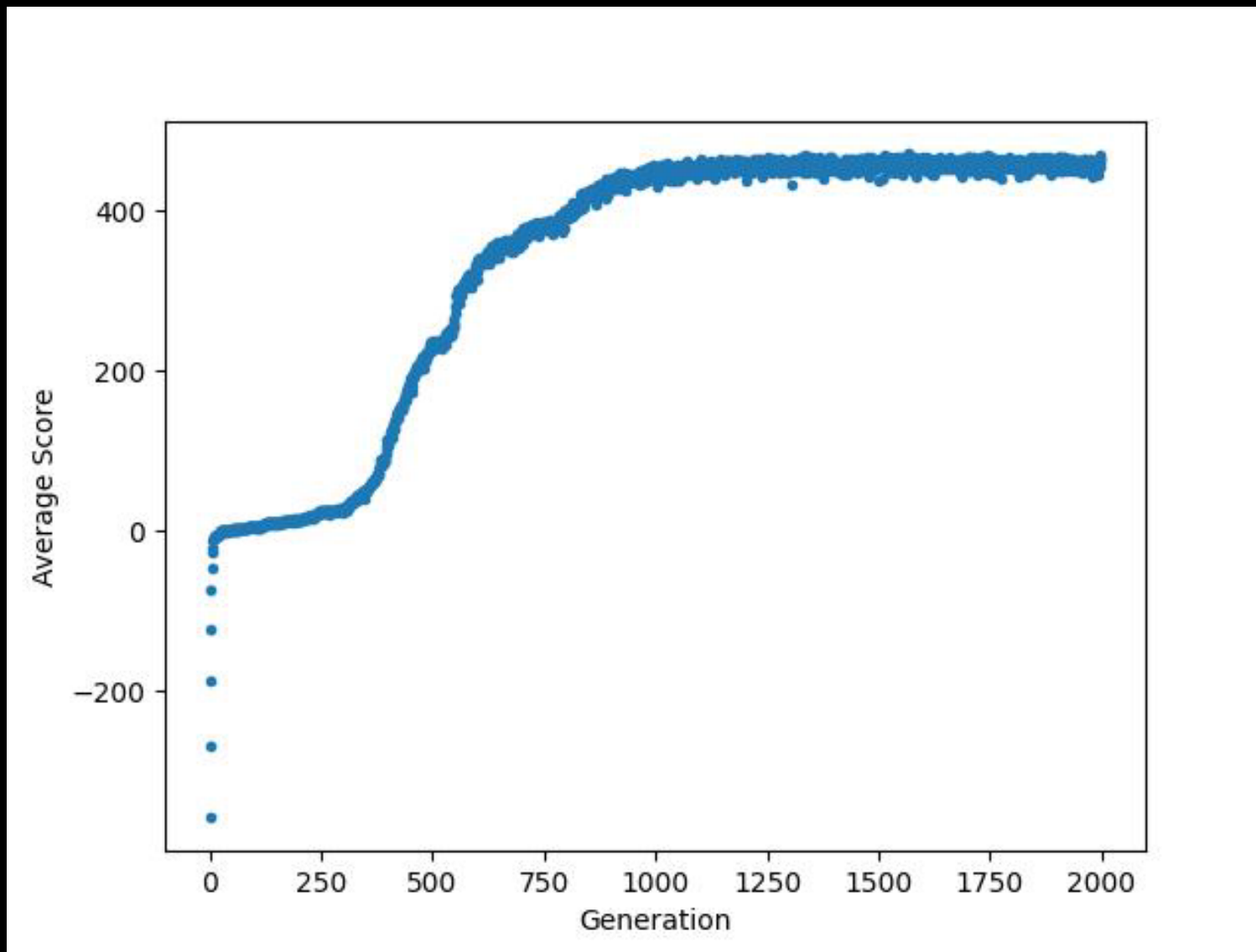
| 超级参数（控制变量：单次随机数种子一致，多种子平均）

超参数是在开始学习过程之前设置值的参数，而不是通过训练得到的参数数据。通常情况下，需要对超参数进行优化，给学习机选择一组最优超参数，以提高学习的性能和效果。

```
#define BATCH_SIZE 200           //一批孵多少只小黄
#define MAP_VOLUME 100           //一批孵多少只小黄
#define ITERATIONS 1000          //孵多少批小黄
#define SLICING 60                //基因重组片段大小
#define INVERSED_MUTATION_CHANCE 1000 //突变概率倒数
```

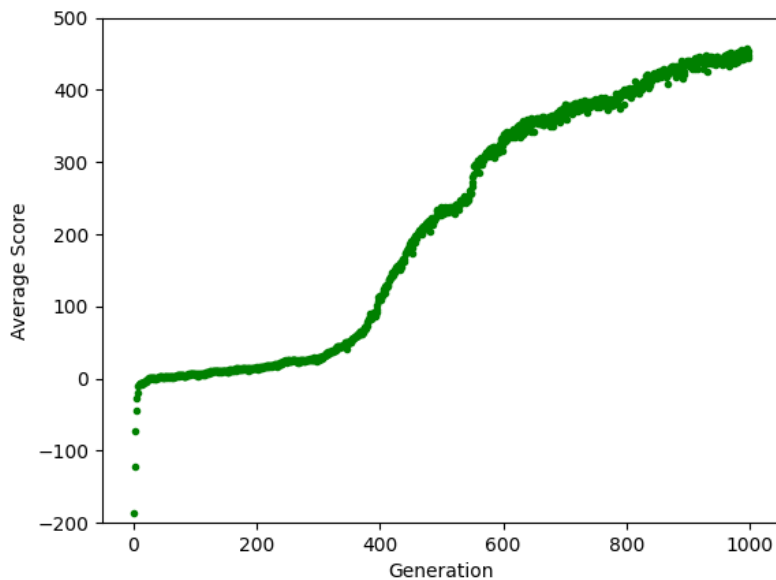
# 调整参数：Generation

| 超级参数（控制变量：单次随机数种子一致，多种子平均）

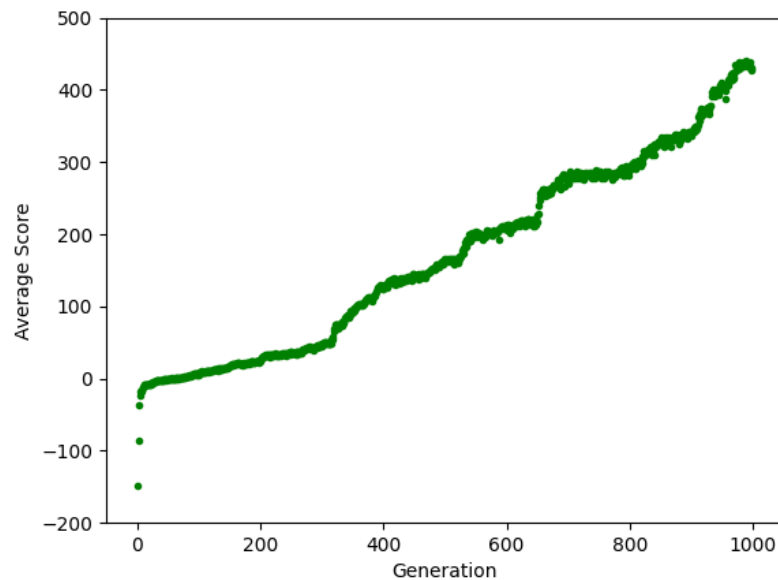


# 调整参数: BATCH\_SIZE

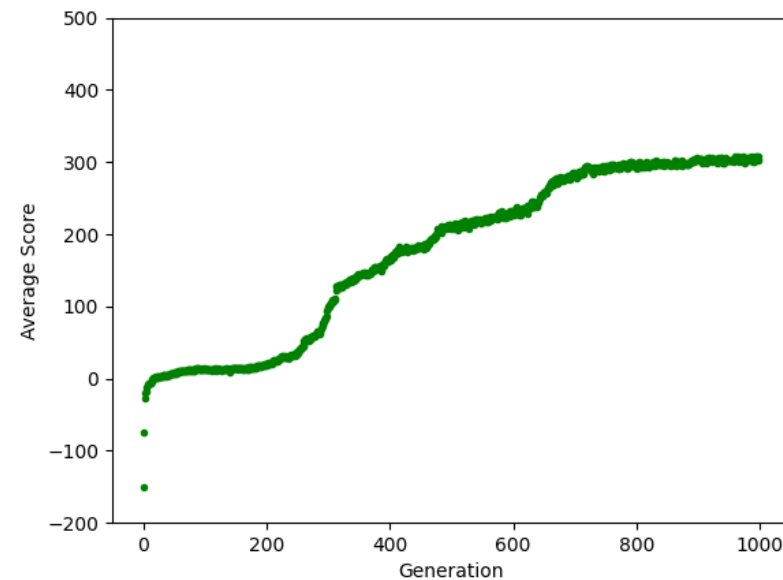
| 超级参数 (控制变量: 单次随机数种子一致, 多种子平均)



200



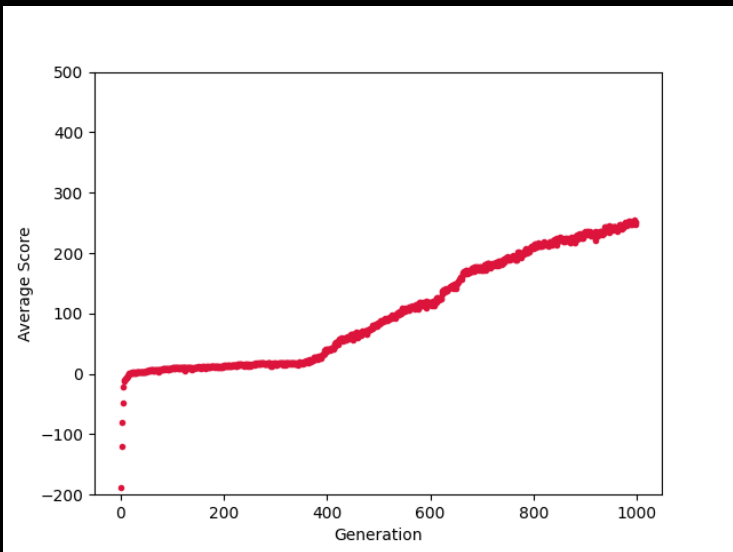
300



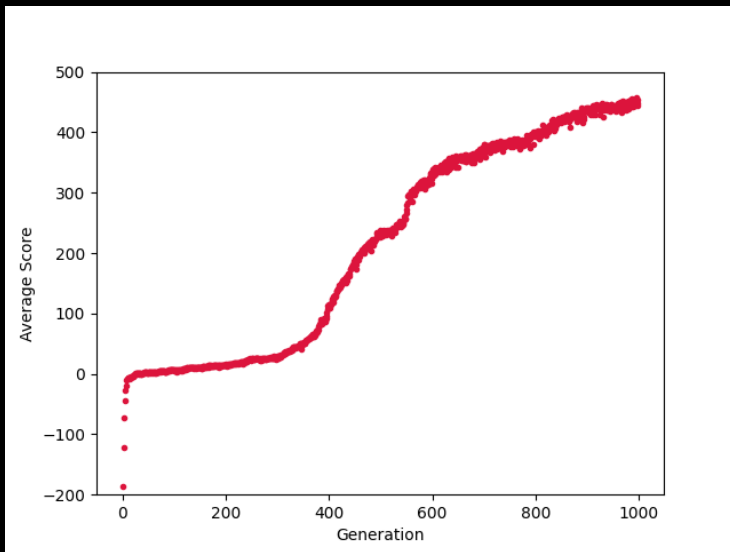
400

# 调整参数：变异率

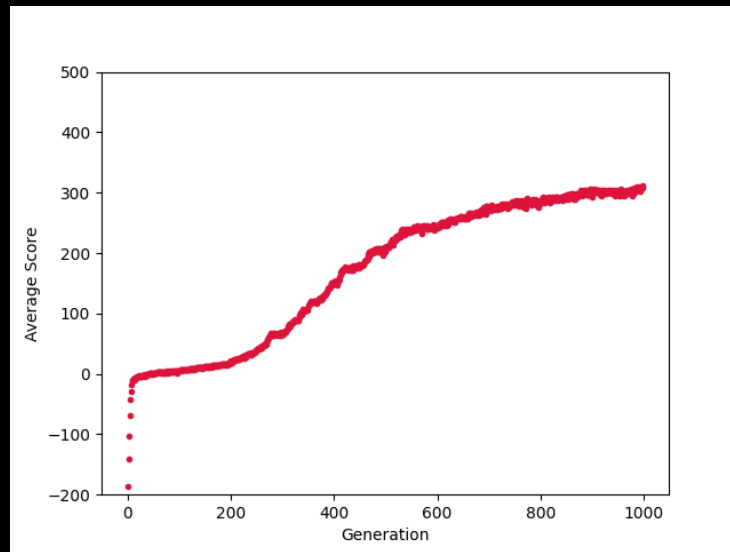
| 超级参数 (控制变量：单次随机数种子一致，多种子平均)



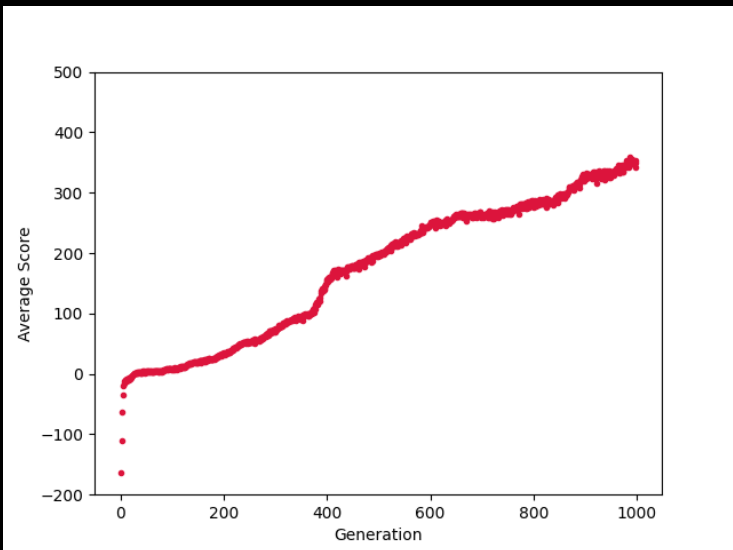
1/900



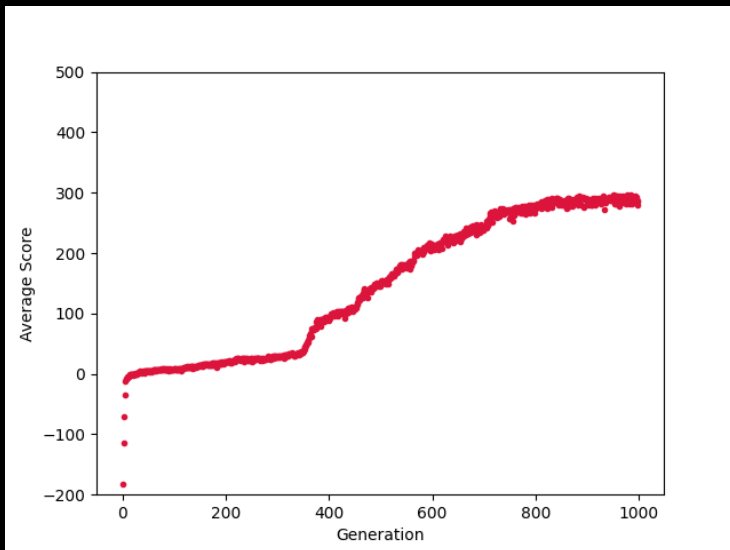
1/950



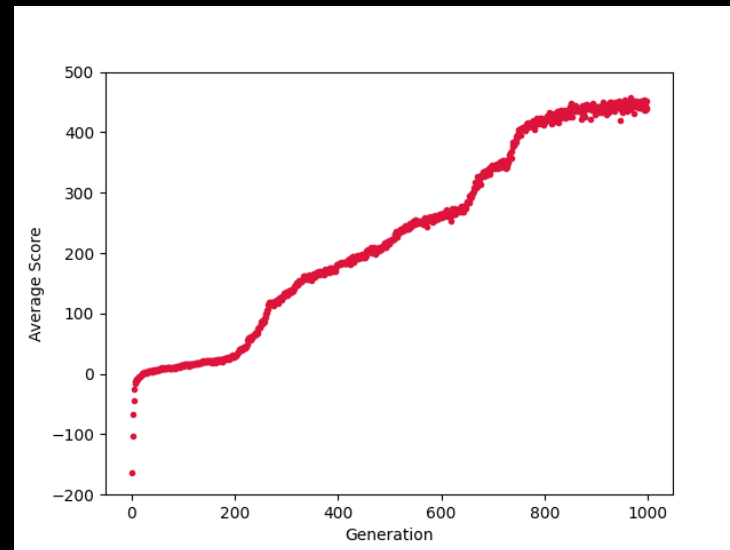
1/1000



1/750



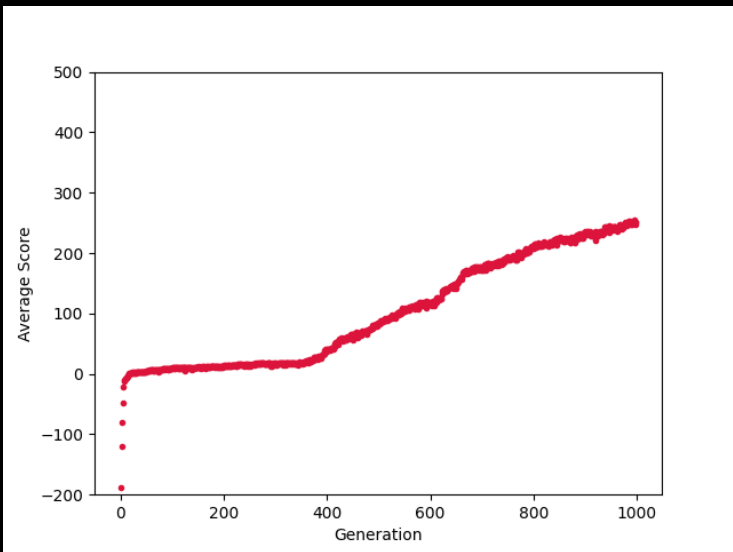
1/800



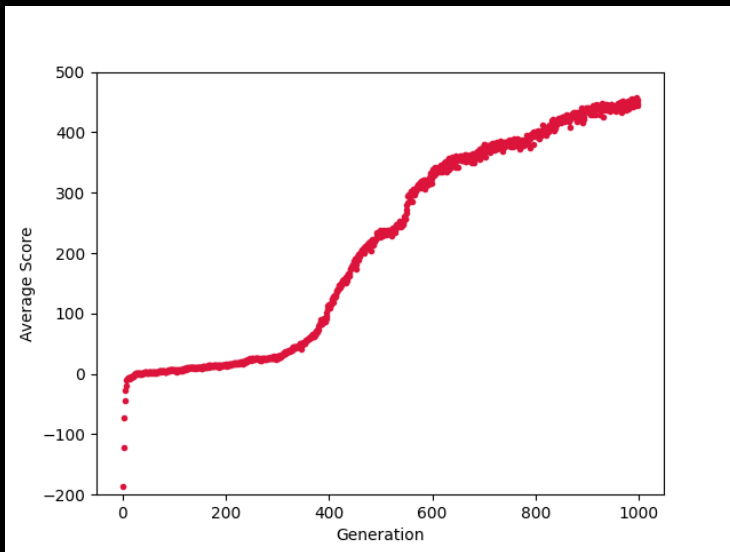
1/850

# 调整参数：变异率

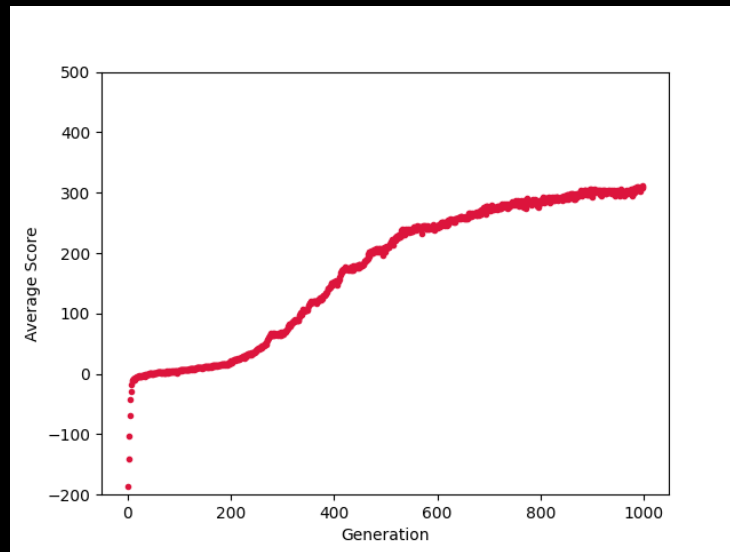
| 超级参数 (控制变量: 单次随机数种子一致, 多种子平均)



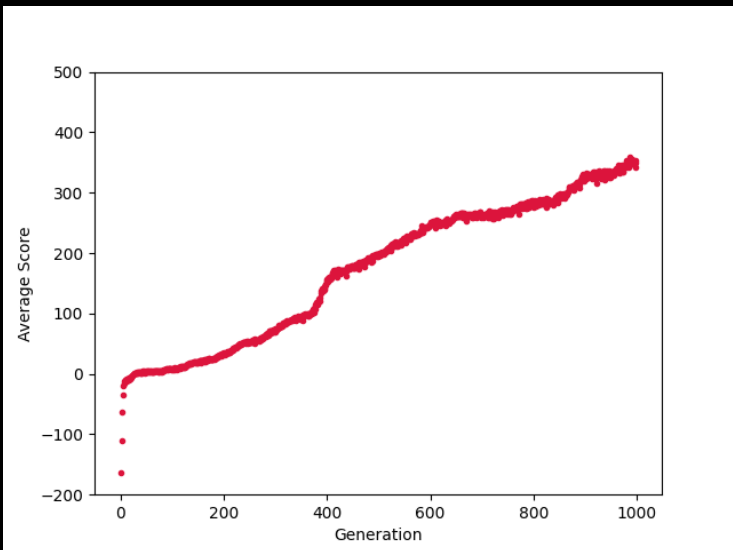
1/900



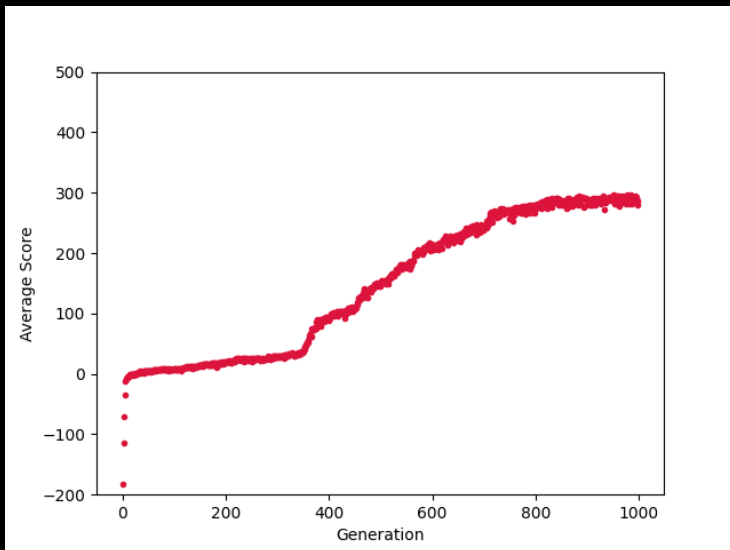
1/950



1/1000



1/750



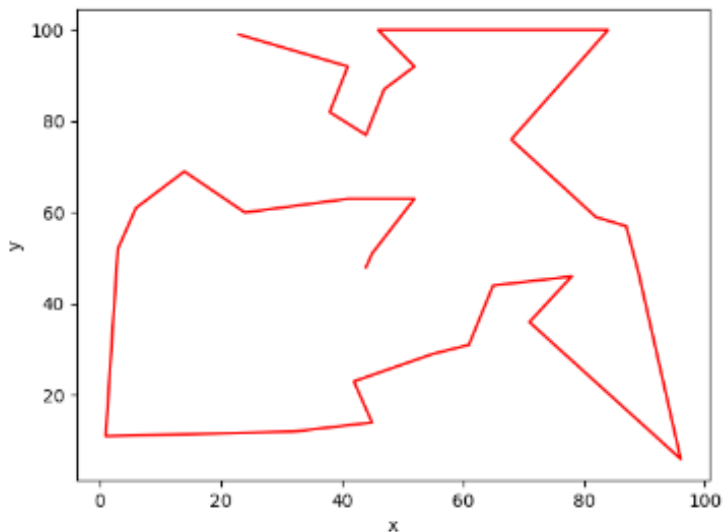
1/800

引入动态变异率...

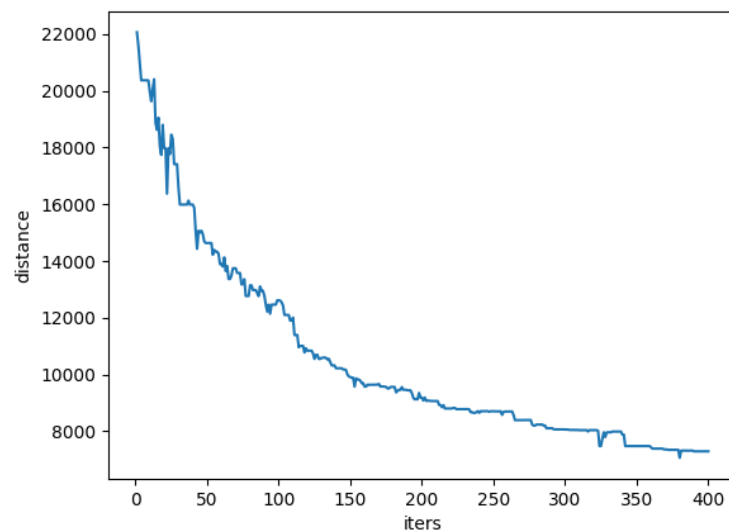
# 遗传算法的其它应用

## | TSP问题

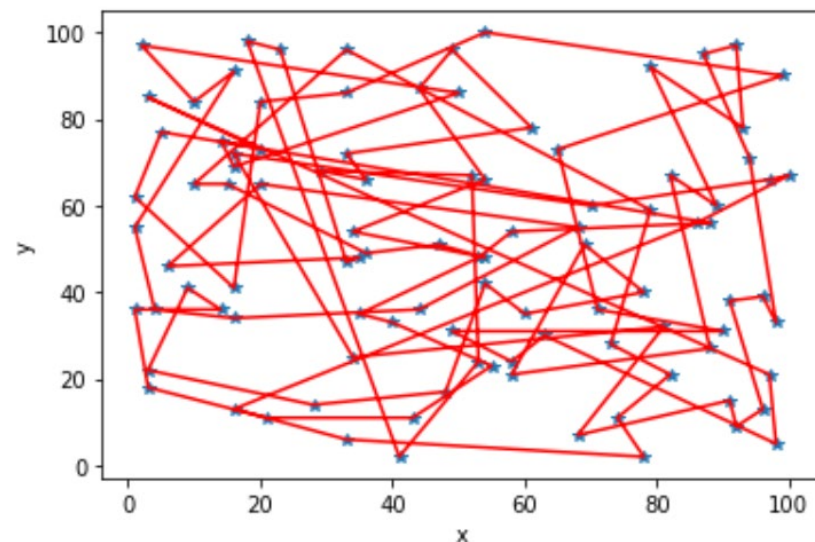
旅行商问题（travelling salesman problem, TSP）：给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路。



20dots



100dots 训练散点图

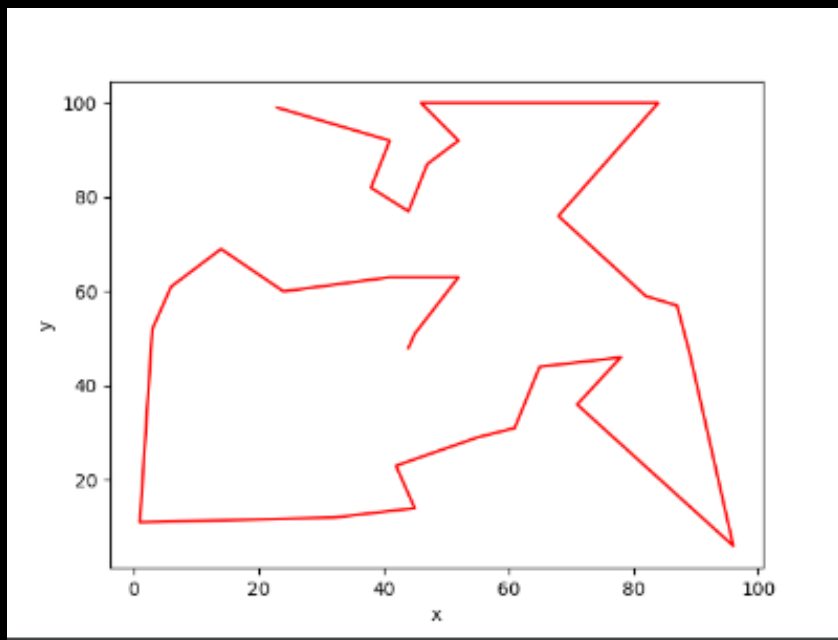


100dots

# 遗传算法的其它应用

## | TSP问题

旅行商问题（travelling salesman problem, TSP）：给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路。



20dots

部分匹配交叉（Partial-Mapped crossover, PMX）

parent1

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

parent2

3	5	8	1	7	4	2	6
---	---	---	---	---	---	---	---

无重复交叉算法



希望你和小黄玩的开心!

