Table of Contents

1. 22秋程序设计PJ:小黄和它的罐子 ——简单的遗传算法实例

1.1. 项目背景

在学习程序设计的基本语法之后，我们可以尝试使用c语言编写、设计一些小程序，我们设定一个角色和一个地图，设计角色的行为，执行我们输入的命令，让角色执行特定的动作。

在本次项目中，我们设想这样一个场景：

很久很久以前,有一群小黄快乐地生活在活动空间10 * 10，共100格活动空间的二维世界里，且世界的最外围有一层围墙包围着。 在100个格子里散布着一些装着宝物的罐子，小黄不知道罐子具体出现在哪里，但是小黄知道每个格子里会有50%的概率出现罐子。

小黄们想要探索世界，并在有限的步数里捡起尽量多的罐子。要注意，它们的视野范围很小，只能看自己所在的格子和周围上、下、左、右的格子，共五个格子；每个格子只有三种可能的状态:有罐子、空格子、墙。

1.1.1. 涉及知识点

- 变量与基本数据类型
- 表达式与语句、运算符
- 程序流程控制，条件判断与循环
- 函数声明与定义
- 数组与多维数组
- 指针与引用
- 读写文件

1.2. 项目要求

1.2.1. PJ1

1. 随机初始化的棋盘

为每一只小黄准备10*10的有围墙的棋盘,棋盘上每一个格子都有50%的概率有罐子.小黄的初始位置为左上角的格子,记为(1,1).游戏开始时你需要生成一张随机棋盘,棋盘满足以下条件

1. 棋盘边缘的四个边被围墙覆盖
2. 围墙以内每个格子有50%的概率出现罐子或空格子

2. 动作

小黄可以做的事包括:

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
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
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
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
6. 不移动
7. 捡罐子[1],如果小黄此时的位置上有罐子,则罐子被捡起,即此格子变为空

3. 得分

这些行为,对应了不同的得分

- 撞墙-5分
- 捡到罐子+10分
- 做了捡罐子的动作但是没有捡到罐子-2分

在PJ1中,你需要实现从标准输入读入命令,让小黄完成动作,并使得分发生相应变化.

4. 功能介绍
    1. 随机生成棋盘 void createMap(args...)
        - 需要随机生成符合要求的棋盘,即一圈围墙包围着10 * 10 的格子空间， 每个格子有50%的概率出现罐.提示：调用 `rand()` 生成随机数来模拟概率，实现一个randomInt(int x, int y)函数返回一个x~y之间的随机数，每个随机数的概率就近似为 `1/(y-x)`
        - 生成的棋盘将在之后的操作流程中被使用，所以棋盘的数据应当被妥善保存，你可能会用指针，局部变量和全局变量相关的知识
    2. 正确打印棋盘 void printMap(args...)

        - 经过上一步，你已经生成棋盘并储存在数组了。接下来可以打印棋盘，用#表示墙壁，@表示有罐子的格子，空白表示没有罐子的格子，！表示小黄当前的位置，初始在左上角. 将上述矩阵打印到标准输出；你也可以在打印棋盘后，打印当前的分数score。为了方便观看，应当注意棋盘的疏密和对齐.

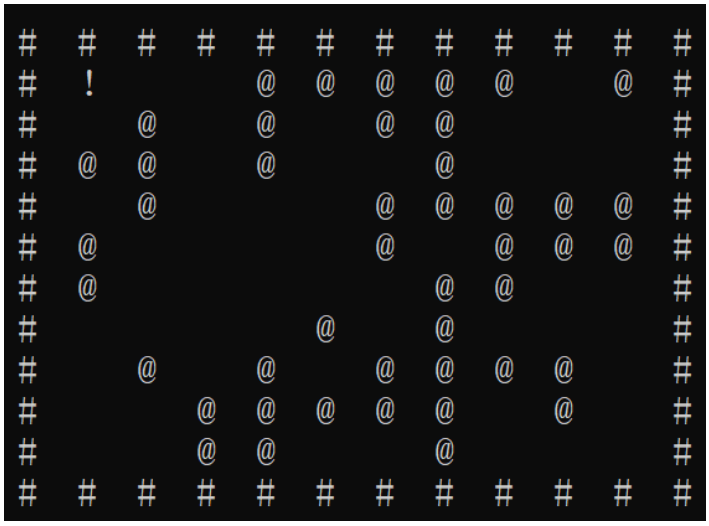

Figure 1: 打印一次随机生成的棋盘

3. 正确处理用户输入 action（args...）

- 我们希望用户可以输入命令来控制小黄。需要实现一个函数，读取用户的输入（数字0-6），让小黄进行7种动作中的一种. 在小黄执行完操作后，它的位置、整个棋盘的状态、分数都可能发生改变，因此需要把棋盘和分数的数据更新，并打印出来.

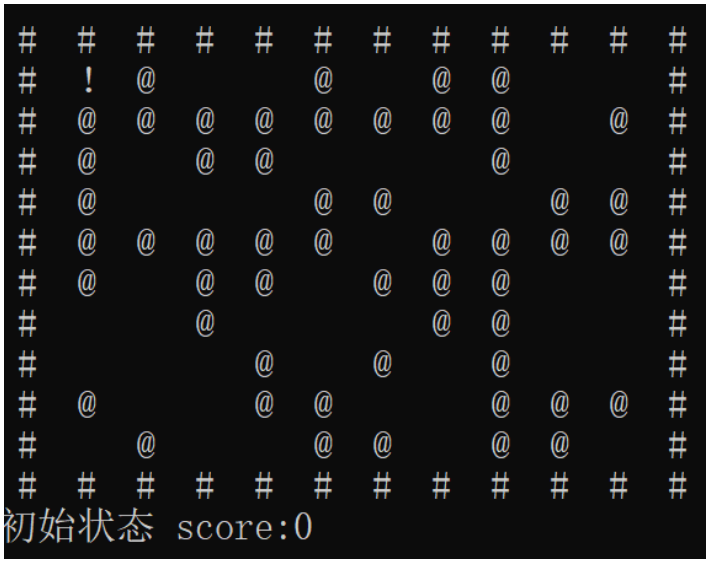

Figure 2: 初始状态，小黄在左上角

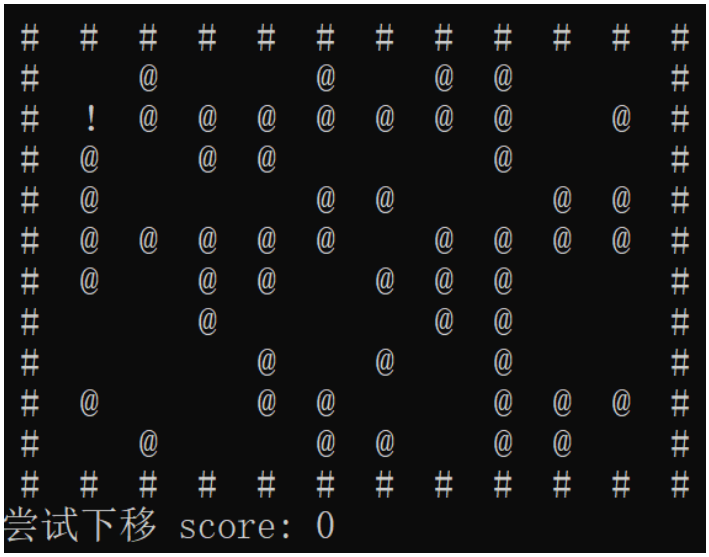

Figure 3: 键入命令，小黄向下移动

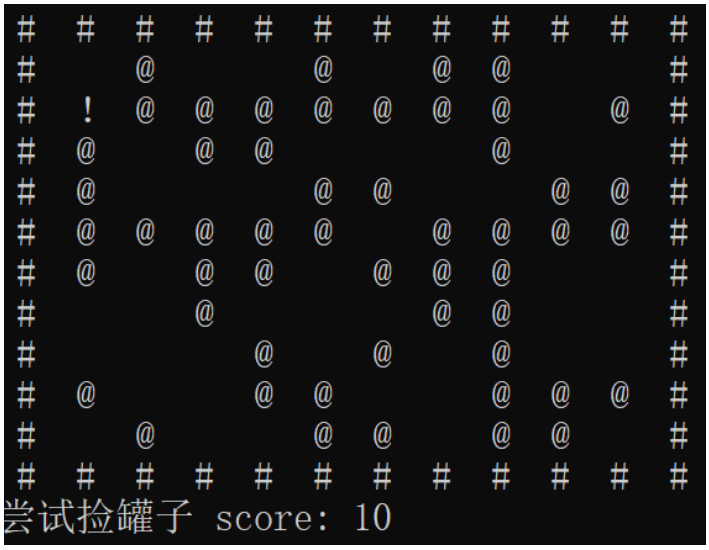

Figure 4: 键入pick命令，小黄做捡起罐子的动作，因为这里刚好有罐子，所以分数增加了

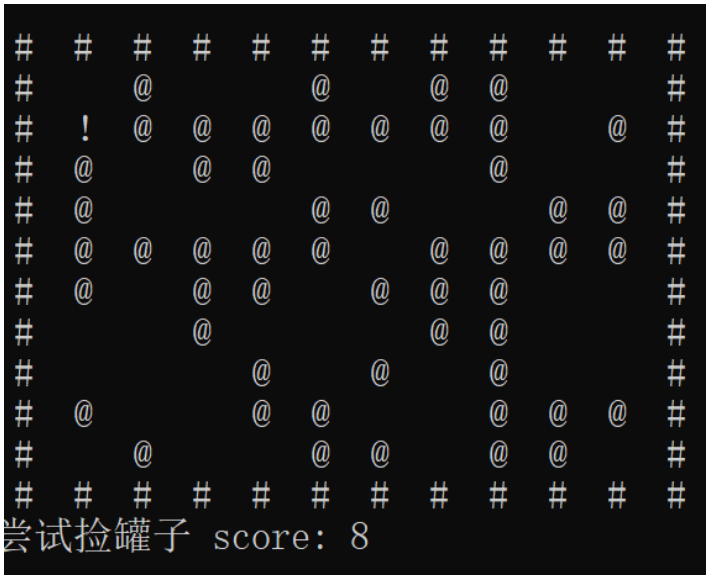

Figure 5: 再次键入pick命令，小黄在原地再次做捡起罐子的动作，但是这里已经空了，所以分数减少

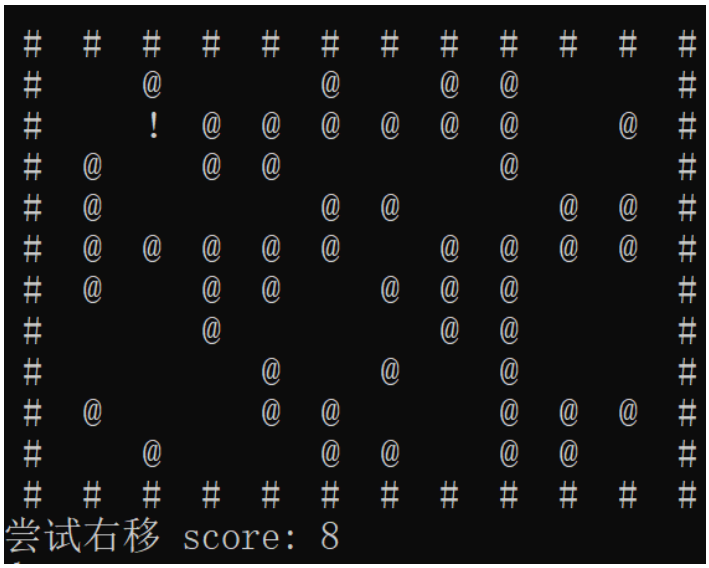

Figure 6: 键入命令，小黄向右移动，之前的罐子被捡走的地方变成了空地

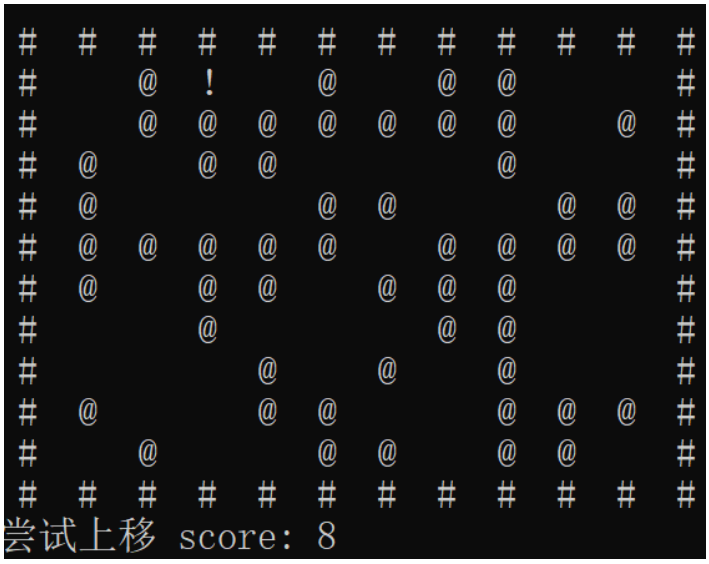

Figure 7: 小黄不断移动，直到上边界...

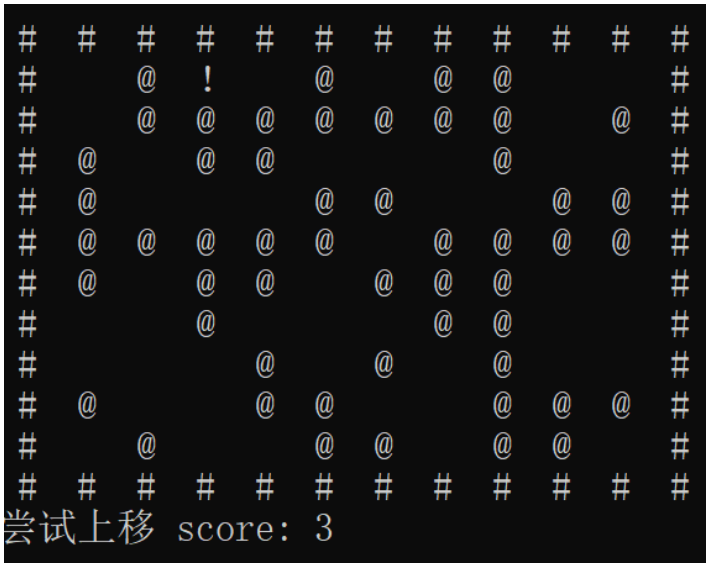

Figure 8: 小黄想继续向上移动，但是撞到了墙，位置没有改变，分数减少

- 仔细判断每种情况的边界条件（比如不要让小黄冲出墙壁，或者数组越界的error）.
- 在每一步操作成功后，正确计算分数.
- 用户可能进行不符合要求的输入，比如输入了单词或一段乱码。面对这些情况，跳过这些无效的操作，并提示用户合法的输入有哪些.

Figure 9: 无效输入

4. 合理的交互
   - 我们期望用户可以理解程序并合理的使用程序，所以可以适当加入一些提示，告诉用户下一步应该怎样做
   - 可以使用一个循环读入用户的输入，直到用户键入某个特定的内容，如‘exit’后结束，并输出最终的棋盘结果与得分

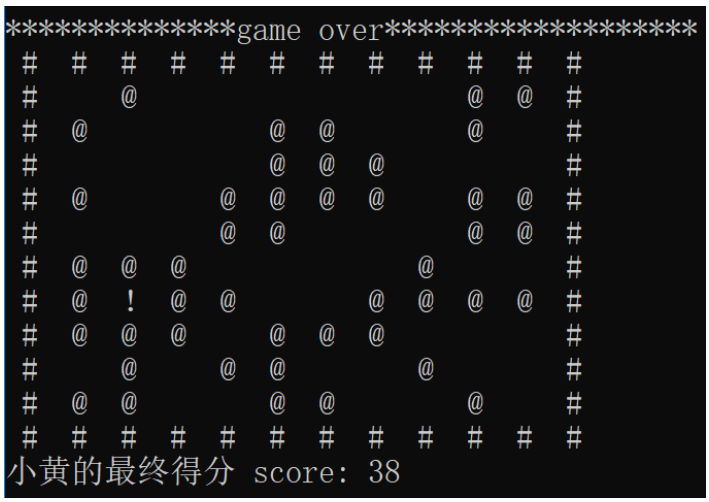

Figure 10: 最终的结果

1.2.2. PJ2

在PJ1中,你用上帝视角(能看见所有格子)一步一步指导小黄捡罐子.在PJ2中,我们要为小黄找到最佳策略,使得他们可以根据有限的已知信息作出好的决定.每一个小黄的策略表长度为243,对应小黄可能看到的 3^5 种情况.每个策略的值为0~6的整数,对应小黄可以做的7种动作.也就是说,有了这个策略表,小黄可以根据自己看到的情况做出唯一选择.

一个关键问题是，如何将每种格子分布情况与一个特定的数字绑定在一起?或许你还记得二进制，由0和1构成，常用来表示是或非等二元判断.设想这样一个情景:

有三枚不同的硬币,抛出,得到8种结果组合，这个时候可以用一个三位二进制数表示，000表示三个硬币都是反面;001表示第一枚硬币为正面，其余为反面;...111表示三个硬币为正面，用十进制计算为4+2+1=7.这样每种硬币情况就对应了0-7中的一个数字.

同理也可以得到，小黄知道五个格子的信息，而每个格子有三种情况，可以用三进制的5位数来表示,再把三进制转换成十进制，就得到了每种格子情况对应的数字,因此我们可以用一个长度为243的策略表涵盖所有的情况.

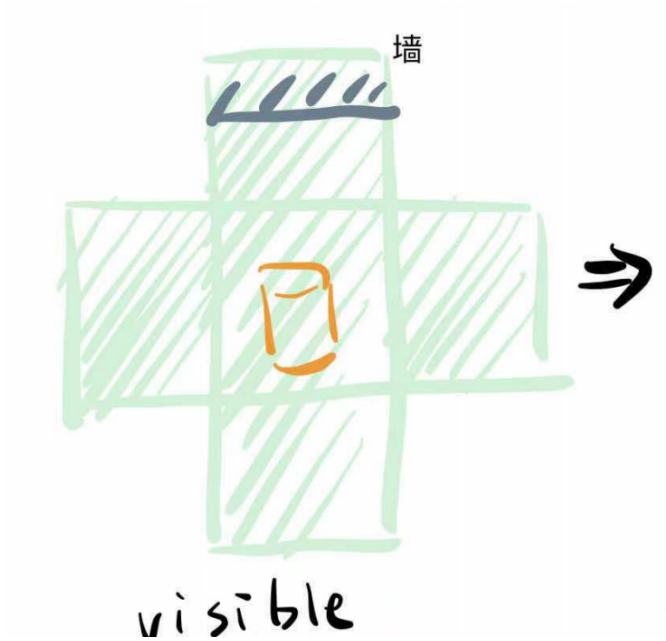

Figure 11: 小黄需要根据自己能看到的五个格子的情况进行反映,面对一种特定的五个格子的组合,小黄只能做出一种固定的决策

1. 读入策略

   在PJ2中,你需要实现从 `.txt` [2]读入策略.读入策略后，小黄就可以根据它所在的五个格子的情况,在策略表中找到它应该怎样做，并根据策略完成动作,使地图和得分发生相应变化. 完成这一步,你应该能为小黄指定一个策略表,并看到小黄按照你给定的策略表自动在棋盘上找罐子.这部分的输出类似PJ1.

2. 用遗传算法[3]找到最优策略

   如何找到最优的策略表？我们使用遗传算法来实现.

   我们随机生成第一批小黄(200只)的策略表.我们也为每一只小黄准备了100个棋盘,每一只小黄在它自己的100个棋盘上，按照自己的策略表,各走200步, 记下它在每个棋盘上的分数的平均值作为它的最终分数.

   如此得到了200只小黄的得分.由于我们的策略表是随机生成的,一开始小黄的得分并不理想.

   为了优化小黄的决策,我们循环执行以下步骤1000代:

   1. 对小黄按照得分进行排序, `rank` 为排名, `i` 为 `201-rank`. 生成 `1~20100` 之间的随机数 `rand`,根据 `rand` 的值抽取小黄. 这样做,可以使得分最高的(`rank1`)小黄(`i` 值为200),被抽到的概率为 `200/20100`；得分最低的(`rank200`)小黄 `i` 值为1,被抽到的概率为 `1/20100`. 这样能保证得分高的小黄更容易被抽到，在第2步留下"后代"

(rank=201-i)

i	rand
1	1
2	2,3
3	4,5,6
....	...
200	19900~20100

Figure 12: 生成随机数 `rand`

2. 按照1.中方法抽取2个小黄，“交配”得到两个子代.交配,即将亲本的决策表每隔长度60一切割(最后一段大于60),并交替拼接得到子代.
    1. 子代1的决策表: `亲本1片段+亲本2片段+亲本1片段+...`
    2. 子代2的决策表: `亲本2片段+亲本1片段+亲本2片段+...`

Figure 13: 交配

3. 如同基因突变的机制，在生成子代决策表的时候,子代决策表的每一个决策都有一定概率发生"突变".(可以尝试变异率的倒数 `MU` 的值分别为200,400,800时的不同训练效果).

Tips:
变异：利用随机数。设定变异率的倒数MU，在复制亲本的每一个策略（数字0～6）时
if random_int (0,MU) != 0:
list_decision_child[i]=list_decision_parent[i]
else:
list_decision_child[i]=random_int(0,7)

Figure 14: 变异

4. 利用2.,3.描述的交配繁殖方法,抽取95对亲本(亲本可以重复),完成190个子代的生成.另外10个子代直接复制前十名的亲本的决策表，最终得到200只子代.
5. 每一只子代小黄在它自己的100个棋盘上各走200步,记下它在每个棋盘上的分数的平均值作为它的最终分数.之后回到1成为亲本继续参与迭代.

3. 迭代结果

我们将每一代200个小黄的平均得分输出,如下图所示,可以看出,到最后小黄的得分已经非常高了

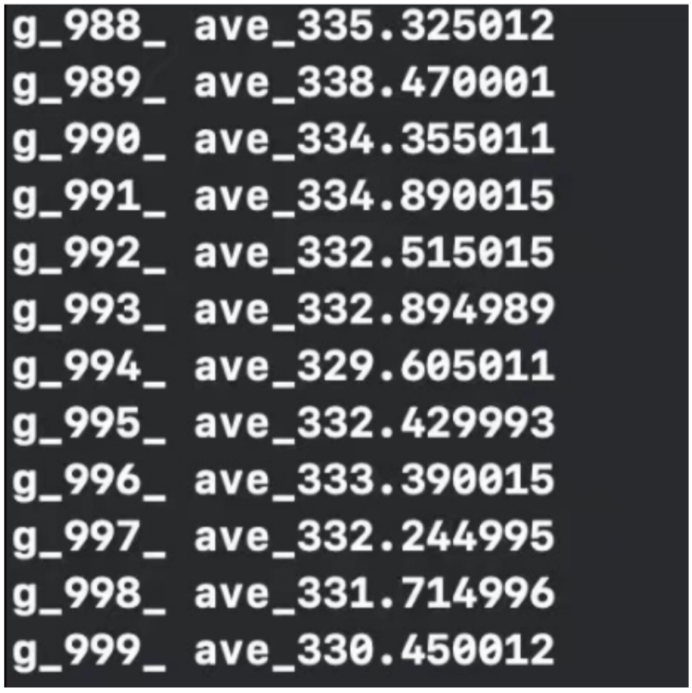

Figure 15: 迭代结果

我们也可以从更加直观的统计散点图上看出小黄的进步(选做)

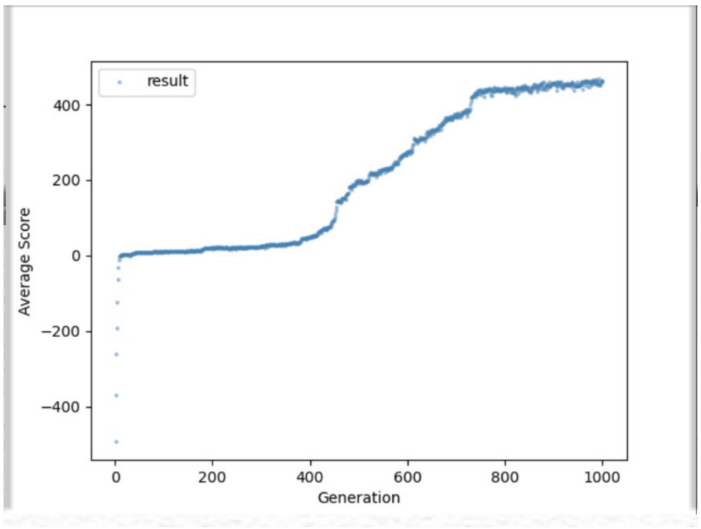

Figure 16: 散点图

4. 最终结果的可视化
    1. 将最后一代小黄的最高分决策写入txt中,数字之间用制表符隔开
    2. 将此小黄的决策表作为我们采纳的最终决策表
    3. 利用此决策表在一个棋盘上走200步

    4. 输出初始棋盘:

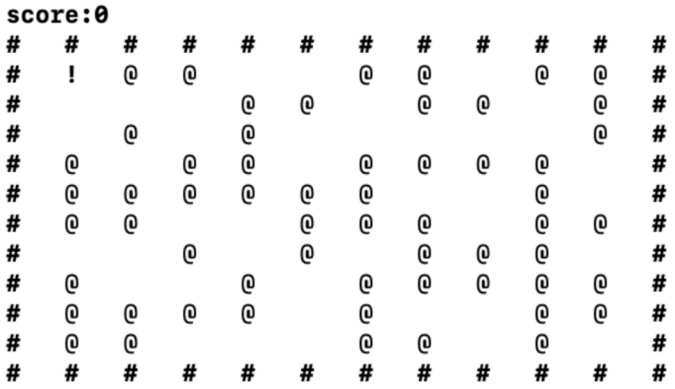

Figure 17: 初始棋盘,其中 `#` 为墙壁, `@` 为罐子, `score` 表示分数, `!` 为小黄当前位置

    5. 按照最终决策表执行,并且每执行一步输出一次棋盘.例如:执行了一次向右走和捡罐子之后的输出:

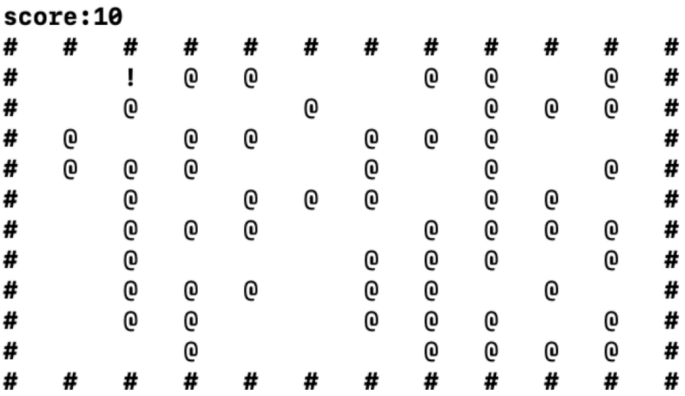

Figure 18: 执行了一次向右走和捡罐子之后的输出

5. Bonus
    1. 尝试其他算法(例如人工神经网络[4])
    2. GUI 图形界面实现[5]
    3. 绘制散点图,每一代200个小黄的平均得分为一个点

1.3. 评分标准

1.3.1. PJ1

需要提交[6]

- 源代码和可执行文件
- 项目说明文档

实现一个键盘控制的捡罐子游戏,正确计算用户操作后的得分变化

得分点	得分	评分标准
随机生成棋盘	10	需要随机生成符合要求的棋盘,即每个坐标出现的物体随机
正确打印棋盘	15	将棋盘按照 `#为墙壁,@为罐子,score表示分数,!为小黄当前位置` 的标志打印到标准输出
正确处理用户输入	25	针对用户输入的7种[7]可能,正确改变棋盘状态
正确计算分数	25	根据打分规则[8],正确改变当前得分
良好且一致的代码风格	10	可参考 `Google Style Guides`[9]和其他流行规范文档[10]
合理的模块化(和复用)	10	按照不同功能,将PJ实现分成不同的文件
清晰的说明文档	10	实现思路,实验结果,遇到的困难和解决过程,对PJ的改进建议(可选),参考资料(可选),其他(可选)

1.3.2. PJ2

需要提交[11]

- 源代码和可执行文件(有两个可执行文件.一个用于训练,运行后则开始训练,训练过程中输出每一代小黄的得分,最终创建并得到一个包含策略表的 `.txt` 文件;另一个用于测试策略,运行后读入策略表,打印地图,开始自动找罐子)
- 项目说明文档
- 你得到的最优策略表
- 你的训练迭代结果(每一代的得分)
- **Bonus**[12]: 其他算法实现的源代码和可执行文件
- **Bonus**[12]: 图形化界面的可执行文件
- **Bonus**[12]: 散点图

实现文件读取策略表,通过遗传算法训练得到最优策略,输出迭代结果,将策略可视化.

得分点	得分	评分标准
读取策略表文件	15	从 `.txt` 文件读取策略表,根据策略表自动选择动作
你得到的最优策略表	15	根据你的策略,小黄得分在400以上计5分

得分点	得分	评分标准
遗传算法实现	20	实现遗传算法,通过迭代训练找到最优策略
训练迭代结果	20	打印每一轮迭代的训练结果
良好且一致的代码风格	10	可参考 Google Style Guides[9]和其他流行规范文档[10]
规范的项目结构	10	根据需要提交的文件组织提交的文件夹的结构[11]
清晰的说明文档	10	实现思路,实验结果,遇到的困难和解决过程,对PJ的改进建议(可选),参考资料(可选),其他(可选)
Bouns1	10	尝试其他算法(遗传算法必做/不可替代)
Bouns2	10	GUI图形化界面
Bouns3	5	绘制散点图,将训练过程的每代得分绘制成散点图,体现策略的进步

1.4. 注意事项

1. 合理安排时间,尽早动手,完成每个功能点任务,不犯拖延症
2. 推荐使用 IDE 进行编写,调试,维护项目结构
3. 如果有任何疑问,及时和TA们联系
4. 注意自己的代码结构和风格(选择一种风格后需要保持一致),PJ2在PJ1基础上进行编写
5. 鼓励同学互相讨论,和TA讨论,找寻解决问题的办法,但严禁抄袭,一旦发现抄袭者和被抄袭者都作0分处理

Footnotes:

[1] 走到罐子所在的格子不会使罐子自动被捡起,需要发出指令

[2] 文件格式为一行243个字符,每个字符可能是$[0,6]$

[3] 遗传算法 - 维基百科,自由的百科全书

[4] 人工神经网络 - 维基百科,自由的百科全书

[5] 推荐一个为C语言提供图形化接口的库:Simple DirectMedia Layer - Wikipedia,这是一个简单教程SDL library in C/C++ with examples - GeeksforGeeks,项目主页Simple DirectMedia Layer - Homepage

[6] PJ1推荐提交文件夹按如下结构组织,压缩成 `.zip` 格式后提交

```
PJ1Handin_19302010001张三
├── PJ1.out
├── README.pdf
└── src #游戏源代码,src文件夹下具体的文件名可以不同
    ├── map.h
    ├── map.c
    ├── ...
    ├── main.h
    └── main.c
```

[7] 合法输入有

1. 随机移动,即随机出现2~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;
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;
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;
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;
6. 不移动;
7. 捡罐子,如果小黄此时的位置上有罐子,则罐子被捡起,即此格子变为空;

[8] 撞墙-5分,捡到罐子+10分,做了捡罐子的动作但是没有捡到罐子-2分

[9] Google Style Guides | styleguide

[10] Pintos Projects: Coding Standards, GNU Coding Standards - GNU Project - Free Software Foundation...

[11] PJ2推荐提交文件夹按如下结构组织,压缩成 `.zip` 格式后提交

```
PJ2Handin_19302010001张三
├── README.pdf
├── auto.out #读入策略表后开始自动寻找的可执行文件
├── bonus
│   ├── Bonus.out #其他获得最优策略表的算法的训练程序(对应train.out)
│   ├── GUI.out #auto.out 的GUI版本
```

```
│       ├── scatter.png #散点图
│       └── src #其他算法的源代码,src文件夹下具体的文件名可以不同
│           ├── main.c
│           ├── main.h
│           ├── ...
│           ├── map.c
│           └── map.h
├── iteration.txt #训练过程的输出
├── src #遗传算法的源代码,src文件夹下具体的文件名可以不同
│       ├── main.c
│       ├── main.h
│       ├── ...
│       ├── map.c
│       └── map.h
├── strategy.txt #你最终的最优策略
└── train.out #遗传算法的训练程序

3 directories, 16 files
```

12 可选项

Author: 左晓蕊 周淇 刘可伊 叶浩宁
Created: 2022-10-01 周六 05:27
Validate