QT 版本的人工智能贪食蛇 SNAKE AI 编程实践

1、 概述

在 2020 年 3 月份的某一天，脑子里突然闪现当年 Nokia 手机上的经典贪食蛇游戏来，通过上、下、左、右四个按键控制贪食蛇的前进方向，在保证贪食蛇不碰撞到边界和自身身体的前提下，吃到更多的食物，从而得到更高的分数。于是，很自然地就想到是否可以引入一套智能算法，让计算机综合贪食蛇和食物在地图中的位置等信息，计算出贪食蛇的最优路径，并实时控制贪食蛇下一步的前进方向，使贪食蛇能吃到最多的食物。

有了这个想法之后，就按捺不住好奇心，准备将想法付诸实践，用计算机进行验证。那么是否已经有人做过类似的工作呢？上网搜索一下，果然已经有人实现了这个想法，而且时间还是早在 2013 年，有一个俄罗斯人发布游戏测试效果，竟然可以吃到满屏。（俄罗斯人咋老是搞一些无聊的东西来娱乐自我，真是战斗民族的童心未泯）智能算法的效果简直刷新我的认识，我原本以为能够玩的比我手动玩的更厉害就不错啦，没想到这么厉害。

这里发布下俄罗斯人做的 GIF 动图，简单粗暴，赏心悦目。

之后搜索到国内还有几个人做的，比如 Hawstein 做的 Python 版本，效果还不错。

研究了一下前人写的一些实践经验文档和源码，对于这个问题也有了更多的理解和认识，在后续编程中起到借鉴作用。

2、 开发平台

生命苦短，请用 Python。虽然老前辈的话一直在耳边回响，但是作为一名长期用 C 语言写程序的算法工程师，还是要顽固地表达我对于 C 的热爱，抗拒一下社会观念，选择 C 作为本次实践的语言。

操作系统：Windows 10　64 位操作系统

开发平台：QT　版本号：5.9.8

编译器：mingw 5.3.0 32 bit

3、 算法思路：

3.1、暴力解决方式

经过思考，发现可以有一种很暴力的方式解决贪食蛇问题，就是让贪食蛇一直按照如下方式行进，肯定能够吃满全屏，但是这样子玩的话，这个游戏就显得很无聊了，也起不到任何锻炼的效果。果断抛弃。

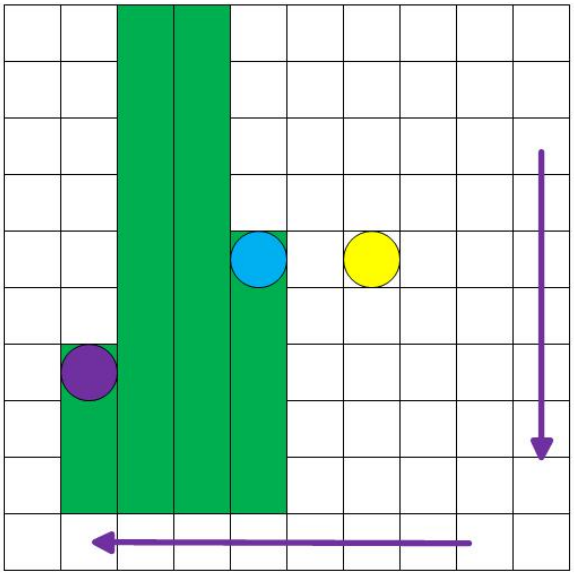

图 1

如图 1 所示，贪食蛇的地图为 10*10 的矩形框，贪食蛇每一步走一个小方格。贪食蛇身体为绿色部分，蓝色圆圈为蛇头，紫色圆圈为蛇尾，黄色圆圈代表食物。贪食蛇在图中曲折前进，走 S 型路线，将最右侧的一列和最下方的一行方格留作逃生通道，当贪食蛇从左侧曲折前进到达右侧后，通过逃生通道再次回到最左侧。循环进行，直到最后填满整个地图。

这个行进路线可以看成贪食蛇一直追着自己的尾巴跑，且走的路线为最长路线。

3.2、正规算法

3.2.1、通过 BFS 算法查找是否有从贪食蛇头部到达食物的最短路径，且当贪食蛇头部到达食物后，必须能够找到贪食蛇头部到贪食蛇尾部的路径。如果条件满足，则按照最短路径让贪食蛇前进一步，否则，跳到 3.2.2。

BFS（breadth first search），广度优先搜索路径算法，是一种图形搜索算法。简单地说，BFS 是从根节点开始，沿着路的宽度遍历树的节点，如果发现目标或者搜索完所有节点，则演算终止。

BFS 是一种盲目搜索法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。

本项目中，BFS 的算法实现参考了 red blob games 的博客，算法的详细介绍参见博文，这里不再赘述，请自行跳转到博客网页，链接详见后续参考文献部分。

如何在贪食蛇游戏中应用 BFS 算法？要找到从贪食蛇头部到达食物的最短路径，首先应用 BFS 算法，算出从食物开始到达地图中的各个方格的权重，两个相邻方格之间的权重设为 1。最短路径就是权重最小的路线。示例如图 2 所示，贪食蛇蛇头的上、下、左、右四个方向的权重分别为 4、4、INFI、2，由于权重 2 是最小值，所以贪食蛇的下一步的前进方向为向右前进。

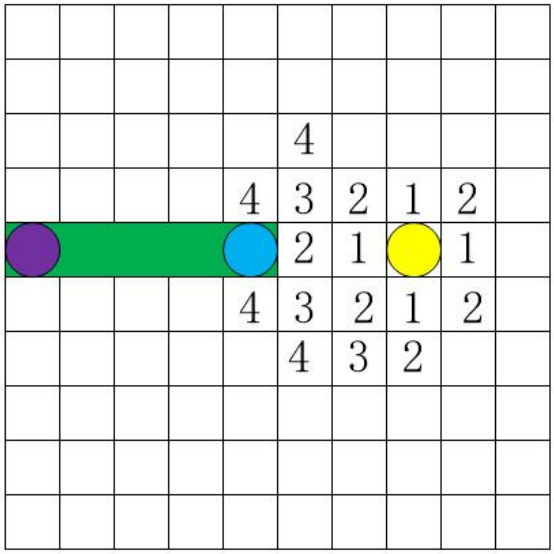

图 2

如果是应用 BFS 算法，计算从贪食蛇头部到达地图中其他方格的权重，能不能达到我们计算最短路径的目的呢？示例如图 3 所示，很容易看出，同样可以算出最短路径。但是由于贪食蛇蛇头的上、下、左、右四个方向的权重分别为 1、1、INFI、1，无法直接确定下一步贪食蛇的前进方向，必须输出最短路径才能找到下一步贪食蛇的前进方向，步骤相对比较麻烦，所以不采用这种方法。

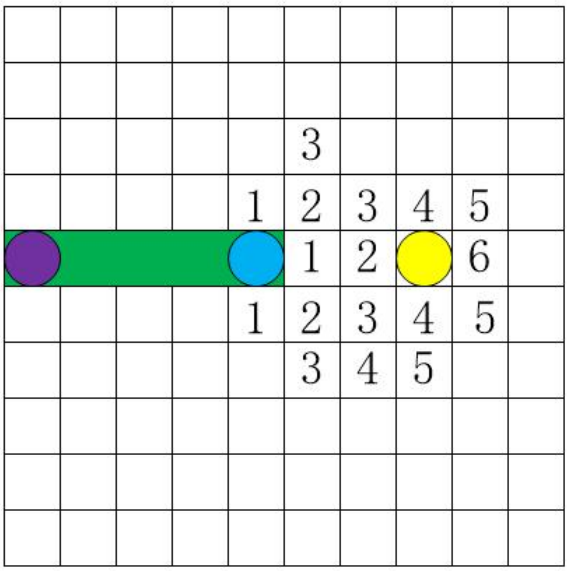

图 3

当采用 BFS 找到最短路径时，为什么需要验证当贪食蛇的蛇头到达食物位置后，是否能够找到从贪食蛇头部到达贪食蛇尾部的路径呢？因为当贪食蛇按照最短路径前进，到达食物所在位置后，如果找不到贪食蛇头部到达尾部的路线，说明贪食蛇会把自己困住，游戏就会结束，所以不能采用此最短路径。示例如图 4 所示，当贪食蛇处于这种状态时，按照最短路径的原则，贪食蛇会按照紫色箭头标注的路线行进，贪食蛇会把自己困在自己身体里面，游戏结束。

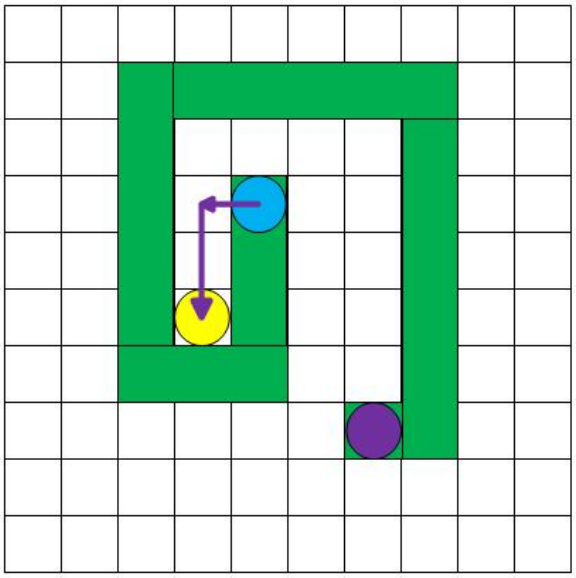

图 4

3.2.2、如果贪食蛇能够规划出从贪食蛇头部到贪食蛇尾部的路径，则选择最长路径，并且让贪食蛇按照最长路径前进一步。

示例如图 5 所示，贪食蛇的最长路径就是权重最大的路线，由于贪食蛇蛇头的上、下、左、右四个方向的权重分别为 7、INFI、9、5，最大权重为 9，因此贪食蛇下一步向左侧前进。

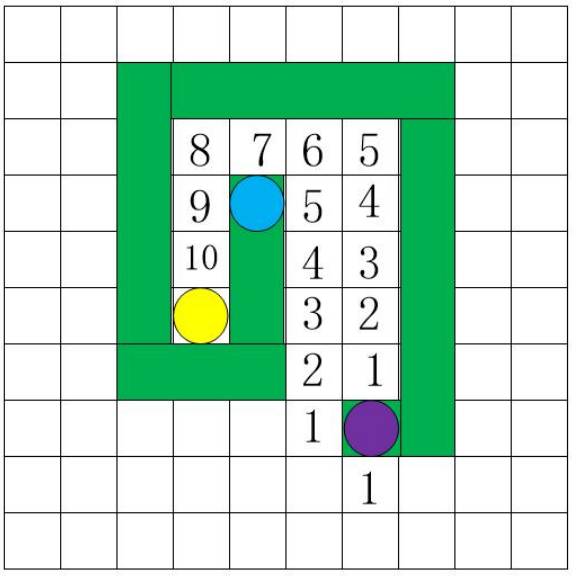

图 5

为什么贪食蛇跟随蛇尾时，需要选择最长路径？因为贪食蛇会不断地走 S 型路线，将空闲区域更多地填满，贪食蛇结构也就更加紧凑，可以争取更多的有效行进空间。

3.2.3、如果前两者都不能满足，即不能规划出有效的最短路径，且不能规划出从贪食蛇头部到达贪食蛇尾部的路径，那么随意前进一步。

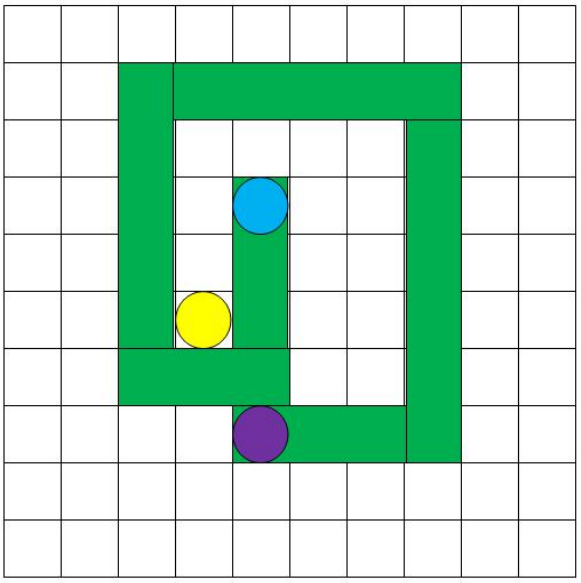

图 6

随意走一步，后续测试的时候发现出现这种情况比较少见，所以就暂时留一个空函数。如果贪食蛇不能找到一条有效到达食物的最短路径，且贪食蛇的蛇头无法跟踪蛇尾，那么游戏就直接结束。

4、 算法测试

一共测试了 3 张不同尺寸大小的地图测试，测试结果算法的效果不错，基本上能够填满 90%以上的地图，但是很难将地图填满。

1、5*5 地图

2、10*10 地图

3、15*15 地图

5、 总结

刚开始觉得问题很简单，没有做好应对困难的准备，结果现实告诉我，事情远远没有想象中那么简单。贪食蛇想要吃满屏真是难呀，特别是地图变大之后，留下的方格越来越多，说明目前的算法还不是最优算法，需要继续钻研，找出一个更好的算法。

同时发现，地图越大，留下的空格数越多，是否留下空格数和地图尺寸有一个比例关系，需要分析。

程序运行有时会崩溃退出，目前还没有找到问题原因。

6、 代码下载

https://github.com/crystalascii/AI.git

7、 参考文献

http://hawstein.com/2013/04/15/snake-ai/

https://www.redblobgames.com/pathfinding/a-star/implementation.html#cpp-breadth-first