# 用Python写一个贪吃蛇游戏

**邓罡 2013301020104 物基2班**

**摘要：**关于贪吃蛇小游戏，想必大多数人都玩过。本文将引导我们理解什么是Python，以及用Python编写一个贪吃蛇游戏的步骤和思路。

**关键词：**贪吃蛇游戏 Python

**正文：**

1. **介绍：**一个完善的贪吃蛇游戏肯定会达到几点要求，比如吃掉每一个食物，和在将屏幕填满之前不会蛇头撞上自己的的蛇身等。那么想要让贪吃蛇真的很贪吃，让它将屏幕内所有食物吃光，并且华丽丽的将屏幕铺满，从而成功结束游戏，我们首先会想到这一定是程序实现的，第二个想到的是写程序该怎么实现，用什么算法。毫无疑问，我们肯定用的是Python。
2. **正文：**直接写完美版本不是什么好出路，因为完美版本往往要考虑很多东西，直接上来就写这个肯定漏洞百出。我们先写一个可以运行起来的贪吃蛇游戏，仅仅是让程序控制贪吃蛇运动起来，去吃食物，然后再去想其他部分。

1.简单版本：

很容易知道最初问题是：在一个矩形中，每一时刻有一个食物， 贪吃蛇要在不撞到自己的条件下，找到一条出路，然后沿这条路走，去吃它的食物。

我们先不去想蛇会越来越长这个事实，问题基本就是，给你一个起点(蛇头)和一个终点(食物)，要避开障碍物(蛇身)，从起点找到一条可行路到达终点。 我们可以用的方法有：

BFS DFS A\*

只要有选择，就先选择最简单的方案，我们现在的目标是要让程序先跑起来， 优化是后话。所以，从BFS开始。我们最初将蛇头位置放入队列，然后只要队列非空， 就将队头位置出队，然后把它四领域内的4个点放入队列，不断地循环操作， 直到到达食物的位置。这个过程中，我们需要注意几点：1.访问过的点不再访问。 2.保存每个点的父结点(即每个位置是从哪个位置走到它的， 这样我们才能把可行路径找出来)。3.蛇身所在位置和四面墙不可访问。

通过BFS找到食物后，只需要让蛇沿着可行路径运动即可。这个简单版本写完后， 贪吃蛇就可以很欢快地运行一段时间了。

上一节的简单版本跑起来后，我们认识到，只教贪吃蛇一种策略是不行的。 它这么笨一条蛇，你不多教它一点，它分分钟就会挂掉的。 所以，我写了个Wander函数，顾名思义，当贪吃蛇陷入困境后， 就别让它再BFS了，而是让它随便四处走走，散散心，等待转机。

Wander函数怎么写都行，我写的是：就让贪吃蛇贪到底。在BFS无解后， 告诉蛇一个步数step(随机产生step)，让它在空白区域以S形运动step步。 这回运动方向就不随机了，而是有组织有纪律地运动。然而它最终还是会挂掉，S型运动也无法让贪吃蛇避免死亡的命运。它可以靠S型运动多存活一段时间，但是由于它的策略是：

While 没有按下ESC键：

If 蛇和食物间有路径：

吃食物去

Else：

Wander 一段时间

问题就出在蛇发现它自己和食物间有路径，就二话不说跑去吃食物了。它没有考虑到，你这一去把食物给吃了后形成的局势(蛇身布局)， 完全就可能让你挂掉。(比如进入了一个自己蛇身围起来的封闭小空间)所以，为了能让蛇活得久一些，我们还要考虑的更全面才行。

1. 全面版本：

现在已经有了一个比较低端的版本，而且对问题的认识也稍微深入了一些。 现在可以进行一些比较慎密和严谨的分析了。首先，让我们罗列一些问题：

* 蛇和食物间有路径直接就去吃，不可取。那该怎么办？
* 如果蛇去吃食物后，布局是安全的，是否就直接去吃？(这样最优吗？)
* 怎样定义布局是否安全？
* 蛇和食物之间如果没有路径，怎么办？
* 最短路径是否最优？(这个明显不是了)
* 那么，如果布局安全的情况下，最短路径是否最优？
* 除了最短路径，我们还可以怎么走？S形？最长？
* 怎么应对蛇身越来越长这个问题？

把思路理一理：一开始，蛇很短(初始化长度为1)，它看到了一个食物， 使用BFS得到矩形中每个位置到达食物的最短路径长度。在没有蛇身阻挡下， 就是曼哈顿距离。然后，我要先判断一下，贪吃蛇这一去是否安全。 所以我需要一条虚拟的蛇，它每次负责去探路。如果安全，才让真正的蛇去跑。 当然，虚拟的蛇是不会绘制出来的，它只负责模拟探路。那么， 怎么定义一个布局是安全的呢？ 仔细想想，会发现即使到最后蛇身已经很长了，它仍然没事一般地走出了一条路。而且，是跟着蛇尾走的！嗯，这个其实不难解释，蛇在运动的过程中，消耗蛇身， 蛇尾后面总是不断地出现新的空间。蛇短的时候还无所谓，当蛇一长， 就会发现，要想活下来，基本就只能追着蛇尾跑了。在追着蛇尾跑的过程中， 再去考虑能否安全地吃到食物。

经过上面的分析，我们可以将布局是否安全定义为蛇是否可以跟着蛇尾运动， 也就是蛇吃完食物后，蛇头和蛇尾间是否存在路径，如果存在，我就认为是安全的。OK，继续。真蛇派出虚拟蛇去探路后，发现吃完食物后的布局是安全的。那么， 真蛇就直奔食物了。这样的策略好吗？未必。因为蛇每运动一步， 布局就变化一次。布局一变就意味着可能存在更优解。比如因为蛇尾的消耗，原本需要绕路才能吃到的食物，突然就出现在蛇眼前了。所以，真蛇走一步后，更好的做法是，重新做BFS。然后和上面一样进行安全判断，然后再走。

上面列的好几个问题里都涉及到蛇的行走策略，一般而言， 我们会让蛇每次都走最短路径。这是针对蛇去吃食物的时候， 可是蛇在追自己的尾巴的时候就不能这么考虑了。我们希望的是蛇头在追蛇尾的过程中， 尽可能地慢。这样蛇头和蛇尾间才能腾出更多的空间，空间多才有得发展。 所以蛇的行走策略主要分为两种：1. 目标是食物时，走最短路径2.目标是蛇尾时，走最长路径。

最后，我们编写出源代码，就可以做出了一条完美的贪吃蛇了！