《算法设计与分析》

考核报告

|  |  |
| --- | --- |
| 专业： | 人工智能 |
| 班级： | 2020185 |
| 学号： | 202018526 |
| 姓名： | 高树林 |

**说 明**

本报告需完成下列项目（二选一）：

**一、迷宫问题：实现任意迷宫求解及可视化**

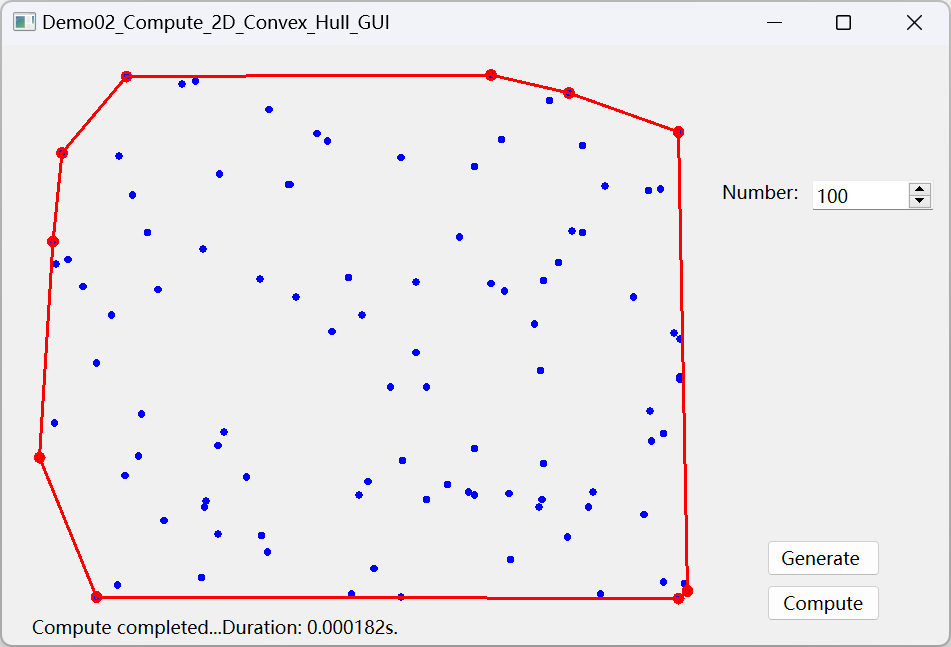
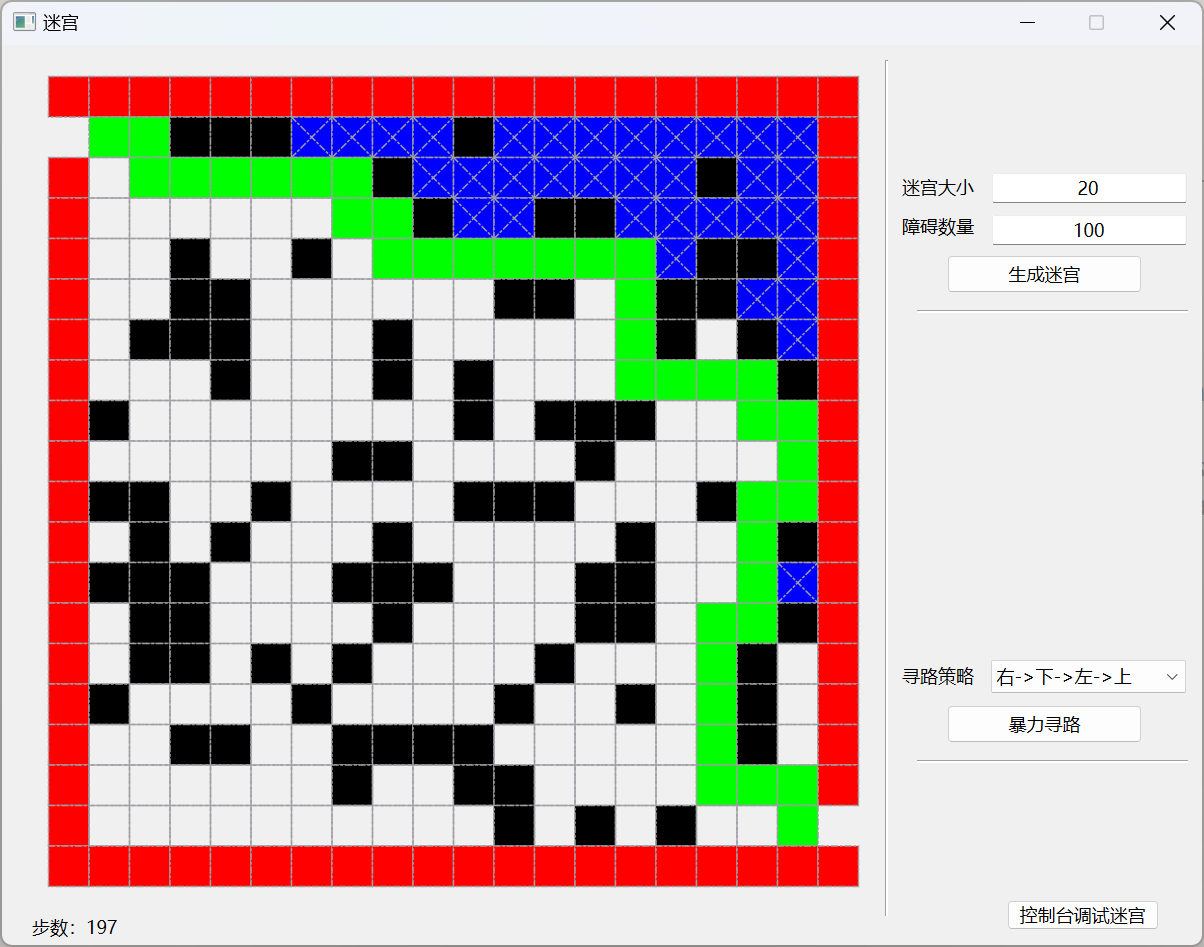
迷宫入口位于左上角，迷宫出口位于右下角，随机生成不同数量的障碍物，算法可从迷宫入口开始，自动寻找到一条到达迷宫出口的通路，如下图（a）所示。

**二、凸包问题：计算二维平面中点的凸包并可视化**

随机生成一个包含个平面坐标点的集合，并将结果可视化，如下图（b）所示。

1）利用分治思想计算二维点集凸包；

2）利用Graham's Scan法计算二维点集凸包（选做）。



(a)

(b)

（上图仅作参考）

**要 求**

1）报告需包含详细的算法描述、程序代码与运行截图；

2）算法描述及代码与网络或他人雷同者，均按0分计算；

3）考核报告中仅粘贴**核心代码**，全部程序源码与电子版考核报告打包提交。（**自己定义核心代码，对代码封装能力有一定要求**）

4）纸质版与电子版同时提交（电子版命名格式：**完整学号-姓名-期末考核报告，不得省略“-”**，如：2088166-乔峰-期末考核报告，**打包压缩命名格式与电子版报告命名格式一致**）；

5）算法描述思路明确、代码清晰、格式正确美观；

6）结合代码质量、算法实现方式、编程技巧、高级数据结构、报告格式等方面综合评分；

7）纸质版双面打印；

8）编程语言自选。

|  |  |
| --- | --- |
| **考核选题** | □ 凸包问题 |
| **算法描述** | 阐述解题思路或算法思想  两种算法思想：   1. 分治思想：选择左右两个端点（相对于X轴），将两个点相连接就将这个问题转化为两个结构相同，规模不同的子问题，即求上半部分的凸包和下半部分的凸包。采用的策略为：将点按x的顺序升序排列，当x相同时按y升序排列，第一个（下标为0的）和最后一个（下标为-1）两点确定一条直线。上半部分的凸包又可以找上半部分的左右两个端点（相对于X轴），将两个点相连接，找到离该直线最远的点，根据数形结合原理可以转化为三点面积最大时，该点离直线最远，找到该点将该点加到边界（boundary）列表里面，按照递归的思想，当迭代到最后在只剩一个点的时候，问题就得到了解决。 2. Graham算法：先找到凸包的纵坐标最小的点，（该点一定是凸包上的点）将该点作为原点，从该点出发，逆时针逐个遍历所有点。然后按照各点与该点连线的极角和极径排序，当极角相同时极径小的排在前面。确定一个点就将它放到边界列表。设当前点为P，连接原点与上一次加入到边界列表最后一个点，确定一条直线，判断P点是否在直线右边，若在，则上次进入边界列表的元素不是凸包上的点，将它删除掉；若在左边，则当前点时凸包上的点，将当前点加入边界列表。 |
| **核心代码** | 仅粘贴核心代码：  分治算法：   1. **def** AreaOfUp(left, right, lists, boundary): 2. area\_max = 0 3. max\_point = () 4. **for** item **in** lists: 5. **if** item == left **or** item == right: 6. **continue** 7. **else**: 8. max\_point = item **if** calc\_area(left, right, item) > area\_max **else** max\_point 9. area\_max = calc\_area(left, right, item) **if** calc\_area(left, right, item) > area\_max **else** area\_max 10. **if** area\_max != 0: 11. boundary.append(max\_point) 12. AreaOfUp(left, max\_point, lists, boundary) 13. AreaOfUp(max\_point, right, lists, boundary)  16. **def** AreaOfDown(left, right, lists, boundary): 17. area\_max = 0 18. max\_point = () 19. **for** item **in** lists: 20. **if** item == left **or** item == right: 21. **continue** 22. **else**: 23. max\_point = item **if** calc\_area(left, right, item) < area\_max **else** max\_point 24. area\_max = calc\_area(left, right, item) **if** calc\_area(left, right,item) < area\_max **else** area\_max 25. **if** area\_max != 0: 26. boundary.append(max\_point) 27. AreaOfDown(left, max\_point, lists, boundary) 28. AreaOfDown(max\_point, right, lists, boundary)   Graham算法：   1. **def** findboundary(): 2. ymin = min(lists\_points, key=**lambda** x: x[1])[1] 3. start = min([i **for** i **in** lists\_points **if** i[1] == ymin], key = **lambda** x:x[0]) 4. boundary=[] 5. lists\_points = angle\_sort(start, lists\_points) 6. boundary.append(lists\_points[0]) 7. boundary.append(lists\_points[1]) 8. i=2 9. **while** len(boundary)!=0 **and** i != len(lists\_points): 10. **if** cross\_product(boundary[len(boundary) - 2], boundary[len(boundary) - 1], lists\_points[i]): 11. boundary.append(lists\_points[i]) 12. i+=1 13. **else**: 14. boundary.pop() 15. **if** len(boundary)<2: 16. boundary.append(lists\_points[i]) 17. i+=1 18. **continue** |
| **运行结果及截图** | 可用多张截图展示中间结果并予以说明  分治算法结果：（可视化用tkinter模块）    Graham算法截图：（matplotlibmo模块） |
| **总结** | 凸包问题有多种解法，通过广泛的查阅资料，我总结了大概于以下几种高级解法：   1. Jarvis步进算法，该算法的时间复杂度为O(n)。 2. 增量算法，该算法的时间复杂度为O(n^2). 3. 快速凸包算法，该算法的时间复杂度为O(nlogn)。 4. 分治算法，该算法的时间复杂度为O(logn)。 5. Graham算法，该算法的时间复杂度为O(nlogn)。 6. 单调连算法，该算法的时间复杂度为O(logn).   题目要求的算法是上述第4和第5种。  对于分治算法是最好理解的，抽象来讲，分治法就是设定一种状态，这种状态有多种可能，将该状态下的所有可能都判断出来，并给出在此状态下它要进行的操作。当递归到递归基时，针对本问题就是当线段上方或者下方只有一个点时，则返回该点，相当与自下而上的返回所有步骤的值，进而实现问题的求解，在这里体现了分治问题的典型特征：最优子结构性质，原问题和他的所有问题的结构都是相同的，仅仅是规模不同，且原问题的最优解包含其子问题的最优解。  Graham算法的实质是将各点按照计较的大小顺序遍历每个点，接着遍历每个点，通过夹角的大小判断凸包上应该有哪些点，将点加入到边界列表，有了边界列表就可以进行可视化了  在本学期学习的众多算法中，我收获最大的就是动态规划和贪心算法两大算法。动态规划的实质就是穷举所有可能，但是他的高明之处就在于，他在进行穷举的时候并不是蛮力的去列举，而是按照一定的策略，此次列举和前几次列举或前面某一次列举是有关系的，而这些列举都是通过下标索引记录在二维数组里面，查找的时间复杂度为O(1)，当所有可能被列举完毕时，就能通过引用下标直接查询到，一般能够找出全局最优解。贪心算法则是具有贪心选择性，从第一开始，每一步的选择都是使当前局部最优，当然这种算法不一定能找到全局最大值，因为当前面某一步选错了，整体的选择就是非最优的，但通过贪心算法选择出的值一般不会比全局最优值差太多，这也是为什么贪心算法运用广泛的原因之一。 |