第5章 chapter 5

图　论

5.1　从图形界面输入图并产生邻接矩阵

1. 案例问题

在某一年的“计算概论”课程中，一位同学提交了一个自选 RAPTOR 图形编程练习（参见 2.4.2 节），用户可以在 RAPTOR 图形界面中建立一个用鼠标设定若干顶点并自动在顶点之间连边的图。老师建议把该例改为可以设置以下几种连接方式：

（1）无向图，全连接。

（2）无向图，随机连接。

（3）有向图（点和线可以用色彩表示），随机形成连通分量。

（4）有向图和无向图的自由构建。

以上几个算法可以为后续算法打下实现的基础。

比较理想的方式是可以指定输入 N 个顶点，顶点之间的边可以指定有向和无向，若两个顶点之间有两条有向边，建议使用弧线连接，以便区分。在用户输入图以后，可以自动产生邻接矩阵和邻接表，这是一项很有趣，也很有价值的大作业。

2. 算法设计思路

实际上，这是为图论算法编制一个实用程序，任何人都可以在此基础上测试自己编写的图论算法。而本算法具体实现的是图形视窗界面下输入有向图与无向图并输出用于后续算法的相关图的邻接矩阵。

本算法需要实现的功能如下：

（1）选择 4 种输入方式之一进行图的输入：mode1 为手动输入无向图，mode2 为随机输入无向图，mode3 为手动输入有向图，mode4 为随机输入有向图。

（2）在手动方式输入下，用户需要在图形界面下按提示依次输入顶点和边，由程序保存到 graph_data.csv 文件中。在随机输入的两种模式下，用户仅需要输入顶点和边的数量即可。

（3）图在视窗界面下的输入或随机产生图的顶点和边：mode1 与 mode3 是通过单击确定顶点位置和顶点之间的边；mode2 与 mode4 随机生成顶点位置和顶点之间的边，坐标输入 graph_data.csv 文件。

(4) 从 graph_data.csv 文件中将图的描述性数据导入算法,产生相应的邻接矩阵导出并写入 adjmatrix.csv,最后在视窗界面中显示。

设置两个文件输出的目的如下:graph_data.csv 中的图的描述性数据可以在大部分图形系统中转换后再现用户原始图的样式,adjmatrix.csv 的目的是为后续计算提供数据结构——邻接矩阵。

需要关注的问题还包括:

(1) 对于有向图需要区分边的不同走向,而 RAPTOR 中的图形线不具备箭头指向。作为一种变通的方案,需要使用随机色彩来描述不同的顶点和边,同色的点与边表示从该点出发的边。如果需要不同色的顶点随机抽到同样色彩,则需要重新抽取。

(2) 在视窗界面上如何实现随机均匀分布随机生成的顶点。

(3) 由于有向图的两个顶点之间的边可能多于一条,所以需要考虑在两个顶点之间安排两条不重合的弧线来区分不同走向的边。

3. 主要算法流程

算法框架如图 5-1 所示。

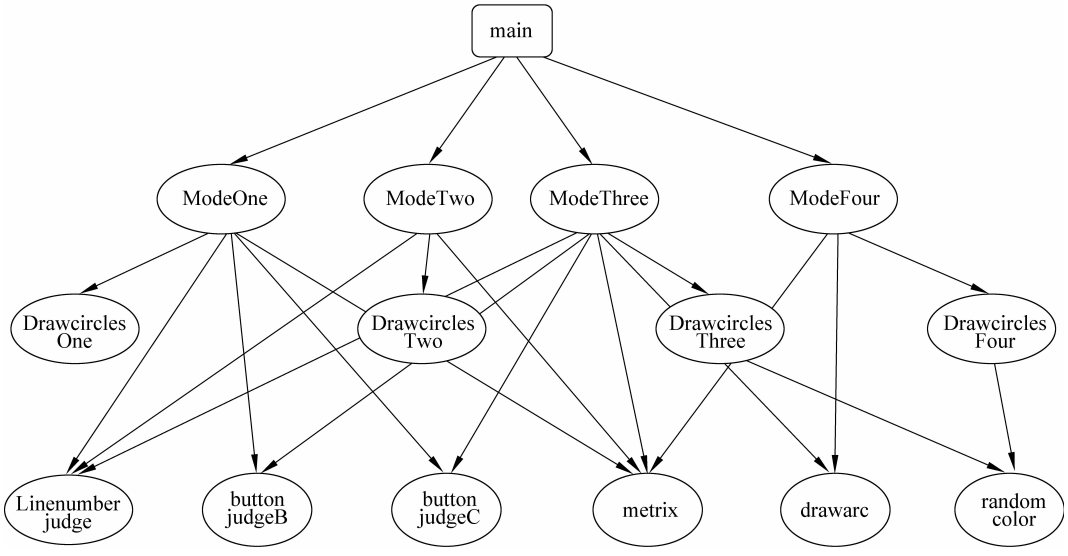


图 5-1 图形界面直接输入图并产生数据结构的算法框架

在 main 子图(见图 5-2)中选择接下来采用的画图模式。mode1 为手动输入无向图,mode2 为随机输入无向图,mode3 为手动输入有向图,mode4 为随机输入有向图。可以看到,判断模式的方法是折半查找法。随机输入有向图(见图 5-3),以颜色代表输出方向,Randomcolor 子程序(见图 5-4)用于选取随机颜色。

Randomcolor 子程序中的核心函数是 Random_Extended_Color。该函数会随机生成一个 0～241 的整数,代表了 242 种不同颜色。Color 在这个子程序中就是代表颜色的整型数组,不允许颜色为黑色或白色。

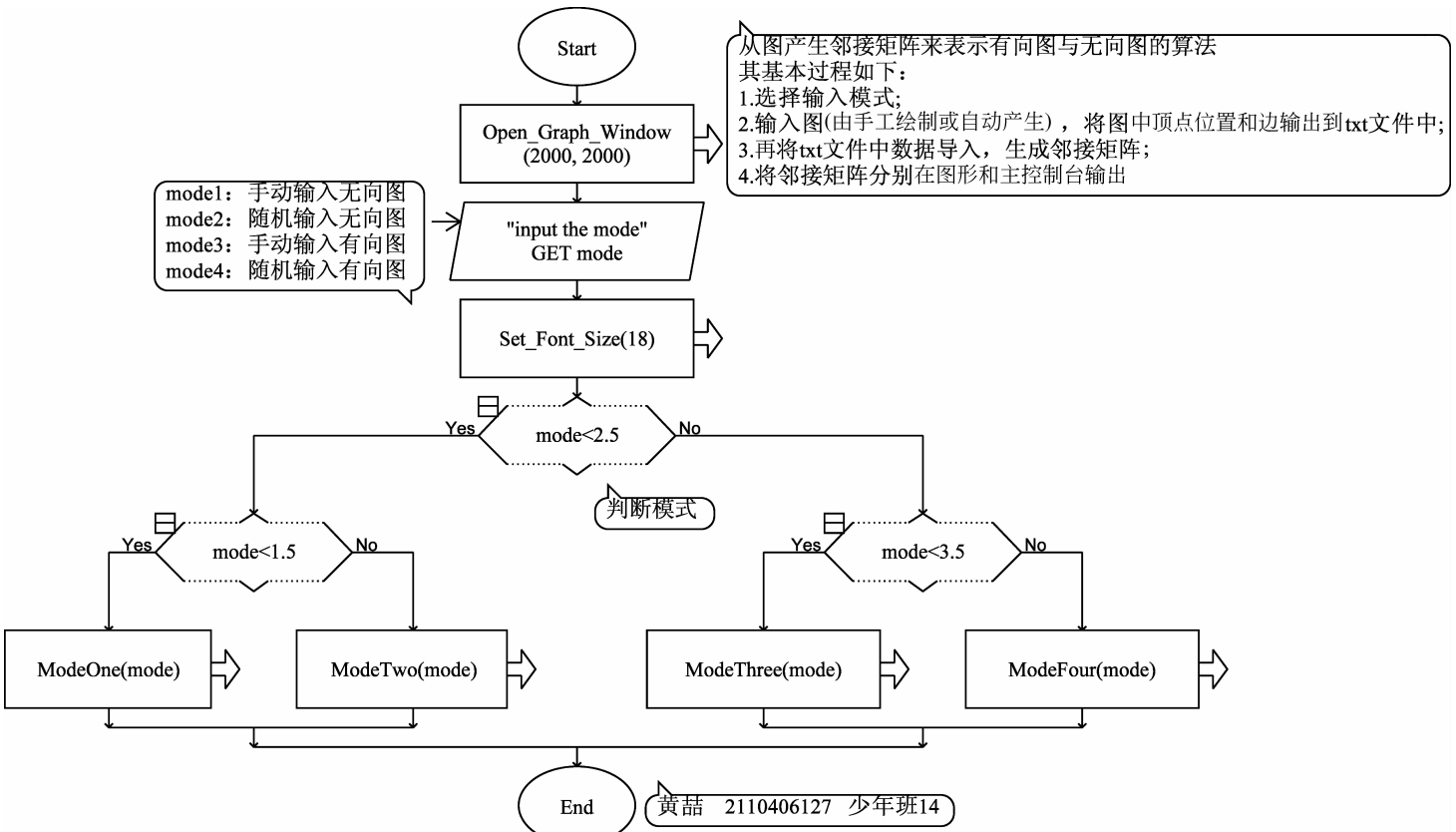


图 5-2 图形界面直接输入图并产生邻接矩阵的 main 子图

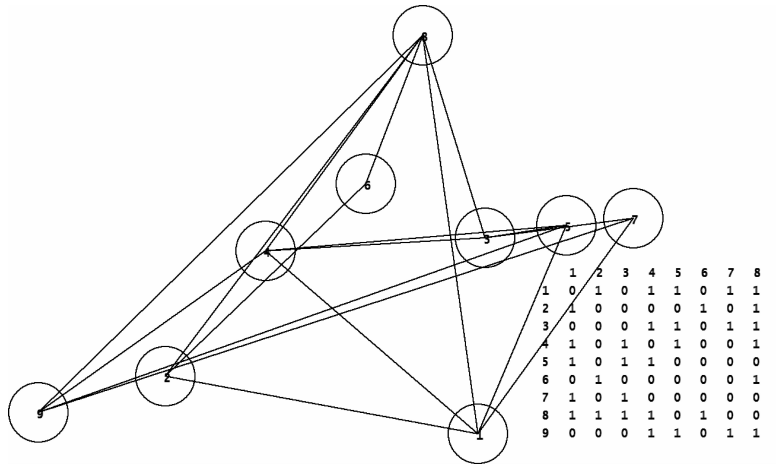


图 5-3　一个随机无向图及邻接矩阵的输出案例

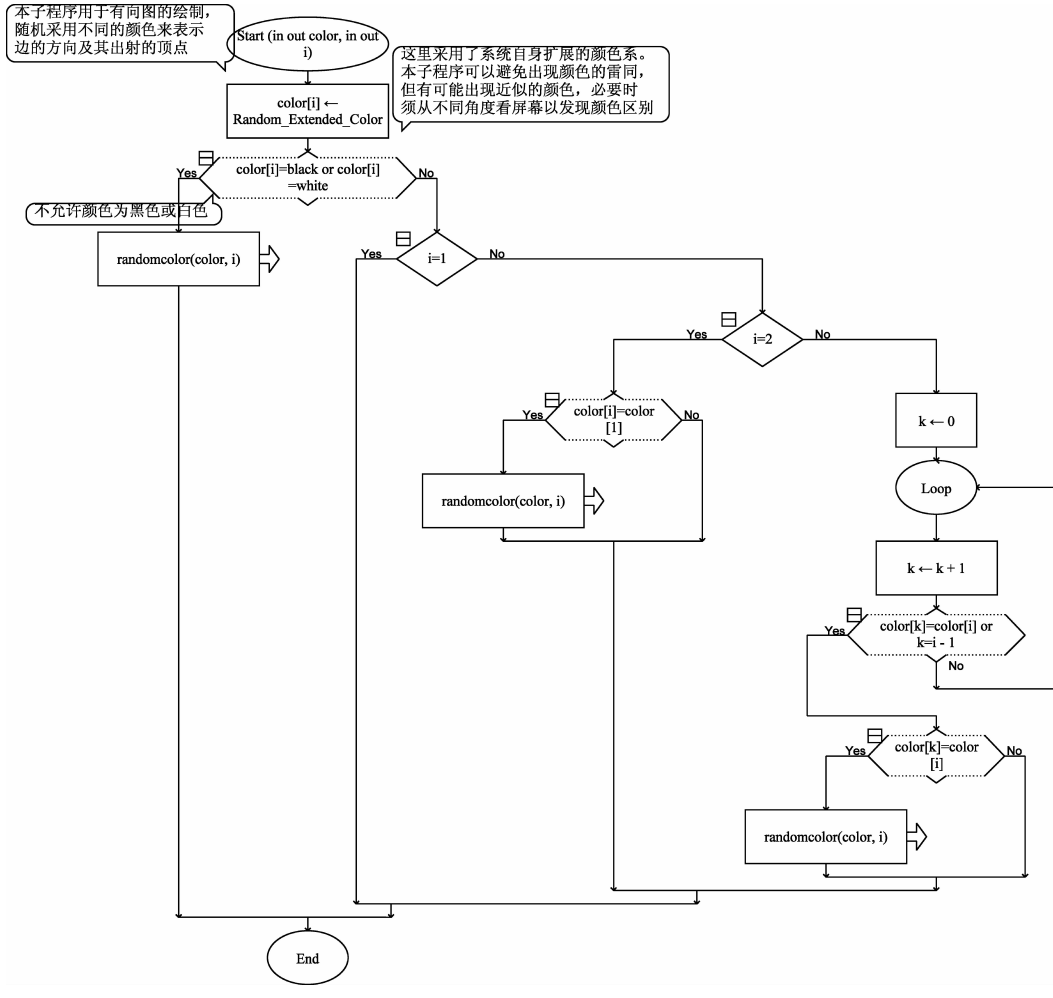
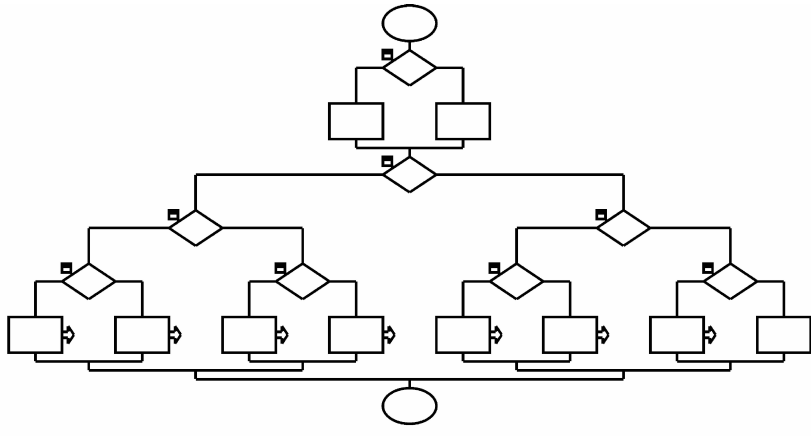
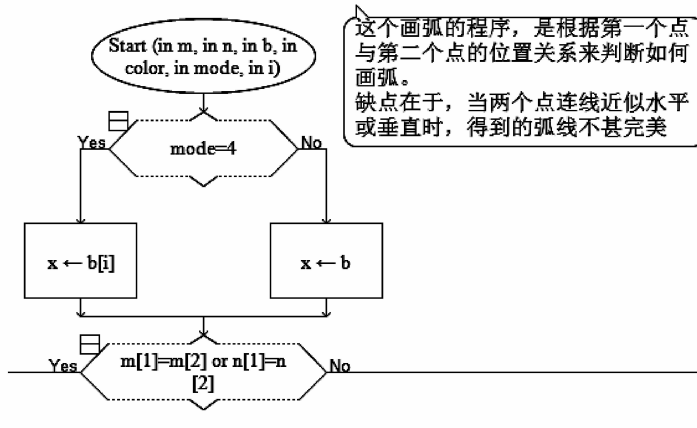


图 5-4　图形界面直接输入图并产生邻接矩阵的 Randomcolor 子程序

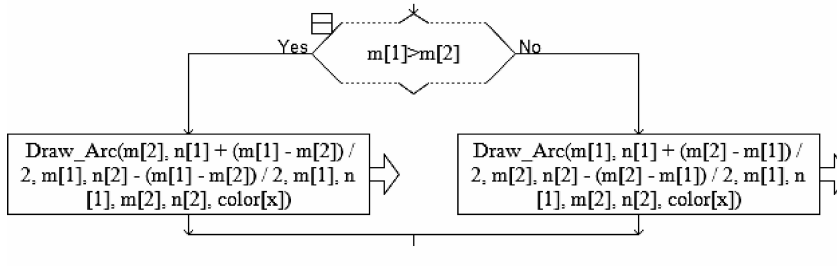
drawarc 子程序，即画弧，专门用于有向图的绘制（见图 5-5）。这是根据两个圆位置的 8 种情况，通过折半来确定，然后以逆时针方向画弧。图 5-6 给出了其中 4 种情况。另有 4 种情况是两圆圆心水平位置相等或垂直位置相等时的特殊情况，会画出半圆。该子程序中 m、n 两个数组分别为两个圆的横坐标与纵坐标。b 为出发圆的颜色，在子程序中用 x 来表示。



(a) 缩略图

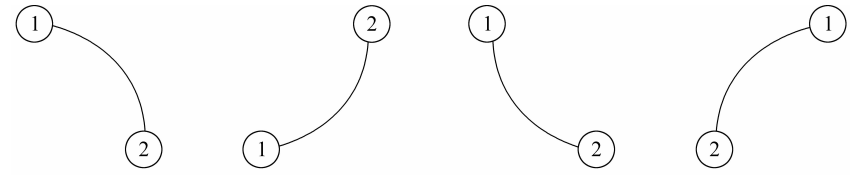


(b) 参数传递



(c) 画弧的语句

图 5-5　图形界面直接输入图并产生邻接矩阵的 drawarc 子程序



(a) 从右下到左上　(b) 从左下到右上　(c) 从左下到右下　(d) 从右上到左下

图 5-6　弧线绘制的 4 种基本案例

4. 图形、交互和 I/O 设计

输入部分如下：

(1) 提示用户输入图的绘制模式。

(2) 手工绘制模式和随机绘制模式的输入方法不同。

(3) 在手工绘制模式(mode1 和 mode3)视窗界面中使用鼠标绘制不同位置上的顶点和两个顶点之间的边。

(4) 在随机绘制模式(mode2 和 mode4)下，提示用户输入顶点和边的数量。

输出部分如下：

(1) 在视窗界面中显示图的绘制结果。

(2) 在视窗界面中显示图的邻接矩阵。

(3) 将图的绘制形态和邻接矩阵分别保存到文件中。

5. 算法测试

本案例的算法测试结果以手动方式产生有向图为例，结果见图 5-7(a)和图 5-7(b)。注意，adjmatrix.csv 文件的内容为自动生成的邻接矩阵，其格式与内容显示在图形输出屏幕上(见图 5-7(a)右下角)，在有向图的邻接矩阵中，∞用 X 替代，读者可按需要进行修改。图 5-7(b)列出了算法输出的 graph_data.csv 文件的格式与内容。

6. 算法小结

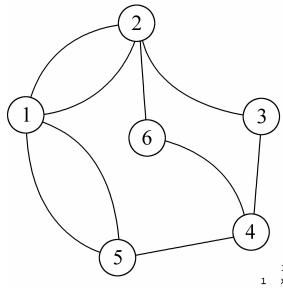
本算法完成了有向图和无向图在图形界面上的直接绘制，以及图的数据结构和绘制形态的文件输出。为后续的图论算法的测试提供了一个十分方便的工具。

根据实践结果可以发现，图论问题的求解和算法测试的一大难点就是将图的数据正确地在文件中进行描述和录入，不仅工作量大，而且十分容易出错。甚至出现有的用户为了避免数据的录入，不加检查就使用他人已经录入且存在错误的数据文件。

本例为图算法的编制和测试提供了简便且可验证的数据源。

7. 改进思路

可以继续改进和完善的内容如下：



	1	2	3	4	5	6
1	x	1	x	x	1	x
2	1	x	1	x	x	x
3	x	x	x	1	x	x
4	x	x	x	x	1	1
5	1	x	x	x	x	x
6	x	1	x	x	x	x

(a) 手动绘制6个顶点9条边的有向图

	A	B	C
1	6		6个顶点
2	1		顶点1
3	470		X坐标
4	740		Y坐标
5	2		
6	722		
7	943		
8	3		第1行是说明
9	1003		有几个顶点；
10	735		第2行到第19
11	4		行说明图的顶
12	980		点坐标序号和
13	473		所在坐标位置
14	5		
15	678		
16	415		
17	6		
18	746		
19	687		第20行是说明有
20	9		几条边；
21	1		从21到38行
22	2		是对有向图边
23	2		走向的依次描述
24	3		
25	3		
26	4		
27	4		
28	5		
29	5		
30	1		
31	2		
32	1		
33	1		
34	5		
35	6		
36	2		
37	4		
38	6		

(b) graph_data.csv文件的格式与内容

图 5-7 手动方式产生有向图的实例

(1) 将图数据结构改变成可以直接计算的形式，例如将以文件形式输出的图数据结构中的 X，Y 轴的数字标示取消，无向图的数据结构可以直接用于各种程序语言编写的图算法输入；而有向图数据结构中的 X 代换以 999 则可以用于大部分后续的有向图算法计算或测试。

(2) 本算法没有考虑两个顶点之间的边多于两条的多图、一个顶点上形成的自环(loop)以及有向网和无向网的情况，必要时，读者可以自行扩展与修改。

(3) 可以考虑修改随机图的产生以配合不同的图论算法的验证。例如，中国邮递员问题需要生成带权连通图，旅行商问题需要产生带权完全图。

(4) 随着顶点的增加，必要时可以调整顶点的直径大小，使得图结构更加清晰。

(5) 如何使用可视化手段表示有向网和无向网中边的权重，仍然需要继续探索。

5.2 用回溯法与空间树求解哈密顿回路的存在问题

1. 案例问题

天文学家哈密顿(William Rowan Hamilton)1857 年提出了如下问题：在一个有多个城市的地图网络中，寻找一条从给定的城市出发回到该城市且沿途恰好经过所有其他

城市一次的路径(见图 5-8)。

这个问题和著名的七桥问题的不同之处在于,某些城市之间的旅行不一定是双向的。例如 A→B 不允许,但 B→A 是允许的。

换一种说法,对于一个给定的网络,确定起点后,如果存在一条路径,最后又回到原起点,且满足每个顶点只被访问一次的情况下能穿过这个网络,我们就说这个网络存在哈密顿回路。

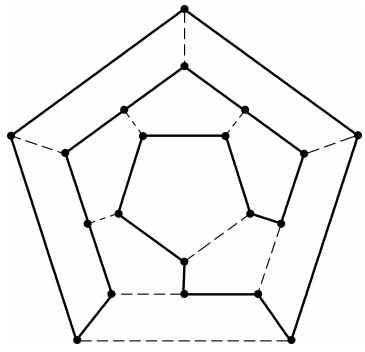


图 5-8 哈密顿回路示例

2. 算法设计思路

本算法是用回溯法求解一个无向图中是否存在哈密顿回路的问题。

算法的主要步骤如下(见图 5-9):

(1) 由文件直接输入图形数据并建立邻接矩阵。

(2) 将每个点的状态初始化(标记为 0),选择源点 A 标记为 1。

(3) 搜索与它相邻的顶点 X,标记为 1。

(4) 依次向下搜索与上一点相邻的顶点,标记为 1。

(5) 无路可走时,判断是否遍历所有的点以及是否是一条回路。若是,直接输出回路;若不是,向前逐步回溯重新搜索。

利用 RAPTOR 图形界面进行算法问题和结果的可视化展示,具体功能如下:

(1) 利用邻接矩阵复原同构的图论问题图,在图形界面上同时输出邻接矩阵。

(2) 将计算结果使用不同于原始图中边的色彩表达实际上探索到的哈密顿回路。

这样做的好处是,方便用户检查输入算法的文件数据以及图形是否存在偏差和问题。当然,输出结果的正确与否也更加容易查对。

3. 主要算法流程

算法框架如图 5-9 所示。算法中的各子图如图 5-10 至图 5-15 所示。

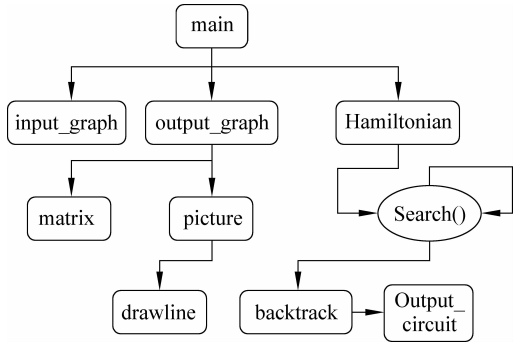


图 5-9 用回溯法求解哈密顿回路的存在问题的算法框架

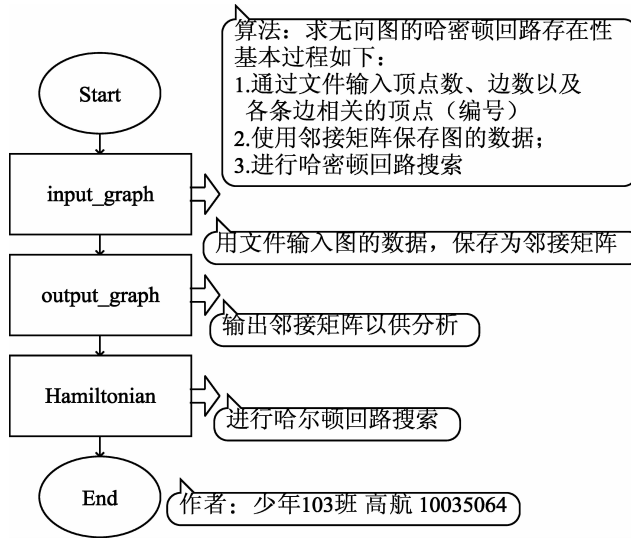
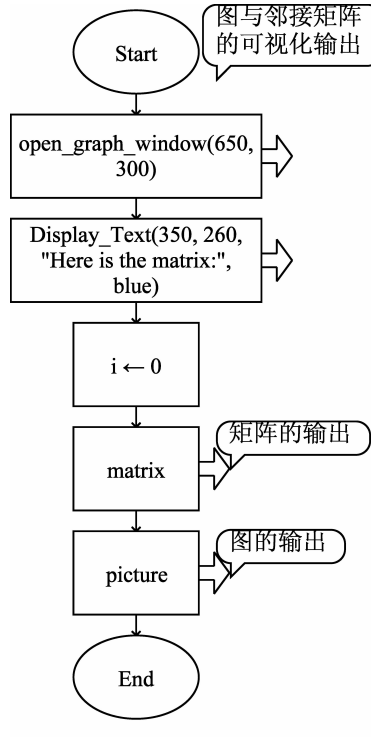
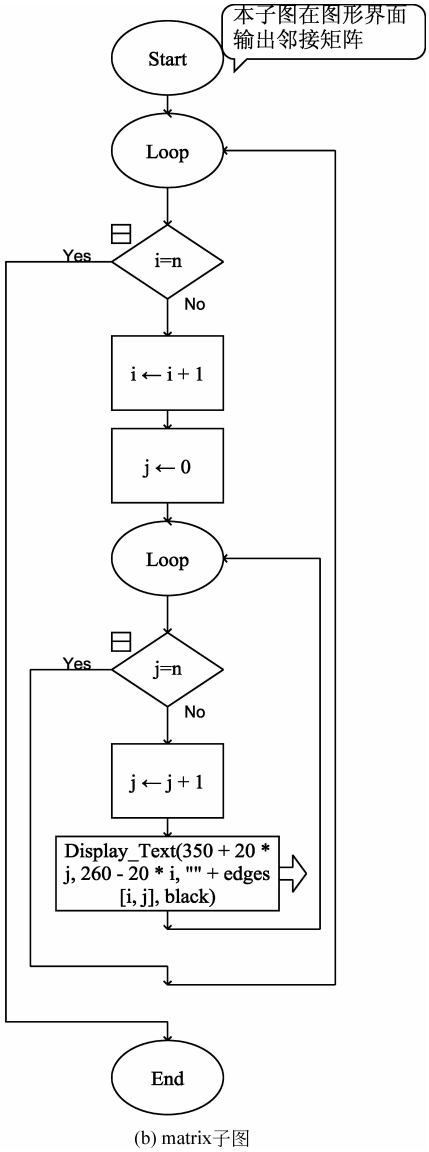


图 5-10　用回溯法求解哈密顿回路存在问题的 main 子图

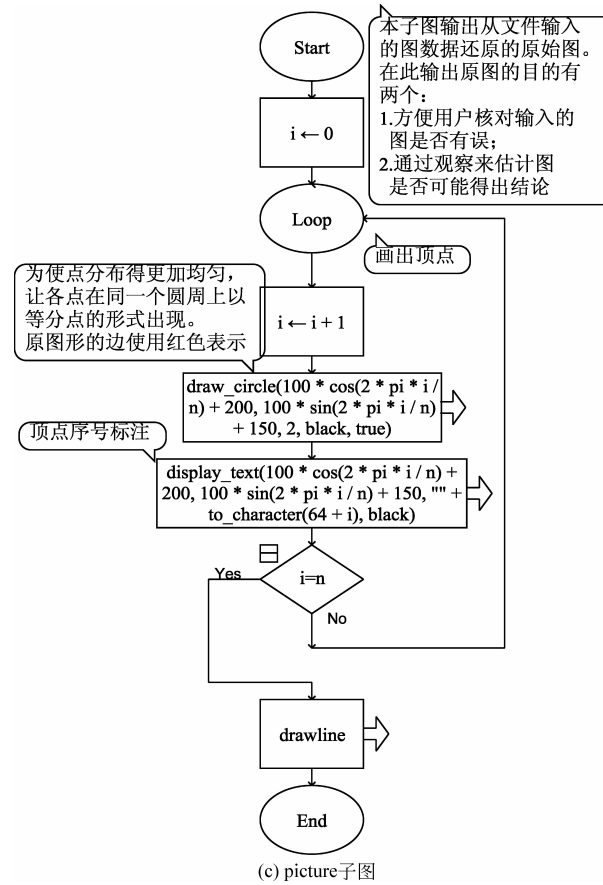


(a) output_graph子图

图 5-11　求解哈密顿回路存在问题的 output_graph 子图、matrix 子图、picture 子图和 drawline 子图



(b) matrix子图



(c) picture子图

图 5-11（续）