迷宫问题实验设计报告

1. 题目要求

输入格式：

从“maze.txt”文件中读取数据，第一行依次为迷宫节点的个数n，迷宫的起点begin，迷宫的终点数endn，随后为endn个数字代表迷宫终点，要求找到路径即可（不要求最短）

接下来若干行都为两个数字，代表这两个节点相连，最后一行为0 0代表输入结束

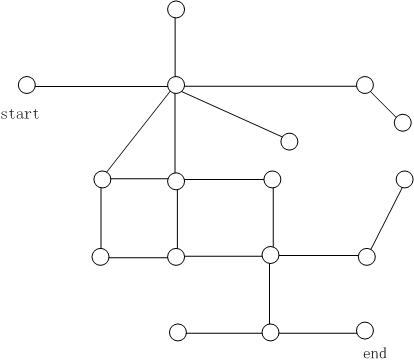
输出格式：

输出共endn行，第i行输出第i个终点的寻路结果，如果找到路径则输出“Path from m to n : ”，m为起点，n为终点，随后输出路径途中的节点，用->相连，最后输出“length is : k ”，k为路径长度。如果没有找到则输出“No path from m to n”。

输入样例： 输出样例：

17 1 1 17 Path from 1 to 17 : 1->3->7->8->12->16->17 length is : 8

1 3



2

1

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

2 3

3 4

3 6

3 7

3 8

4 5

7 8

7 10

8 9

8 11

9 12

10 11

11 12

12 13

12 16

13 14

16 15

16 17

0 0

1. 设计思路

将问题要求分成三个执行阶段

输入阶段：

从“maze.txt”文件中读取数据，用长度为endn的一维数组储存所有终点，用n行n列的二维数组作为邻接矩阵arr[n][n]储存任意两个节点之间的关系，用长度为n的数组x初始化为0储存节点是否被遍历的信息。

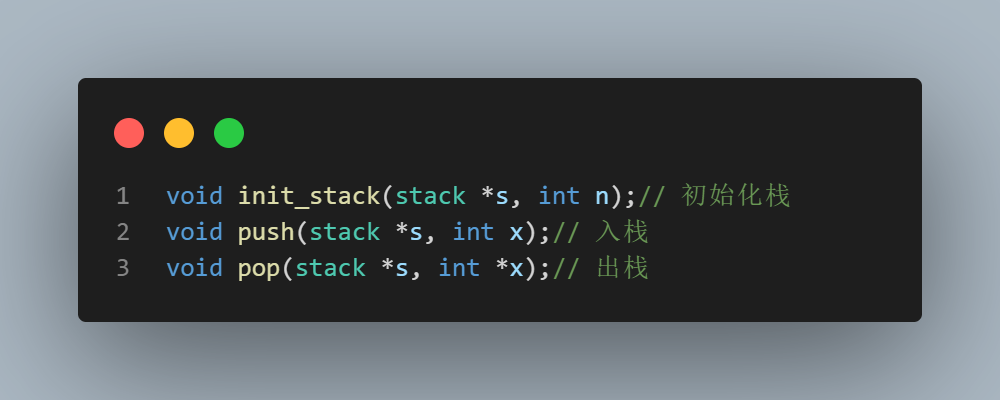
处理阶段：

用栈的相关操作实现寻路逻辑

输出阶段：

依次输出栈里储存的节点信息与路径长度

1. 设计思路
2. 构造栈的结构与功能：首先定义一个栈结构，包含栈底指针和栈顶指针。随后分别构造为栈分配空间操作，元素入栈操作和元素出栈操作的函数。
3. 储存图的信息：用邻接矩阵arr[n][n]储存任意两个节点之间的关系。如若节点1和节点2相连，则设定arr[0][1]与arr[1][0]的值为1。
4. 循环寻路：从二维数组的第begin-1行0列开始遍历该行数组，若x[n]为0且arr[begin-1][i]为1将节点i入栈，将节点关系设置成0，节点信息设置成1，i设置为begin，若此时i为终点节点则跳出循环；若全为0则将栈顶节点出栈，将begin设置为栈顶节点，若此时栈顶指针和栈底指针相同也跳出循环。
5. 输出最终结果：从栈底指针开始逐个输出节点信息，直到指向栈顶指针停止，随后输出栈顶指针减去栈底指针的值代表路径长度
6. 代码说明
7. 结构体定义
8. 函数定义和功能

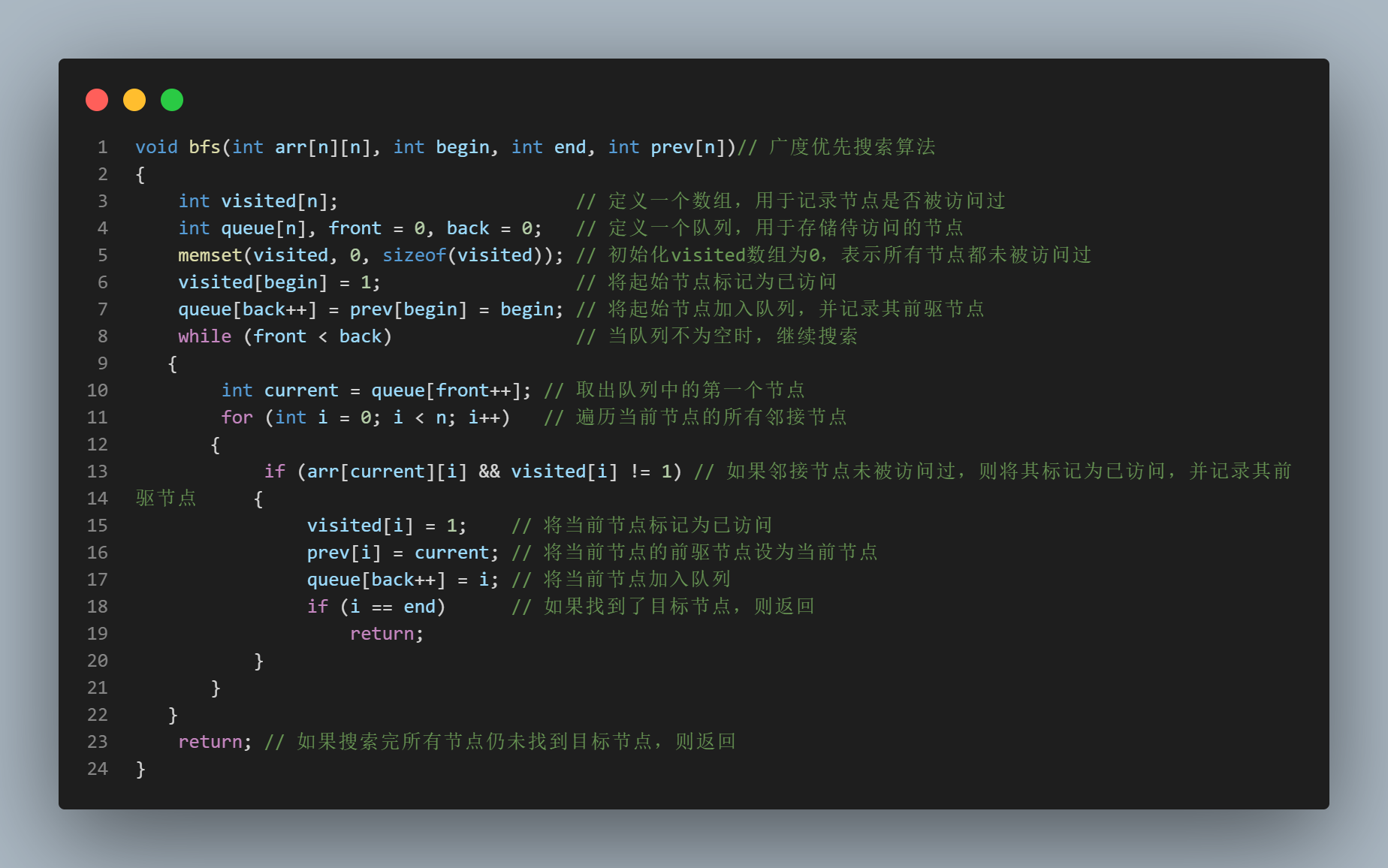


1. 函数复杂度分析

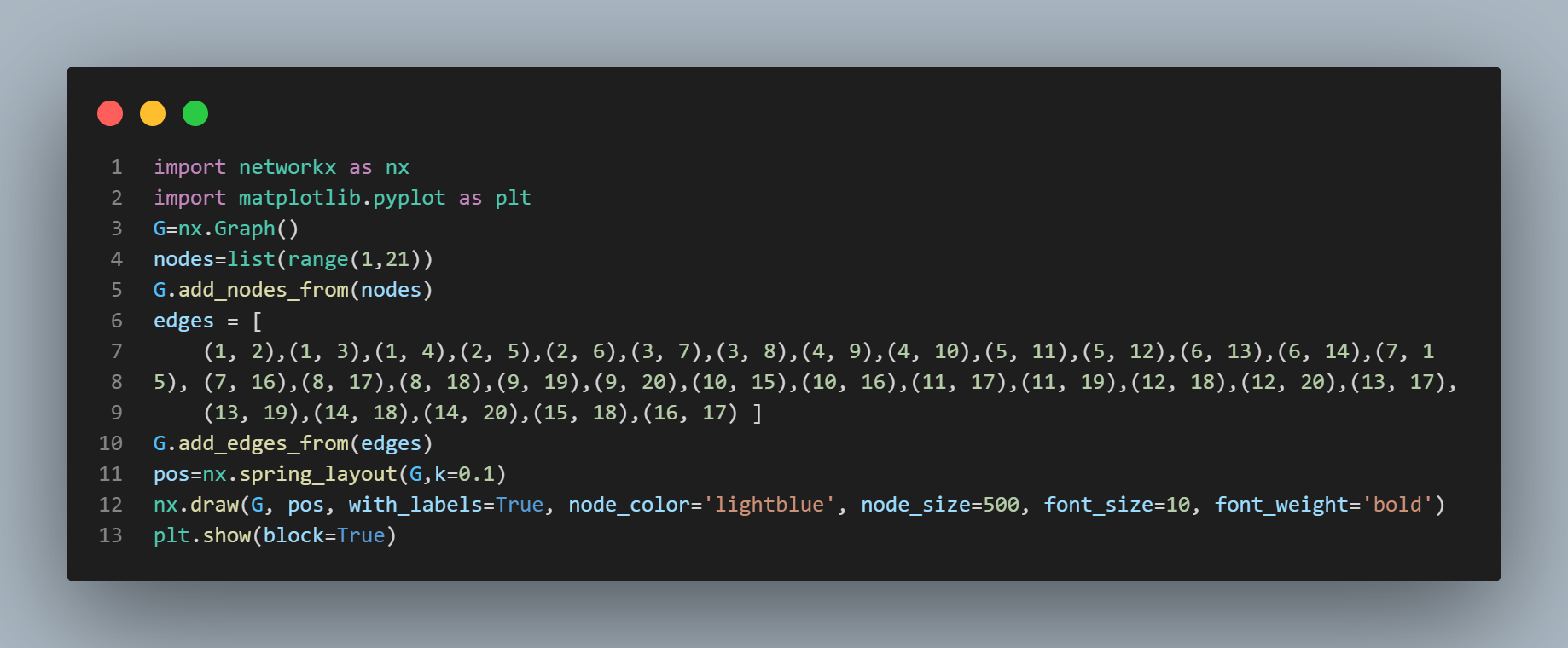
|  |  |  |
| --- | --- | --- |
| 函数 | 时间复杂度 | 空间复杂度 |
| void init\_stack(stack \*s, int n) | O(1) | O(n) |
| void push(stack \*s, int x) | O(1) | O(1) |
| void pop(stack \*s, int \*x) | O(1) | O(1) |
| int main() | O(mn2) | O(n2) |

1. 扩展实验

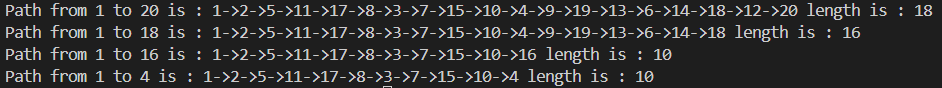
基础实验使用的算法为按照节点的编号大小顺序进行寻路，所以编号顺序会极大地影响路径长度。为找到最短路径，引入广度优先搜索（BFS）算法。该算法在、引入了一个长度为n的数组依次保存被访问到的节点编号和另一个长度为n的数组储存数组下标的前一个节点。具体实现如下：



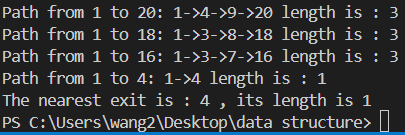
为直观地体现差距，使用以下python程序生成一个图



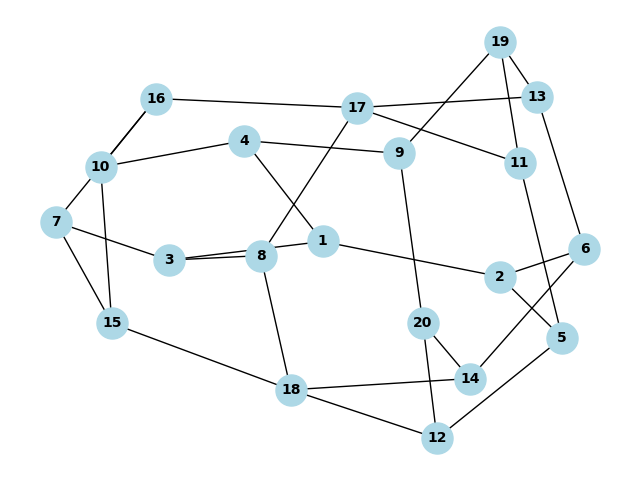
在生成的迷宫中设定从节点1出发，终点有4个，分别为20，18，16，4，对比两种算法的输出：



简单算法：



BFS算法：



生成的图：

由此可见，简单算法仅仅实现了找到路径的功能，而BFS算法则找到了最短的路径。而存在多个出口找到最短出口的功能也能靠依次完成多个出口的寻路后通过比较路径的长度实现