一、调试成功程序及说明

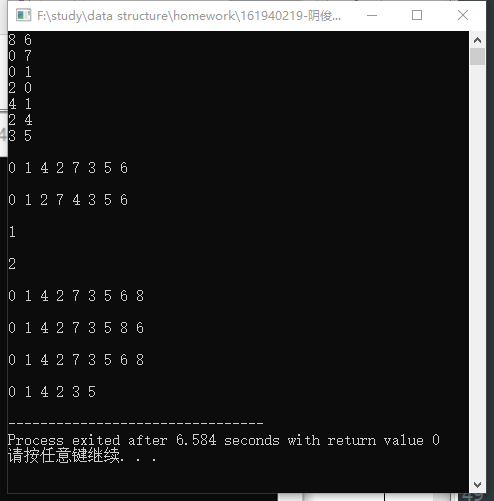
1. 图的深度优先和广度优先遍历；

算法思想：

DFS：类似树的先序遍历，利用递归实现。函数起初先访问该结点，再标记该结点已被访问，然后对于该结点的每一个邻接点，若还未被访问过，再次递归调用DFS访问它。

BFS：类似树的层序遍历，利用队列实现。先令起始结点入队，并标记已访问。只要队不空就出队，并访问它，对于其每一个未被访问过的邻接点，再次入队，并在结点入队时标记其已访问。

运行结果：



结果分析：

T(n) = O()

S(n) = O()

附源程序。

2. 编程实现Dijkstra算法；

算法思想：

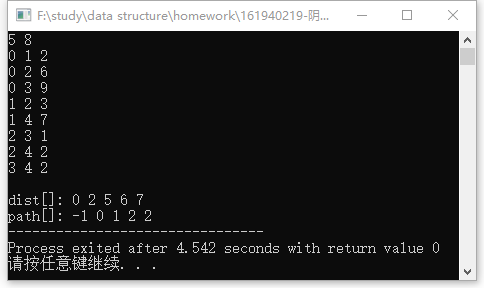
1.起初Dist[]都初始化正无穷，path[]都初始化为-1。

2.每次循环从未收录的顶点中选择一个dist值最小的结点，如果找到，执行步骤3，反之执行步骤4。

3.将该顶点收录，并访问所有未被收录过的邻接点，若有更短路径，更新其dist值，并且设置其前驱结点。

4.输出结果。

运行结果：



结果分析：

T(n) = O()

S(n) = O(n)

附源程序。

3. CSP题目

问题描述 ：Alice和Bob正在玩井字棋游戏。 井字棋游戏的规则很简单：两人轮流往3\*3的棋盘中放棋子，Alice放的是“X”，Bob放的是“O”，Alice执先。当同一种棋子占据一行、一列或一条对角线的三个格子时，游戏结束，该种棋子的持有者获胜。当棋盘被填满的时候，游戏结束，双方平手。   
　　Alice设计了一种对棋局评分的方法：   
　　- 对于Alice已经获胜的局面，评估得分为(棋盘上的空格子数+1)；   
　　- 对于Bob已经获胜的局面，评估得分为 -(棋盘上的空格子数+1)；   
　　- 对于平局的局面，评估得分为0；

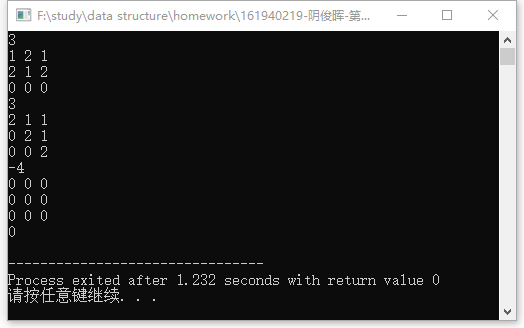
　　例如上图中的局面，Alice已经获胜，同时棋盘上有2个空格，所以局面得分为2+1=3。   
　　由于Alice并不喜欢计算，所以他请教擅长编程的你，如果两人都以最优策略行棋，那么当前局面的最终得分会是多少？   
输入格式   
　　输入的第一行包含一个正整数T，表示数据的组数。   
　　每组数据输入有3行，每行有3个整数，用空格分隔，分别表示棋盘每个格子的状态。0表示格子为空，1表示格子中为“X”，2表示格子中为“O”。保证不会出现其他状态。   
　　保证输入的局面合法。(即保证输入的局面可以通过行棋到达，且保证没有双方同时获胜的情况)   
　　保证输入的局面轮到Alice行棋。   
输出格式   
　　对于每组数据，输出一行一个整数，表示当前局面的得分。   
样例输入   
3   
1 2 1   
2 1 2   
0 0 0   
2 1 1   
0 2 1   
0 0 2   
0 0 0   
0 0 0   
0 0 0   
样例输出   
3   
-4   
0   
样例说明   
　　第一组数据：   
　　Alice将棋子放在左下角(或右下角)后，可以到达问题描述中的局面，得分为3。   
　　3为Alice行棋后能到达的局面中得分的最大值。   
　　第二组数据：

Bob已经获胜(如图)，此局面得分为-(3+1)=-4。   
　　第三组数据：   
　　井字棋中若双方都采用最优策略，游戏平局，最终得分为0。

算法思想：

暴力枚举9！种情况。

运行结果：



结果分析：

T(n) = O()

S(n) = O()

附源程序。

4．CSP题目

问题描述：小刘承包了很多片麦田，为了灌溉这些麦田，小刘在第一个麦田挖了一口很深的水井，所有的麦田都从这口井来引水灌溉。 为了灌溉，小刘需要建立一些水渠，以连接水井和麦田，小刘也可以利用部分麦田作为“中转站”，利用水渠连接不同的麦田，这样只要一片麦田能被灌溉，则与其连接的麦田也能被灌溉。现在小刘知道哪些麦田之间可以建设水渠和建设每个水渠所需要的费用（注意不是所有麦田之间都可以建立水渠）。请问灌溉所有麦田最少需要多少费用来修建水渠。

输入格式：

输入的第一行包含两个正整数n, m，分别表示麦田的片数和小刘可以建立的水渠的数量。麦田使用1, 2, 3, ……依次标号。

接下来m行，每行包含三个整数ai, bi, ci，表示第ai片麦田与第bi片麦田之间可以建立一条水渠，所需要的费用为ci。

输出格式：

输出一个整数，表示灌溉所有麦田所需要的最小费用，及水渠连接说明。

**问题分析：**这个问题可以用最小生成树算法实现。

输入样例:

4 4

1 2 1

2 3 4

2 4 2

3 4 3

输出样例:

6

建立以下3条水渠：麦田1与麦田2、麦田2与麦田4、麦田4与麦田3。

算法思想：

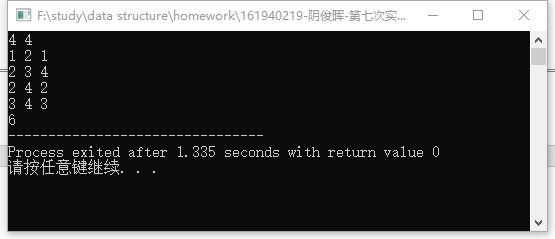
Prim算法：

1. 先将任意一个结点收录入生成树上。
2. 找一个不在生成树上且dist值最小的v。
3. 若找到v，执行步骤4。

若未找到，执行步骤5。

1. 遍历v所有不在生成树上的邻接点，若v到i的权值比dist值小，就更新dist值，并回到步骤2。
2. 输出最小生成树所有边权值之和。

运行结果：



结果分析：

T(n) = O()

S(n) = O(n)

附源程序。

-------------------------------------------------------------------------

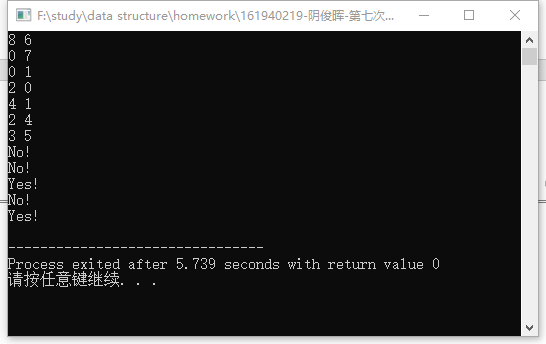
选做题**7.27 采用邻接表存储结构，编写一个判别无向图中任意给定的两个顶点之间是否存在一条长度为k的简单路径的算法。**

算法思想：

核心部分利用DFS递归求解。

1. 递归函数起初判断当前结点是否为终点且路径长度为k，若两者同时满足，执行步骤2，反之执行步骤3。
2. 将flag标记为存在长度为k的简单路径，并执行步骤4。
3. 将当前结点标记为已访问，路径长度加1，并以当前结点为起点，调用递归函数访问其每一个邻接点。
4. 输出结果，程序结束。

运行结果：



结果分析：

T(n) = O()

S(n) = O(n)

附源程序。

选做题**7.34 试编写一算法，给有向无环图G中每个顶点赋以一个整数序号，并满足以下条件：若从顶点i到顶点j有一条弧，则应使i<j。**

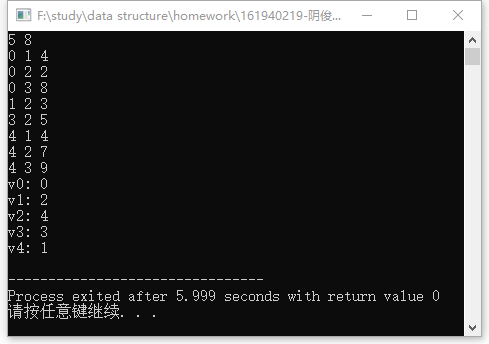
算法思想： （拓扑排序）

1. 找一个未被访问过的且入度为0的结点v。
2. 若找到v，执行步骤3。

若没找到，执行步骤4。

1. 将v标记为已访问。赋予v编号。从图中去除该结点及其所相连的边。
2. 输出每个结点的编号。

运行结果：



结果分析：

T(n) = O(v\*e)

S(n) = O()

附源程序。

选做题**7.37 试设计一个求有向无环图中最长路径的算法，并估计其时间复杂度。**

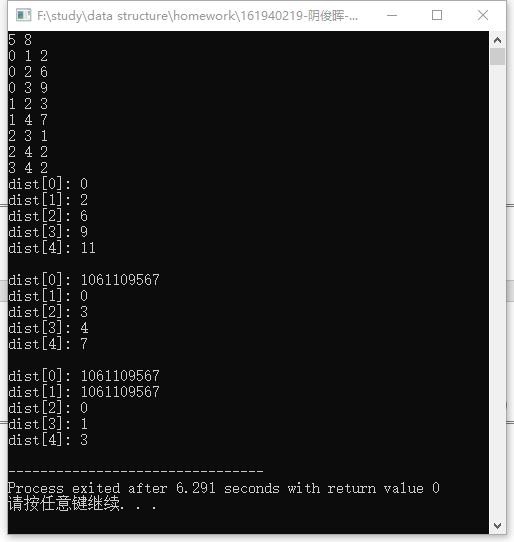
算法思想：

思路基本同拓扑排序。

若从起点到该结点不可达，则将该结点dist值设为正无穷。

若从起点到该结点可达，拓扑排序时每次更新dist值即可。

运行结果：



结果分析：

T(n) = O()

S(n) = O(v)

附源程序。

二、代码行数及小结

|  |  |
| --- | --- |
| 题目 | 代码行数 |
| mygraph.h | 32 |
| mygraph.cpp | 248 |
| 1 | 14 |
| 2 | 12 |
| 3 | 209 |
| 4 | 81 |
| 选7.27 | 72 |
| 选7.34 | 14 |
| 选7.37 | 18 |
| 总计 | 700 |