姓名：冉湖

学号：202322040432

课程作业选题：选题 2

作业详情：

1.编写了三种最短路算法（Dantzig、dijkstra、ford）；

2.编写了两种最小生成树算法测试（Kruskal、Prim）

3.编写了匈牙利算法（hungarian）

4.编写了最优匹配算法（KM_OptimalMatch）

代码说明：

代码共有 7 个文件如下表所陈列：

文件名	简介
Hungarian.cpp	匈牙利算法与最优匹配算法的定义
Hungarian.hpp	匈牙利算法与最优匹配算法的声明
MinSpanTree.cpp	最小生成树算法的函数定义
MinSpanTree.hpp	最小生成树算法的函数声明
ShortestPath.cpp	最短路算法的函数定义
ShortestPath.hpp	最短路算法的函数声明
test_main.cpp	主要测试主文件

算法结构：

1.图的输入形式：

使用一个二维数组表示边的信息，unsigned int edges[edgeCounts][3]。edgeCounts 表示该图一共有多少个边。Edges[i][0]表示第i个边的起点节点，Edges[1][1]表示第i个边的终点节点，Edges[i][0]表示第i个边的权值。

2.图的表达形式：

调用函数 AdjMatrix(edges)，会将该二维数组转化为一个邻接矩阵 adjMat。邻接矩阵 adjMat 将用于最短路算法和最小生成树算法。

调用函数 InitGraph(edges,G)，会将该二维数组转化为一个邻接链表 adjList。邻接链表 adjList 将会用于匈牙利算法和最优匹配算法。

printAdjMatrix(adjMat)和 PrintAdjList(G)两个函数分别用于输出对应图的邻接矩阵和邻接链表。

3.算法测试：

在 test_main.cpp 中会引用上述算法所对应的头文件并对所有的算法进行测试。不同的 unsigned int edges[edgeCounts][3]可以进行多组测试，但是由于本人代码水平有限，每启用一个 edges，需要去往 MinSpanTree.hpp 中手动更改顶点数量 vexCounts 和边数 edgeCounts。

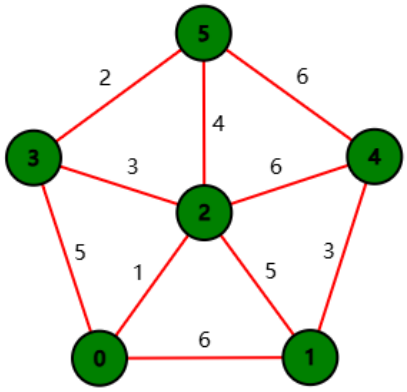
测试用例与测试结果：

1.最短路径算法

测试使用到的边信息：

```
unsigned int edges[edgeCounts][3] = {{0, 1, 6}, {0, 2, 1}, {0, 3, 5}, {1, 2, 5}, {1, 4, 3}, {2, 3, 3}, {2, 4, 6}, {2, 5, 4}, {3, 5, 2}, {4, 5, 6}};
```

对应的图为：



算法运行结果截图如下：

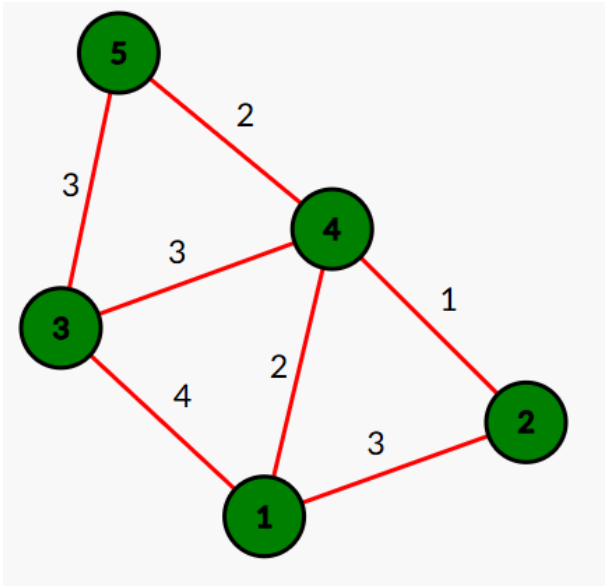
```
**************************最短路算法测试**************************
Dantzig 最短路径:
3 -> 0: 3 -> 2 -> 3, 距离为 4
3 -> 1: 3 -> 2 -> 3, 距离为 8
3 -> 2: 3 -> 3, 距离为 3
3 -> 4: 3 -> 5 -> 3, 距离为 8
3 -> 5: 3 -> 3, 距离为 2
Dijkstra 最短路径:
3 -> 0: 3 -> 2 -> 0, 距离为 4
3 -> 1: 3 -> 2 -> 1, 距离为 8
3 -> 2: 3 -> 2, 距离为 3
3 -> 4: 3 -> 5 -> 4, 距离为 8
3 -> 5: 3 -> 5, 距离为 2

Ford 最短路径:
3 -> 0: 3 -> 2 -> 0, 距离为 4
3 -> 1: 3 -> 2 -> 1, 距离为 8
3 -> 2: 3 -> 2, 距离为 3
3 -> 4: 3 -> 5 -> 4, 距离为 8
3 -> 5: 3 -> 5, 距离为 2
Press any key to exit...
```

2.最小生成树算法：

测试使用到的边信息：

```
unsigned int edges[edgeCounts][3] = {{0, 1, 3}, {0, 3, 2}, {3, 1, 1}, {2, 0, 4}, {2, 3, 3}, {4, 3, 2}, {4, 2, 3}};
```

对应的图为：



算法运行结果截图如下：

```
选择 D:\codeCPP\minimum_span_tree\test_main.exe
**********************************
姓名：冉湖
学号：202322040432
**********************************
邻接矩阵：
INF 3 4 2 INF
3 INF INF 1 INF
4 INF INF 3 3
2 1 3 INF 2
INF INF 3 2 INF
**************************最小生成树算法测试**************************
Kruskal :
4---2
4---1
5---4
4---3
-------------
Prim:
5--4
4--2
4--1
5--3
Press any key to exit...
```

算法运行结果是完全正确的

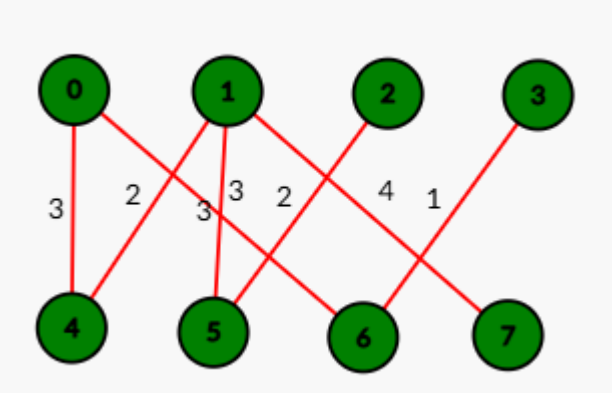
3.匈牙利算法测试：

该算法首先会调用函数 isBipartite(G)，判断输入的邻接链表是否是一个二分图。如果不会直接警告并退出。并会判断得出的算法是否是一个完美匹配。

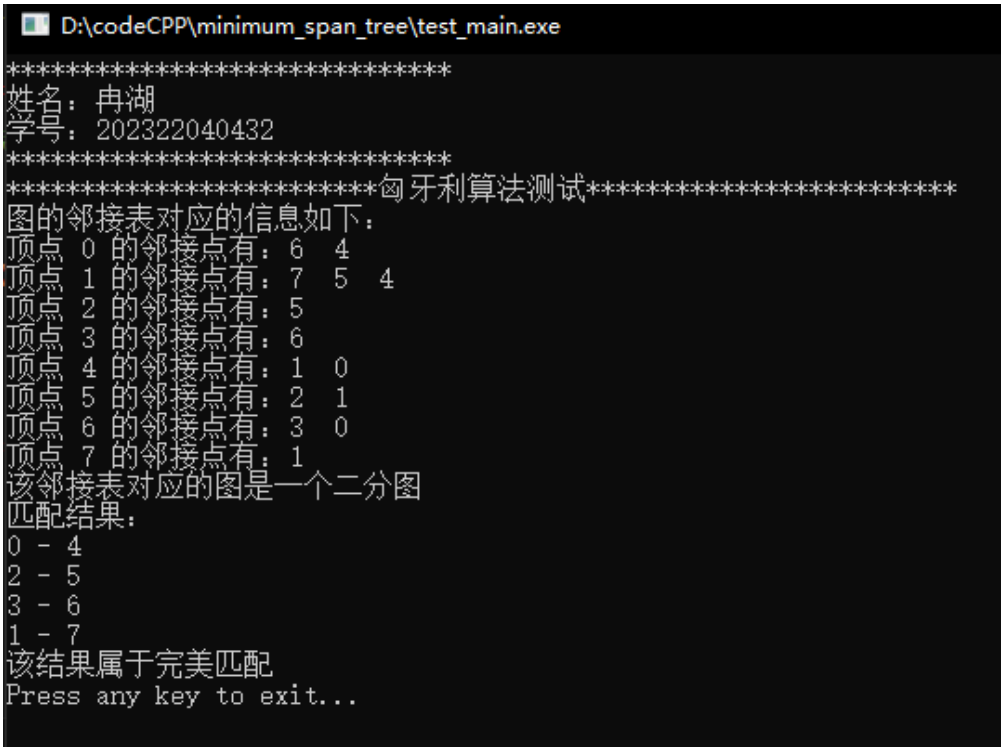
使用偶图的输入作为测试案例：

```
unsigned int edges[edgeCounts][3] = {{0, 4, 3}, {0, 6, 3}, {1, 4, 2}, {1, 5, 3}, {1, 7, 4}, {2, 5, 2}, {3, 6, 1}};
```

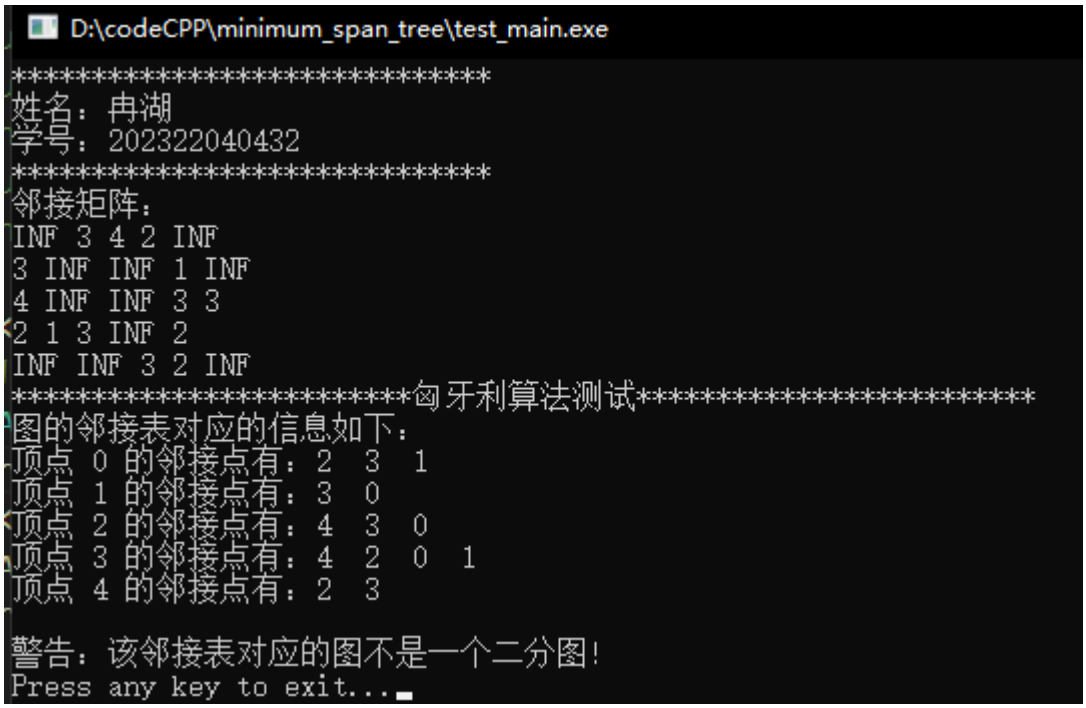
对应的图为：



测试结果如下：



若使用最小生成树的测试案例作为输入，则结果为：

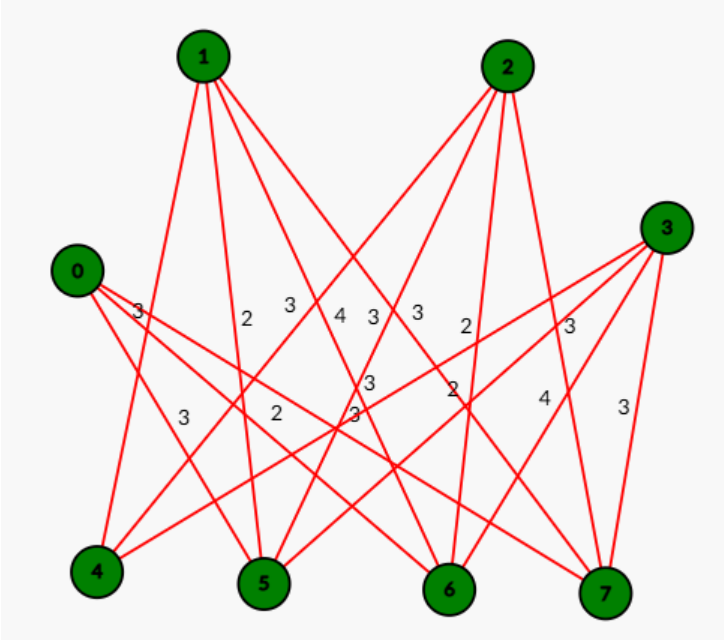


4.最优匹配算法测试：

Kuhn-Munkres 最优匹配针对完成偶图，因此算法会调用 isCompleteBipartite(G)判断输入的图是否是完全偶图。

```
unsigned int edges[edgeCounts][3] = {{0, 4, 3}, {0, 5, 3}, {0, 6, 2}, {0, 7, 3}, {1, 4, 1}, {1, 5, 2}, {1, 6, 4}, {1, 7, 3}, {2, 4, 3}, {2, 5, 3}, {2, 6, 2}, {2, 7, 3},{3, 4, 1}, {3, 5, 2}, {3, 6, 4}, {3, 7, 3}};
```

对应的图为：



测试结果如下：

```
D:\codeCPP\minimum_span_tree\test_main.exe
**********************************
姓名：冉湖
学号：202322040432
**********************************
**************************最优匹配算法测试**************************
顶点 0 的邻接点有：7  6  5  4
顶点 1 的邻接点有：7  6  5  4
顶点 2 的邻接点有：7  6  5  4
顶点 3 的邻接点有：7  6  5  4
顶点 4 的邻接点有：3  2  1  0
顶点 5 的邻接点有：3  2  1  0
顶点 6 的邻接点有：3  2  1  0
顶点 7 的邻接点有：3  2  1  0
属于完美匹配
匹配结果：
0 - 4 (代价：3)
1 - 5 (代价：3)
2 - 6 (代价：3)
3 - 7 (代价：3)
最优匹配的总代价为：12
Press any key to exit...
```