**算法设计与分析**

**第一次上机实验报告**

姓名： 粟锦 学号：U201817053 班级：软工1803班

**完成的排序**：

冒泡排序

插入排序

快速排序

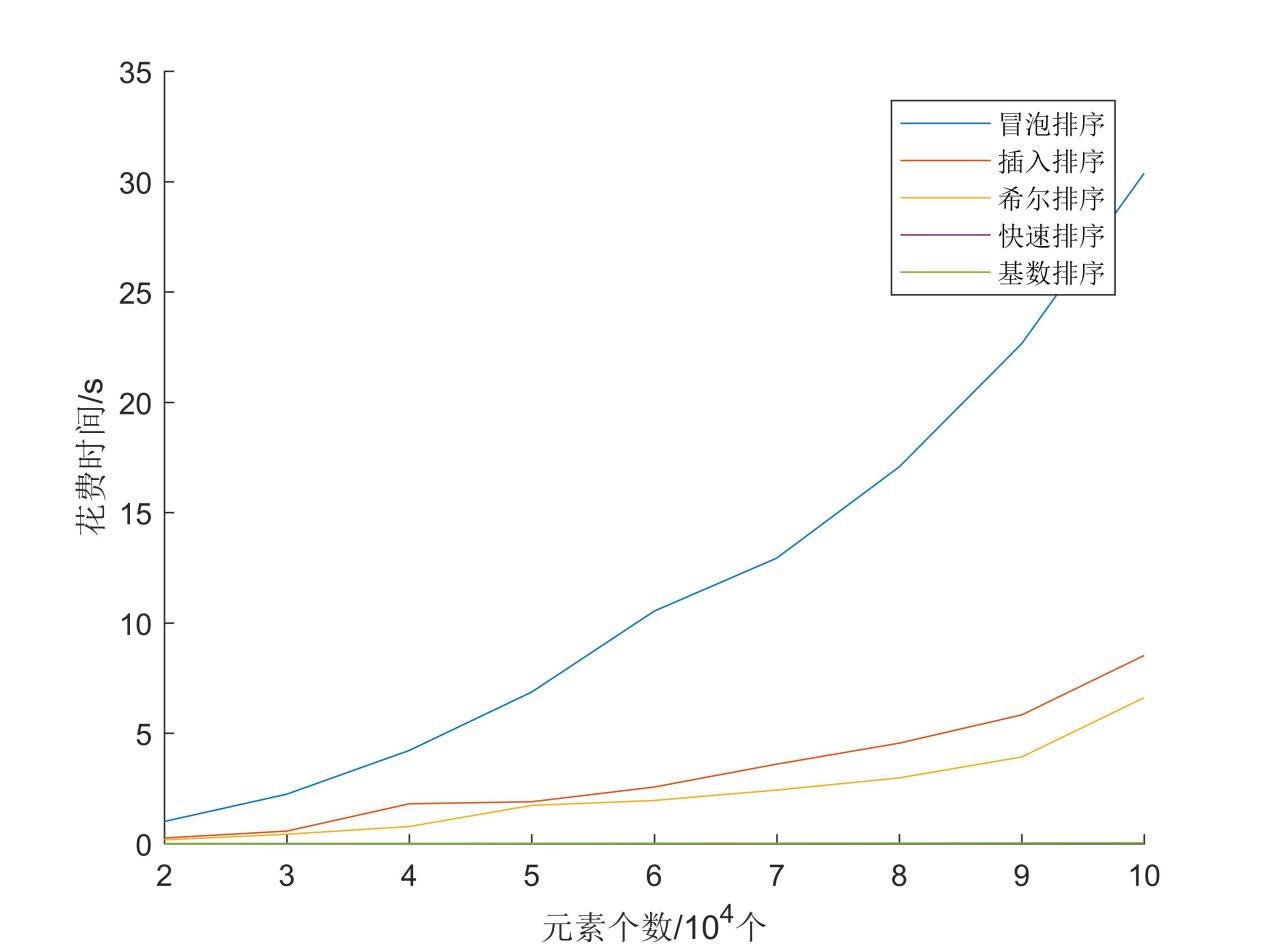
希尔排序

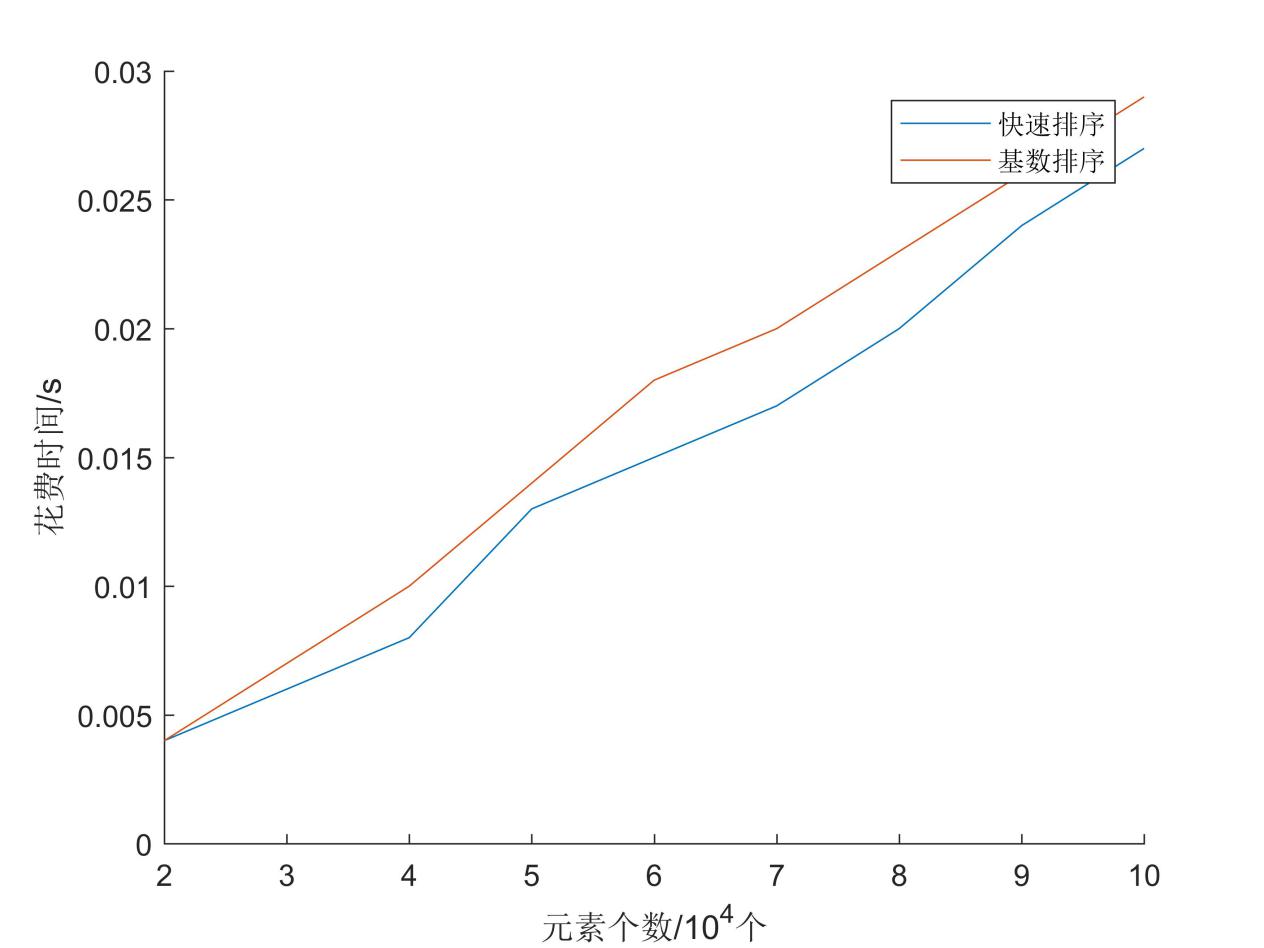
基数排序

经过测试，5中排序算法在不同的元素个数时排序花费的时间如下



绘制出的时间图如下：





数据分析：

由数据和图示可知，冒泡排序作为经典比较排序算法，其运算时间是最长的。而快速排序作为广泛使用的优化排序算法在本次比较中时间最短，且明显优于冒泡、插入、希尔三种排序算法。基数排序与快速排序时长相当，在本次比较中时间略微长于快速排序。

**第二次上机实验报告**

姓名： 粟锦 学号：U201817053 班级：软工1803班

实验内容一：

程序实现

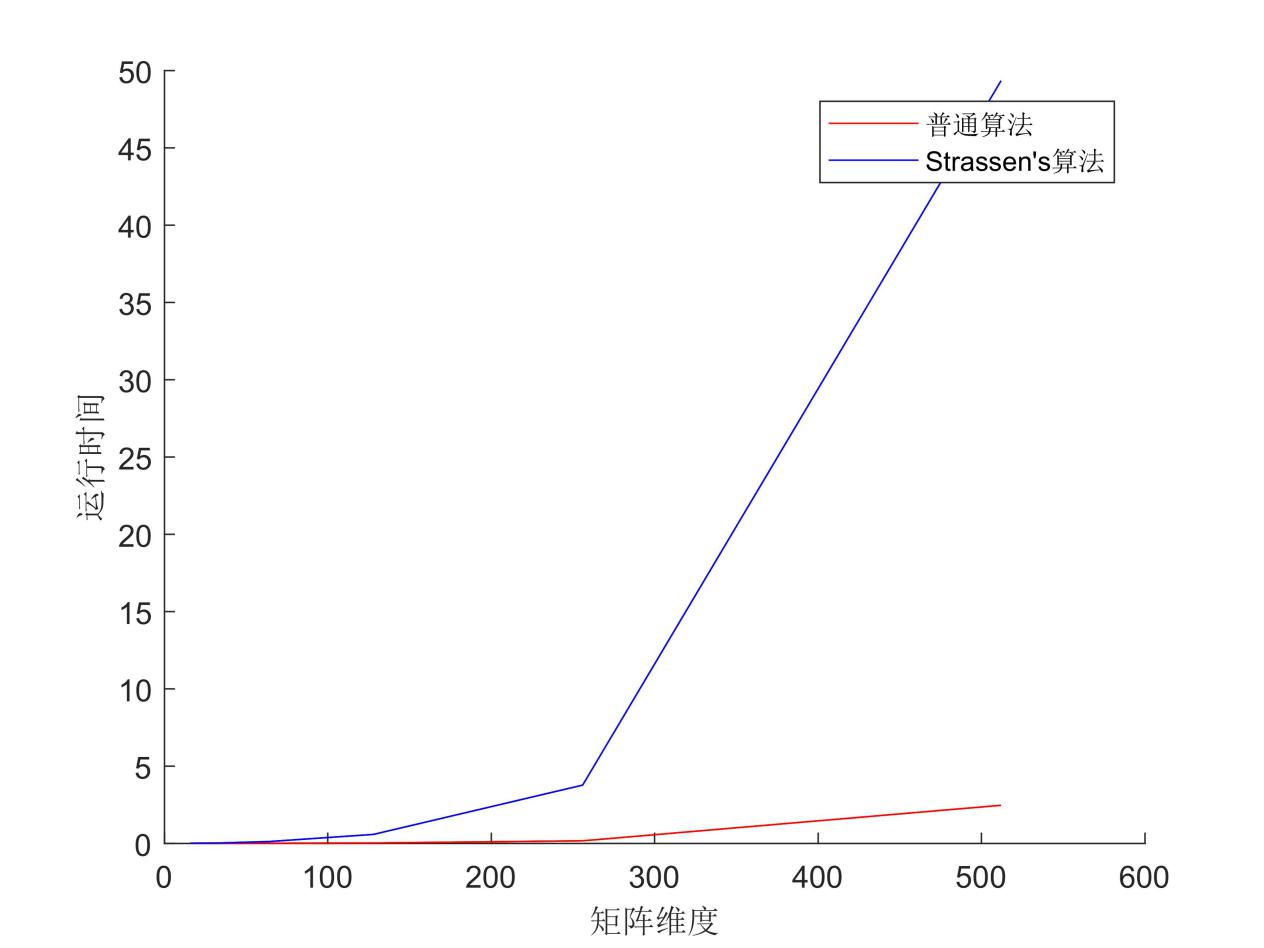
利用matlab程序，编写了基于Strassen算法的矩阵乘法，利用函数的递归调用成功地完成了矩阵乘法的正确运算。

实验内容二：

算法比较

利用matlab，绘制出了随着矩阵维数的增加，数学定义算法与Strassen算法的时间变化的图像。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 矩阵维数 | 16 | 32 | 64 | 128 | 256 | 512 |
| 普通算法 | 0.0004758 | 0.000431 | 0.0025 | 0.0183 | 0.1588 | 2.456 |
| Strassen’s算法 | 0.0049 | 0.0151 | 0.1111 | 0.5741 | 3.7634 | 49.3318 |

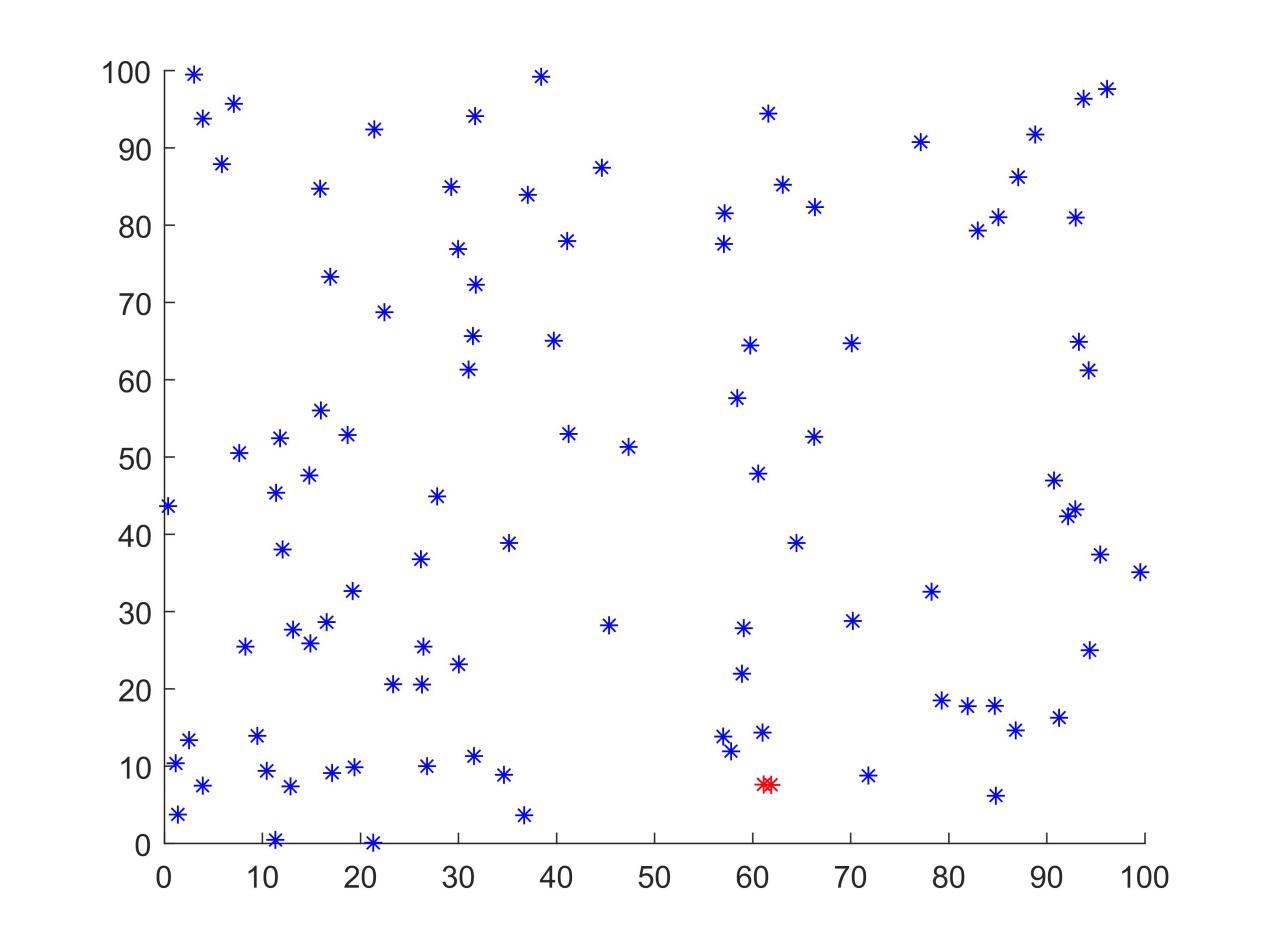


由数据表和图示可以看出，Strassen’s算法明显比普通算法花费的时间更加的长，这与我们原先设想的结果不同。究其原因，是因为Strassen’s利用大量的递归程式，程序在调用递归时花费了大量的时间，因此才会得到如上结果。

实验内容三：

最近点对算法

利用matlab工具进行编程，通过对100个在100\*100

范围内的点进行分治法求解，得出了最近点对。

其中标红的点代表的是最近点对。

思考题：

1. 分治法的三个步骤是：
2. 将问题分解为若干个子问题
3. 对子问题进行递归求解
4. 将子问题的解进行合并
5. 对于主方式求解Strassen’s乘法矩阵，有实验内容二的表格可知，随着矩阵维数的增大，算法花费的时间呈指数递增。

对于分治法求最近点对，根据实验测试可知，随着总点数的增加，递归调用所花费的时间也大大增加，呈指数增长。

3.通过增加元素全为0的行和列将矩阵补充为2的幂次的矩阵。例如 3\*5的矩阵，通过增加3列和5行元素全为0的向量，补充为8\*8的矩阵，然后再利用Strassen’s算法，求出结果后再从8\*8矩阵中取3\*5即可。

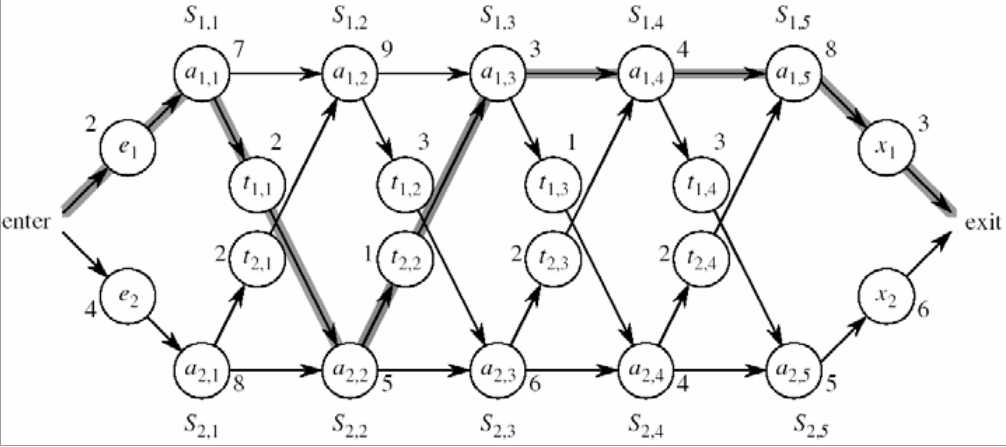
**第三次上机实验报告**

姓名： 粟锦 学号：U201817053 班级：软工1803班

实验内容一：

利用动态规划完成最小排程问题

利用matlab工具，实现了动态规划的编程。经过编程计算可得，最小时间为35，顺序如下图。



思想：

将一个大问题的最优解逐步分解为小问题的最优解。利用小问题的最优解向前推进，可以得出大问题的最优解。例如，我们要得到到exit的最优解，我们可以先求出分别到S1,5和S2,5的最优解，再分别加上x1和x2之后比较大小就可以得出到exit的最优解。S1,5和S2,5又可以分解成更小的子问题的最优解。了解这种思想后，我们自底向上，从最小的子问题的最优解开始，逐步往前推进，依次算出更大一点的问题的最优解，并利用这个最优解推出下一个更大问题的最优解，直到最后得到最终的最优解。

实验内容二：

利用动态规划完成最长共同子序列问题

利用matlab工具，实现了动态规划的编程。经过编程计算可得，S1 = ACCGGTCGAGATGCAG

S2 = GTCGTTCGGAATGCA

最长共同子序列为GGTCGGATGCA。

思想：

将问题分解为子问题。设n(i,j)为有i个元素的S1与有j个元素的S2之间的最长共同子序列的长度。我们拿此时S1的最后一个元素和S2的最后一个元素作比较。若两者最后一个元素相等，则n(i,j) = n(i-1,j-1)+1。若最后一个元素不相等，则n(i,j) = max(n(i,j-1),n(i-1,j))。这样便将问题逐个分解成更小的问题，换言之，可以利用更小的问题最优解推出更大的问题最优解。然后便可自底向上一步一步推出原问题的最优解。

思考题：

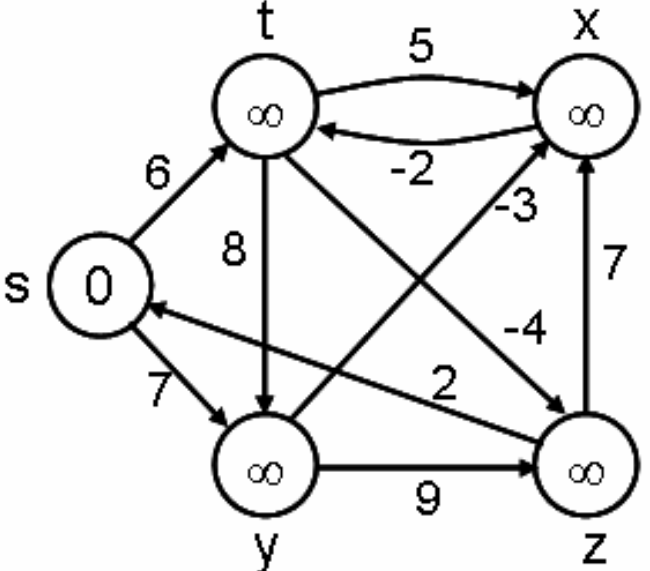
1. 动态规划的范式是通过分析原问题的性质，思考如何将原问题分解成更小的子问题并能够利用子问题的最优解推出原问题的最优解，找到一个递推式。之后可以通过自底向上或自顶向下的方式推出最优解。
2. 给定一个矩阵相乘链，如A1\*A2\*A3\*A4\*A5\*A6，我们首先假设最优解在某个地方有括号，例如(A1\*A2)\*(A3\*A4\*A5\*A6)是最优解最外层的括号（每个括号内部还可能分割），那么原问题便分解成两个括号形成的新的矩阵相乘链的最优解之和加上两个矩阵相乘链相乘的计算次数。这便将问题分成了两个子问题，只需以此类推便可知道递推规律，然后就可以自底向上推出原问题的最优解。

**第四次上机实验报告**

姓名： 粟锦 学号：U201817053 班级：软工1803班

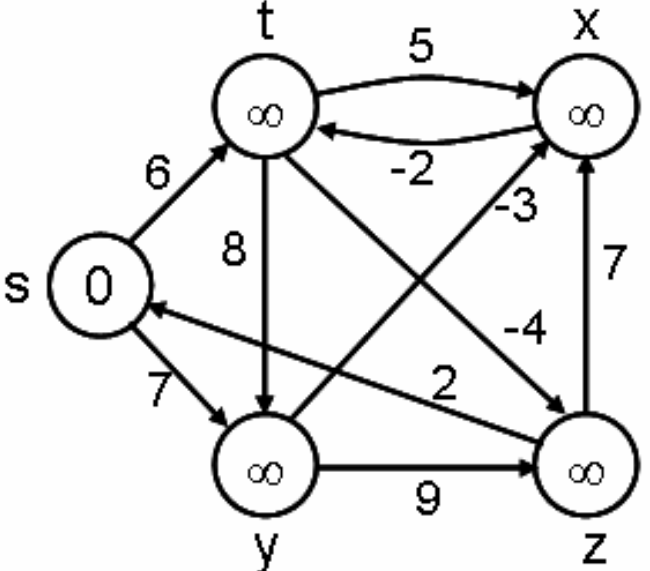
实验内容一：

描述并实现单源最短路径算法，显示在下图上的运算结果



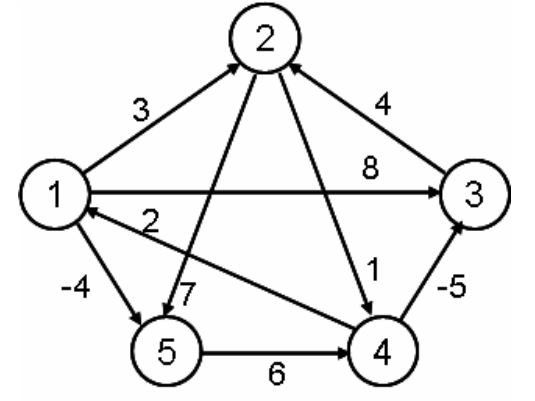
思想：利用Bellman-Ford算法，通过V-1的循环，然后在每次循环中对所有边集进行一次缩短法。最后，再单独对所有边进行一次缩短法便可以判断有向图中是否存在负权回路。

利用matlab编程可以得出，该有向图的最短路径为：



实验内容二：

描述并实现全点对最短路径算法，显示在下图上的运算结果



方法一：延伸法

设D（k）ij为节点i到节点j经过边数不超过k的最小距离。D（1）是初始时节点的邻接矩阵（不相连则值为∞，相连则值为距离大小），然后对D（1）进行延伸，通过比较取最小值得到D（2）。依次类推直到D（k-1），得到的便是全点对最短路径矩阵。

方法二：Floyd-warshall算法

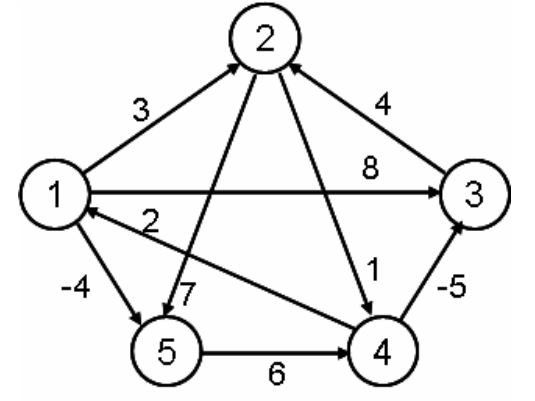
利用动态规划的思想，设D（k）ij为节点i到节点j经过的节点标号不超过k的最短距离，然后利用递推公式：

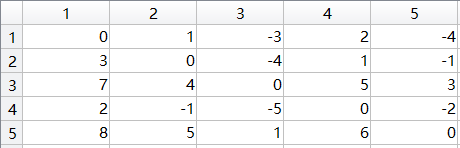
D（k）ij = D(0)(初始距离矩阵) ,k = 0

Min{D(k-1)ij , D(k-1)ik + D(k-1)kj}, k > 0

即可求出全点对最短路径矩阵。

求得结果如下：





思考题：

1. 全点对最短路径的动态规划范式是考虑节点ij之间是否包含最大节点k。如果不包含节点k，则D(k)ij可以等效于D(k-1)ij；如果包含节点k，则因为全局最优路径必定局部最优，可以将ij路径从节点k处分开，分别求ik和kj的最短路径。这就构成了递推公式，之后便可以利用动态规划的思想自底向上进行编程实现。
2. 图的存储一般是通过邻接链表或者邻接矩阵的方式进行的。对于邻接链表，我们在使用的时候很难直接知道两个节点之间的连通关系，需用遍历节点的邻接链表才能知道，这很不方便。而对于邻接矩阵，我们可以通过寻找矩阵对应位置上的元素便可以知道节点之间的关系，比较方便。虽然所占的空间会比较大，但效率相比于邻接链表也有很大的提升。