****

**算法分析与设计预习报告**

****

**学 院： 电子信息与人工智能学院**

**专业名称： 计算机科学与技术**

**班 级： 计算机203**

**学 号： 202007020625**

**姓 名： 於俊涛**

**任课老师： 齐 勇**

# 实验3 回溯算法实验

班级 计算机203 学号 202007020625 姓名 於俊涛 角色 designer

**一、实验目的**

1、掌握拓扑排序、最短路径、Dijkstra算法、Prim算法、Floyd算法相关知识、以及不同算法的性能。

2、了解各自算法的内容，尝试使用不同的编程语言来实现代码。

**二、实验环境**

线上平台sustoj.com

**三、实验内容**

**1. Prim算法**

Prim算法的具体效率，是与图的顶点和边数有关的。这取决于所选择的表示图本身的数据结构，以及表示集合V-Vr的优先队列的数据结构，集合中顶点的优先级就是到最近的树的顶点的距离。

假设总共有n个顶点，那么肯定有n次比较过程！而在第k次比较过程中，又有n-k个点和另外n-k个顶点相互比较，那么总结起来，那么算法复杂度几乎再n的立方级别。n³级别复杂度的确不是很高，但其实prim算法真正效率是介于n²和n³之间的，因为我们如上的分析是一种最坏的情况，就是每个点之间都存在一条路径。

在查找最小生成树这一问题中，Prim算法算是高效的算法，尽管算法的复杂度为n3。举个例子，如果试图用一种穷举法查找算法来构成一棵最小生成树，会遇到两个严重的阻碍，首先，随着图的规模的增长，生成树的数量呈指数增加（至少稠密图是如此）。其次，生成一个给定图的所有生成树也并非易事。事实上，这比用一些高效的算法来求加权图的最小生成树还要困难。

最小生成树算法Prim算法的时间复杂度为O（E\*lgV）。

而Prim算法若采用斐波那契堆实现的话，算法时间复杂度为O（E+V\*lgV），当|V|<<|E|时，E+V\*lgV是一个较大的改进。

**2. 最短路径（两种算法的区别）**

Floyd 算法虽然总体上时间复杂度较高，但可以处理带负权边的图（但不能有负权回路），并且均摊到每一点对上，在所有的算法中还是属于比较优秀的算法。另外，floyd算法较小的编码复杂度也是一大优势，所以，如果要求的是所有点对间的最短路径，或者如果数据范围较小，则floyd算法比较合适。

Dijkstra算法最大的弊端就是他无法处理带有负权边以及负权回路的图，但是Dijkstra算法具有良好的可扩展性，扩展后可以适应很多问题。另外用堆优化的Dijkstra算法的时间复杂度可以达到O（M log N）。当边有负权，甚至存在负权回路时，需要使用Bellman-ford 算法或者队列优化的Bellman-ford算法，因此我们选择最短路径法时，根据实际的需求和每一种算法的特性，选择合适的算法来使用。

Dijkstra算法适用场景：

一般算法书中都说适合于有向无环图（DAG）。但是并不代表无向图就不行，只要将源点和目标点的权重相互之间赋值成一个值即可。如果是有环图的话，对于每个节点都是做了一层遍历，也不会出现死循环和重复计算的问题，因此迪杰斯特拉算法是可以用在无向图和有环图中的，适合于求单源最短路径。如果想适用于多源最短路径，就要将每个节点都进行遍历，那么时间复杂度就是O(n³)。

**3. 拓扑排序（算法优化）**

先将入度为0的顶点放在栈或者队列中。当队列不空时，删除一个顶点v，然后更新与顶点v邻接的顶点的入度。只要有一个顶点的入度降为0，则将之入队列。此时，拓扑排序就是顶点出队的顺序。该算法的时间复杂度为O（V+E）。