

《算法设计与分析》实验教学大纲

课程编号: CS205102

课程名称: 算法设计与分析

英文名称: Design and Analysis of Algorithms

学分/学时: 2.5/40

课程性质: 专业选修

适用专业: 计算机科学与技术

建议开设学期: 5

物联网工程, 卓越, 教改

先修课程: 离散数学, 数据结构

开课单位: 计算机学院

JAVA 程序设计

一、实验简介

本实验要求学生能够综合运用排序、搜索、图处理和字符串处理的基础算法和数据结构, 将算法理论、算法工程和编程实践相结合开发相应的软件, 解决科学、工程和应用环境下的实际问题。使学生能充分运用并掌握算法设计与分析的方法以及算法工程技术, 为从事计算机工程和软件开发等相关工作打下坚实的基础。

二、实验课程目标与毕业要求

通过本课程的学习使学生系统掌握算法设计与分析的基本概念和基本原理, 理解排序、搜索、图处理和字符串处理的算法设计理论及性能分析方法, 掌握排序、搜索、图处理和字符串处理的数据结构与算法实现技术。课程强调算法的开发及 Java 实现, 理解相应算法的性能特征, 评估算法在应用程序中的潜在性能。

三、实验内容及基本要求

(三) 地图路由 (Map Routing)

实现经典的 Dijkstra 最短路径算法, 并对其进行优化。这种算法广泛应用于地理信息系统 (GIS), 包括 MapQuest 和基于 GPS 的汽车导航系统。

地图。本次实验对象是图 *maps* 或 *graphs*, 其中顶点为平面上的点, 这些点由权值为欧氏距离的边相连成图。可将顶点视为城市, 将边视为相连的道路。在文件示例中表示地图, 首先列出了其中顶点数和边数, 然后列出顶点 (索引后跟其 x 和 y 坐标), 然后列出边 (顶点对), 最后列出源点和汇点。例如, 如下左图信息表示右图:

```

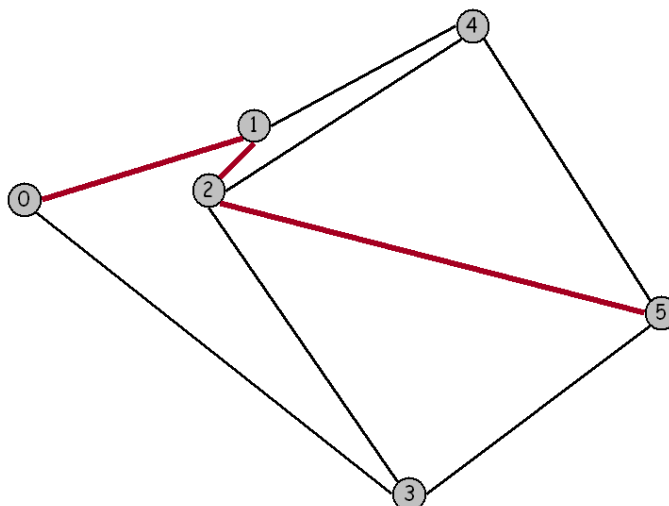
6 9
0 1000 2400
1 2800 3000
2 2400 2500
3 4000 0
4 4500 3800
5 6000 1500

```

```

0 1
0 3
1 2
1 4
2 4
2 3
2 5
3 5
4 5
0 5

```



Dijkstra 算法。 Dijkstra 算法是最短路径问题的经典解决方案。教科书 4.4 节描述了该算法。基本思路不难理解。对于图中的每个顶点，我们维护从源点到该顶点的最短已知的路径长度，并且将这些长度保持在优先队列（*priority queue*, PQ ）中。初始时，我们把所有的顶点放在这个队列中，并设置高优先级，然后将源点的优先级设为 0.0。算法通过从 PQ 中取出最低优先级的顶点，然后检查可从该顶点经由一条边可达的所有顶点，以查看这条边是否提供了从源点到那个顶点较之之前已知最短路径的更短路径。如果是这样，它会降低优先级来反映这种新的信息。

这里给出了 Dijkstra 算法计算从 0 到 5 的最短路径 0-1-2-5 的详细过程。

```

process 0 (0.0)
    lower 3 to 3841.9
    lower 1 to 1897.4
process 1 (1897.4)
    lower 4 to 3776.2
    lower 2 to 2537.7
process 2 (2537.7)
    lower 5 to 6274.0
process 4 (3776.2)
process 3 (3841.9)
process 5 (6274.0)

```

该方法计算最短路径的长度。为了记录路径，我们还保持每个顶点的源点到该顶点最短路径上的前驱。文件 `Euclidean Graph.java`, `Point.java`, `IndexPQ.java`, `IntIterator.java` 和 `Dijkstra.java` 提供了针对 `map` 的 Dijkstra 算法的基本框架实现，你可以以此作为起点。客户端程序 `ShortestPath.java` 求解一个单源点最短路径问题，并使用图形绘制了结果。客户端程序 `Paths.java` 求解了许多最短路径问题，并将最短路径打印到标准输出。客户端程序 `Distances.java` 求解了许多最短路径问题，仅将距离打印到标准输出。

目标。 优化 Dijkstra 算法，使其可以处理给定图的数万条最短路径查询。一旦你读取图（并可选地预处理），你的程序应该在**亚线性时间内解决最短路径问题**。一种方法是预先计算出所有顶点对的最短路径；然而，你无法承受存储所有这些所需的信息。你的目标是减少每次最短路径计算所涉及的工作量，而不会占用过多的空间。建议你选择下面的一些潜在想法来实现，或者你可以开发和实现自己的想法。

想法 1. Dijkstra 算法的朴素实现检查图中的所有 V 个顶点。减少检查的顶点数量的一种策略是一旦发现目的地的最短路径就停止搜索。通过这种方法，可以使每个最短路径查询的运行时间与 $E' \log V'$ 成比例，其中 E' 和 V' 是 Dijkstra 算法检查的边和顶点数。然而，这需要一些小心，因为只是重新初始化所有距离为 ∞

就需要与 V 成正比的时间。由于你在不断执行查询，因而只需重新初始化在先前查询中改变的那些值来大大加速查询。

想法 2. 你可以利用问题的欧式几何来进一步降低搜索时间，这在算法书的第 4.4 节描述过。对于一般图，Dijkstra 通过将 $d[w]$ 更新为 $d[v] +$ 从 v 到 w 的距离来松弛边 $v-w$ 。对于地图，则将 $d[w]$ 更新为 $d[v] +$ 从 v 到 w 的距离 $+ 从 w 到 d 的欧式距离 - 从 v 到 d 的欧式距离$ 。这种方法称之为 A* 算法。这种启发式方法会有性能上的变化，但不会影响正确性。

想法 3. 使用更快的优先队列。在提供的优先队列中有一些优化空间。你也可以考虑使用 Sedgewick 程序中的多路堆（Multiway heaps, Section 2.4）。

测试。 美国大陆文件 [usa.txt](#) 包含 87,575 个交叉口和 121,961 条道路。图形非常稀疏 - 平均的度为 2.8。你的主要目标应该是快速回答这个网络上的顶点对的最短路径查询。你的算法可能会有不同执行时间，这取决于两个顶点是否在附近或相距较远。我们提供测试这两种情况的输入文件。你可以假设所有的 x 和 y 坐标都是 0 到 10,000 之间的整数。

注：这个问题的实验由 Bob Sedgewick 和 Kevin Wayne 设计开发（Copyright © 2004）。更多信息可参考 <http://algs4.cs.princeton.edu/>。