**最短路径**

找到从一个顶点到达另一个顶点的成本最小的路径。

在一幅加权有向图中，从顶点s到顶点t的最短路径是所有从s到t的路径中的权重最小者。

单点最短路径：从s到给定的目的顶点v是否存在一条有向路径？如果有，找出来。

**4.4.1 最短路径的性质**

给定一副加权有向图和一个顶点s，以s为起点的一棵最短路径树是图的一幅子图，它包含s和从s可达的所有顶点。这棵有向树的根节点为s，树的每条路径都是有向图中的一条最短路径。

**4.4.2 加权有向图的数据结构**

加权有向边的API

构造函数：DirectedEdge(int v, int w, double weight)

边的权重：double weight()

指出这条边的顶点：int from()

这条边指向的顶点：int to()

对象的字符串描述：toString()

加权有向图的API

构造函数：EdgeWeightedDigraph(int V)

顶点总数：int V()

边的总数：int E()

将e添加到该有向图中：addEdge(DirectedEdge e)

从v指出的边：Iterable<DirectedEdge> adj(int v)

该有向图中的所有边：Iterable<DirectedEdge> edges()

对象的字符串表示：toString()

编程：加权有向边的数据类型

编程：加权有向图的数据类型

最短路径的API

构造函数：SP(EdgeWeightedDiagraph G, int s)

从顶点s到v的距离，如果不存在则为无穷大：double distTo(int v)

是否存在从顶点s到v的路径：boolean hasPathTo(int v)

从顶点s到v的路径：Iterable<DirectedEdge> pathTo(int v)

放松边v->w意味着检查从s到w的最短路径是否是先从s到v，然后再由v到w。

**4.4.3 最短路径算法的理论基础**

最短路径的最优性条件：令G为一幅加权有向图，顶点s是G中的起点，distTo[]是一个由顶点索引的数组，保存的是G中路径的长度。对于从s可达的所有顶点v，distTo[v]的值是从s到v的某条路径的长度，对于从s不可达的所有顶点v，该值为无穷大。当且仅当对于从v到w的任意一条边e，这些值满足distTo[w]<=distTo[v]+e.weight()时(换句话说，不存在有效边时)，它们是最短路径的长度。

通用最短路径算法：将distTo[s]初始化为0，其他distTo[]元素初始化为无穷大，继续如下操作：放松G中的任意边，直到不存在有效边为止。对于任意从s可达的顶点w，在进行这些操作之后，distTo[w]的值即为从s到w的最短路径的长度。

**4.4.4 Dijkstra算法**

Dijkstra算法：首先将distTo[s]初始化为0，distTo[]中的其他元素初始化为正无穷，然后将distTo[]最小的非树顶点放松并加入树中，如此这般，直到所有的顶点都在树中或者所有的非树顶点的distTo[]值均为无穷大。

在一幅含有V个顶点和E条边的加权有向图中，使用Dijkstra算法计算根节点为给定起点的最短路径树所需的空间与V成正比，时间与ElogV成正比。

编程：最短路径的Dijkstra算法

**4.4.5 无环加权有向图中的最短路径算法**

按照拓扑顺序放松顶点，就能在和E+V成正比的时间内解决无环加权有向图的单点最短路径。

编程：无环加权有向图的最短路径算法

解决无环加权有向图的最长路径问题所需的时间与E+V成正比。

解决并行任务调度问题的关键路径方法的步骤如下：创建一幅无环加权有向图，其中包含一个起点s和一个终点t且每个任务都对应着两个顶点（一个起始顶点和一个结束顶点）。对于每个任务都有一条从它的起始顶点指向结束顶点的边，边的权重为任务所需的时间。对于每条优先级限制v->w，添加一条从v的结束顶点指向w的起始顶点的权重为零的边。我们还需要为每个任务添加一条从起点指向该任务的起始顶点的权重为零的边以及一条从该任务的结束顶点到终点的权重为零的边。这样，每个任务预计的开始时间即为从起点到它的起始顶点的最长距离。

编程：优先级限制下的并行任务调度问题的关键路径方法

解决优先级限制下的并行任务调度问题的关键路径法所需的时间为线性级别。

相对最后期限限制下的并行任务调度问题是一个加权有向图中的最短路径问题（可能存在环和负权重边）。

Dijkstra算法只适用于正或零权重的边，算法4.10要求有向图是无环的。

4.4.6 一般加权有向图中的最短路径问题

加权有向图中的负权重环是也一个总权重为负的有向环。

在负权重环存在的情况下，最短路径问题是没有意义的。

Bellman-Ford算法：在任意含有V个顶点的加权有向图中给定起点s，从s无法到达任何负权重环，以下算法能够解决其中的单点最短路径问题：将distTo[s]初始化为0，其他distTo[]元素初始化为无穷大。以任意肾虚放松有向图的所有边，重复V轮。

Bellman-Ford算法在最坏情况下所需的时间和EV成正比，空间和V成正比。

编程：基于队列的Bellman-Ford算法

套汇问题等价于加权有向图中的负权重环的检测问题。

编程：货币兑换中的套汇