**《数据结构》上机报告**

**2018 年 12 月 21 日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **姓名：** | **赵得泽** | **学号：** | **1753642** | **班级：** | **电子2班** | **得分：** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **实验题目** | 图（单源最短路径） | | |
| **问题描述** | 给定一张 n 个点的无向带权图，节点的编号从 1 至 n，求从 S 到 T 的最短路径长度。 | | |
| **基本要求** | 1. 给定一张 n 个点的无向带权图，节点的编号从 1 至 n，求从 S 到 T 的最短路径长度。 | | |
| **已完成基本内容（序号）：** | 1 | |
| **选做要求** |  | | |
| **已完成选做内容（序号）** | |  |
| **数据结构**  **设计** | **typedef long long ShortPathTable[MAX\_VERTEX\_NUM]; //最短路径长度**  **int first[MAX\_VERTEX\_NUM];//顶点表**  **int u[MAX\_VERTEX\_NUM];//起始顶点表；**  **int v[MAX\_VERTEX\_NUM];//终止顶点表；**  **int w[MAX\_VERTEX\_NUM];//权值表；**  **int \_next[MAX\_VERTEX\_NUM \* 2];//邻接点表**  本次实验主要是对单源最短路径的考察，也就是如何用迪杰斯特拉算法对单源最短路径求解。在此数据结构主要是顺序表，其中有first[]顶点表，u[],待输入的起始顶点表，v[]待输入的终止顶点表，w[]待输入的权值表，\_next[]邻接顶点表（静态链表方式存储）。三个待输入的表通过邻接表的方式进行构建。 | | |
| **功能(函数)说明** | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **函数功能：用迪杰斯特拉算法求单源最短路径**  **函数说明：在此算法中，我利用的是静态链表创建的一张无向网，它的**  **算法效率较邻接链表、邻接矩阵高；具体的算法步骤解释见注释区域。**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**  **void ShortPath\_DIJ(int n, int m, int v0, ShortPathTable &D)**  **{**  **//用Dijkstra算法求有向网v0到其余各顶点带权长度D[v]；**  **//final[v]=true当且仅当v∈S,即已经求得v0到v的最短路径**  **int final[MAX\_VERTEX\_NUM] = { 0 };**  **long long min;**  **int i, j, l;**  **for (int i = 1; i <= n; i++)**  **{**  **D[i] = INFINITY;**  **first[i] = -1;**  **}**  **j = 1;**  **for (i = 1; i <= m; i++)**  **{**  **/\*构建无向网\*/**  **/\*相当于输入2\*m条边，其中u[],v[]代表的是起始顶点表**  **及终止顶点表，即u->v;v->u 并同时将它们存储在静态链**  **表为基础的邻接表中，在此要特别注意创建逆邻接表的时候**  **我是在邻接表\_next[]的基础上进行创建的，就可以按照静**  **态链表的方式找到一个顶点相关联的其他所有顶点\*/**  **cin >> u[i] >> v[i] >> w[i];**  **\_next[j] = first[u[i]];**  **first[u[i]] = j;//邻接表**  **\_next[j + 1] = first[v[i]];**  **first[v[i]] = ++j;//在邻接表的下一个位置创建逆邻接表**  **if (u[i] == v0)//将v0到其他所有顶点的权值表D[]初始化**  **D[v[i]] = w[i];**  **if (v[i] == v0)**  **D[u[i]] = w[i];**  **j++;**  **}**  **final[v0] = 1;//表示v0已经访问**  **D[v0] = 0;**  **int k;**  **for (k = 1; k <= n; k++)**  **{**  **min = INFINITY;**  **for (j = 1; j <= n; j++)**  **{**  **/\*找到最离源点最近的一个点l\*/**  **if (final[j] == 0 && min > D[j])**  **{**  **l = j;**  **min = D[j];**  **}**  **}**  **final[l] = 1;//l顶点已经访问**  **int p;**  **for (i = first[l]; i != -1; i = \_next[i])//在静态链表中按照静态链表的存储方式进行寻找**  **{**  **/\*在静态邻接表中遍历更新D[]\*/**  **p = (i + 1) / 2;//因为权值只有n个，在建立逆邻接表的的时候使用的权值**  **//依旧是与之相邻的邻接表的权值**  **if (i % 2 == 0 && !final[u[p]])//若i为偶数，则证明该顶点是邻接表中的点**  **{**  **if (D[l] + w[p] < D[u[p]])**  **D[u[p]] = D[l] + w[p];**  **}**  **else if (!final[v[p]] && i % 2 == 1)//若i为奇数，则证明该点是逆邻接表中的点**  **{**  **if (D[l] + w[p] < D[v[p]])**  **D[v[p]] = D[l] + w[p];**  **}**  **}**  **}**  **}** | | |
| **开发环境** | **Win10,vs2017,C++高级程序语言设计** | | |
| **调试分析** |  | | |
| **心得体会** | 本次实验是图的简单应用，也是对**单源最短路径算法**——**迪杰斯特拉算法**的优化应用，因为测试数据量比较大，所以算法的优化成为了首要任务，尽管用邻接矩阵和邻接链表能做出来，但是由于两种算法在空间和时间上的利用效率不太高，所以在此我采用了**静态邻接表法。**  静态邻接表指的就是利用静态邻接表创建一张图，之后在静态邻接表的基础上进行搜索，从而达到空间和时间上的优化。首先静态邻接表都是一维数组，相对于邻接矩阵，其空间开销大大降低；其次，在一维数组上查找比较，其时间效率也大大提高。  迪杰斯特拉算法的优化方式还有很多种，目前我们接触的还只是其最原本的算法，所以还要深入学习才能够在面对问题时真正地做出最好的选择。 | | |