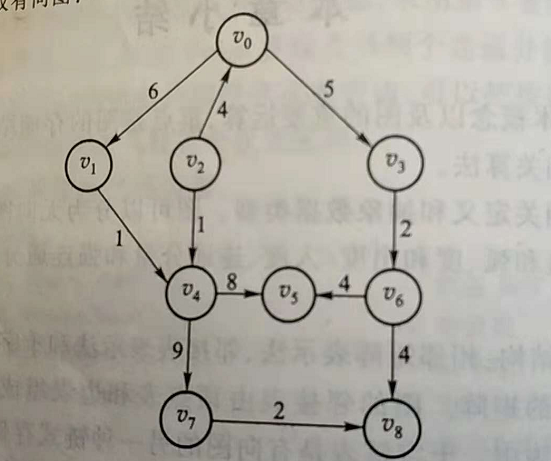
**20181116数据结构作业**

**1800022769 张靖昆 20181116**

**我承诺诚实作业，没有抄袭他人！**



1. 对于上图中的带权有向图：
2. **写出其相邻矩阵**

从图中可知，共有9个顶点，从而建立9×9的矩阵，得到如下相邻矩阵：（由于0比较多，我将非0的部分全部以**深红加粗**醒目表示）

1. **画出其邻接表表示**

这里以出度表示。

顶点表 边表

1

6

3

5

\

0

0

4

4

1

\

2

4

1

\

1

6

2

\

3

7

9

\

5

8

4

\

5

8

4

\

5

4

6

8

2

\

\

8

7

1. **计算每个顶点的入度和出度**

出度和入度表如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **顶点** |  |  |  |  |  |  |  |  |  |
| **出度** | 2 | 1 | 2 | 1 | 2 | 0 | 2 | 1 | 0 |
| **入度** | 1 | 1 | 0 | 1 | 2 | 2 | 1 | 1 | 2 |

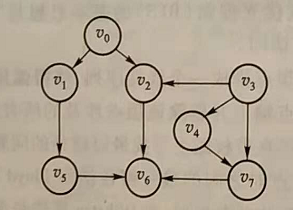
1. **如果每个指针需要4个字节，每个顶点的标号需要2个字节，每条边的权需要2个字节，则此图采用哪种表示法需要的空间较少**？

**计算邻接矩阵的空间**：由于邻接矩阵只记录权值，从而所占据空间为9\*9\*2=162字节；

**计算出度邻接表空间**：顶点表中每个顶点占据2+4=6字节，从而9个顶点共9\*6=54字节；边表中每个边结点占据2+2+4=8字节，从而11个边结点占据11\*8=88字节，则邻接表所占总空间为88+54=142字节。

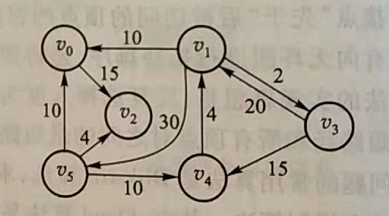
很显然，**邻接表占据空间较少。**

1. 对于下面的有向图，从顶点出发，分别画出其深度优先搜索和广度优先搜索生成的森林。



深度优先遍历生成的森林（其中一种） 广度优先遍历生成的森林（唯一一种）

1. 求下面的有向图中从顶点到其他顶点的全部最短路径及长度



1. **声明**：
2. 设置长度为6的path[]用于记录每个顶点最短路径中的前一个顶点的下标。
3. 设置长度为6的disc[]数组记录每个顶点最短路径的长度。
4. 设置长度为6的visit[]数组标记每个顶点是否被访问过。
5. 设置顶点集合vertexes[]，用于记录被加入最短路径区的顶点，主要是用于表示。
6. 以邻接矩阵表示该有向图，记邻接矩阵为weight[][]，weight[i][j]表示顶点i到顶点j的路径的权值，为∞表示不可达。
7. **规则\***：每次处理中，对于新加入的顶点，应当根据其所连接的顶点刷新数组中每个可达顶点的最短路径。即假设中间顶点为k，则对于如果k对i顶点可达，且disc[i] > disc[k]+weight[k][i]，则应当更新disc[i] = disc[k]+weight[k][i]。
8. **规则\*\***：对path[]的处理规则为，经过选取权值最小的步骤后得到的新顶点k，添加入vertexes[]后，根据规则\*应当对所有由顶点k可达的顶点的disc[]进行更新，此时如若某个顶点i的disc[]被更新，那么path[i]=k。
9. **初始状态**：设置各数组的初值。  
   visit[]：令visit[4]=1，其余全部为0，代表初始状态下首顶点已被访问；  
   disc[]：搜索顶点可达顶点的权值，令disc[1]=4，其他全部为∞；  
   path[]：全部为-1。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **下标** | 0 | 1 | 2 | 3 | 4 | 5 |
| **path[]** | -1 | -1 | -1 | -1 | -1 | -1 |
| **disc[]** | ∞ | 4 | ∞ | ∞ | ∞ | ∞ |
| **visit[]** | 0 | 0 | 0 | 0 | 1 | 0 |
| **vertexes[]** | {} | | | | | |

1. **第一次搜索**：从disc[]中且没被访问的顶点中选出最小的权值为4，对应顶点为。更新：  
   更新visit[]：visit[1] = 1；  
   更新disc[]：由顶点可达的顶点有、、，根据规则\*更新disc[0] = 14，disc[3] = 6，disc[5] = 34。  
   更新path[]：根据规则\*\*修改path[]数组，得到path[1] = 4，path[0] = 1, path[3] = 1, path[5] = 1；

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **下标** | 0 | 1 | 2 | 3 | 4 | 5 |
| **path[]** | 1 | 4 | -1 | 1 | -1 | 1 |
| **disc[]** | 14 | 4 | ∞ | 6 | ∞ | 34 |
| **visit[]** | 0 | 1 | 0 | 0 | 1 | 0 |
| **vertexes[]** | {，} | | | | | |

1. **第二次搜索**：从disc[]中且没被访问的顶点中选出最小的权值为6，对应顶点为。更新：  
   更新visit[]：visit[3] = 1；  
   更新disc[]：由顶点可达的顶点有，根据规则\*，发现disc[1]＜disc[3]+weight[3][1]，又顶点是出发顶点，从而不用更新；  
   更新path[]：根据规则\*\*，更新path[3] = 1。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **下标** | 0 | 1 | 2 | 3 | 4 | 5 |
| **path[]** | 1 | 4 | -1 | 1 | -1 | 1 |
| **disc[]** | 14 | 4 | ∞ | 6 | ∞ | 34 |
| **visit[]** | 0 | 1 | 0 | 1 | 1 | 0 |
| **vertexes[]** | {，，} | | | | | |

1. **第三次搜索**：从disc[]中且没被访问的顶点中选出最小的权值为14，对应顶点为。更新：  
   更新visit[]：visit[0] = 1；  
   更新disc[]：由顶点可达的顶点只有，根据规则\*，更新disc[2] = 29。  
   更新path[]：根据规则\*\*，更新path[2] = 0；

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **下标** | 0 | 1 | 2 | 3 | 4 | 5 |
| **path[]** | 1 | 4 | 0 | 1 | -1 | 1 |
| **disc[]** | 14 | 4 | 29 | 6 | ∞ | 34 |
| **visit[]** | 1 | 1 | 0 | 1 | 1 | 0 |
| **vertexes[]** | {，，，} | | | | | |

1. **第四次搜索**：从disc[]中且没被访问的顶点中选出最小的权值为29，对应顶点为。更新：  
   更新visit[]：visit[2] = 1；  
   更新disc[]：由于出度为0，从而disc[]不用更新；  
   更新path[]：由于disc[]没有更新，从而path[]不用更新。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **下标** | 0 | 1 | 2 | 3 | 4 | 5 |
| **path[]** | 1 | 4 | 0 | 1 | -1 | 1 |
| **disc[]** | 14 | 4 | 29 | 6 | ∞ | 34 |
| **visit[]** | 1 | 1 | 1 | 1 | 1 | 0 |
| **vertexes[]** | {，，，，} | | | | | |

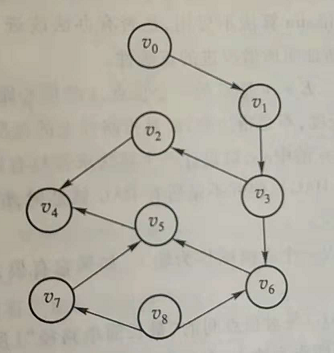
1. **第五次搜索**：从disc[]中且没被访问的顶点中选出最小的权值为34，对应顶点为。更新：  
   更新visit[]：visit[5] = 1；  
   更新disc[]：由于从顶点可达的顶点、、，根据规则\*\*，发现disc[0]＜disc[5]+weight[5][0]，disc[2]＜disc[5]+weight[5][2]，又顶点是源顶点，从而disc[]不用更新。  
   更新path[]：由于disc[]没有更新，从而不需要更新。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **下标** | 0 | 1 | 2 | 3 | 4 | 5 |
| **path[]** | 1 | 4 | 0 | 1 | -1 | 1 |
| **disc[]** | 14 | 4 | 29 | 6 | ∞ | 34 |
| **visit[]** | 1 | 1 | 1 | 1 | 1 | 1 |
| **vertexes[]** | {，，，，，} | | | | | |

经过上述查找过程，由顶点到达其他顶点的最短路径以及长度如下表所示：

|  |  |  |
| --- | --- | --- |
| **到达顶点** | **最短路径** | **长度** |
|  | → | 4 |
|  | →→ | 6 |
|  | →→ | 14 |
|  | →→→ | 29 |
|  | →→ | 34 |

1. 拓扑排序的结果不是唯一的，对于下面有向图中的顶点进行拓扑排序，能够得到多少个不同的拓扑序列。



**拓扑排序不唯一的说明**：设置一个队列，每次从顶点中找到一个入度为0的顶点，添加入队列，并删除该顶点所有出边，循环前述操作直至队列中已经存满所有顶点，得到的序列就是拓扑序列。**拓扑序列的不唯一就来源于当存在多个入度为0的顶点时，选取顺序不一样，排序结果就不唯一。**

下表中的“列”表示每个顶点在拓扑排序中的序列位置，一一讨论所有的情况。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **序号** |
|  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  | 2 |
|  |  |  |  |  |  | 3 |
|  |  |  |  | 4 |
|  |  |  |  |  | 5 |
|  |  |  |  | 6 |
|  |  |  |  | 7 |
|  |  |  |  |  | 8 |
|  |  |  |  | 9 |
|  |  |  |  | 10 |
|  |  |  |  |  |  |  | 11 |
|  |  |  |  | 12 |
|  |  |  |  |  | 13 |
|  |  |  |  | 14 |
|  |  |  |  | 15 |
|  |  |  |  |  | 16 |
|  |  |  |  | 17 |
|  |  |  |  | 18 |
|  |  |  |  |  |  | 19 |
|  |  |  |  | 20 |
|  |  |  | 21 |
|  |  |  |  |  |  |  |  | 22 |
|  |  |  |  | 23 |
|  |  |  |  |  | 24 |
|  |  |  |  | 25 |
|  |  |  |  | 26 |
|  |  |  |  |  | 27 |
|  |  |  |  | 28 |
|  |  |  |  | 29 |
|  |  |  |  |  |  | 30 |
|  |  |  | 31 |
|  |  |  |  | 32 |
|  |  |  |  |  |  |  | 33 |
|  |  |  | 34 |
|  |  |  |  | 35 |
|  |  |  |  |  |  |  |  |  | 36 |
|  |  |  |  | 37 |
|  |  |  |  |  | 38 |
|  |  |  |  | 39 |
|  |  |  | 40 |
|  |  |  |  |  | 41 |
|  |  |  |  | 42 |
|  |  |  | 43 |
|  |  |  |  |  |  | 44 |
|  |  |  |  | 45 |
|  |  |  | 46 |
|  |  |  |  |  |  |  | 47 |
|  |  |  |  | 48 |
|  |  |  | 49 |
|  |  |  |  |  |  |  |  | 50 |
|  |  |  |  | 51 |
|  |  |  | 52 |

**非合并表格请查看一并提交的EXCEL表格，所有情况都在其中**。**总共52种**。

1. 第9题请看DEV C++项目
2. **第14题**：证明：对于一个无向图G = <V, E>，若G中各顶点的度均大于等于2，则G中必有回路。

证明：利用数学归纳法进行证明，顶点数n≥2：

1. 当n=2时，易得唯一的两个顶点和度均为2，即和的入度和出度均为1，即→且→均可达，因此从出发可以回到，存在回路；
2. 假设当n=k时，存在以下回路→→→…→，假设路径中的不重复顶点数目为m，则m≤k；
3. n=k+1时，新加入一个顶点，则该顶点至少与图中2个顶点相连。对与其他顶点的相连情况分为以下几中情况讨论：
4. 顶点与回路→→→…→中任一顶点都不相连，那么此时图中仍存在回路，即→→→…→；
5. 顶点的**出**边中有一部分或者全部与回路→→→…→中的顶点相连，此时回路→→→…→依旧存在，且由于顶点只有出边与该回路中顶点相连，因此不会构成包含的新回路，但依旧存在回路。
6. 顶点的**入**边中有一部分或者全部与回路→→→…→中的顶点相连，此时回路→→→…→依旧存在，且由于顶点只有入边与该回路中顶点相连，因此不会构成包含新回路，但依旧存在回路。
7. 顶点有一条出边和一条入边与回路→→→…→中的任一顶点相连，那么显然可以代替构成新回路→→→…→…→，并且不会破坏原回路，即依旧存在回路。
8. 顶点有一条出边与回路→→→…→中的任一顶点相连，有一条入边与与回路→→→…→中的任一顶点相连，那么可以构成新回路→→→…→→→→…→，且原回路依然存在，即依然存在回路。
9. 顶点有多个出边和多个入边与回路→→→…→中的多个顶点任意互连，由d)和e)可知，可以构成至少一个包含顶点新回路，且原回路依旧存在，即依然存在回路。

综上，当G中各顶点的度均大于等于2，则G中必有回路。

1. **第19题**：对于一个具有n个顶点和e条边的有向图G = <V, E>，证明：求其强连通分量的算法所需的时间复杂度是O(n + e)。

证明：

1. 首先证明确定一个顶点数为n’和边数为e’的图G’=(V’,E’)是强连通分量的算法的时间复杂度为O(n’ + e’)。
2. 设顶点集合set(n)为由互相可达的顶点所组成，即对∀。
3. 初始状态下加入顶点进set(n)，并记录顶点已被访问。对可达的任意顶点，判断如若可达且不在集合set(n)中，则加入集合set(n)。可以看到加入集合set(n)的顶点以及他们的出边都会被访问，而且没有加入集合set(n)即可达但不可达的情况，到的边也会被访问，这就使得即使两个顶点不是直接可达，那么他们之间的边也会被访问到。
4. 再对新加入集合set(n)的顶点做同样的操作，循环往复，直至最后一个顶点加入集合set(n)；
5. 由上述过程可以确定，求解过程中每个顶点、它们的所有出边以及出边所连顶点，都会被至少访问一次，易得求解算法的复杂度为O(n’ + e’)。
6. 假设有向图G中存在m个强连通分量也是m个子图，分别为，记它们的顶点数分别为，记它们的和为；记它们的边数分别为，记它们的和为；则求解每个强连通分量所用时间复杂度为O()，O()…O()，那么总时间复杂度为
7. 再记剩余的顶点和边构成子图，其顶点数为，其边数为，易得。由于求解算法并不能也没必要区分一个顶点是属于强连通分量集合(…)的某一个，还是属于，因此求解算法对于子图的求解策略同求解强连通分量，从而子图中的所有顶点以及每个顶点的所有出边都会被至少访问一次，从而对于子图上算法所花费的时间复杂度为。
8. 综上，求解强连通分量的算法所花费的时间复杂度为。命题得证。