第四章作业2参考答案

**1、简答题**

1-1带权图（权值非负，表示边连接的两顶点间的距离）的最短路径问题是找出从初始顶点到目标顶点之间的一条最短路径。假设从初始顶点到目标顶点之间存在路径，现有一种解决该问题的方法：

① 设最短路径初始时仅包含初始顶点，令当前顶点*u*为初始顶点；

② 选择离*u*最近且尚未在最短路径中的一个顶点*v*，加入到最短路径中，修改当前顶点*u*= *v*；

③ 重复步骤②，直到*u*是目标顶点时为止。

请问上述方法能否求得最短路径？若该方法可行，请证明之；否则，请举例说明。

【参考答案】

该方法不一定能（或不能）求得最短路径。

举例说明：

1

1

2

1

图a

1

2

图b

图a中，设初始顶点为1，目标顶点为4，欲求从顶点1到顶点4之间的最短路径。显然这两点之间的最短路径长度为2。但利用给定方法求得的路径长度为3，这条路径并不是这两点之间的最短路径。

图b中，设初始顶点为1，目标顶点为3，欲求从顶点1到顶点3之间的最短路径。利用给定的方法，无法求出顶点1到顶点3的路径。

1-2 已知有6个顶点（顶点编号为0 ~ 5）的有向带权图*G*，其邻接矩阵*A*为上三角矩阵，按行为主序（行优先）保存在如下的一维数组中。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 6 | ∞ | ∞ | ∞ | 5 | ∞ | ∞ | ∞ | 4 | 3 | ∞ | ∞ | 3 | 3 |

要求：

（1）写出图*G*的邻接矩阵*A*。

（2）画出有向带权图*G*。

（3）求图*G*的关键路径，并计算该关键路径的长度。

【参考答案】

（1）图*G*的邻接矩阵A如下：



（2）图*G*如下：

4

5

4

6

3

3

3

（3）下图中双线箭头所标识的4个活动组成图*G*的关键路径。

4

5

4

6

3

3

3

图*G*的关键路径的长度为16。

1-3 使用*Prim（*普里姆）算法求带权连通图的最小（代价）生成树（*MST*）。请回答下列问题。

6

5

6

4

5

4

4

（1）对右列图*G*，从顶点*A*开始求*G*的*MST*，依次给出按算法选出的边。

（2）图*G*的*MST*是唯一的吗？

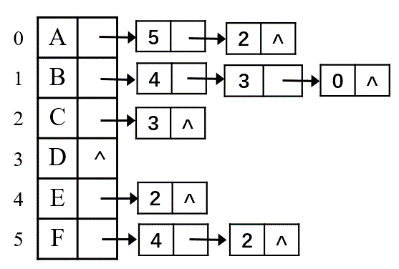
（3）对任意的带权连通图，满足什么条件时，其*MST*是唯一的？

【参考答案】

（1）依次选出的边为：

(*A*,*D*),(*D*,*E*),(*C*,*E*),(*B*,*C*)

（2）图*G*的*MST*是唯一的。

（3）当带权连通图的任意一个环中所包含的边的权值均不相同时，其*MST*是唯一的。

1-4已知图的邻接表如图所示，给出以顶点*A*为起点的一次深度优先（先深，*DFS*）和广度优先（先广，*BFS*）的搜索序列。

【参考答案】

DFS：AFEDCB

BFS：AFCEEB

**2、算法设计**

1. 采用C或C++语言设计数据结构；
2. 给出算法的基本设计思想；
3. 根据设计思想，采用C或C++语言描述算法，关键之处给出注释；
4. 说明你所设计算法的时间复杂度和空间复杂度。
   1. 一个连通图采用邻接表作为存储结构，设计一个算法，实现从顶点v出发的深度优先遍历的非递归过程。

【参考答案】

Void DFSn(Graph G,int v)

{ //从第v个顶点出发非递归实现深度优先遍历图G

Stack s;

SetEmpty(s);

Push(s,v);

While(!StackEmpty(s))

{ //栈空时第v个顶点所在的连通分量已遍历完

Pop(s,k);

If(!visited[k])

{ visited[k]=TRUE;

VisitFunc(k); //访问第k个顶点

//将第k个顶点的所有邻接点进栈

for(w=FirstAdjVex(G,k);w;w=NextAdjVex(G,k,w))

{

if(!visited[w]&&w!=GetTop(s)) Push(s,w);

//图中有环时w==GetTop(s)

}

}

}

* 1. 已知邻接表表示的有向图，请编程判断从第*u*顶点至第*v*顶点是否有简单路径，若有则印出该路径上的顶点。

【参考答案】

void Allpath (AdjList g, vertype u, vertype v)

//求有向图g中顶点U到顶点v的所有简单路径，初始调用形式

{ int top = 0 , s [ ] ;

s [++top ] = u; visited[u]= 1 ;

while (top>0|| p)

{ p = g[s[top]].first are ; //第一邻接点

while (p!=null&&visited[p->adjvex]==1) p=->next;

//下一个访问邻接点表

if (p = = null) top--; //退栈

else { i = p->adjvex; //取邻接点(编号)

if (i = = v)//找到从U到v的一条简单路径，输出

{ for (k = 1 ; k < = top; k++)

printf ( "% 3 d"，s[ k] ) ;

printf ( "% 3 d\n" ，v) ;

} //if

else { visited[i] = 1;

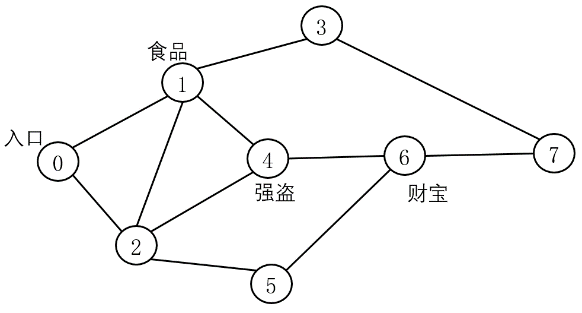
s[++top ] = i;

}// else深度优先遍历

}// else

} //while

}// AllSPdfs



* 1. 有这样一个洞穴探宝问题：针对如右图所示的藏宝图，要找到从入口到出口的一条路径，该路径必须经过“食品”和“财宝”的地方以补充食物并得到财宝，但要绕开“强盗”居住地。

（注：此题不要求写出C/C++代码，文字描述求解步骤即可）

【参考答案】

1. 不要求写成程序源码，算法可以任意方式描述。
2. 这是深度搜索的一个改进，采用邻接矩阵和邻接表均可，但要明确顶点信息且包含食品/财宝/强盗标志。
3. 遍历的时候把途径的点用flag 数组标记。
4. 设访问过标记为0，访问过标记为1，特别强调把强盗点也标记为1。
5. 每次访问标记为0的邻接点，最后当访问7点时候暂停，找到一条新路。
6. 新路如果过财宝，食品点，说明新路成功并且合法，回溯，继续查找下一条合法的路。
7. 如果没有经过财宝和食品点都经过说明新路非法，回溯，继续查找下一条合法的路。