数据结构

深圳技术大学
大数据与互联网学院

第七章 图

7.1 图的定义和术语
7.2 图的存储结构
7.3 图的遍历
7.4 图的连通
7.5 有向无环图及其应用
7.6 最短路径

上节复习

- 图的邻接矩阵
  - 二维数组表示顶点之间的连接，有连接为1，无连接为0
  - 网的邻接矩阵，有连接则用权值表示，无连接为无穷大或0
- 邻接矩阵的性质
  - 无向图的邻接矩阵是对称的，有向图的则不一定
  - 无向图中，顶点i的度等于矩阵第i行的非0元素个数
  - 有向图中，顶点i的出度等矩阵第i行的非0元素个数，顶点i的入度等矩阵第i列的非0元素个数
- 图的邻接表
  - 顺序表+单链表来表示顶点间的连接，顺序表表示顶点，单链表表示每个顶点相连接的其他顶点
  - 邻接表，根据顶点的出度建立
  - 逆邻接表，根据顶点的入度建立

练习

- 设图G=(V,E)，其中V={a, b, c, d, e}，E={<a, b>, <a, d>, <b, a>, <c, b>, <c, d>, <d, e>, <e, a>, <e, b>, <e, c>}
  - 画出该图
  - 画出该图的邻接矩阵
  - 画出该图的正邻接链表和逆邻接链表

7.2 图的存储结构

三. 十字链表

- 十字链表是有向图的另一种存储结构
- 十字链表是将有向图的邻接表和逆邻接表结合起来的一种存储结构

7.2 图的存储结构

三. 十字链表

- 顶点的结点结构
  - data;   // 与顶点相关的信息
  - firstin; // 指向以顶点为弧头的第一个弧结点
  - firstout;// 指向以顶点为弧尾的第一个弧结点

data	firstin	firstout

7.2 图的存储结构

三. 十字链表

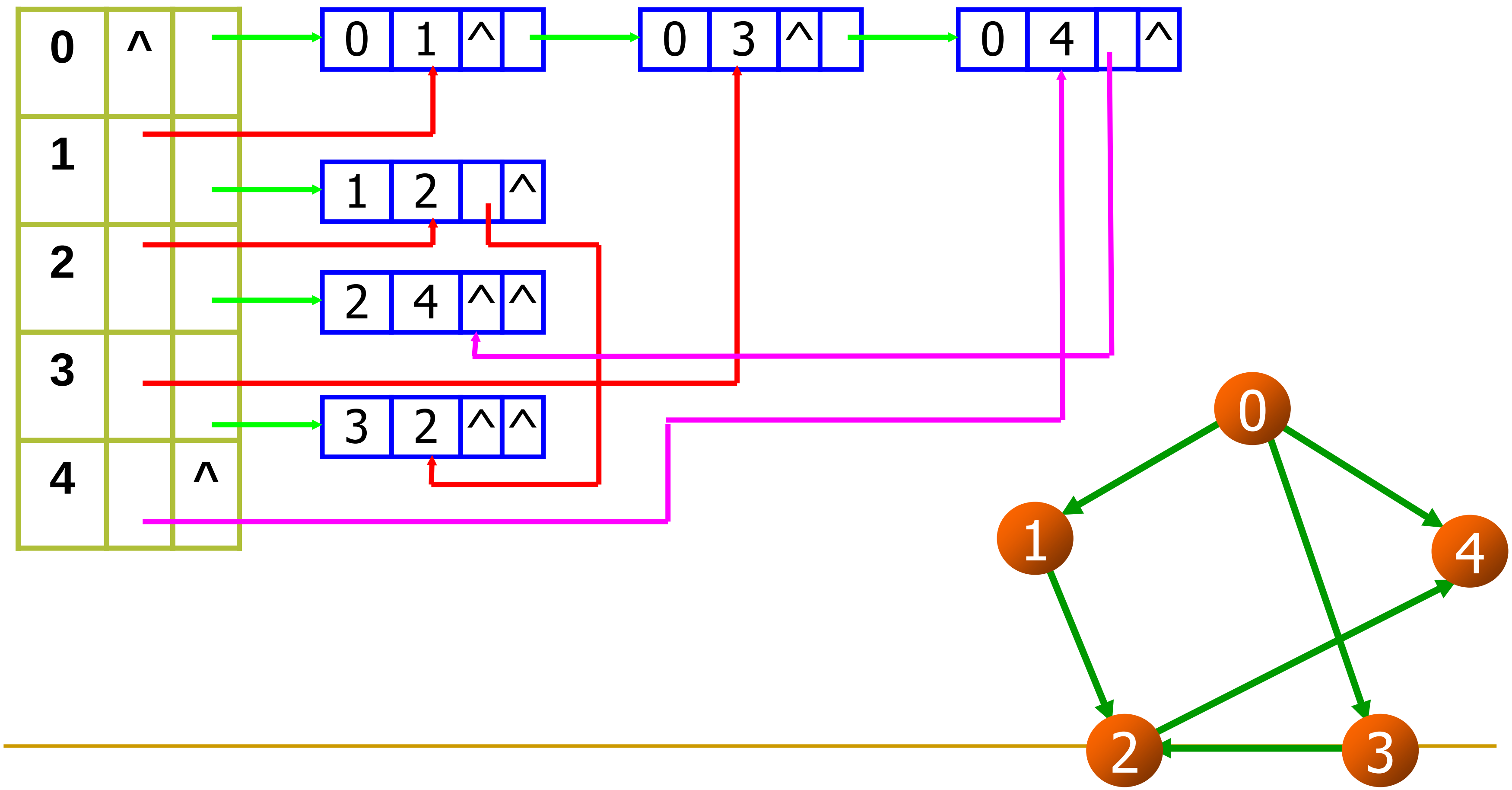
- 弧的结点结构
  - tailvex;// 弧尾顶点的位置
  - headvex;// 弧头顶点的位置
  - hlink; // 指向弧头相同的下一条弧
  - tlink; // 指向弧尾相同的下一条弧
  - info; // 该弧相关信息的指针或权值

tailvex	headvex	hlink	tlink	info

7.2 图的存储结构

三. 十字链表

- 十字链表举例



7.2 图的存储结构

四. 邻接多重表

- 邻接多重表，Adjacency Multilist，是无向图的另一种存储结构
- 在无向图邻接表中，一条边要用2个结点表示(分别从2个顶点的角度看)
- 在邻接多重表中，一条边只用一个结点表示
- 将所有具有某顶点的结点，全部用链连结起来，链所在的域为该顶点对应的指针域

7.2 图的存储结构

四. 邻接多重表

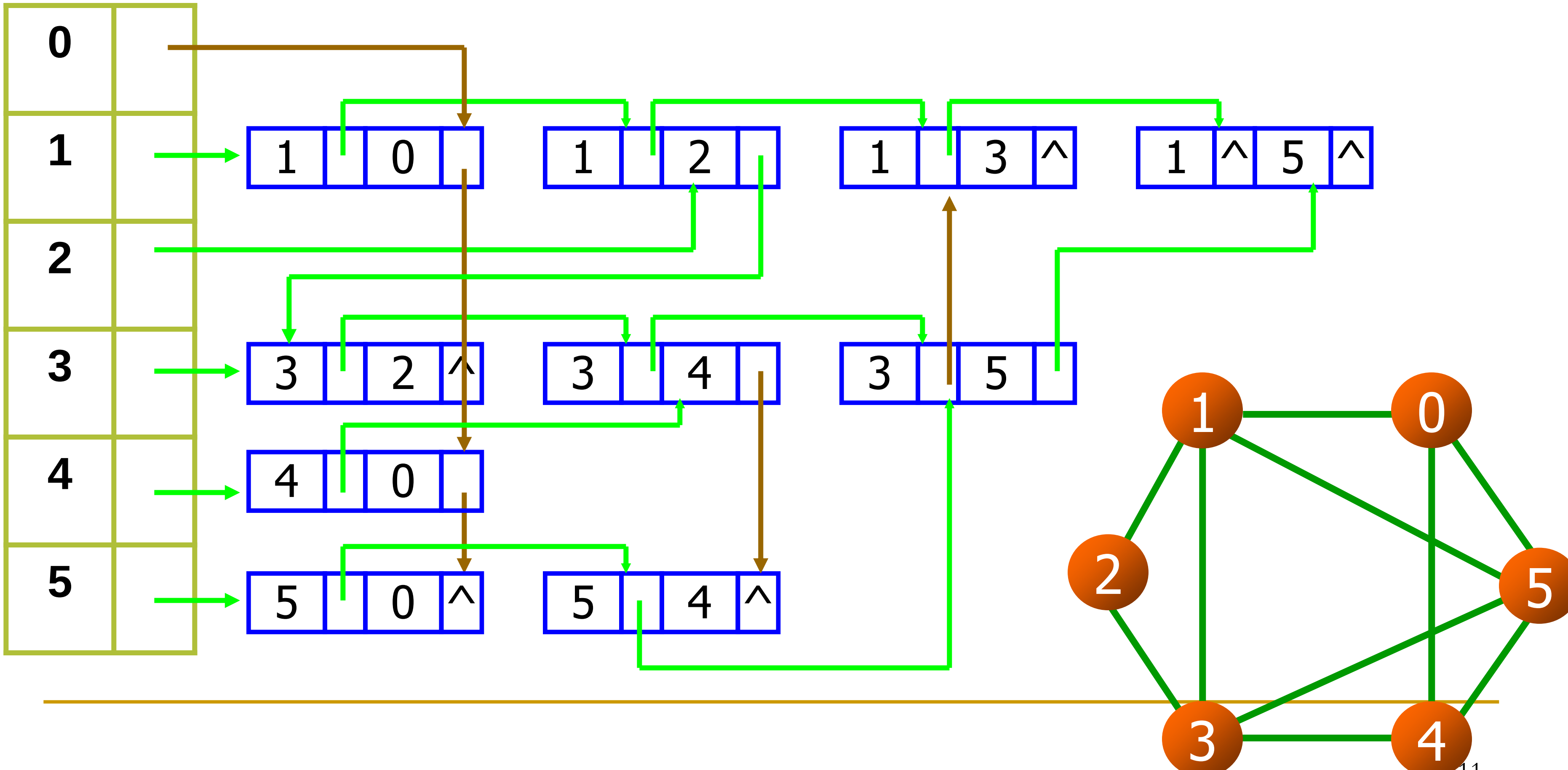
- 边的结点结构
  - mark; // 标记域，如指示该边是否被搜索过
  - ivex, jvex;// 该边所依附的两个顶点的位置
  - ilink;// 指向下一条依附于ivex的边
  - jlink;// 指向下一条依附于jvex的边
  - info; // 该边相关信息的指针或权值

mark	ivex	ilink	jvex	jlink	info

7.2 图的存储结构

四. 邻接多重表

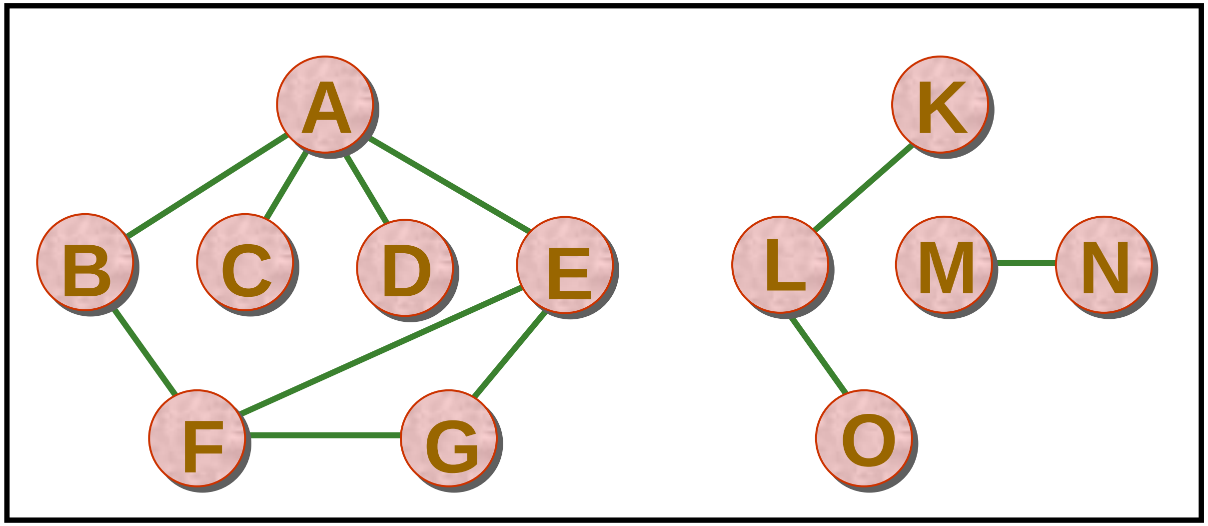
- 该链表从左上角结点1开始建立，优先建立度大的结点
  - 先建立顶点1，再建立顶点3，以此类推



7.4 图的连通

四. 无向图的连通性

- 如果无向图中，存在不连通的顶点，则该图称为非连通图



7.4 图的连通

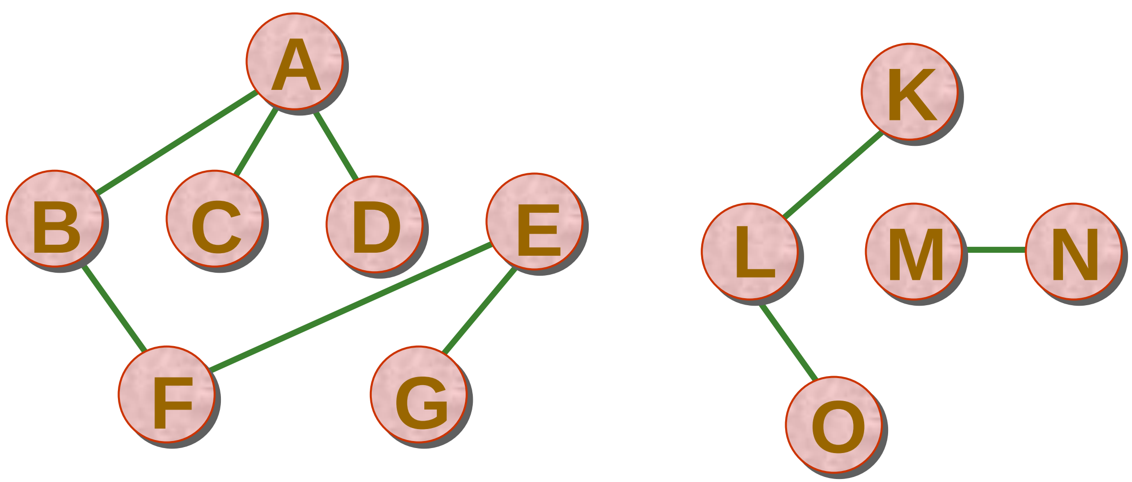
四. 无向图的连通性

- 非连通图的极大连通子图叫做连通分量
- 若从无向图的每一个连通分量中的一个顶点出发进行DFS或BFS遍历，可求得无向图的所有连通分量的生成树(DFS或BFS生成树)
- 所有连通分量的生成树组成了非连通图的生成森林

7.4 图的连通

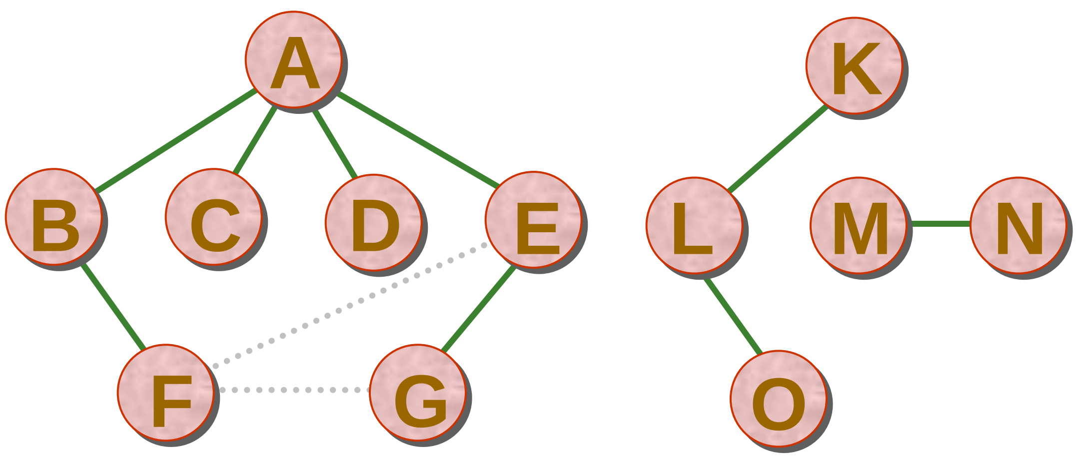
四. 无向图的生成树

- 无向图由DFS遍历，求得连通分量称为DFS生成树
  - 下图的三棵DFS生成树组成一个生成森林
    - A-B-F-E-G-C-D
    - K-L-O
    - M-N



四. 无向图的生成树

- 无向图由BFS遍历，求得连通分量称为BFS生成树
  - 下图的三棵BFS生成树组成一个生成森林
    - A-B-C-D-E-F-G
    - K-L-O
    - M-N



五. 最小生成树

- 如果无向图中，边上有权值，则称该无向图为无向网
- 如果无向网中的每个顶点都相通，称为连通网
- 最小生成树(Minimum Cost Spanning Tree)是代价最小的连通网的生成树，即该生成树上的边的权值和最小

7.4 图的连通

五. 最小生成树

- 最小生成树准则
  - 必须使用且仅使用连通网中的n-1条边来联结网络中的n个顶点；
  - 不能使用产生回路的边；
  - 各边上的权值的总和达到最小。

7.4 图的连通

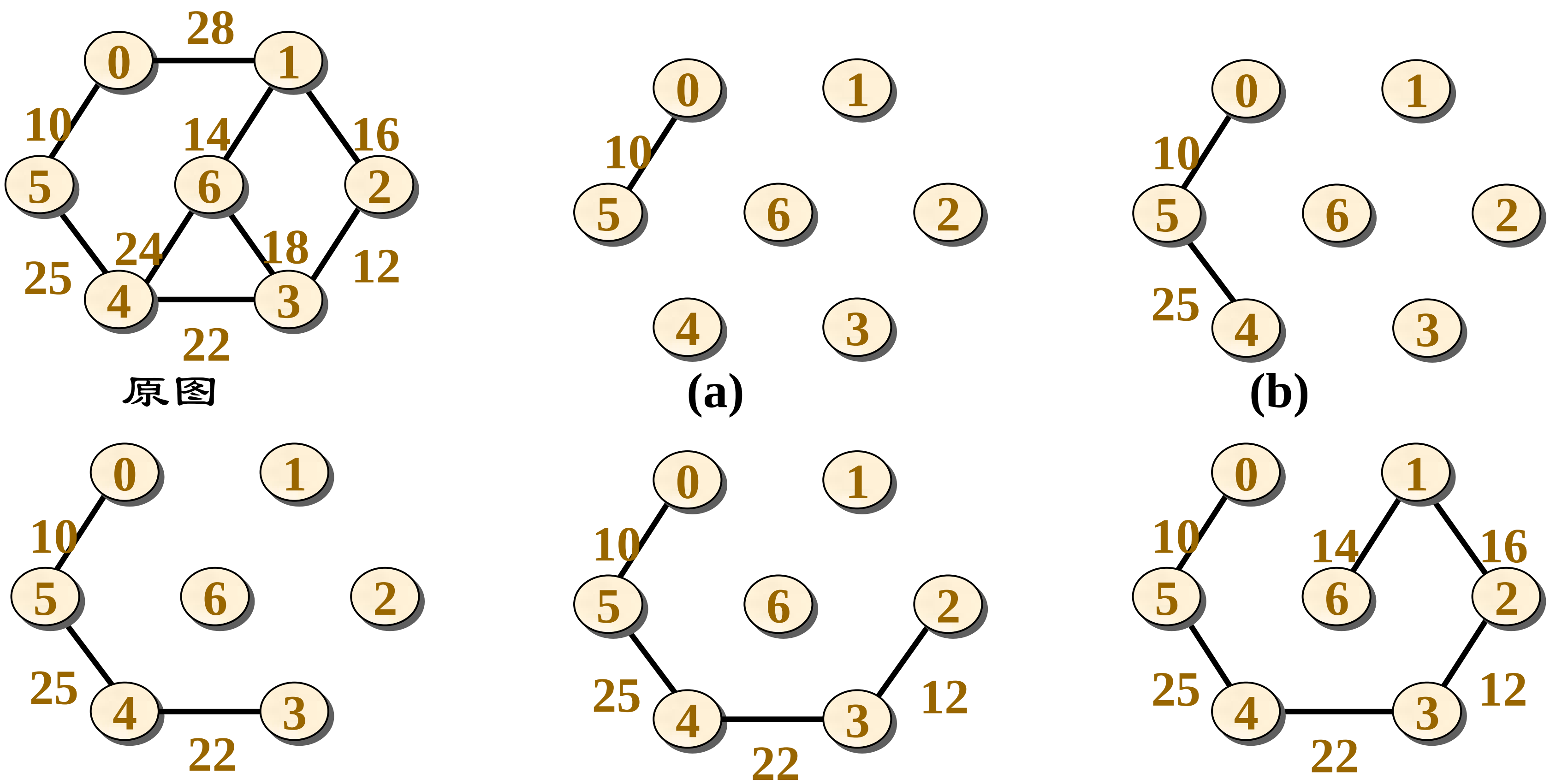
五. 最小生成树

- 最小生成树生成算法——普里姆(Prim)算法
  - 假设N=(V, E)是连通网
  - TE是N上最小生成树中边的集合
- 1. U={u_0}，（u_0∈V），TE={}
- 2. 在所有u∈U, v∈V-U的边(u, v)∈E中找一条代价最小的边(u, v_0)并入集合TE，同时v_0并入U
- 3. 重复2，直到U=V

7.4 图的连通

五. 最小生成树

- 普里姆(Prim)算法举例



原图

(a)

(b)

7.4 图的连通

```
void MiniSpanTree_PRIM(MGraph G, VertexType u) {  // 算法7.9
  int i,j,k;
  k = LocateVex ( G, u );
  for ( j=0; j<G.vexnum; ++j ) {      // 辅助数组初始化
    if (j!=k)
     { closedge[j].adjvex=u; closedge[j].lowcost=G.arcs[k][j].adj; }
  }
  closedge[k].lowcost = 0;      // 初始， U＝{u}
  for (i=1; i<G.vexnum; ++i) {  // 选择其余G.vexnum-1个顶点
    k = minimum(closedge);      // 求出T的下一个结点：第k顶点
      // 此时closedge[k].lowcost =
      // MIN{ closedge[vi].lowcost | closedge[vi].lowcost>0, vi∈V-U }
    printf(closedge[k].adjvex, G.vexs[k]);   // 输出生成树的边
    closedge[k].lowcost = 0;    // 第k顶点并入U集
    for (j=0; j<G.vexnum; ++j)
      if (G.arcs[k][j].adj < closedge[j].lowcost) {
         // 新顶点并入U后重新选择最小边
         // closedge[j] = { G.vexs[k], G.arcs[k][j].adj };
        closedge[j].adjvex=G.vexs[k];
        closedge[j].lowcost=G.arcs[k][j].adj;
      }//end for
  }//end for
} // MiniSpanTree
```

7.4 图的连通

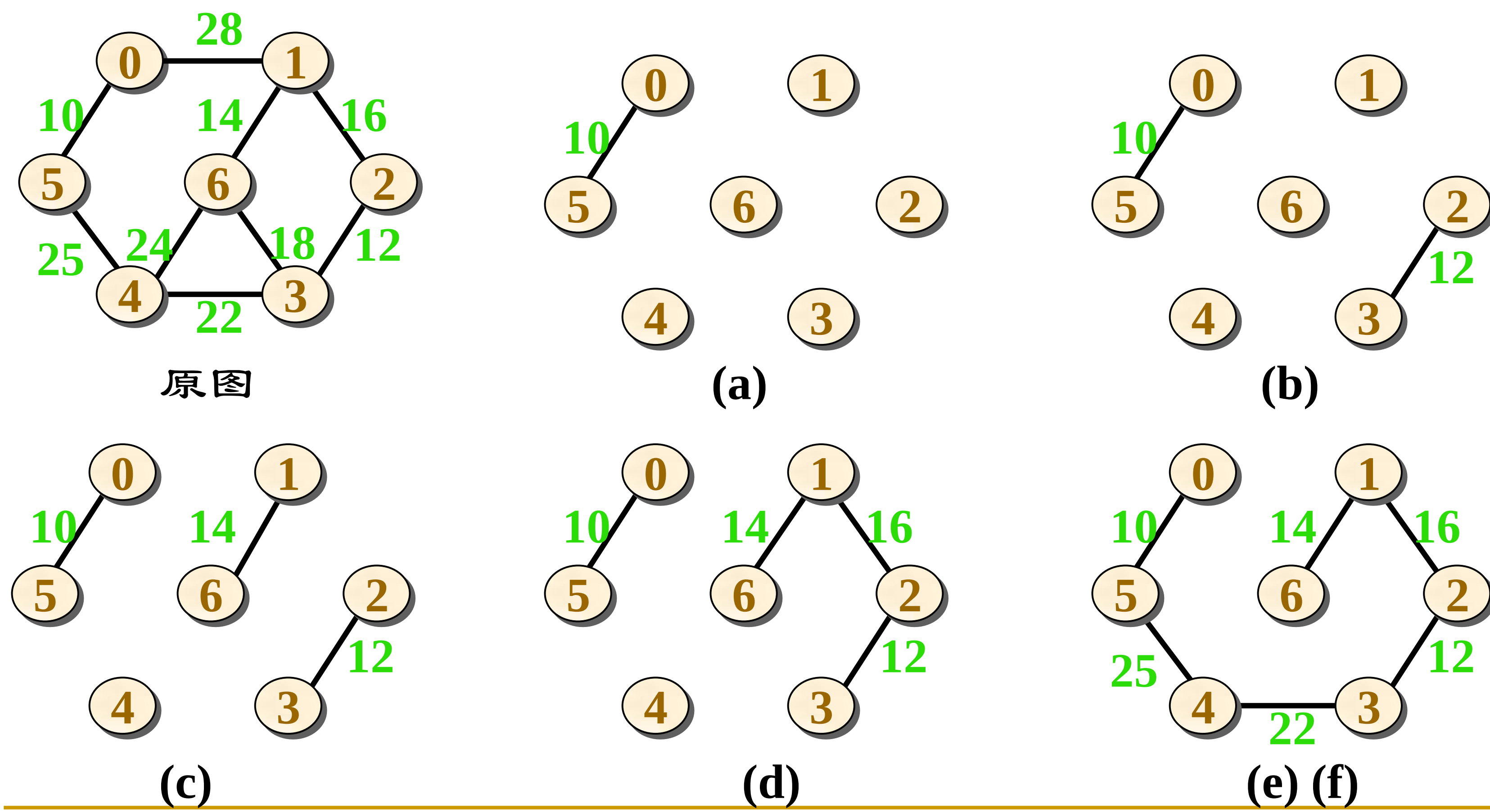
五. 最小生成树

- 最小生成树生成算法——克鲁斯卡尔(Kruskal)算法
  - 假设N=(V,E)是连通网
- 1.非连通图T={V,{}}，图中每个顶点自成一个连通分量
- 2.在E中找一条代价最小，且其两个顶点分别依附不同的连通分量的边，将其加入T中
- 3.重复2，直到T中所有顶点都在同一连通分量上

7.4 图的连通

五. 最小生成树

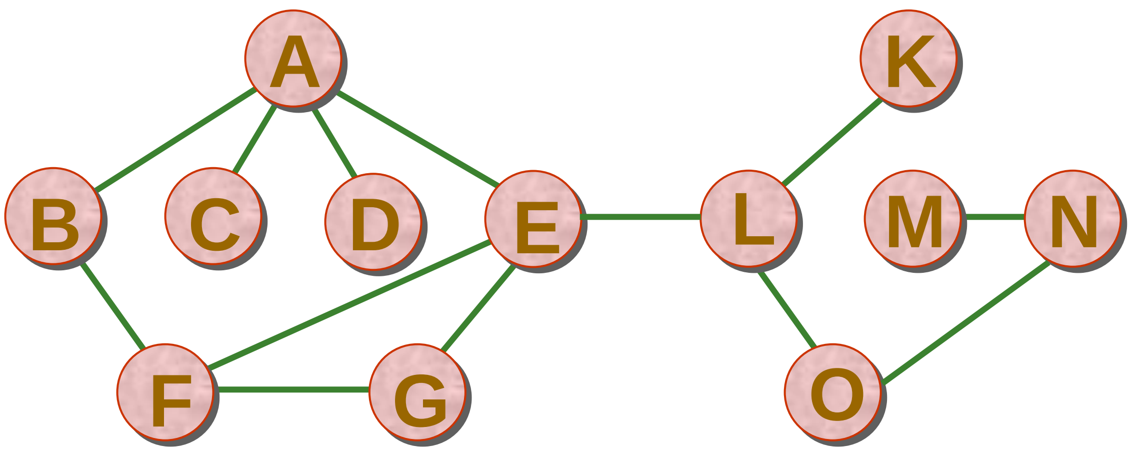
- 克鲁斯卡尔(Kruskal)算法举例



7.4 图的连通

六. 关节点和重连通图

- 关节点：若把顶点v和相关的边删除，则图的一个连通分量变成两个以上分量，则v是图的一个关节点
  - 下图关节点：A、E、L、O、N



7.4 图的连通

六. 关节点和重连通图

- 在连通图中，如果没有关节点，则为重连通图，
  - 在重连通图中，任意一对顶点都至少存在两条路径