8.5 生成树和最小生成树

8.5.1 生成树的概念

一个连通图的生成树是一个极小连通子图，它含有图中全部n个顶点和构成一棵树的$(n-1)$条边。

命题：如果在一棵生成树上添加一条边，必定构成一个环。

可以通过遍历方法产生生成树：

- 由深度优先遍历得到的生成树称为深度优先生成树。

- 由广度优先遍历得到的生成树称为广度优先生成树。

一个连通图的生成树不一定是唯一的！

最小生成树的概念

- 对于带权连通图G （每条边上的权均为大于零的实数），可能有多棵不同生成树。
- 每棵生成树的所有边的权值之和可能不同。
- 其中权值之和最小的生成树称为图的最小生成树。

8.5.2 非连通图和生成树

连通图：仅需调用遍历过程（DFS或BFS）一次，从图中任一顶点出发，便可以遍历图中的各个顶点，产生相应的生成树。

非连通图：需多次调用遍历过程。每个连通分量中的顶点集和遍历时走过的边一起构成一棵生成树。所有连通分量的生成树组成非连通图的生成森林。

重点：求带权连通图的最小生成树

8.5.3 普里姆算法

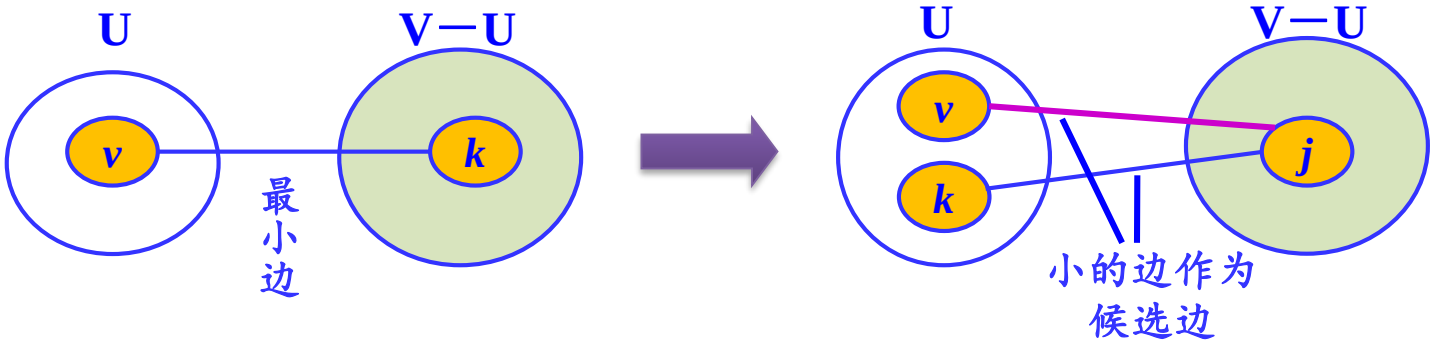
普里姆（Prim）算法是一种构造性算法，用于构造最小生成树。

（1）初始化U={*v*}。*v*到其他顶点的所有边为候选边；

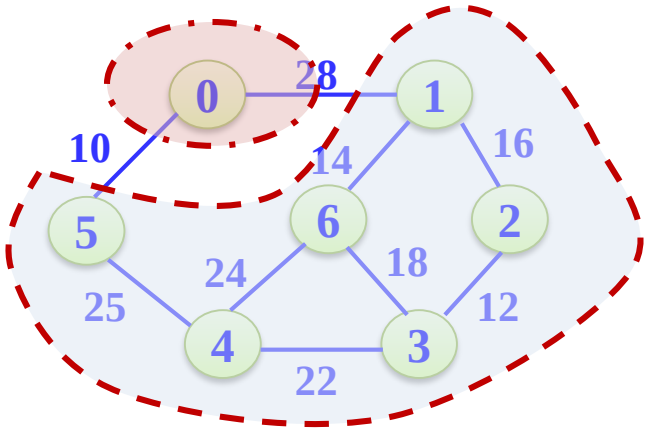
（2）重复以下步骤*n*−1次，使得其他*n*−1个顶点被加入到U中：

❶ 从候选边中挑选权值最小的边输出，设该边在V−U中的顶点是*k*，将*k*加入U中；

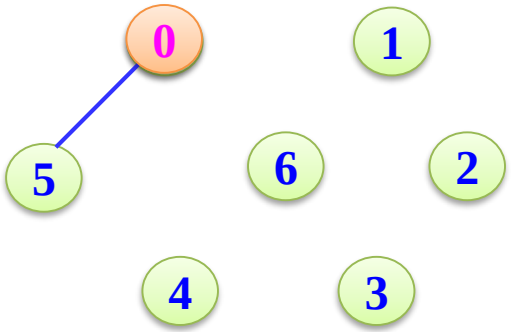
❷ 考察当前V−U中的所有顶点*j*，修改候选边：若(*j*，*k*)的权值小于原来和顶点*k*关联的候选边，则用(*k*，*j*)取代后者作为候选边。



Prim算法示例演示（起点0）



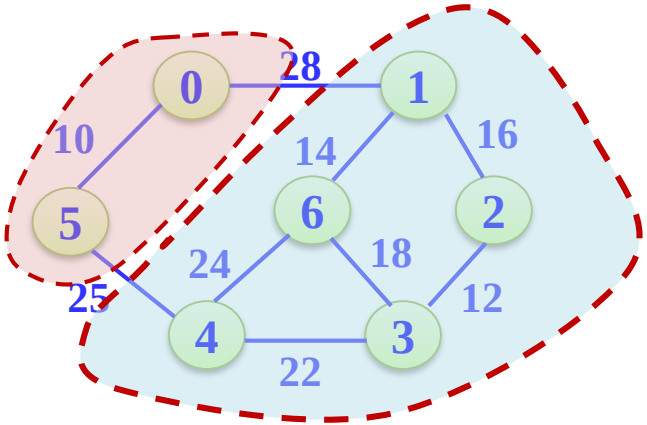
图G



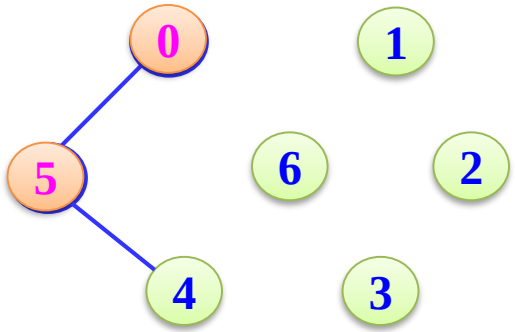
U={ 0 }

普里姆算法求解最小生成树的过程

Prim算法示例演示（起点0）



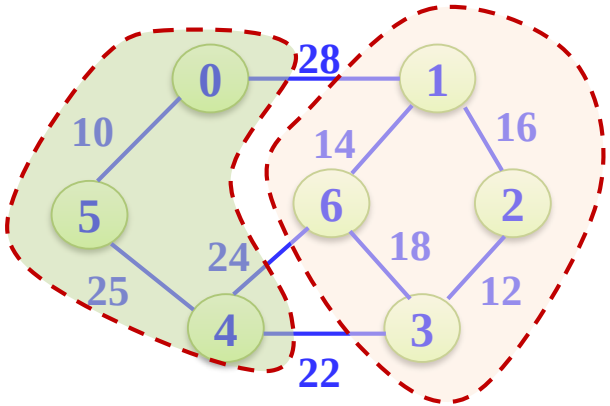
图G



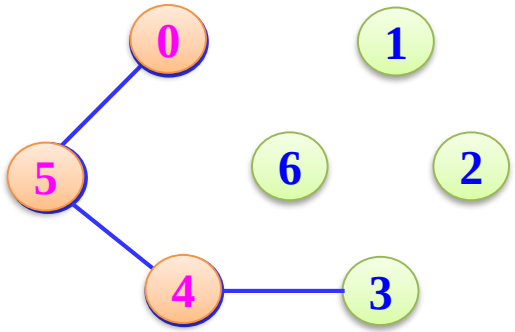
U={ 0， 5 }

普里姆算法求解最小生成树的过程

Prim算法示例演示（起点0）



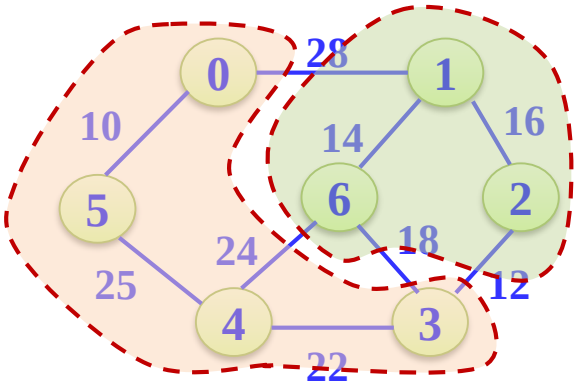
图G



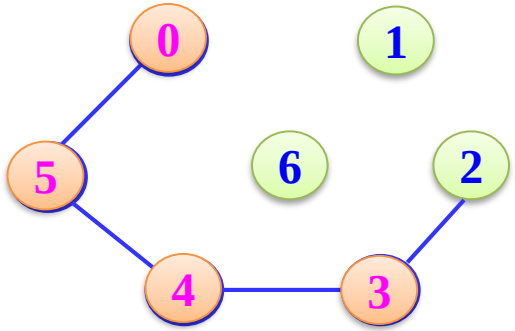
U={ 0， 5 ， 4 }

普里姆算法求解最小生成树的过程

Prim算法示例演示（起点0）



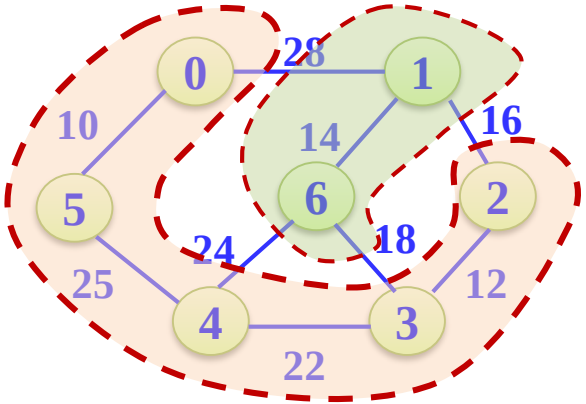
图G



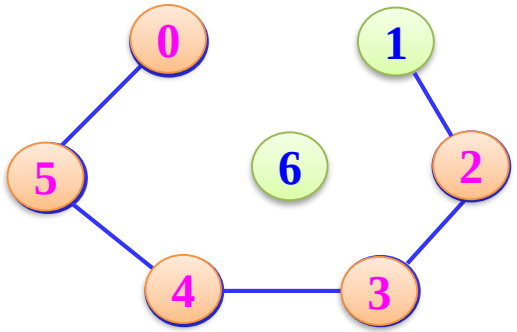
U={ 0， 5， 4， 3 }

普里姆算法求解最小生成树的过程

Prim算法示例演示（起点0）



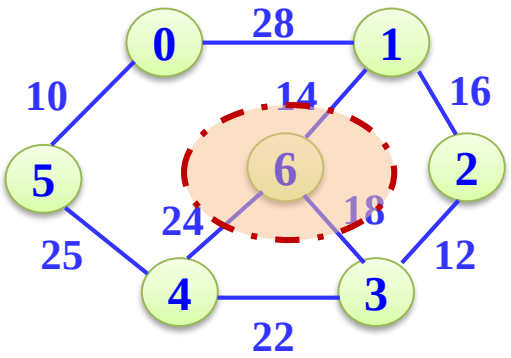
图G



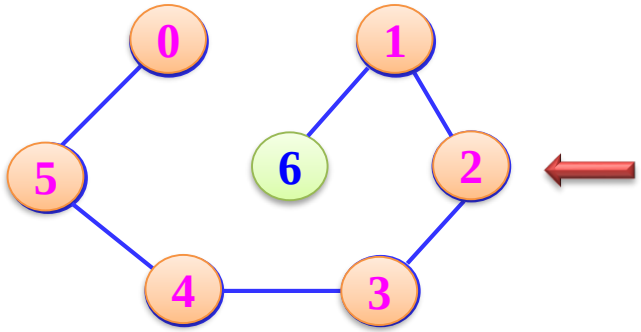
U={ 0，5，4，3，2 }

普里姆算法求解最小生成树的过程

Prim算法示例演示（起点0）



图G



最小生成树

U={ 0，5，4，3，2，1，6 }

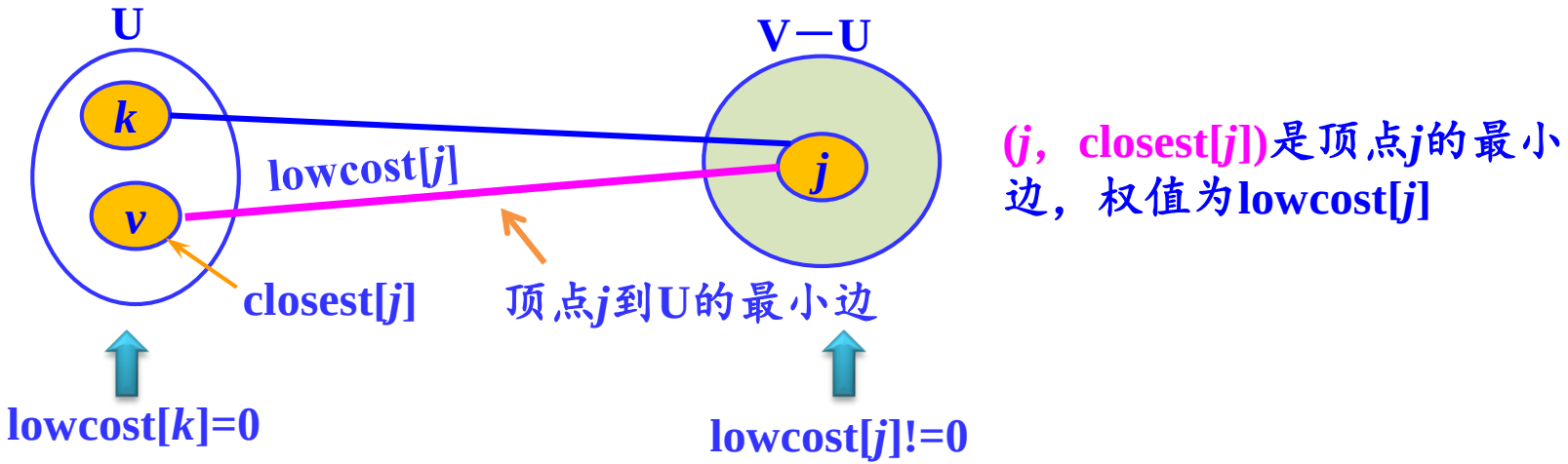
普里姆算法求解最小生成树的过程

算法设计（解决4个问题）：

- 如何求U、V-U两个顶点集之间的最小边？（只求一条）

  只考虑V-U中顶点*j*到U顶点集的最小边（无向图），比较来找最小边

- 如何存储顶点*j*到U顶点集的最小边？

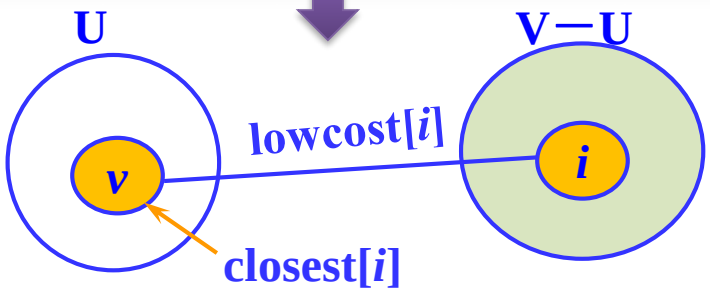


(*j*，closest[*j*])是顶点*j*的最小边，权值为lowcost[*j*]

- 一个顶点属于哪个集合？
- 图采用哪种存储结构更合适？ 邻接矩阵

普里姆（Prim）算法如下：

```
#define INF 32767                          //INF表示∞
void  Prim(MGraph g,int v)
{     int lowcost[MAXV];
      int min;
      int closest[MAXV], i, j, k;
      for (i=0;i<g.n;i++)                  //给lowcost[]和closest[]置初值
      {       lowcost[i]=g.edges[v][i];
              closest[i]=v;
      }
```

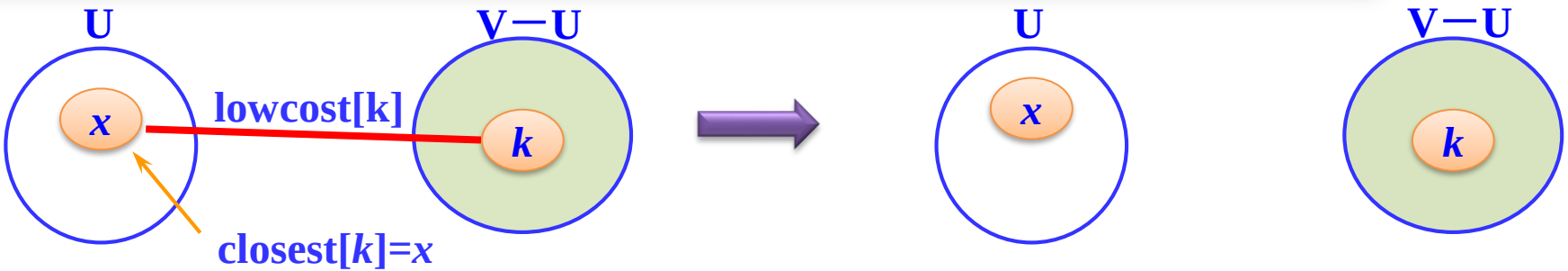


U中只有一个顶点v

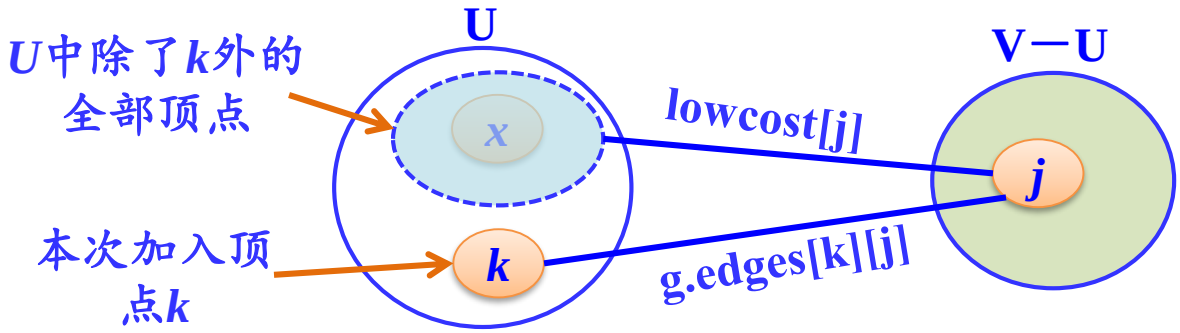
顶点i到U的最小边：

(v，g.edges[v][i])

```
for (i=1;i<g.n;i++)                    //输出(n-1)条边
{     min=INF;
      for (j=0;j<g.n;j++)              //在(V-U)中找出离U最近的顶点k
          if (lowcost[j]!=0 && lowcost[j]<min)
          {       min=lowcost[j];
                  k=j;                 //k记录最近顶点编号
          }
      printf(" 边(%d,%d)权为:%d\n",closest[k],k,min);
      lowcost[k]=0;                    //标记k已经加入U
```



输出： (closest[k], k)




```
        for (j=0;j<g.n;j++) //修改数组lowcost和closest
            if (lowcost[j]!=0 && g.edges[k][j]<lowcost[j])
            {   lowcost[j]=g.edges[k][j];
                closest[j]=k;
            }
    }
}
```

仅仅考虑V-U中的顶点

普里姆算法思路

局部最优 + 调整 ＝ 全局最优

贪心算法思想　　最优结果

普里姆算法分析

Prim()算法中有两重for循环，所以时间复杂度为$O(n^2)$。

思考题

为什么说Prim算法更适合稠密图求最小生成树。

——本讲完——