石家莊鐵道大學
SHIJIAZHUANG TIEDAO UNIVERSITY

在线开放课程

数据结构— 图

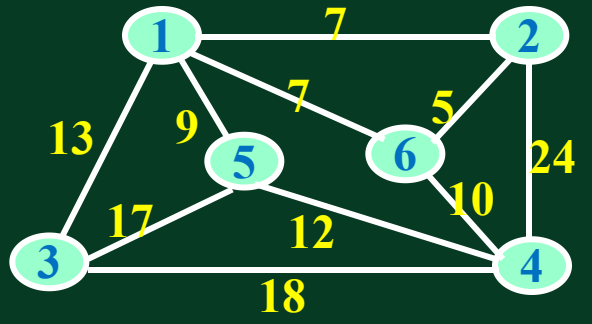
图的应用—最小生成树
–普里姆算法

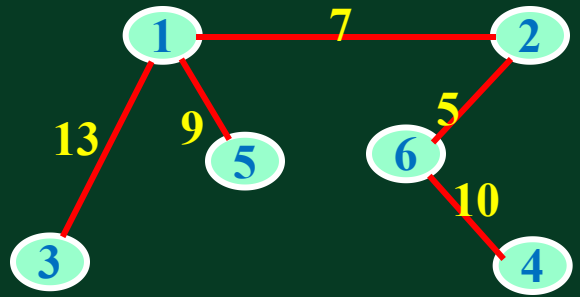
主讲 ： 石玉晶

目录

- 1、最小生成树的概念
- 2、普里姆算法的构造过程
- 3、普里姆算法的实现

一、最小生成树的概念





- 问题描述
  - 用一个连通网表示 n 个居民点和各个居民点之间可能架设的通讯线路，网中边上的权值表示架设这条线路所需经费。
  - 在 n 个居民点间构建通讯网只需架设 n-1 条线路，则工程队面临的问题是架设哪几条线路能使总的工程费用最低？
  - 问题均等价于：在含有 n 个顶点的连通网中选择 n-1 条边，构成一棵极小连通子图，并使该连通子图中 n-1 条边上权值之和达到最小，则称这棵连通子图为连通网的最小生成树。
  - 类似此类的问题很多。

二、普里姆算法的构造过程

- **基本思想**
  - 首先选取图中任意一个顶点 v 作为生成树的根，之后继续往生成树中添加顶点 w，要求在顶点 w 和树的顶点之间必须有边，且该边上的权值应在所有和 树 相邻接的边中权最小。

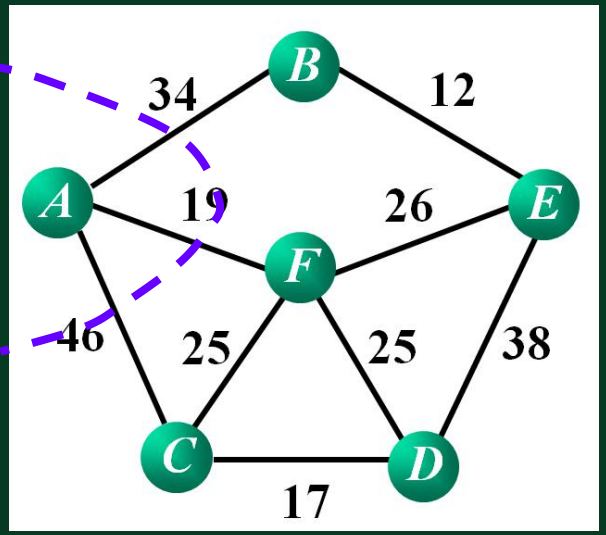
二、普里姆算法的构造过程

具体作法

✓ TE是最小生成树T中边的集合，U为最小生成树T的顶点集合。

- 初始令U={u0}，(u0∈V)， TE=Φ；
- 在所有u∈U，v∈V-U的边(u, v)∈E中，找一条代价最小的边(u0, v0)；
- 将(u0, v0)并入集合TE，同时v0并入U；
- 重复上述操作直至U=V为止，则T=(V, {TE})为最小生成树。

二、普里姆算法的构造过程

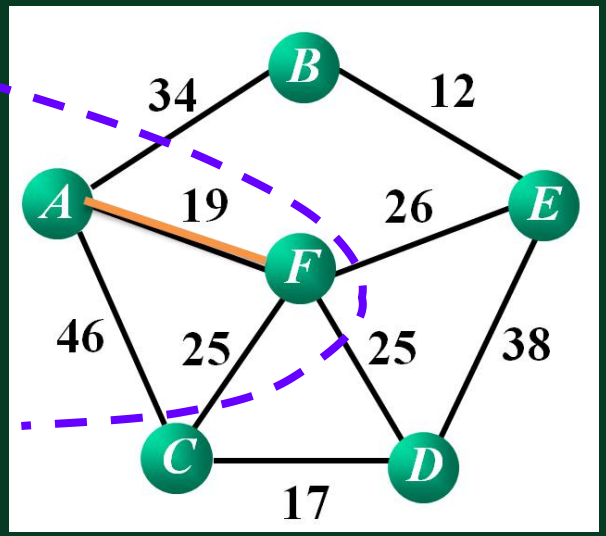


U={A}

V-U={B, C, D, E, F}

cost={(*A*, *B*)34, (*A*, *C*)46,
(*A*, *D*)∞,(*A*, *E*)∞,(*A*, *F*)19}

二、普里姆算法的构造过程



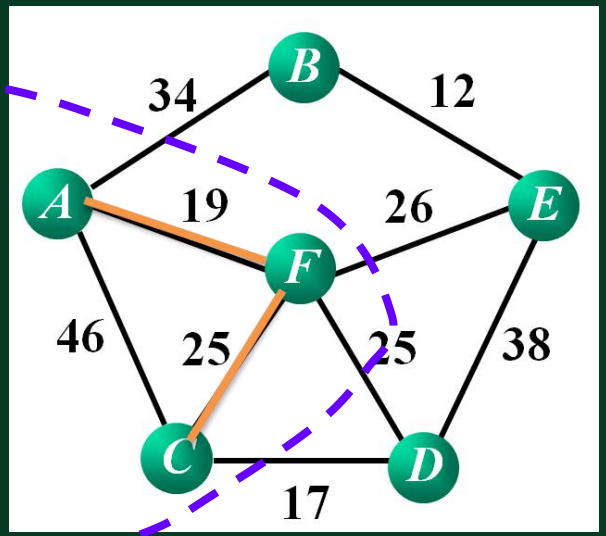
U={A, F}

V-U={B, C, D, E}

cost={(*A*, *B*)34, (A, C)46,

(F, *C*)25, (*F*, *D*)25,(*F*, *E*)26}

二、普里姆算法的构造过程



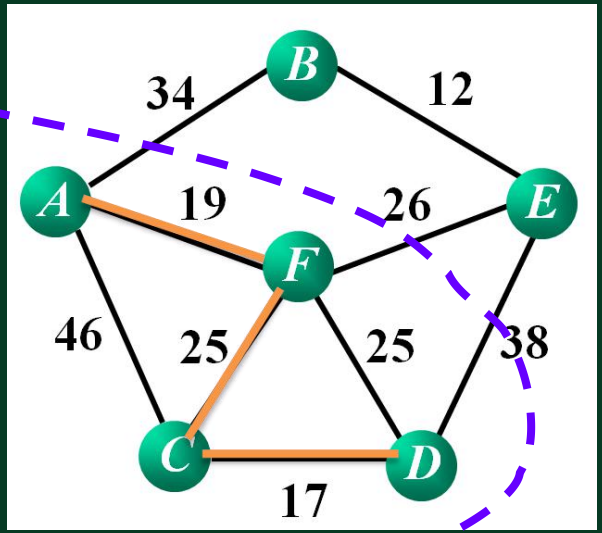
U={A, F, C}

V-U={B, D, E}

cost={(*A*, *B*)34, (F,D)25,

(*C*, *D*)17,(*F*, *E*)26}

二、普里姆算法的构造过程

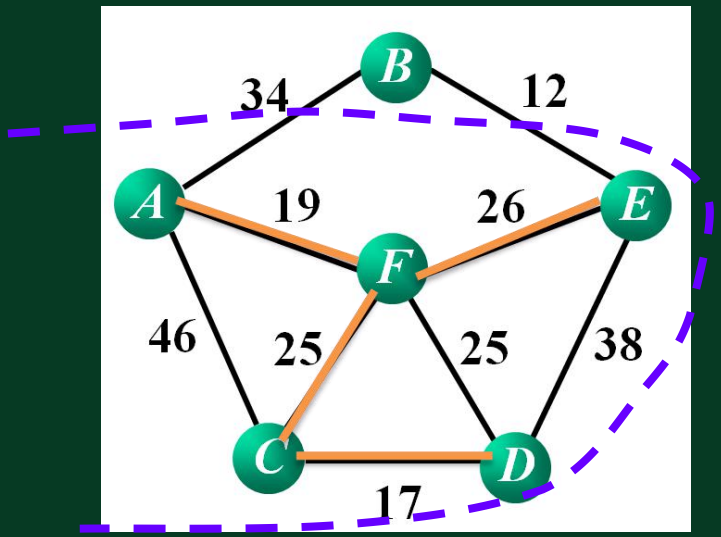


U={A, F, C, D}

V-U={B, E}

cost={(*A*, *B*)34, (*F*, *E*)26, (D,E)38}

二、普里姆算法的构造过程

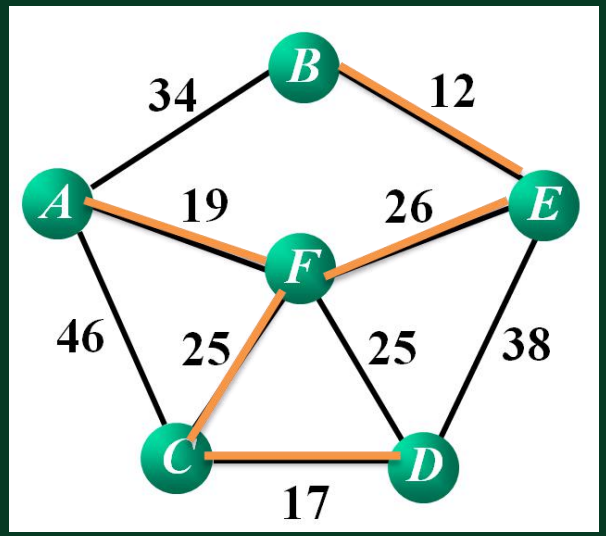


U={A, F, C, D, E}

V-U={B}

cost={(*A*, *B*)34,(*E*, *B*)12}

二、普里姆算法的构造过程



U={A, F, C, D, E, B}

V-U={ }

三、普里姆算法的实现

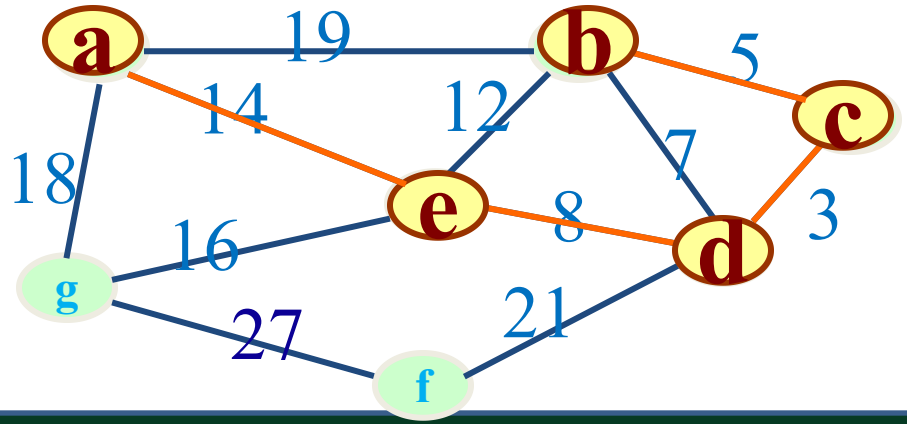
- 辅助数据结构

```
Struct
{     VerTexType   adjvex;
      Arctype      lowcost;
}closedge[MVNUM];  //记录图中各顶点V到U的权值最小的边
```

closedge[i].adjvex

closedge[i].lowcost

vi顶点到U中最短距离的那个点及边长，如果lowcost=0,表示vi已经进入U



closedge	0a	1b	2c	3d	4e	5f	6g
Adjvex		c	d	e	a	d	e
Lowcost		5	3	8	14	21	16

三、普里姆算法的实现

- 【算法步骤】
  - 1.初始顶点u进入U，并对V中顶点初始化closedge[i].lowcost和closedge[i]. adjvex；
  - 2. 重复执行下列操作n−1次
    - 2.1 从lowcost中选择最小边closedge[k]，输出该边(closedge[k].adjvex,k, closedge[k].lowcost);
    - 2.2 k加入U中；
    - 2.3 更新数组closedge的lowcost和adjvex，使得closedge[i].lowcost为vi到U的最小值。

```
*prim 算法(加点法)*/
MiniSpanTree_Prim(AMGraph G, VerTexType u) {
/*从顶点 u 出发，按 prim 算法构造连通网 G 的最小生成树，并输出生成树的每条边*/
    k=LocateVex(G,u);
    closedge[k].lowcost = 0;   closedge[k].adjvex =k;    /*初始化*，  U = {u}*/
    for(i = 0; i < G.vexnum; i++)
        if(i != k)          /*对 V-U 的顶点 vi，初始化 closedge[i]*/
            closedge[i]={ k, G.arcs[k][i]};
        for(i= 1; i < G.vexnum; i++) {      /*找 n-1 条边*/
         k = Mimium(closedge);       /*closedge 中存有当前最小边(u，v)的信息*/
         u0=closedge[k].adjvex;
         v0=G.vexs[k];
         cout<<u0<<v0;  /*输出生成树的当前最小边(u0，v0)*/
         closedge[k].lowcost = 0;        /*将顶点 v0 纳入 U 集合*/
        for(j = 0; j< G.vexnum; j++)       /*顶点 v0 纳入 U 集合后,更新 closedge*/
            if(G.arcs[k][j] < closedge[j].lowcost)
                closedge[j] = {G.vexs[k],G.arcs[k][j] };
    }
}
```

小结

- 图的应用——最小生成树——prim算法

小结

石家庄铁道大学 SHIJIAZHUANG TIEDAO UNIVERSITY

在线开放课程

- 接下来学习：
- 图的应用—最小生成树—克鲁斯卡尔算法

谢谢！