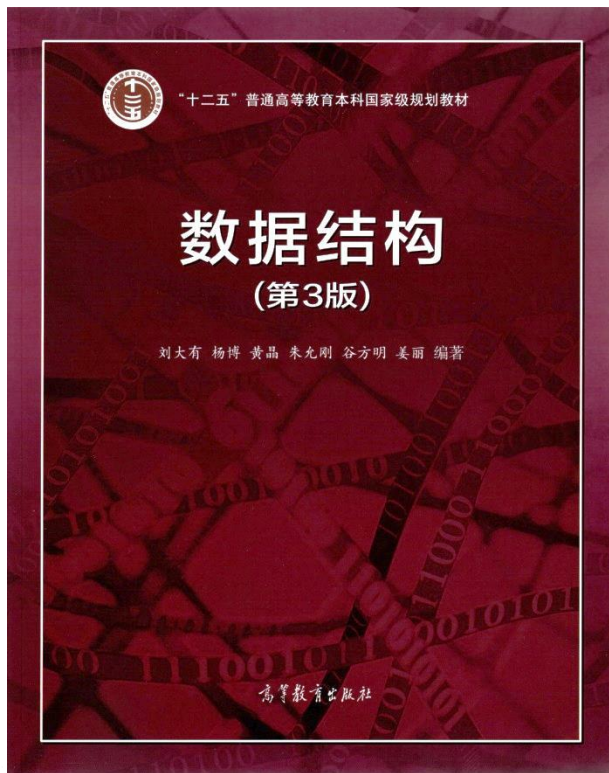




计算机学院王湘浩班
2024级



最小支撑树及图应用

- 最小支撑树的概念
- Prim算法
- Kruskal算法
- 强连通分量

数据之法
结构之美
算法之道



戴文渊

上海交通大学本科硕士

香港科技大学博士

ACM/ICPC全球总决赛冠军

百度主任架构师、华为主任科学家

第四范式创始人兼CEO

身价85亿元人民币（据胡润财富榜）

学习达不到自己预期时，不用慌。每个人的预期总会或多或少的带点理想色彩，所以实际的效果一般来说总是不如预期的。有时甚至会出现只有预期的三分之一到二分之一之一的情况。

但是，只要你坚持不懈地去做，不动摇自己的信念，最后的成绩一定不会差。

记住，你遇到的困难，你的对手也会遇到的。



最小支撑树及图应用

- 最小支撑树概念
- Prim算法
- Kruskal算法

数据之美
结构之美
算法之道

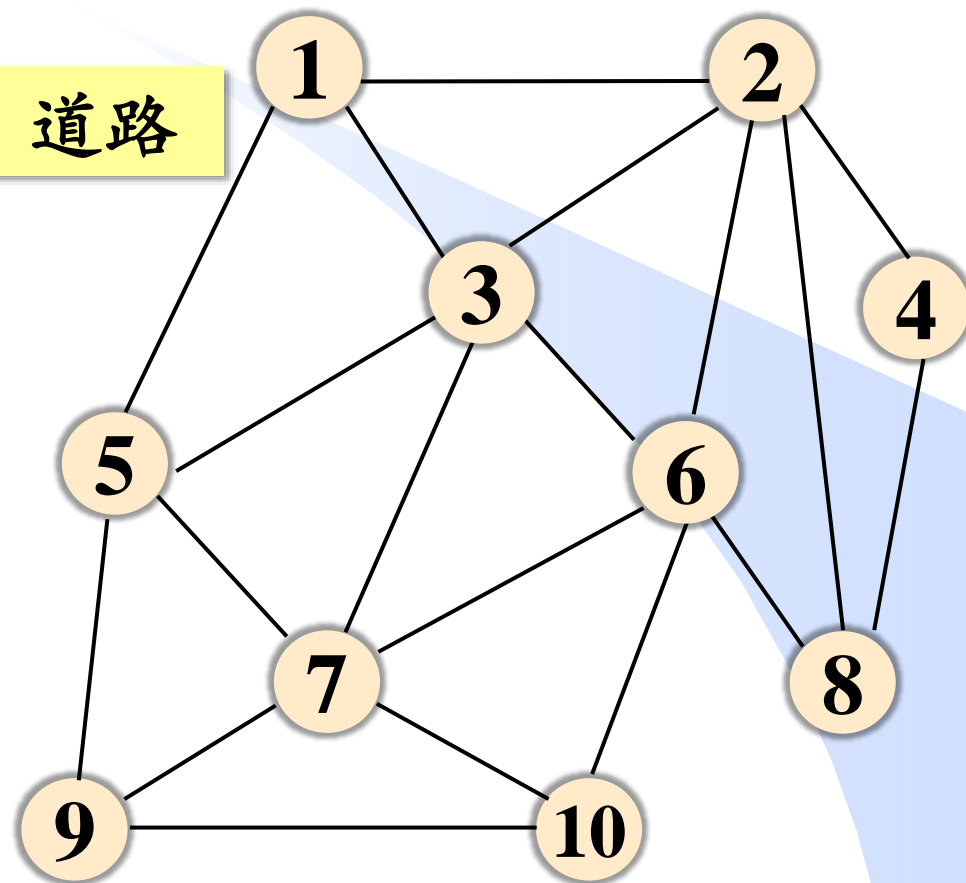
zhuyungang@jlu.edu.cn

问题引入

在 n 个地点间修缮造价最低的连通道路。



边：道路

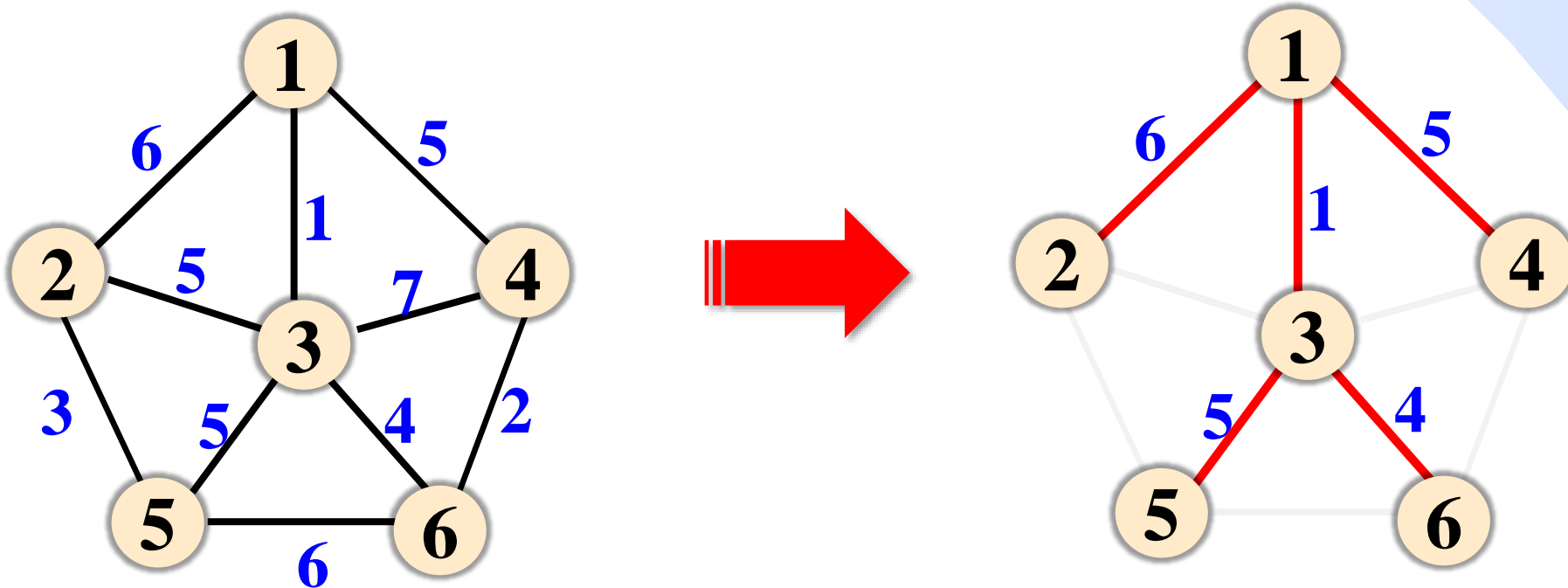


边的权值：修缮道路的费用

支撑树 (Spanning Tree)

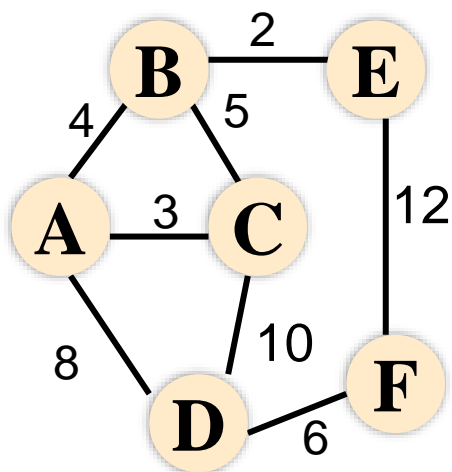
- 给定包含 n 个顶点的无向带权连通图 G ，由 G 中 n 个顶点和 $n-1$ 条边构成的连通子图，称为 G 的一棵支撑树，亦称生成树。
- 支撑树包含图 G 的全部顶点，且包含使子图连通所需的最少的边（ $n-1$ 条边）。

课下思考：支撑树中有环么？

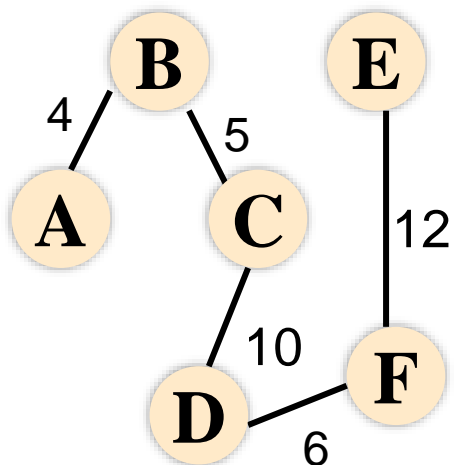


最小支撑树 (Minimum Spanning Tree, MST)

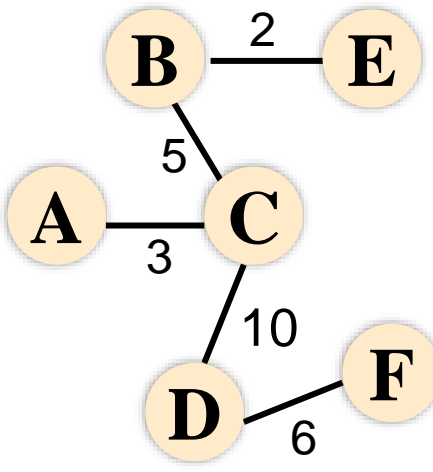
➤ 边权之和最小的支撑树称为G的最小支撑树。



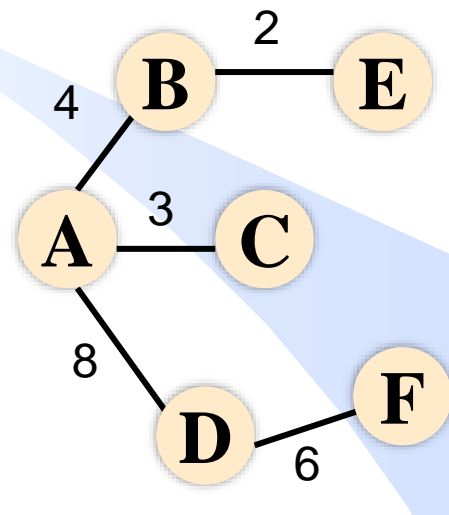
无向带权
连通图G



G的支撑树
边权和 37



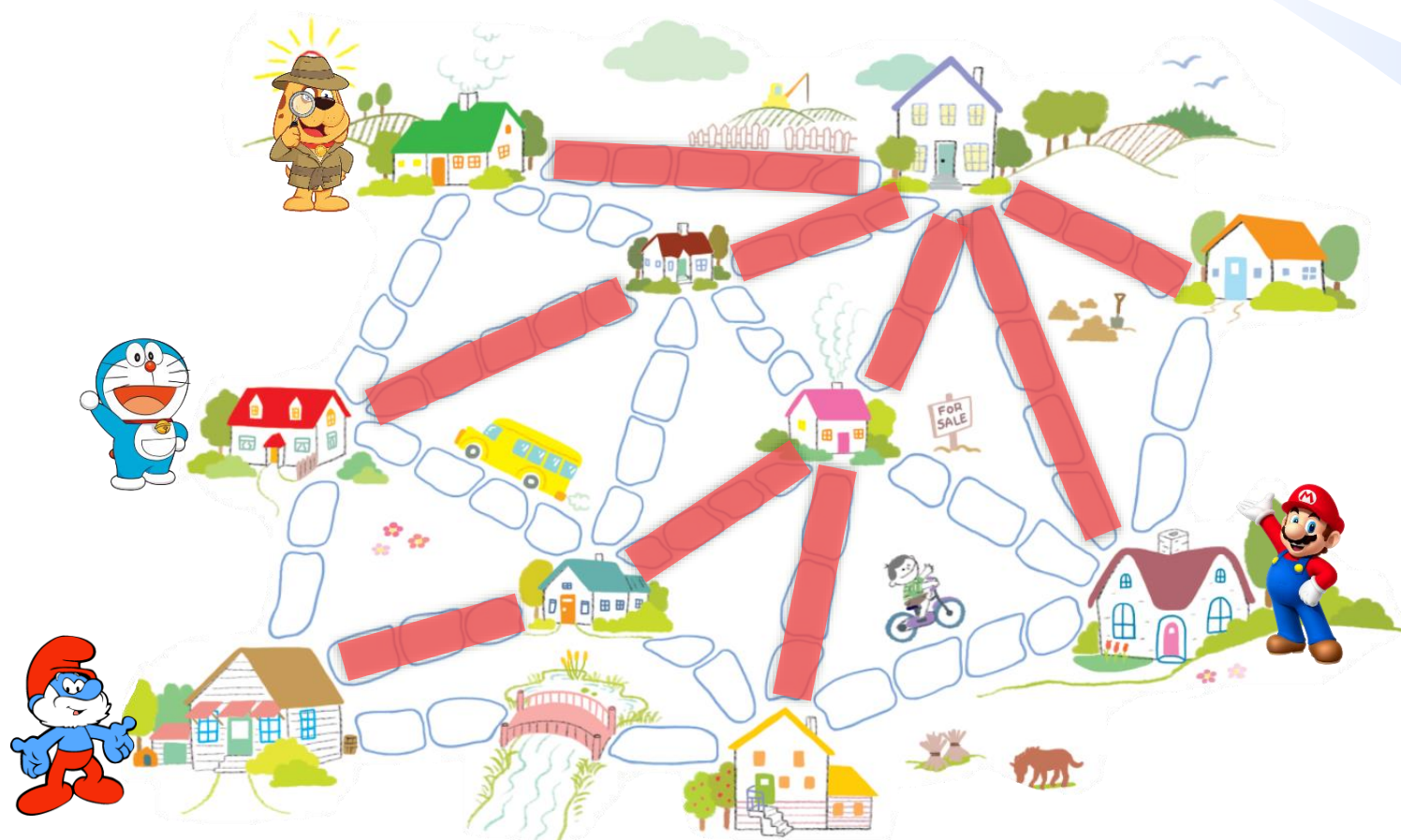
G的支撑树
边权和 26




G的最小支撑树
边权和23

最小支撑树的典型应用

网络设计：如交通网络、通信网络等。



应用广泛



检索范围: 学术期刊 (篇名: 最小生成树)

☐ 全选 已选: 0 清除 批量

☐ 1 基于最小生成树的高分辨率;

☐ 2 基于最小生成树原理的矿井;

☐ 3 基于LINGO的最小支撑树问

☐ 4 一种求解双目标最小生成树;

☐ 5 最小支撑树博弈:重新审视Bi

☐ 6 基于度约束最小生成树的域

☐ 7 基于最小生成树的船舶运输;

☐ 8 基于局部差异最小生成树脑

☐ 9 面向装配序列规划的最小生

☐ 10 一种基于最小生成树的无人

☐ 11 最小生成树算法应用于蛋白

<input type="checkbox"/> 38 基于互近邻相对距离的最小生成树聚类算法	程汝峰; 刘奕志; 梁永全	郑州大学学报(理学版)	2017-07-24 17:28	2019-09-15
<input type="checkbox"/> 39 基于聚类-最小生成树的沥青路面裂缝检测方法研究	张栋冰	中山大学学报(自然科学版)	2017-07-15	2019-01-15
<input type="checkbox"/> 40 一种基于统计学习理论的最小生成树图像分割准则	王平; 魏征; 崔卫红; 林志勇	武汉大学学报(信息科学版)	2017-07-05	2019-01-05 08:50
<input type="checkbox"/> 41 一种融合超像素与最小生成树的高分辨率遥感影像分割方法	董志鹏; 王密; 李德仁	测绘学报	2017-06-15	2018-12-15
<input type="checkbox"/> 42 基于广义最小生成树的多微网源荷储恢复顺序优化策略	许志荣; 杨苹; 曾智基; 何婷; 彭嘉俊	电力系统自动化	2017-04-25	2018-12-15
<input type="checkbox"/> 43 基于自然邻居和最小生成树的原型选择算法	朱庆生; 段浪军; 杨力军	计算机科学	2017-04-15	2018-11-16 13:47
<input type="checkbox"/> 44 基于GPU的Bor?vka最小生成树改进算法	吴永存; 刘成安; 印茂伟	科技通报	2017-02-28	2018-09-26
<input type="checkbox"/> 45 基于最小生成树的多视图特征点快速匹配算法	李丹; 朱玲玲; 胡迎松	华中科技大学学报(自然科学版)	2017-01-20 13:53	2018-09-15
<input type="checkbox"/> 46 基于最小生成树的渠道系统优化布局模型	许自昌	农业工程学报	2017-01-08	2018-07-15
<input type="checkbox"/> 47 基于最小生成树的含分布式电源的配电网重构算法	侯成滨; 王瑞峰	郑州大学学报(理学版)	2016-12-21 13:58	2018-07-11 17:19
<input type="checkbox"/> 48 粒子寻优和最小生成树聚类下的WSN能量优化	郑淼; 郑成增	计算机工程与应用	2016-12-16 14:00	2018-06-29 10:33
<input type="checkbox"/> 49 基于粒子群优化和最小生成树聚类的能耗均衡算法	李凯佳; 袁凌云; 俞锐刚	微电子学与计算机	2016-12-05	2018-06-16
<input type="checkbox"/> 50 基于直觉模糊集的随机最小支撑树选取	王肖霞; 杨风暴; 袁华	计算机工程	2016-10-15	2018-05-08
				2018-04-11 21:29

1

2

3

4

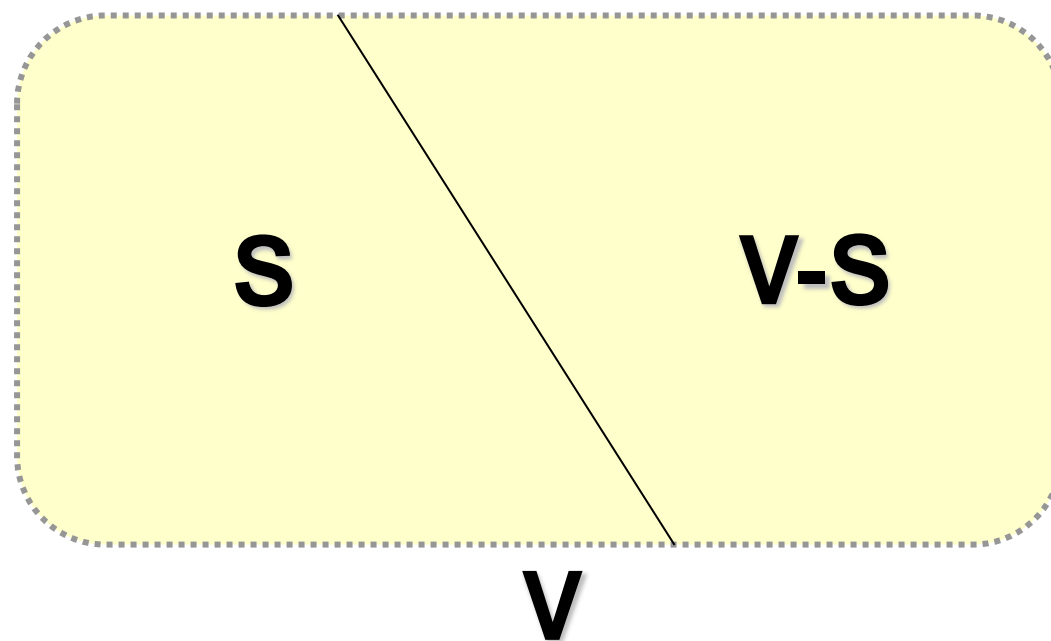
5

下一页



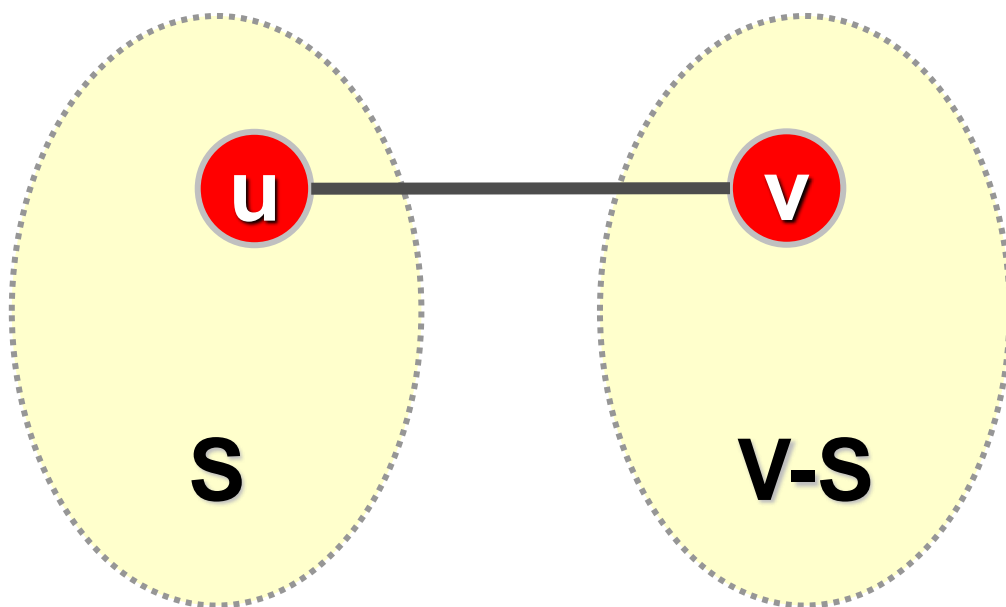
跨集合边

$G=(V,E)$ 是一个连通图， S 是顶点集 V 的一个非空子集，则图 G 的顶点被分为 S 和 $V-S$ 两个集合。



跨集合边

$G=(V,E)$ 是一个连通图， S 是 V 的一个非空子集，若边 (u,v) 满足 $u \in S, v \in V-S$ ，则称边 (u,v) 为跨集合边，简称跨边。



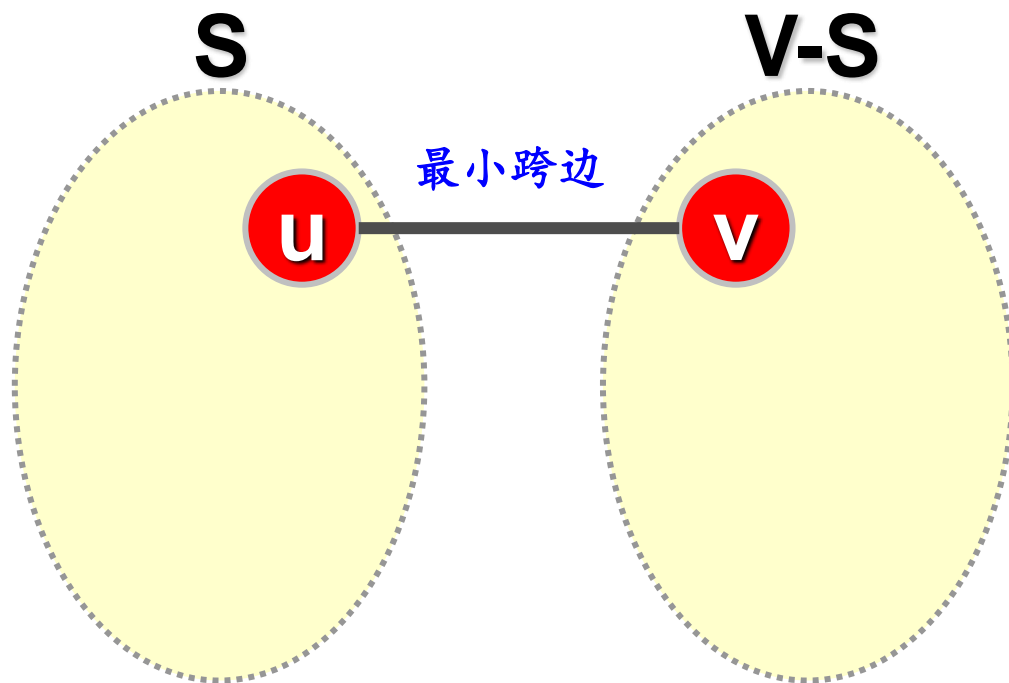
端点分别属于两个
集合的边

跨集合边

跨边

性质：最小跨边一定在某棵最小支撑树里

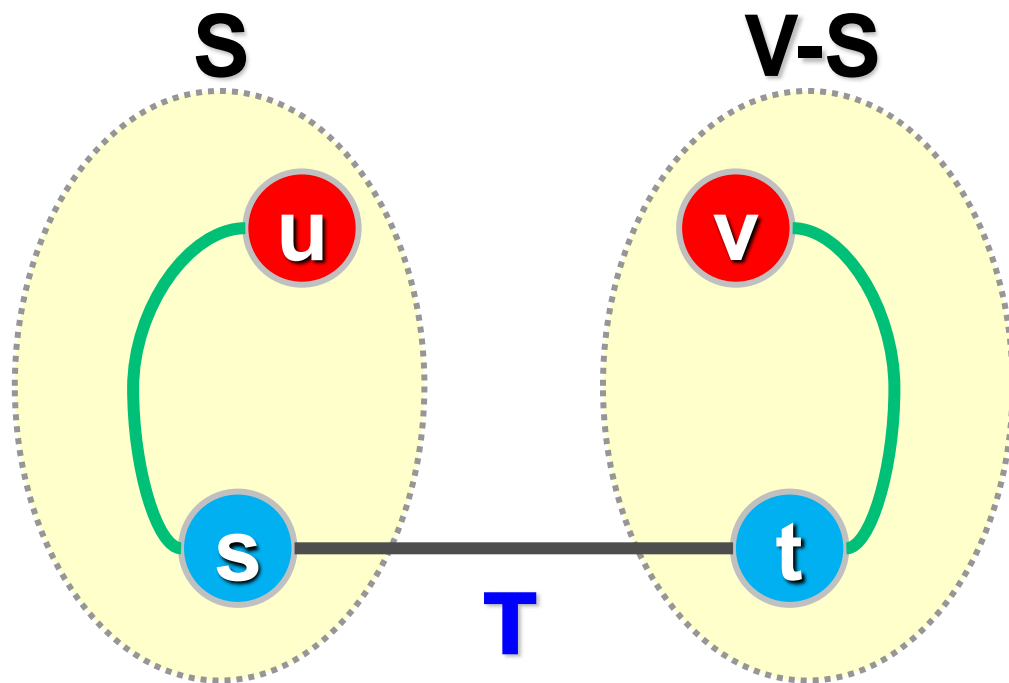
定理：连通图 G 中权值最小的跨集合边（简称**最小跨边**）**一定在 G 的一棵最小支撑树里**。即若边 (u,v) 满足 $weight(u,v)=\min\{weight(u_0,v_0) \mid u_0 \in S, v_0 \in V-S\}$ ，则必存在 G 的一棵最小支撑树 T ， T 包含边 (u,v) 。



证明：反证法，假定最小支撑树 T 不包含最小跨边 (u,v) ， T 是连通的，故 u 和 v 之间必有一条路径，该路径必包含一条跨集合边 (s,t) 。

性质：最小跨边一定在某棵最小支撑树里

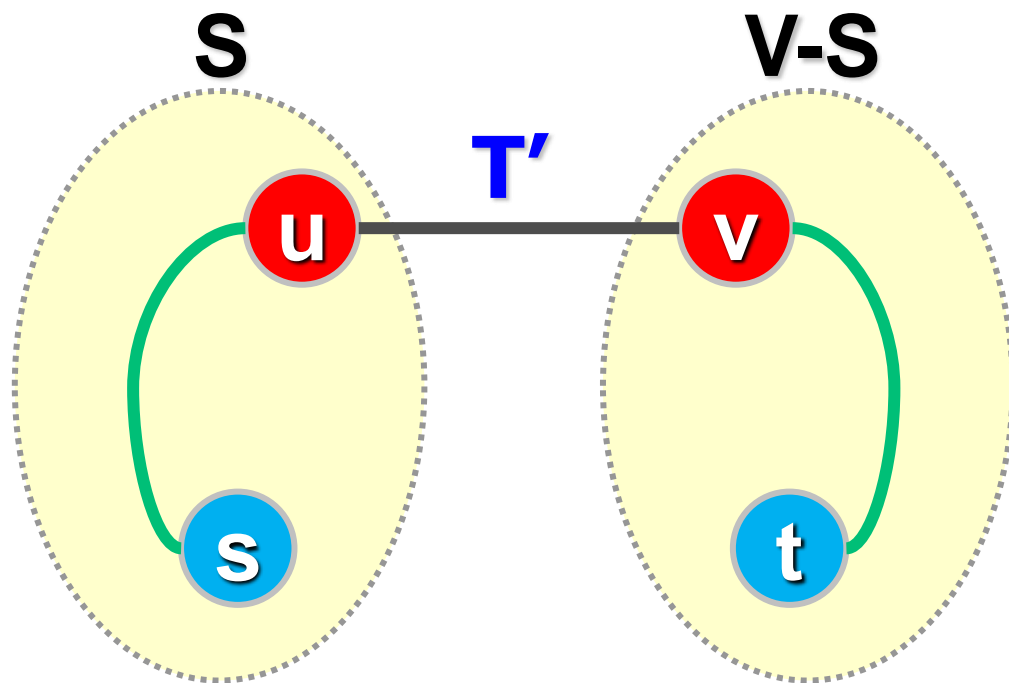
定理：连通图 G 中权值最小的跨集合边（简称**最小跨边**）**一定在 G 的一棵最小支撑树里**。即若边 (u,v) 满足 $weight(u,v)=\min\{weight(u_0,v_0) \mid u_0 \in S, v_0 \in V-S\}$ ，则必存在 G 的一棵最小支撑树 T ， T 包含边 (u,v) 。



证明：反证法，假定最小支撑树 T 不包含最小跨边 (u,v) ， T 是连通的，故 u 和 v 之间必有一条路径，该路径必包含一条跨集合边 (s,t) 。

性质：最小跨边一定在某棵最小支撑树里

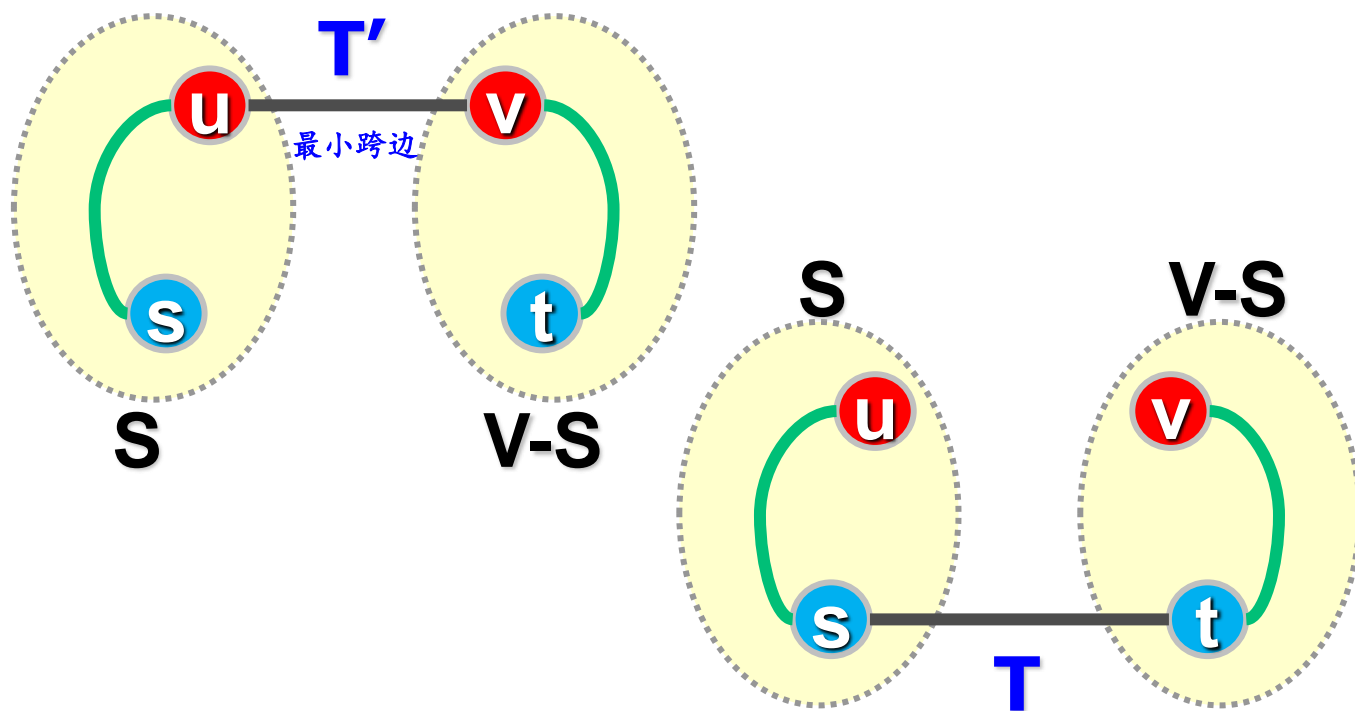
定理：连通图 G 中权值最小的跨集合边（简称**最小跨边**）一定在 G 的一棵**最小支撑树里**。即若边 (u,v) 满足 $weight(u,v) = \min\{ weight(u_0, v_0) \mid u_0 \in S, v_0 \in V-S \}$ ，则必存在 G 的一棵最小支撑树 T ， T 包含边 (u,v) 。



对 T 删去边 (s, t) ，加入边 (u, v) ，得到一棵新的支撑树 T'

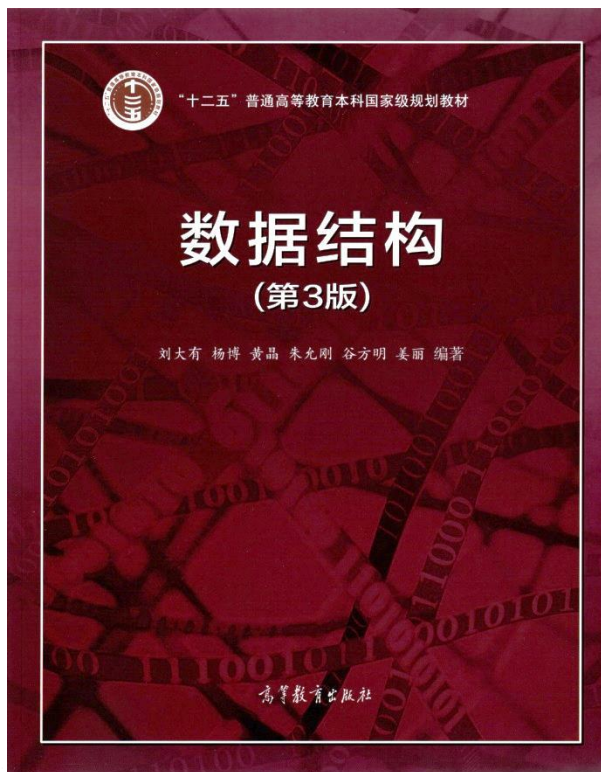
性质：最小跨边一定在某棵最小支撑树里

定理：连通图 G 中权值最小的跨集合边（简称**最小跨边**）**一定在 G 的一棵最小支撑树里**。即若边 (u,v) 满足 $weight(u,v)=\min\{weight(u_0,v_0) \mid u_0 \in S, v_0 \in V-S\}$ ，则必存在 G 的一棵最小支撑树 T ， T 包含边 (u,v) 。



由于 $w(u,v) < w(s,t)$ ，故 T' 的边权之和小于 T ，这与 T 是最小支撑树矛盾。

该定理是最小支撑树算法的依据

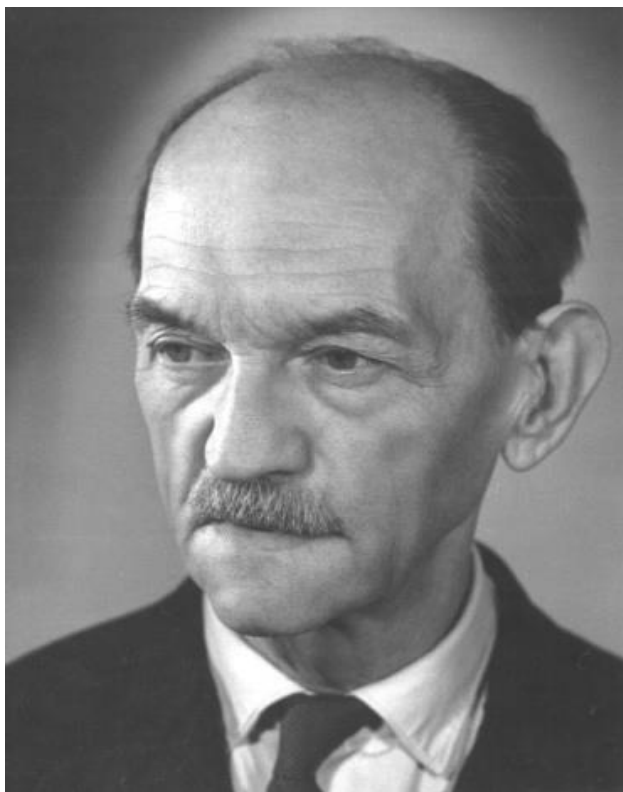


最小支撑树及图应用

- 最小支撑树的概念
- **Prim算法**
- Kruskal算法

数据之法
结构之美
算法之道

Prim算法

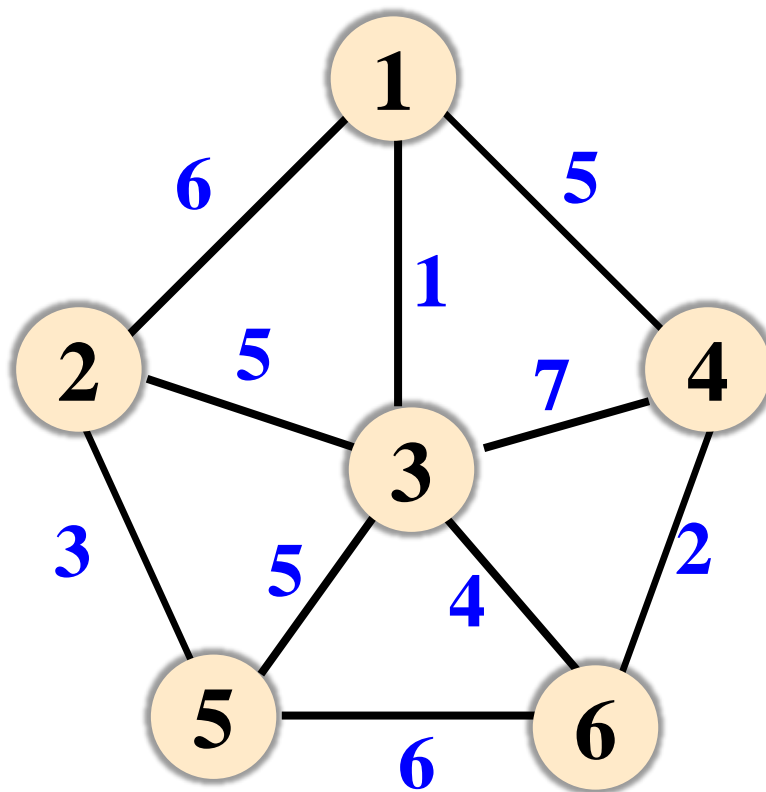


Vojtěch Jarník
布拉格大学教授
捷克科学院院士



Robert Prim
贝尔实验室数学研究中心主任

Prim算法举例



Prim算法举例

集合S:

1

集合V-S:

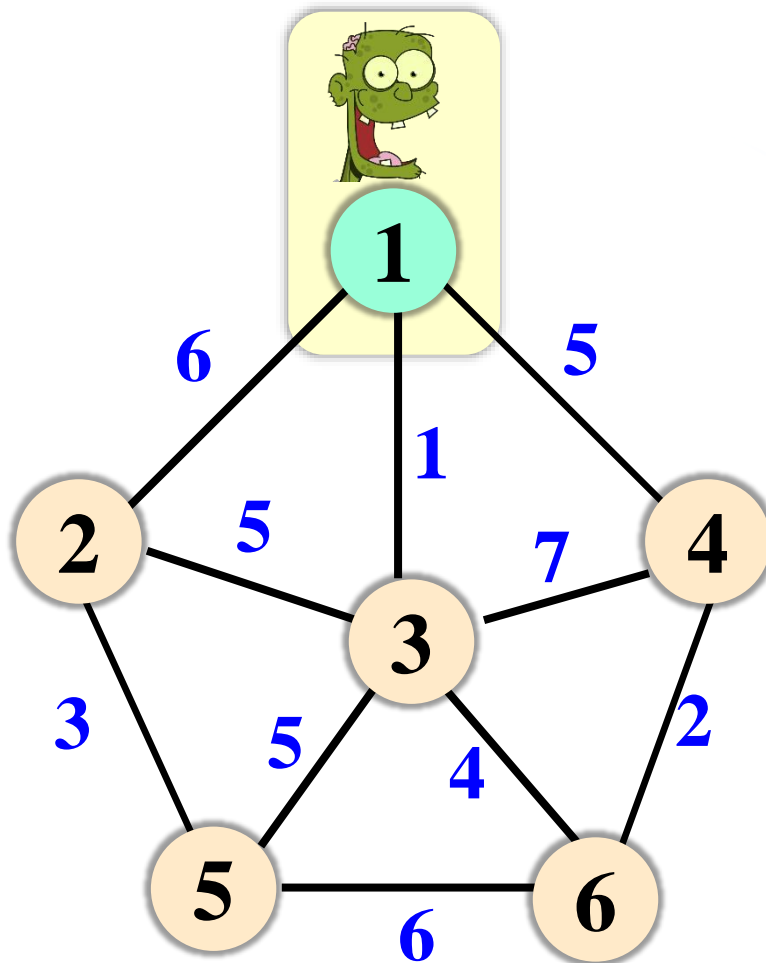
2

3

4

5

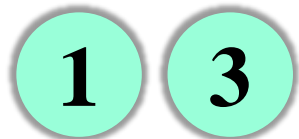
6



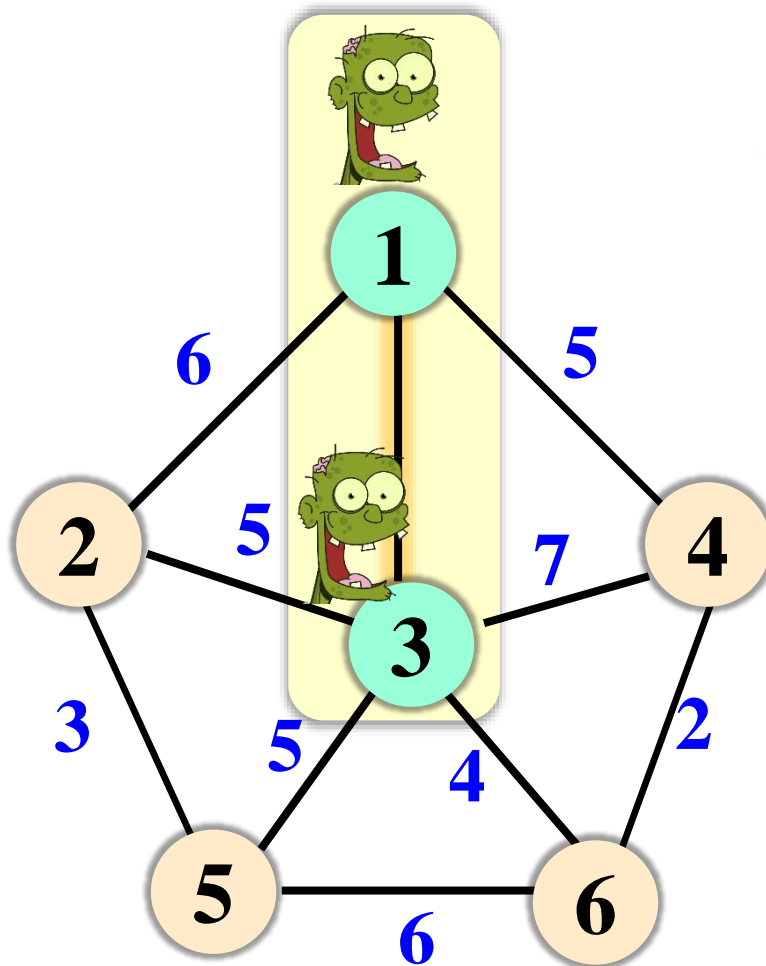
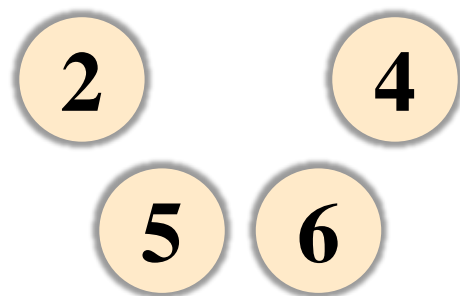
最小跨边
1—3

Prim算法举例

集合S:



集合V-S:



最小跨边

1—3

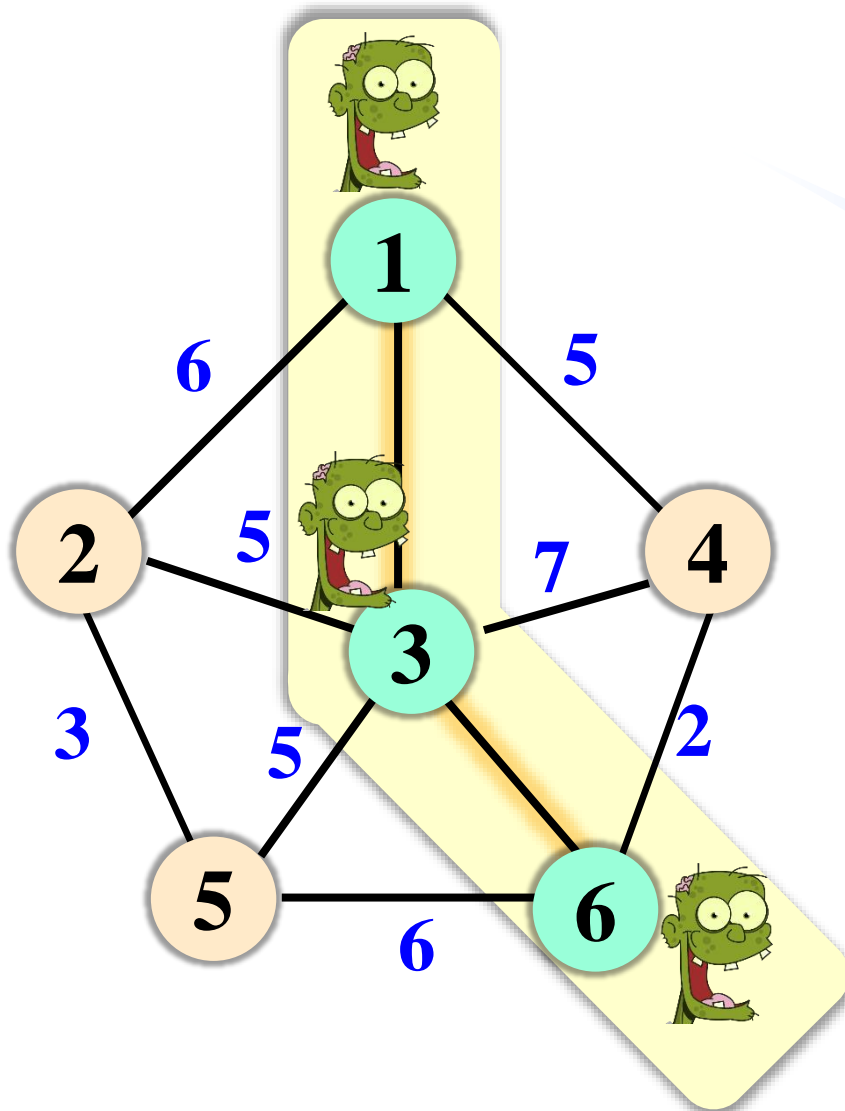
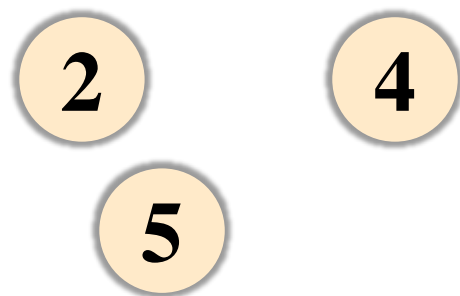
3—6

Prim算法举例

集合S:



集合V-S:



最小跨边

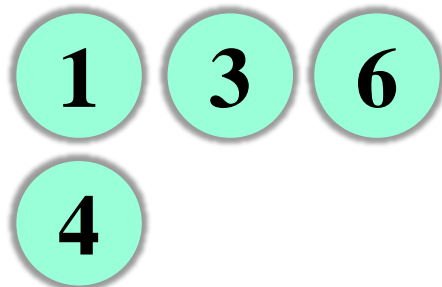
1—3

3—6

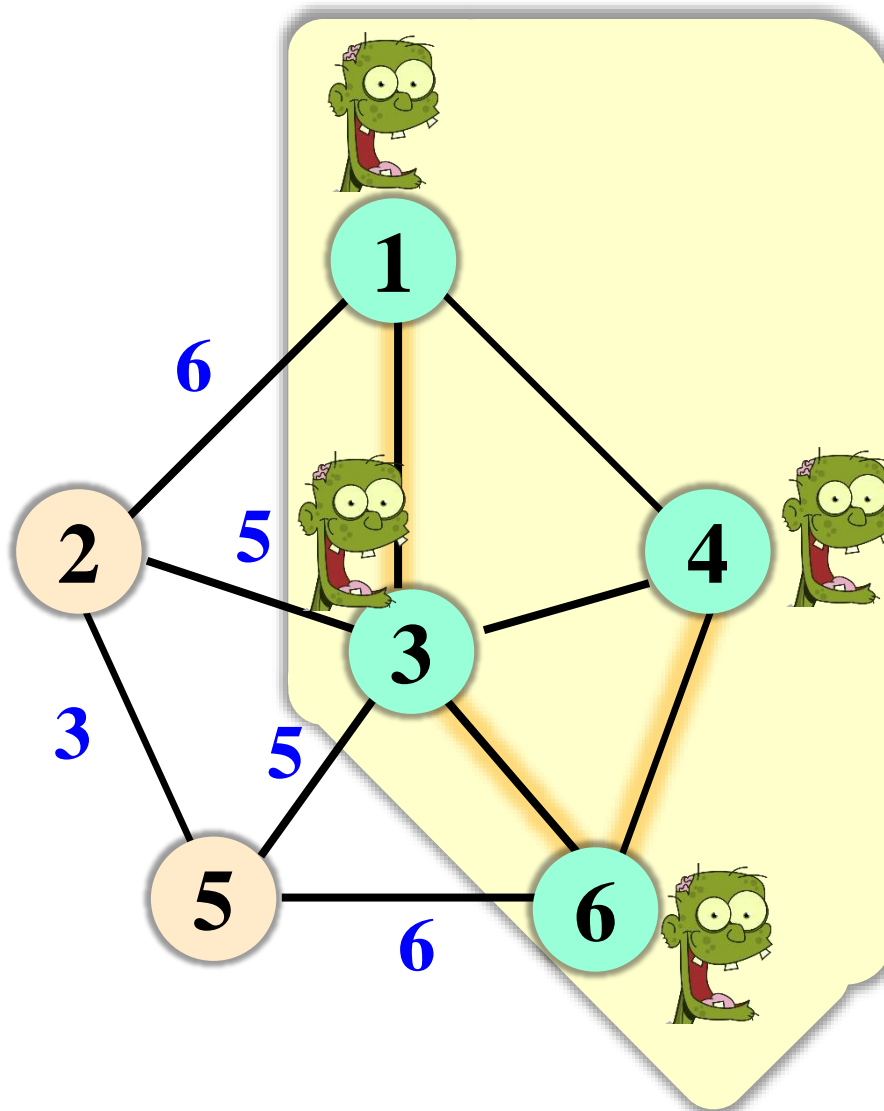
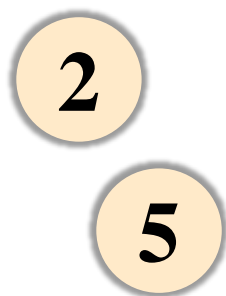
6—4

Prim算法举例

集合S:



集合V-S:



最小跨边

1—3

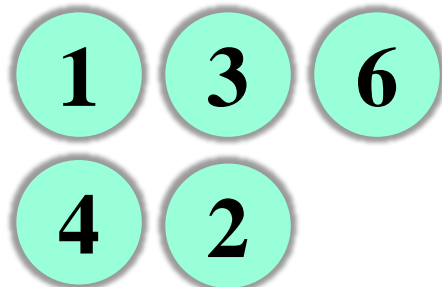
3—6

6—4

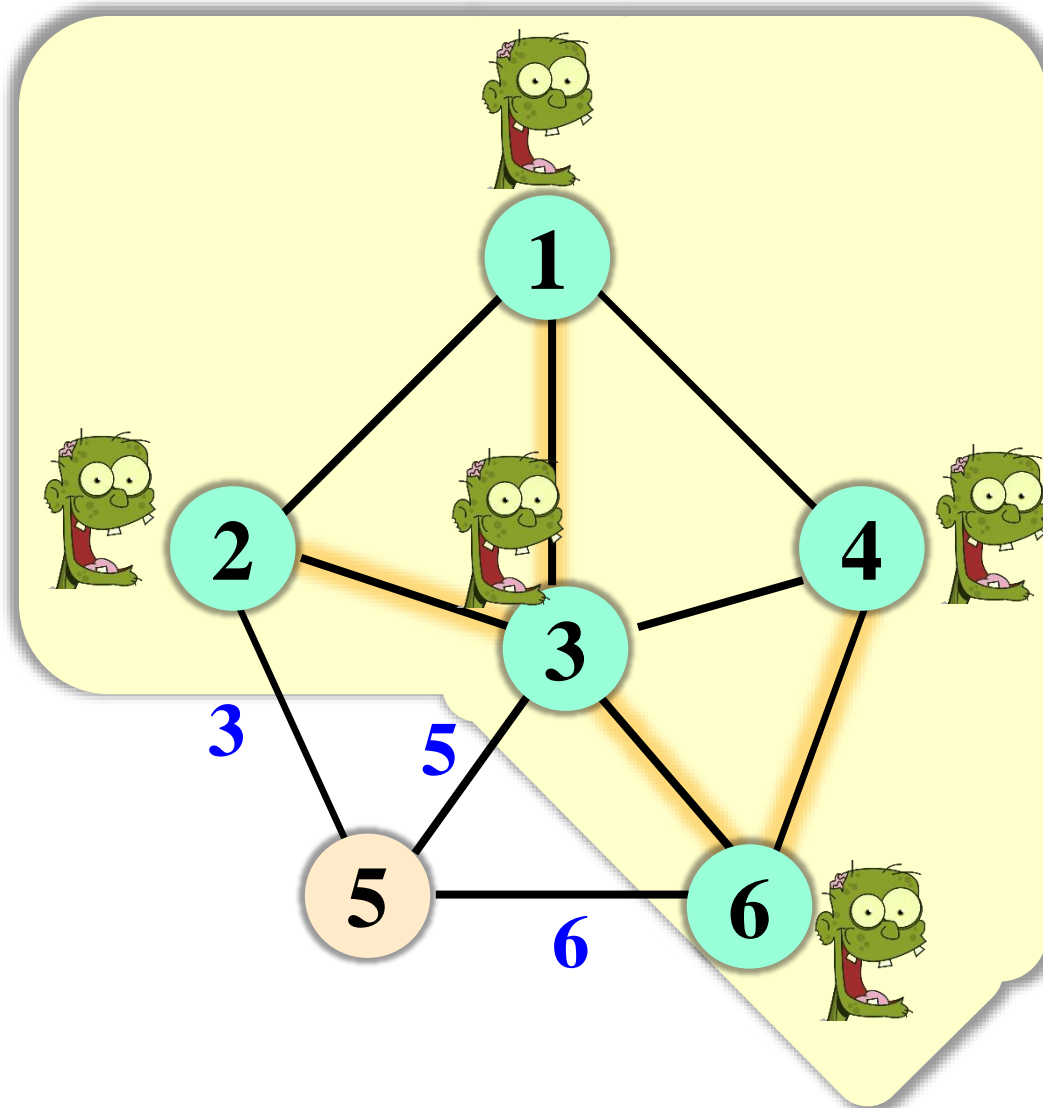
3—2

Prim算法举例

集合S:



集合V-S:



最小跨边

1—3

3—6

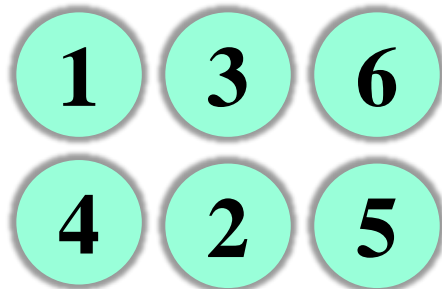
6—4

3—2

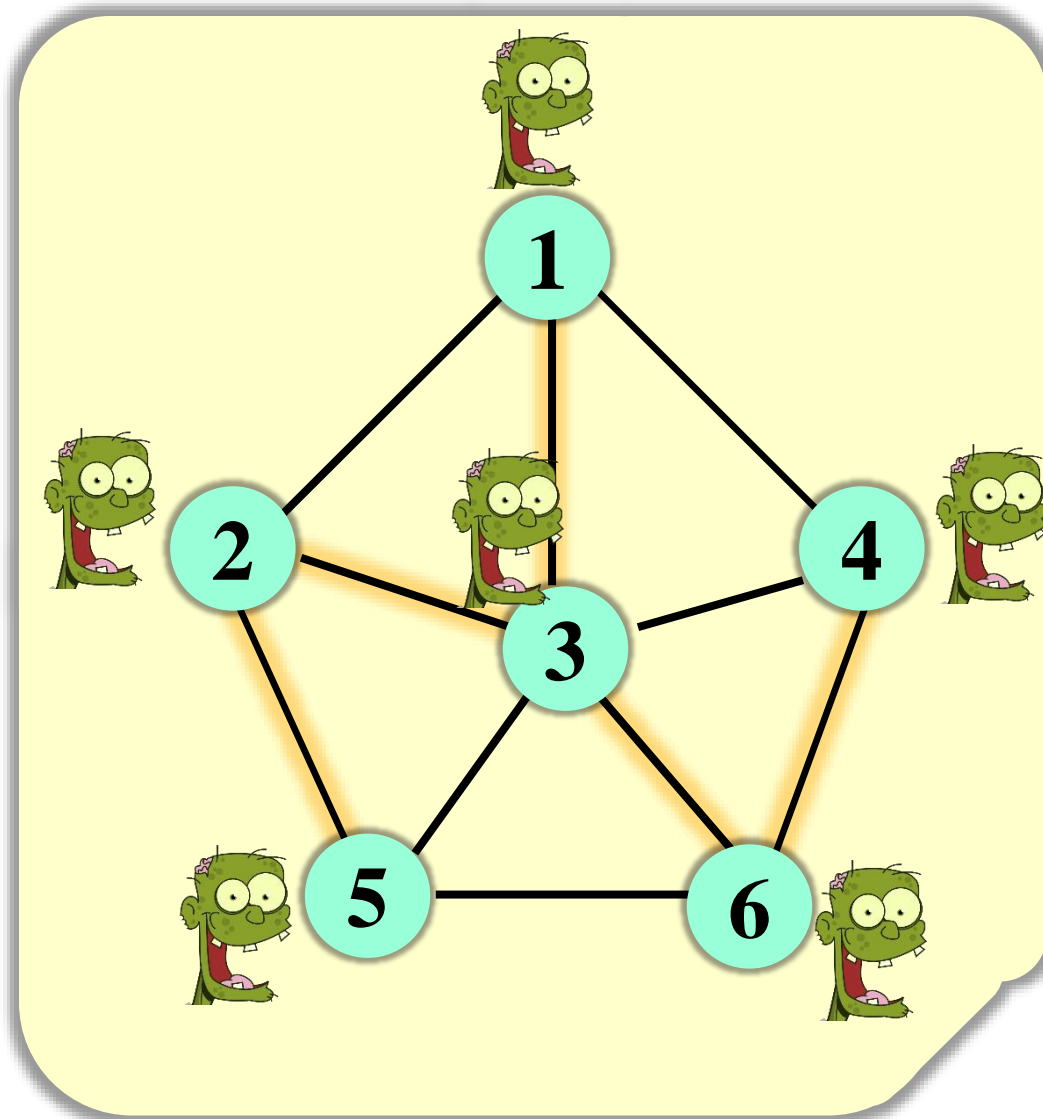
2—5

Prim算法举例

集合S:



集合V-S:



最小跨边

1—3

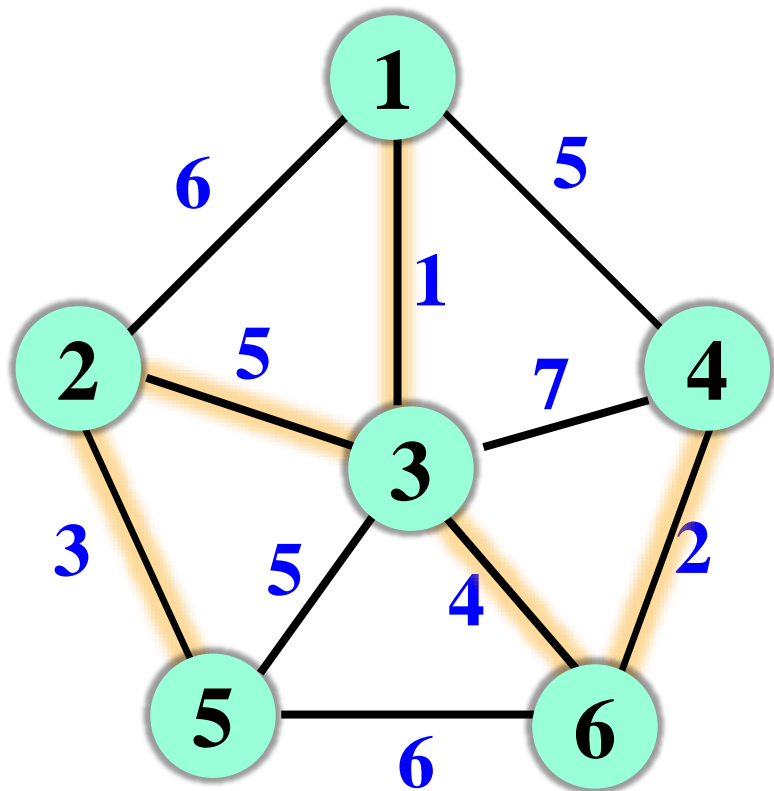
3—6

6—4

3—2

2—5

Prim算法举例



最小跨边

1—3

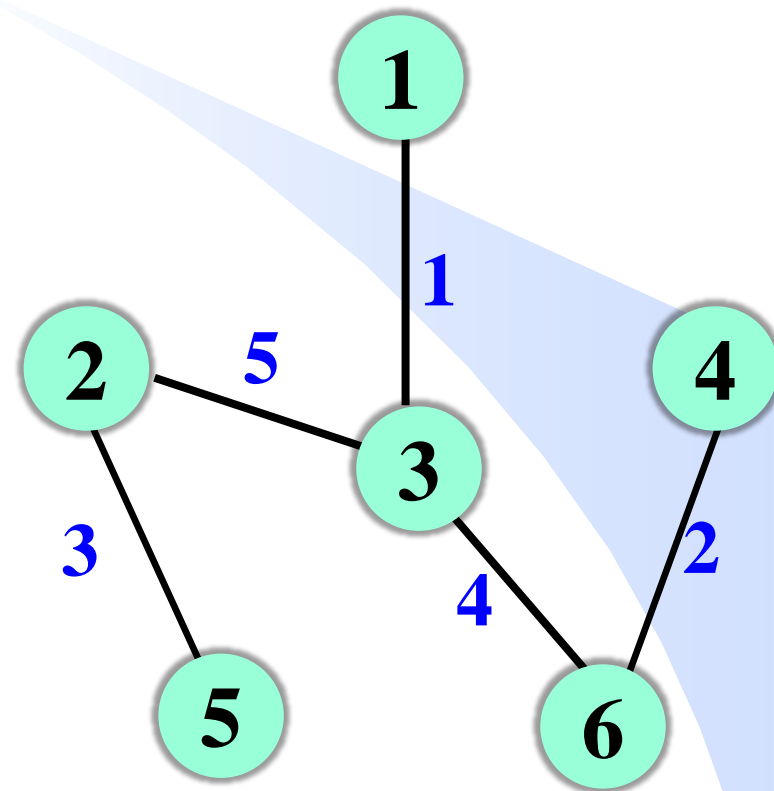
3—6

6—4

3—2

2—5

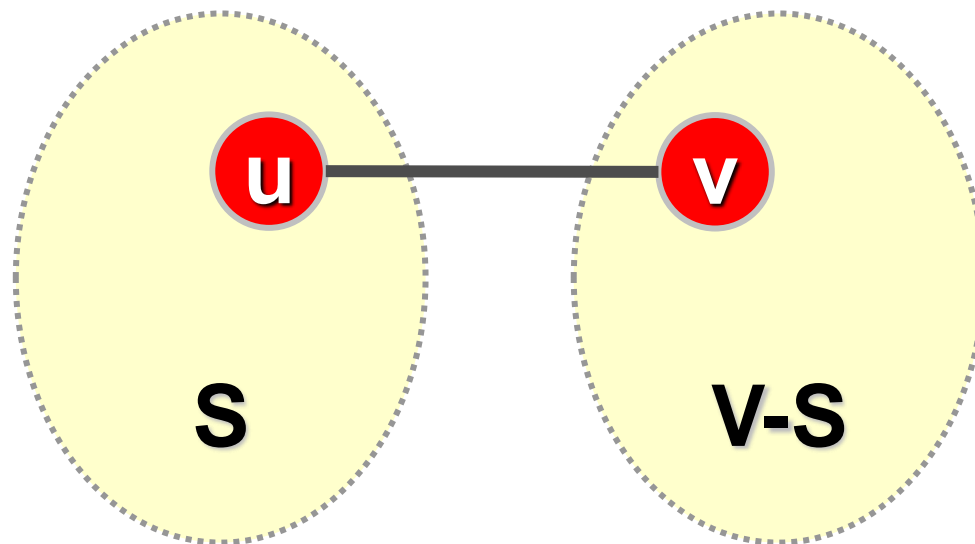
最小支撑树



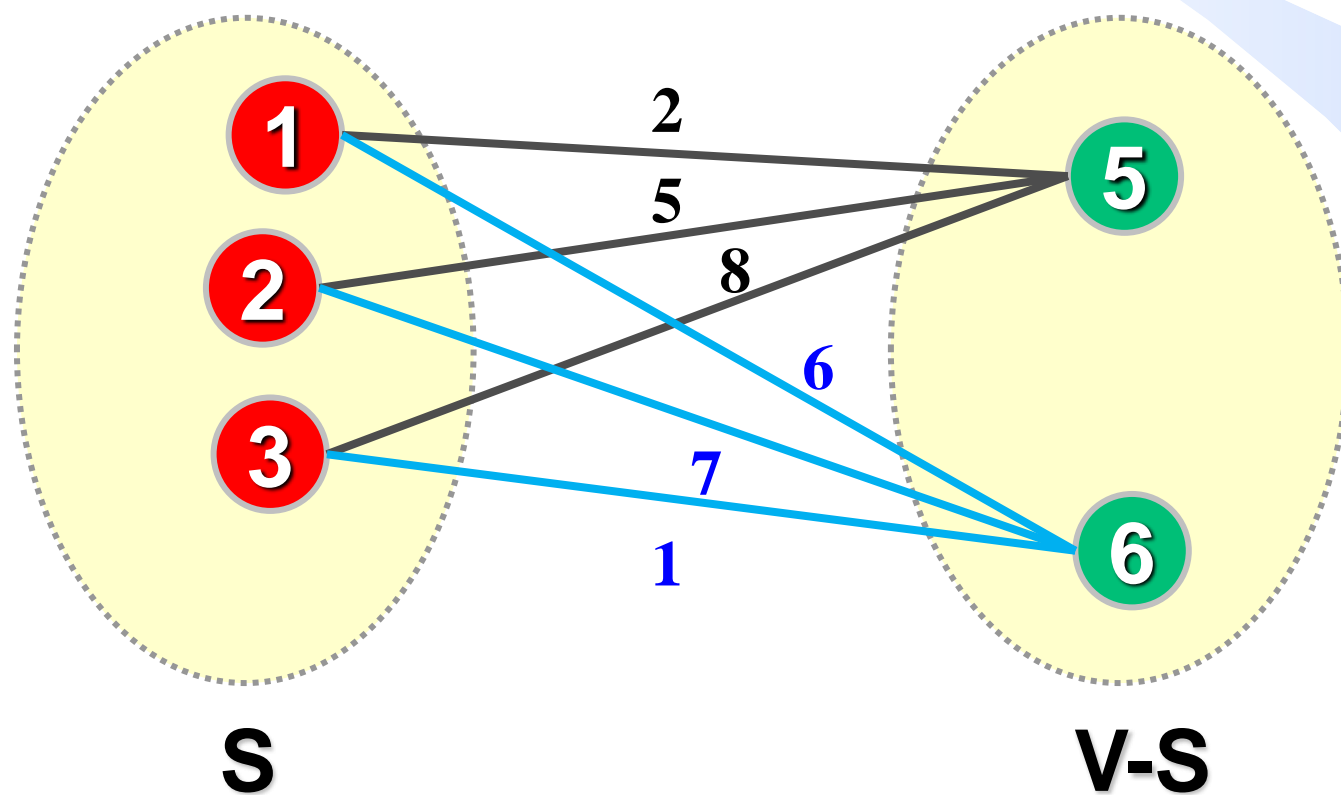
Prim算法

设 $G=(V, E)$ 为连通图， S 为最小支撑树的顶点集。

- ① 选择任一点 u 做为起点，放入集合 S ，即令 $S=\{u\}$ ($u \in V$)；
- ② 找最小跨集合边 (u, v) ，即端点分别属于集合 S 和 $V-S$ 且权值最小的边，将该边加入最小支撑树，并将点 v 放入 S ；
- ③ 执行②，直至 $S=V$ 。

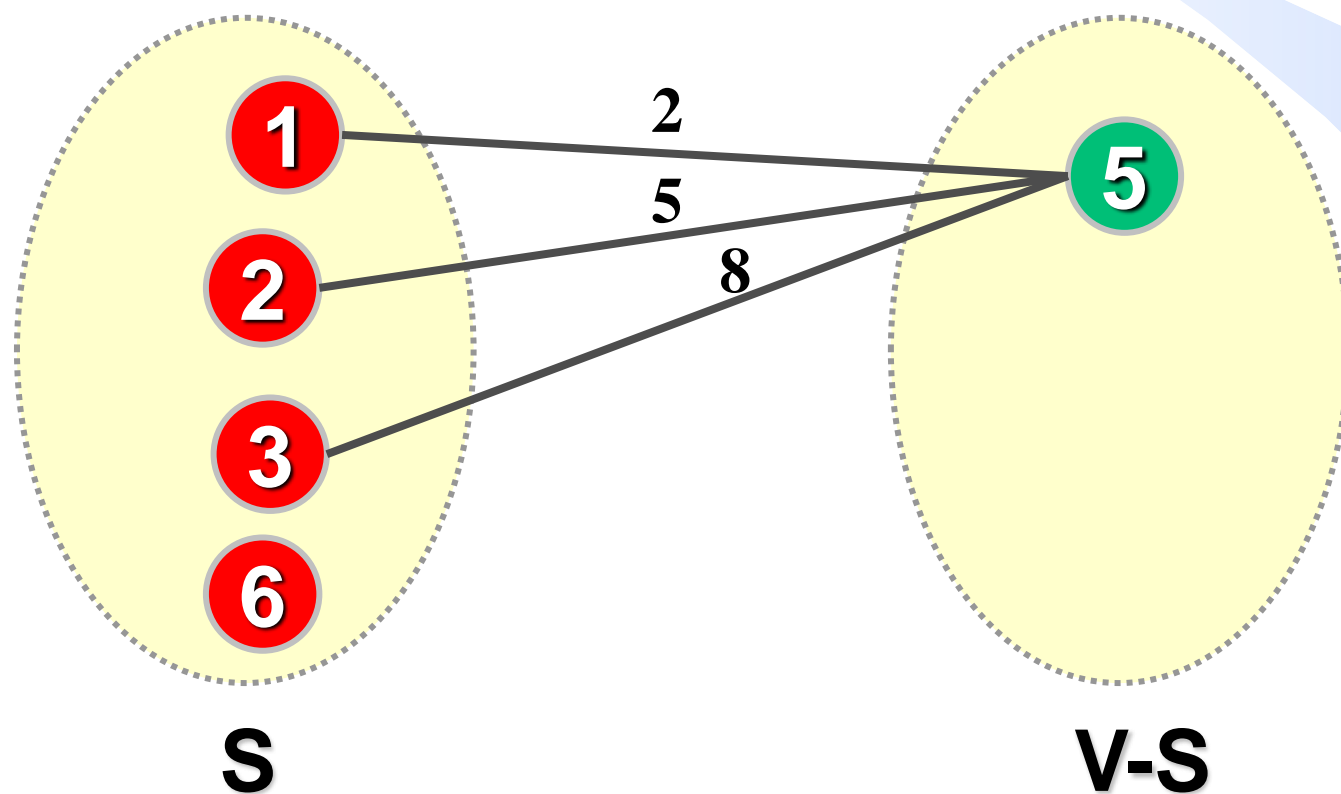


如何找最小跨集合边?



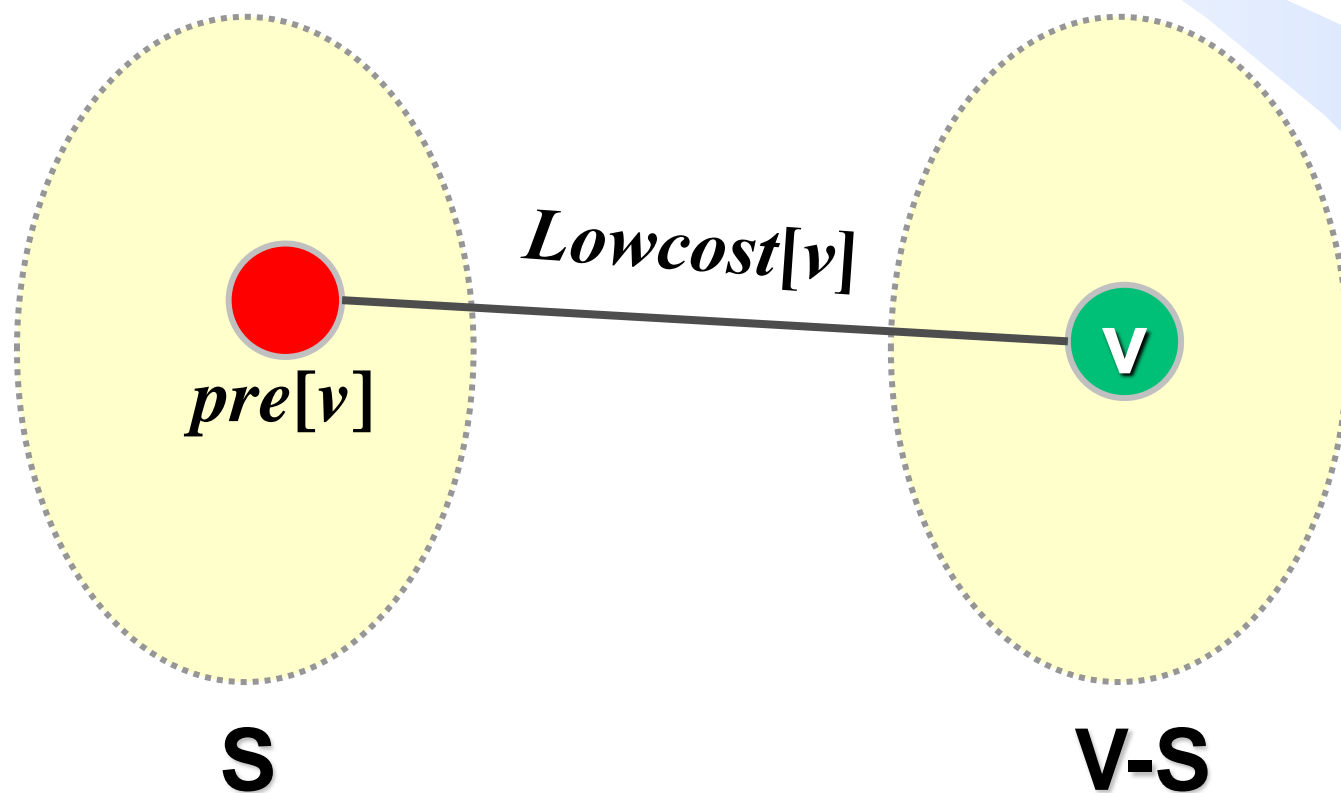
如何找最小跨集合边？

扫描所有跨集合边找权值最小的边，涉及重复运算

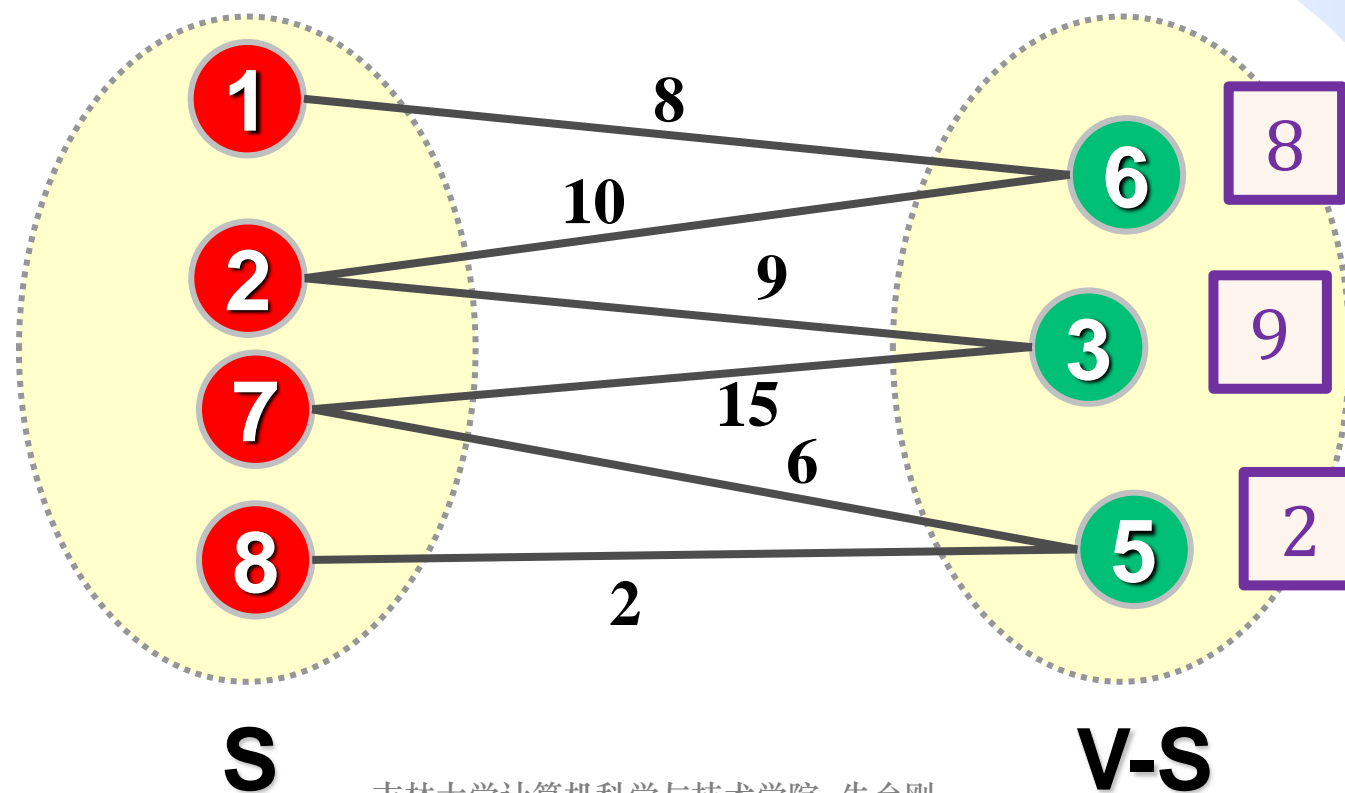


如何找最小跨集合边？

把顶点 v 到集合 S 的最小跨边保存起来，权值存入 $Lowcost[v]$ ，边在集合 S 中的端点存入 $pre[v]$ 。即 $Lowcost[v] = \min\{weight(u, v) | u \in S, v \in V-S\}$, $pre[v] = u, u \in S$ 。



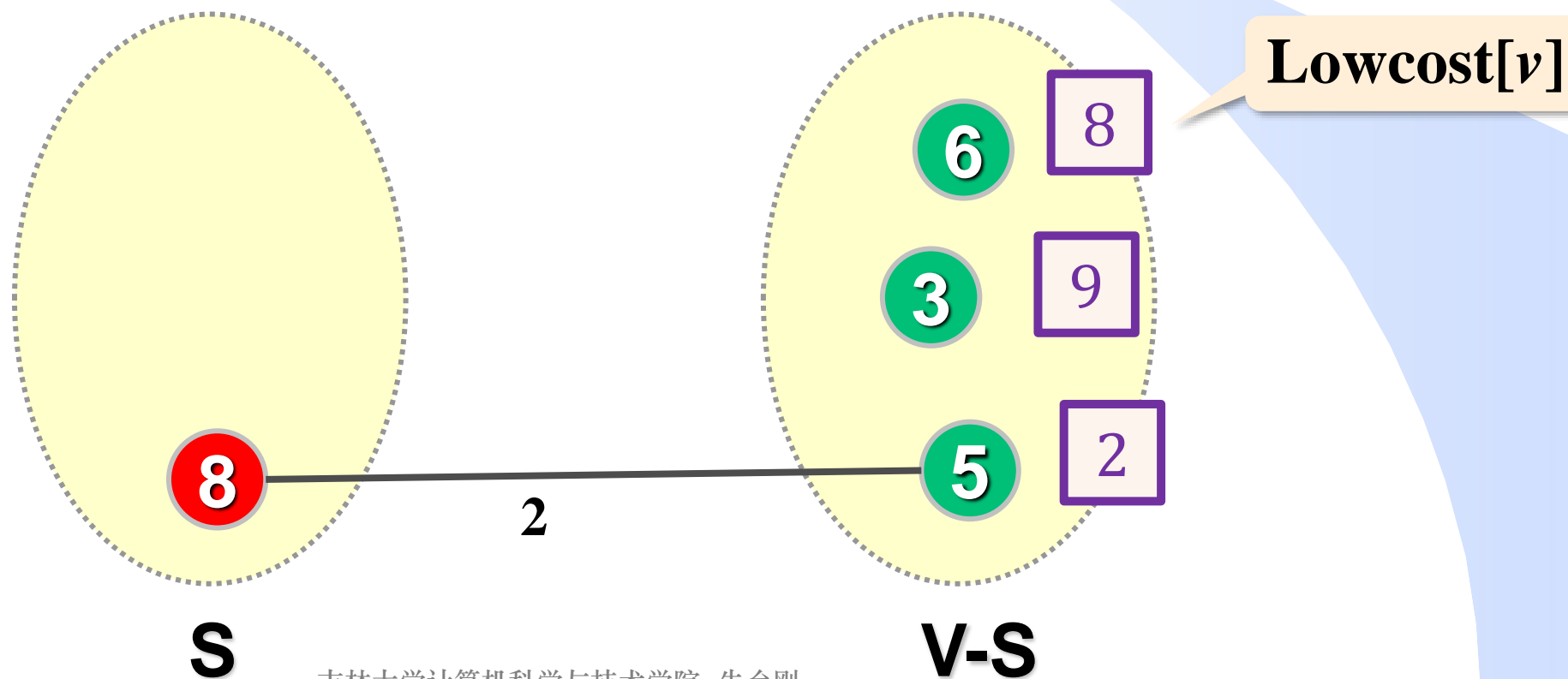
如何找最小跨集合边?



如何找最小跨集合边？

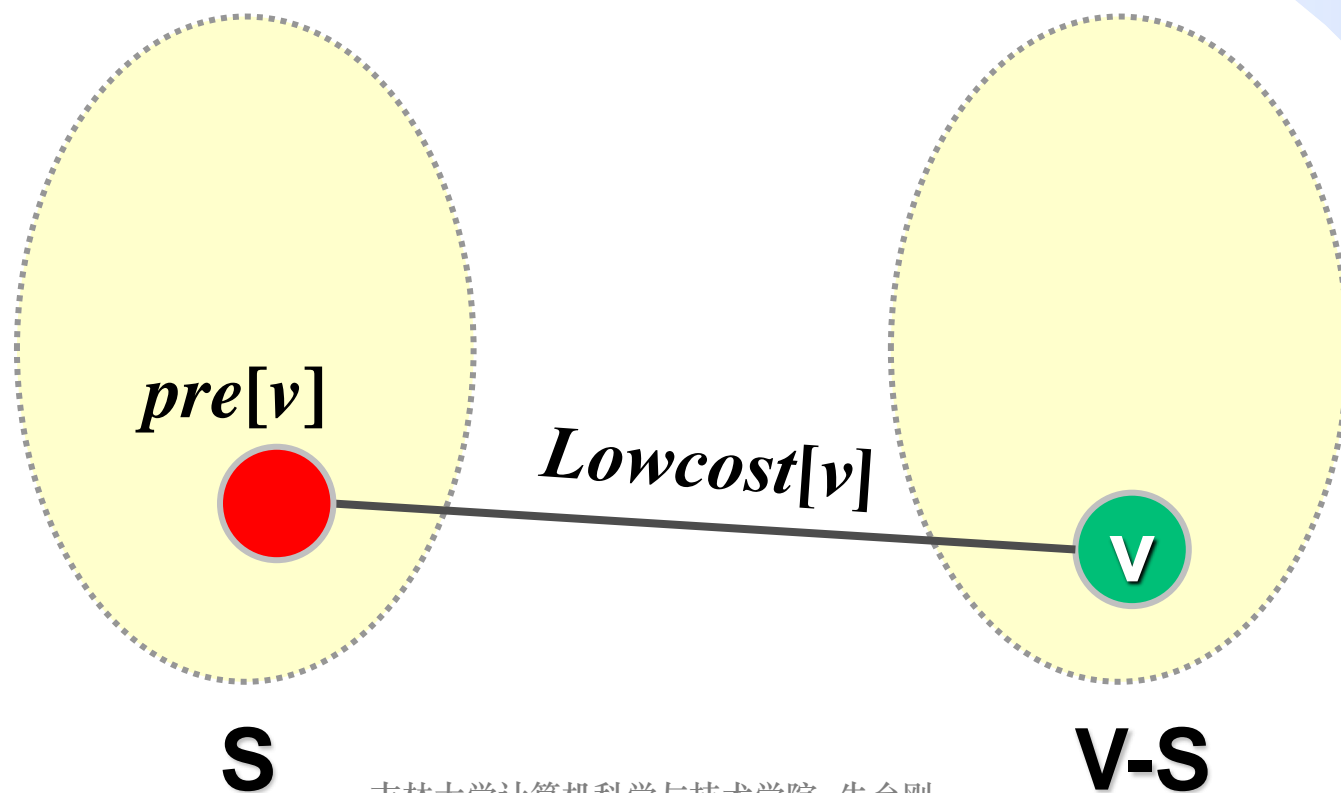
每次选择 $Lowcost$ 值最小的顶点 v ，把边 $(pre[v], v)$ 加入最小支撑树，把点 v 放入集合 S 。

整个图的最小跨边的权值： $\min_{i \in V-U} \{Lowcost[i]\}$



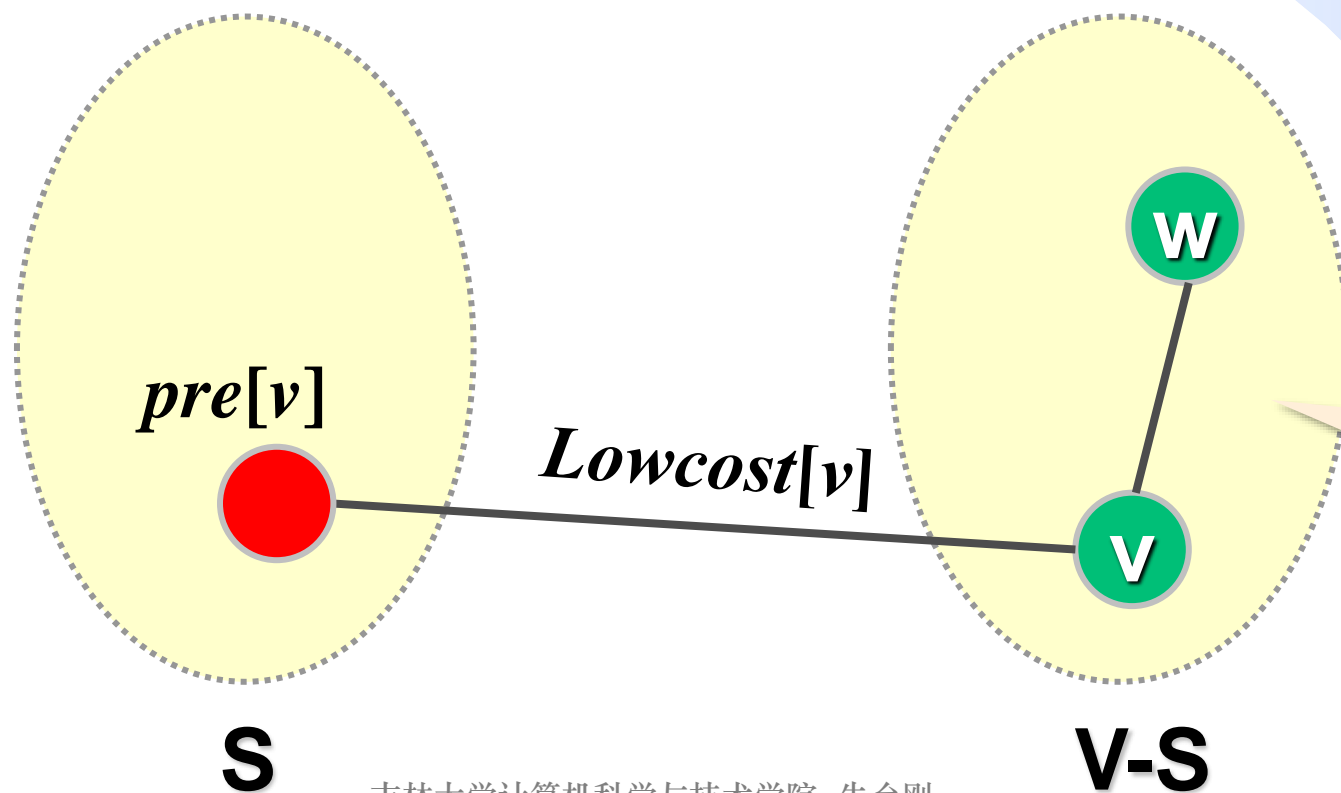
如何找最小跨集合边？

每次选择 $Lowcost$ 值最小的顶点 v ，把边 $(pre[v], v)$ 加入最小支撑树，把点 v 放入集合 S 。



如何找最小跨集合边？

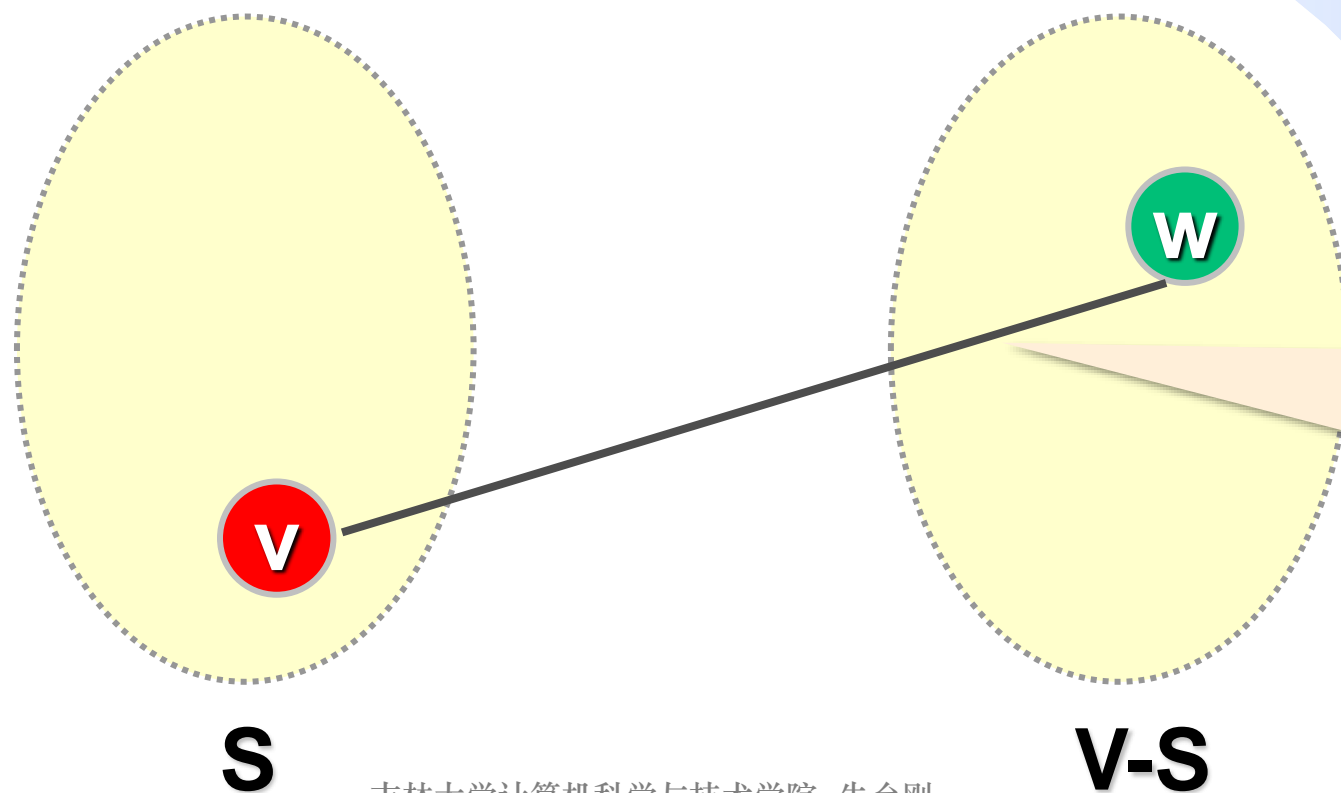
每次选择 $Lowcost$ 值最小的顶点 v ，把边 $(pre[v], v)$ 加入最小支撑树，把点 v 放入集合 S 。



如果 v 有邻接
顶点 $w...$

如何找最小跨集合边？

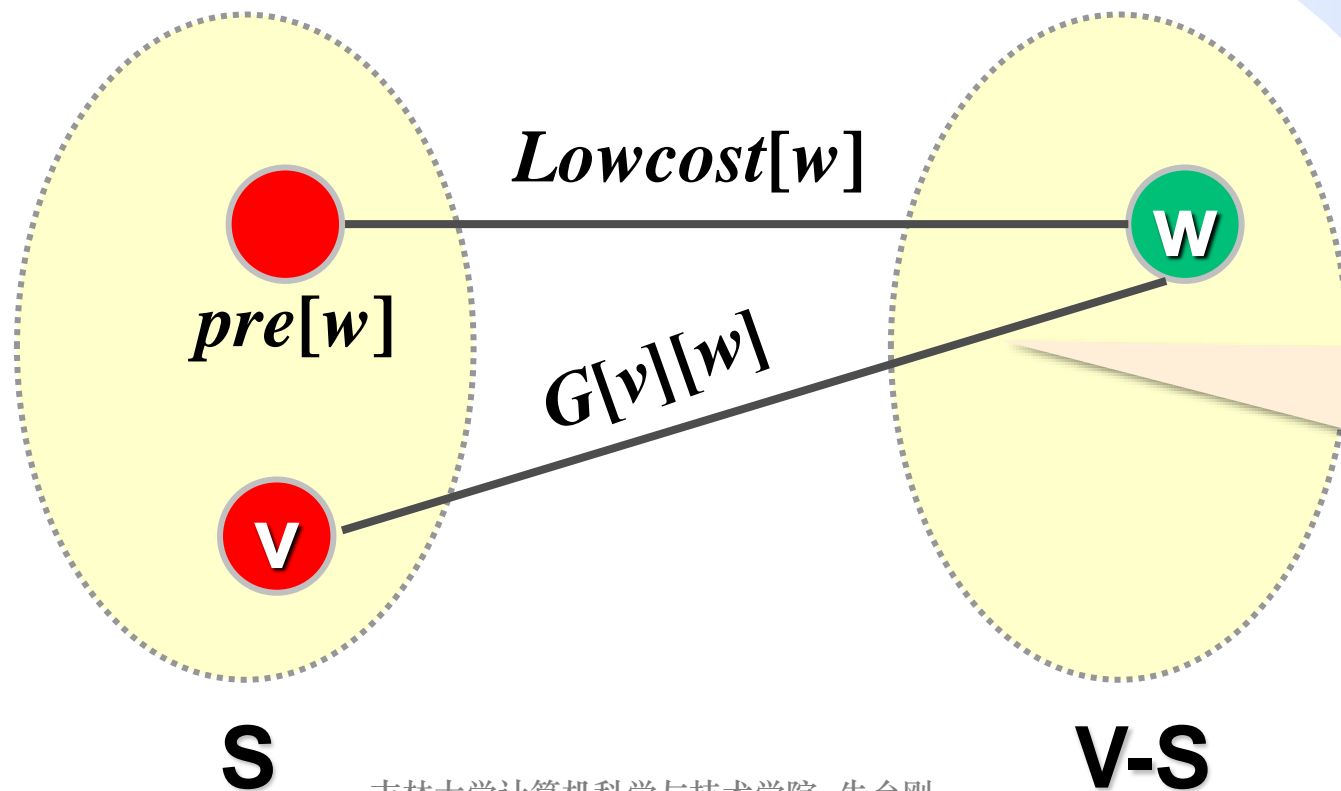
每次选择 $Lowcost$ 值最小的顶点 v ，把边 $(pre[v], v)$ 加入最小支撑树，把点 v 放入集合 S 。



将 v 加入集合 S 后将产生新的跨集合边 (v, w)

如何找最小跨集合边？

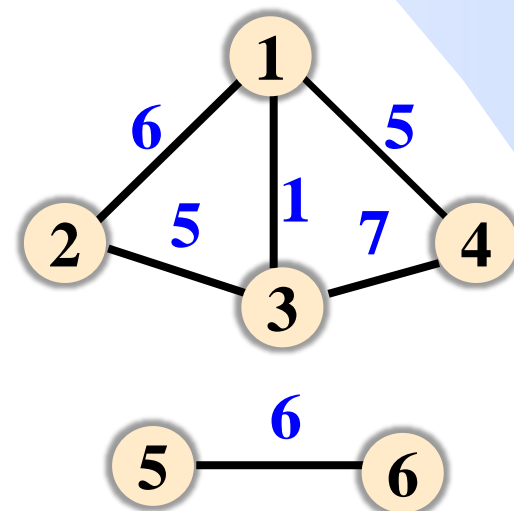
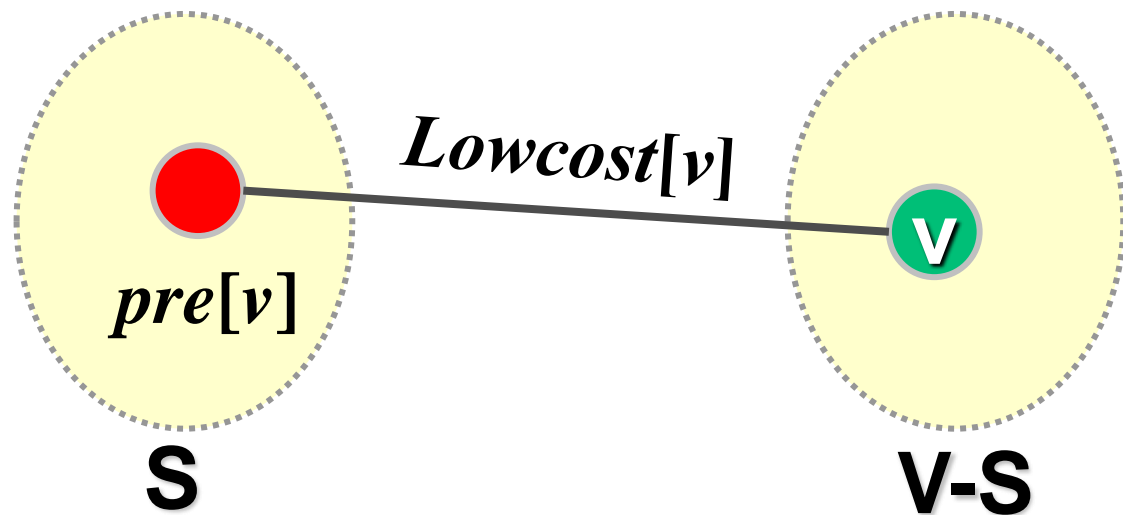
- ① 每次选择 **Lowcost** 值最小的顶点 v ，把边 $(pre[v], v)$ 加入最小支撑树，把点 v 放入集合 S 。
- ② 更新顶点 v 的邻居 w 的 **Lowcost** 值：若 $G[v][w] < Lowcost[w]$ ，则 $Lowcost[w] \leftarrow G[v][w]$ 。 $pre[w] \leftarrow v$ 。



将 v 加入集合 S 后将产生新的跨集合边 (v, w)

Prim算法实现——准备工作

- $Lowcost[v]$: 顶点 v 到集合 S 的最小跨边的权值。初始时 $Lowcost[u]=0$, $Lowcost[i]=\infty$ ($\forall i \neq u$), 其中 u 为起点;
- $pre[v]$: 顶点 v 到集合 S 的最小跨边在集合 S 中的端点, 初值-1;
- $S[v]$: 顶点 v 是否在集合 S 中, 初值0。
- 若存在 MST , 求出 MST 中的边并返回边权和, 否则 (图不连通) 求出包含起点 u 的连通子图的 MST 并返回其边权和。
- 采用邻接矩阵 G 存图。

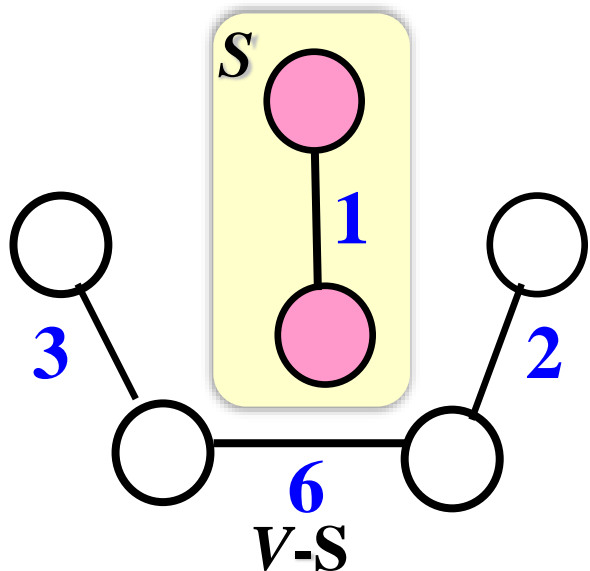


Prim算法的实现

```
int Prim(int G[N][N], int n, int u, int Lowcost[], int pre[]){
    int S[N]={0}, sum=0; //sum存MST边权之和, 作为函数返回值
    for(int i=1; i<=n; i++){pre[i]=-1; Lowcost[i] = (i==u)? 0: INF;}
    for(int i=1; i<=n; i++){ //把n个点逐个加入集合S
        int v=FindMin(S, Lowcost, n); //从不在S中的点里选Lowcost值最小的点
        if(v==-1) return sum; //不存在跨边, 图不连通
    }
    //未完待续...
```

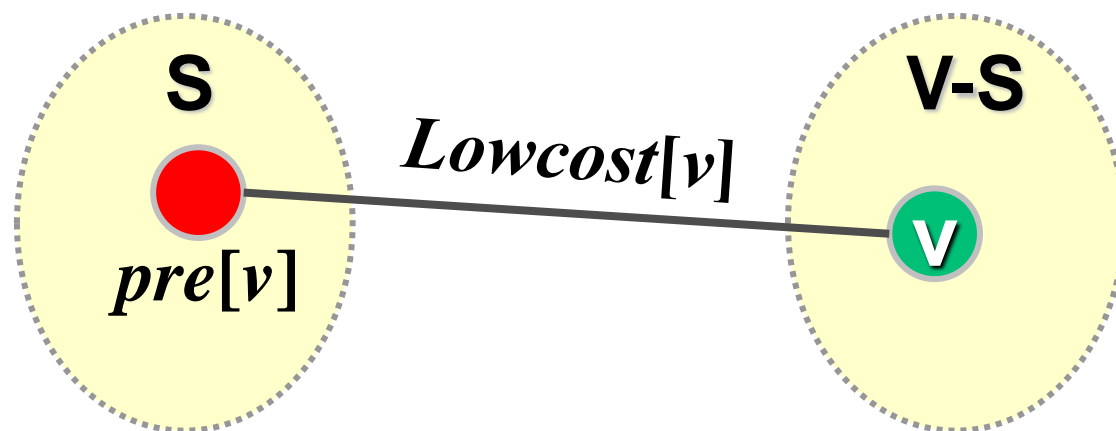
```
int FindMin(int S[], int Lowcost[], int n){
    int v=-1, min=INF;
    for(int i=1; i<=n; i++){
        if(S[i]==0 && Lowcost[i]<min){
            min=Lowcost[i]; v=i;
        }
    }
    return v;
}
```

时间复杂度
 $O(n)$



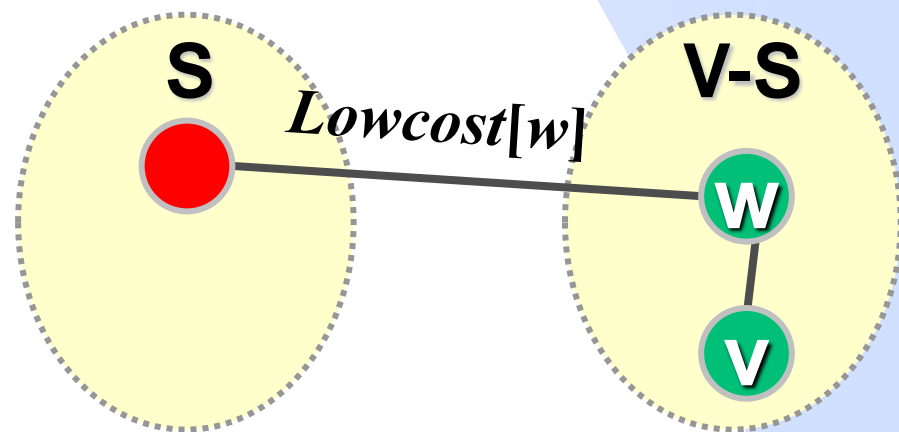
Prim算法的实现

```
int Prim(int G[N][N], int n, int u, int Lowcost[], int pre[]){
    int S[N]={0}, sum=0; //sum存MST边权之和, 作为函数返回值
    for(int i=1; i<=n; i++){pre[i]=-1; Lowcost[i] = (i==u)? 0: INF;}
    for(int i=1; i<=n; i++){ //把n个点逐个加入集合S
        int v=FindMin(S, Lowcost, n); //从不在S中的点里选Lowcost值最小的点
        if(v==-1) return sum; //不存在跨边, 图不连通
        S[v]=1; sum+=Lowcost[v]; //找到一条MST的边 (pre[v],v), v加入集合S
    }
    //未完待续...
```



Prim算法的实现

```
int Prim(int G[N][N], int n, int u, int Lowcost[], int pre[]){
    int S[N]={0}, sum=0; // sum存MST边权之和, 作为函数返回值
    for(int i=1; i<=n; i++){pre[i]=-1; Lowcost[i] = (i==u)? 0: INF;}
    for(int i=1; i<=n; i++){ //把n个点逐个加入集合S
        int v=FindMin(S, Lowcost, n); //从不在S中的点里选Lowcost值最小的点
        if(v==-1) return sum; //不存在跨边, 图不连通
        S[v]=1; sum+=Lowcost[v]; //找到一条MST的边 (pre[v],v), v加入集合S
    }
    //未完待续...
```

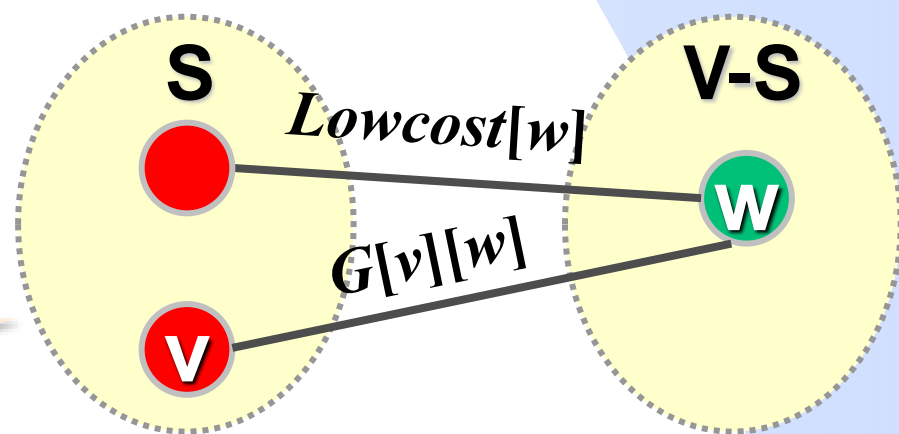


Prim算法的实现

```
int Prim(int G[N][N], int n, int u, int Lowcost[], int pre[]){
    int S[N]={0}, sum=0; // sum存MST边权之和, 作为函数返回值
    for(int i=1; i<=n; i++){pre[i]=-1; Lowcost[i] = (i==u)? 0: INF;}
    for(int i=1; i<=n; i++){ //把n个点逐个加入集合S
        int v=FindMin(S, Lowcost, n); //从不在S中的点里选Lowcost值最小的点
        if(v==-1) return sum; //不存在跨边, 图不连通
        S[v]=1; sum+=Lowcost[v]; //找到一条MST的边 (pre[v],v), v加入集合S
        for(int w=1; w<=n; w++) //更新v邻接顶点的Lowcost和pre值
            if(S[w]==0 && G[v][w]<Lowcost[w]){
                Lowcost[w]=G[v][w]; pre[w]=v;
            }
    }
    return sum;
}
```

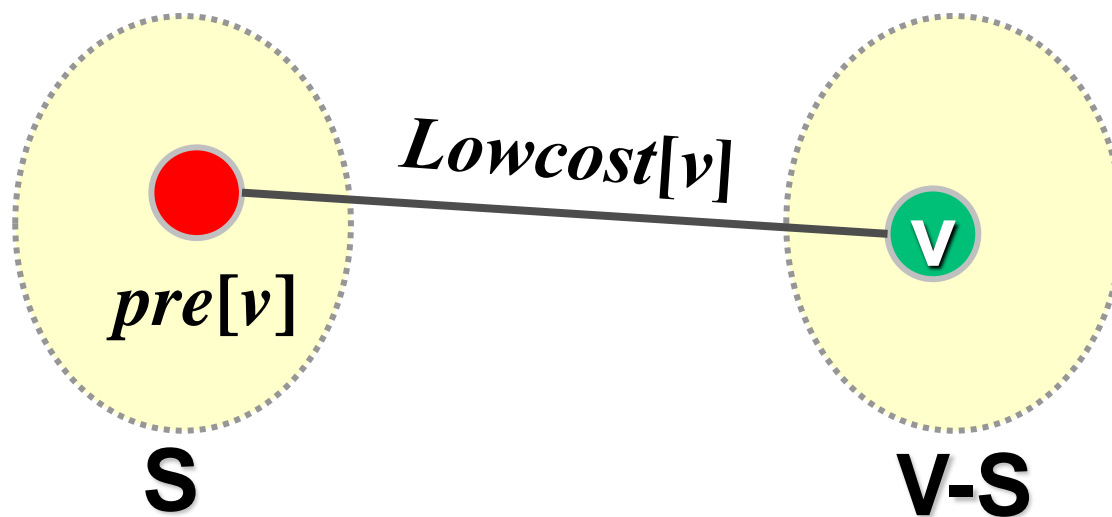
时间复杂度
 $O(n^2)$

将v加入集合S后可能
产生新跨边(v,w)



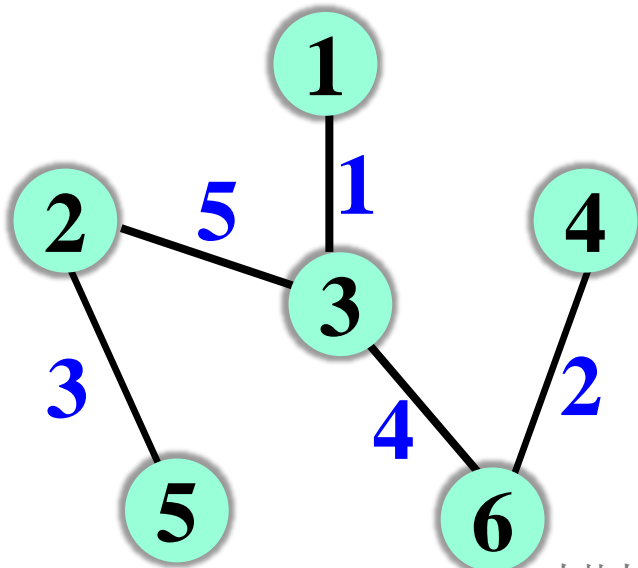
如何使用pre数组——直接输出MST的边

```
void OutputMST(int Lowcost[], int pre[], int n){ //输出MST中的边
    for(int v=1; v<=n; v++) //输出MST中的边 (pre[v],v)及权值
        if(pre[v]!=-1)
            printf("edge:%d-%d,weight:%d\n",pre[v],v,Lowcost[v]);
}
```



课下阅读——使用pre数组构建MST的邻接表

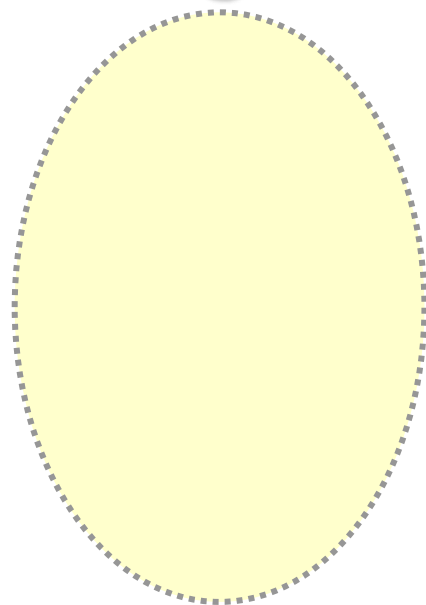
```
void BuildMST(Vertex Head[], int n, int Lowcost[], int pre[]){
    //通过Lowcost和pre数组构建MST的邻接表
    for(int v=1; v<=n; v++){
        Edge *p = Head[v].adjacent; //将v的邻接顶点pre[v]插入邻接表
        Head[v].adjacent = new Edge(pre[v], Lowcost[v], p);
        p = Head[pre[v]].adjacent; //将pre[v]的邻接顶点v插入邻接表
        Head[pre[v]].adjacent = new Edge(v, Lowcost[v], p);
    }
}
```



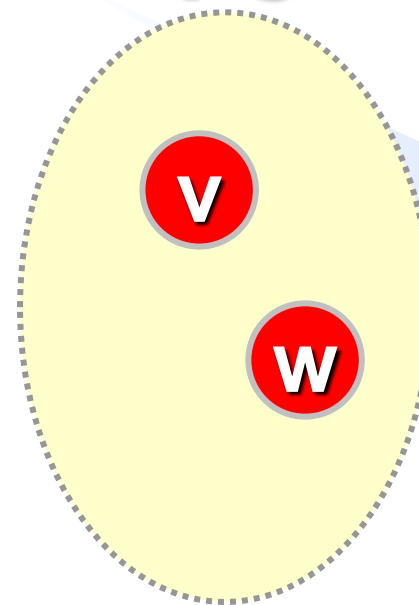
```
struct Edge{
    int VerAdj;
    int cost;
    Edge *link=NULL;
    Edge(int a, int c, Edge *p){
        VerAdj=a; cost=c; link=p;
    }
};
```

Dijkstra算法 VS Prim算法

S



V-S



Dijkstra算法

从集合 $V-S$ 中选择 $Dist$ 值最小的顶点放入集合 S ，并更新其邻居的 $Dist$ 值

Prim算法

从集合 $V-S$ 中选择 $Lowcost$ 值最小的顶点放入集合 S ，并更新其邻居的 $Lowcost$ 值

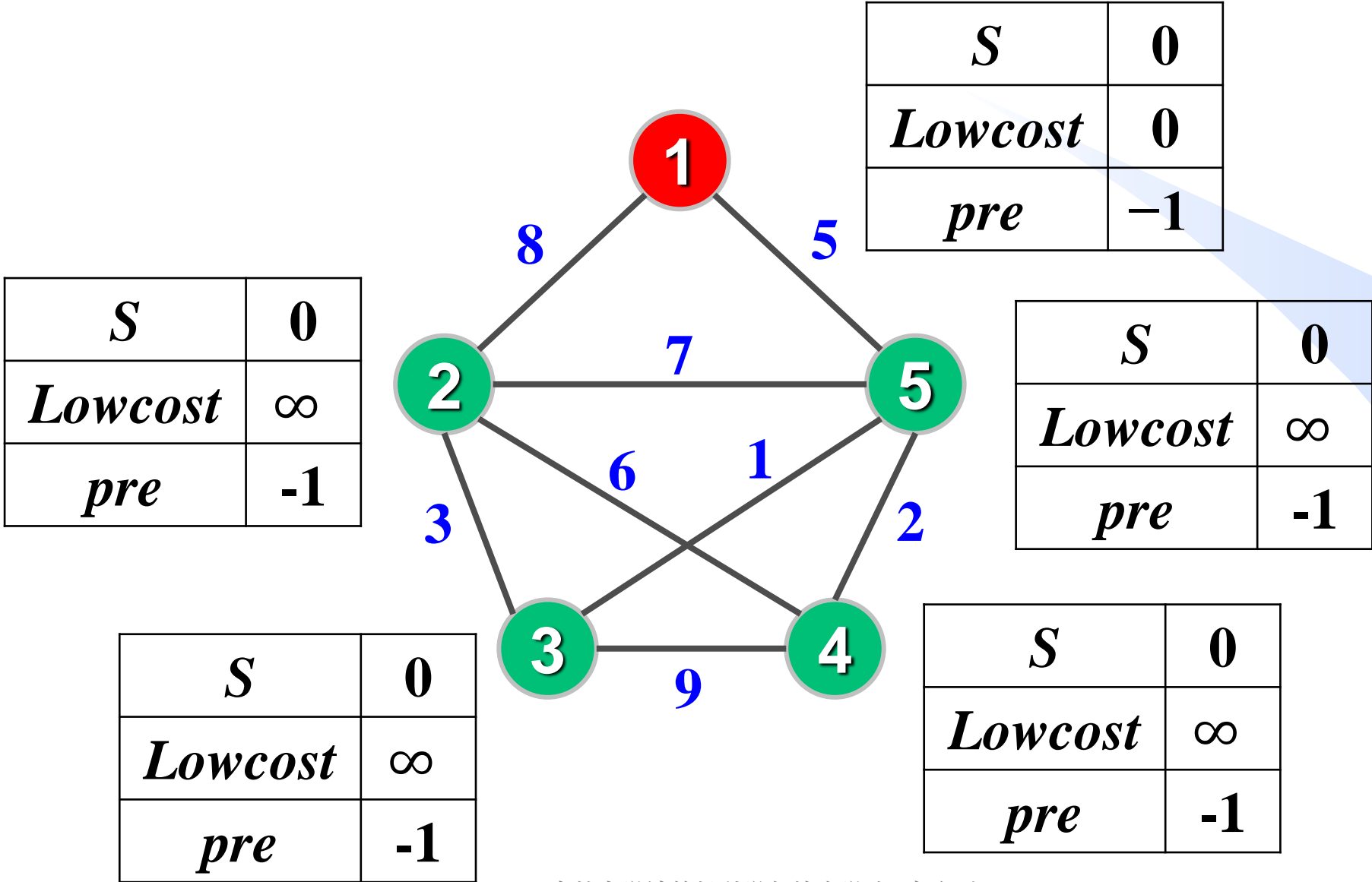
Dijkstra算法 VS Prim算法

```
void Dijkstra(int G[N][N],int n,int u,int dist[],int pre[]){
    int S[N]={0};
    for(int i=1;i<=n;i++){dist[i]=(i==u)?0:INF; pre[i]=-1;}
    for(int i=1;i<=n;i++){
        int v = FindMin(S, dist, n);
        if(v==-1) return;
        S[v]=1;
        for(int w=1; w<=n; w++){
            if(S[w]==0 && dist[v]+G[v][w]<dist[w]){
                dist[w]=dist[v]+G[v][w]; pre[w]=v;
            }
        }
    }
}
```

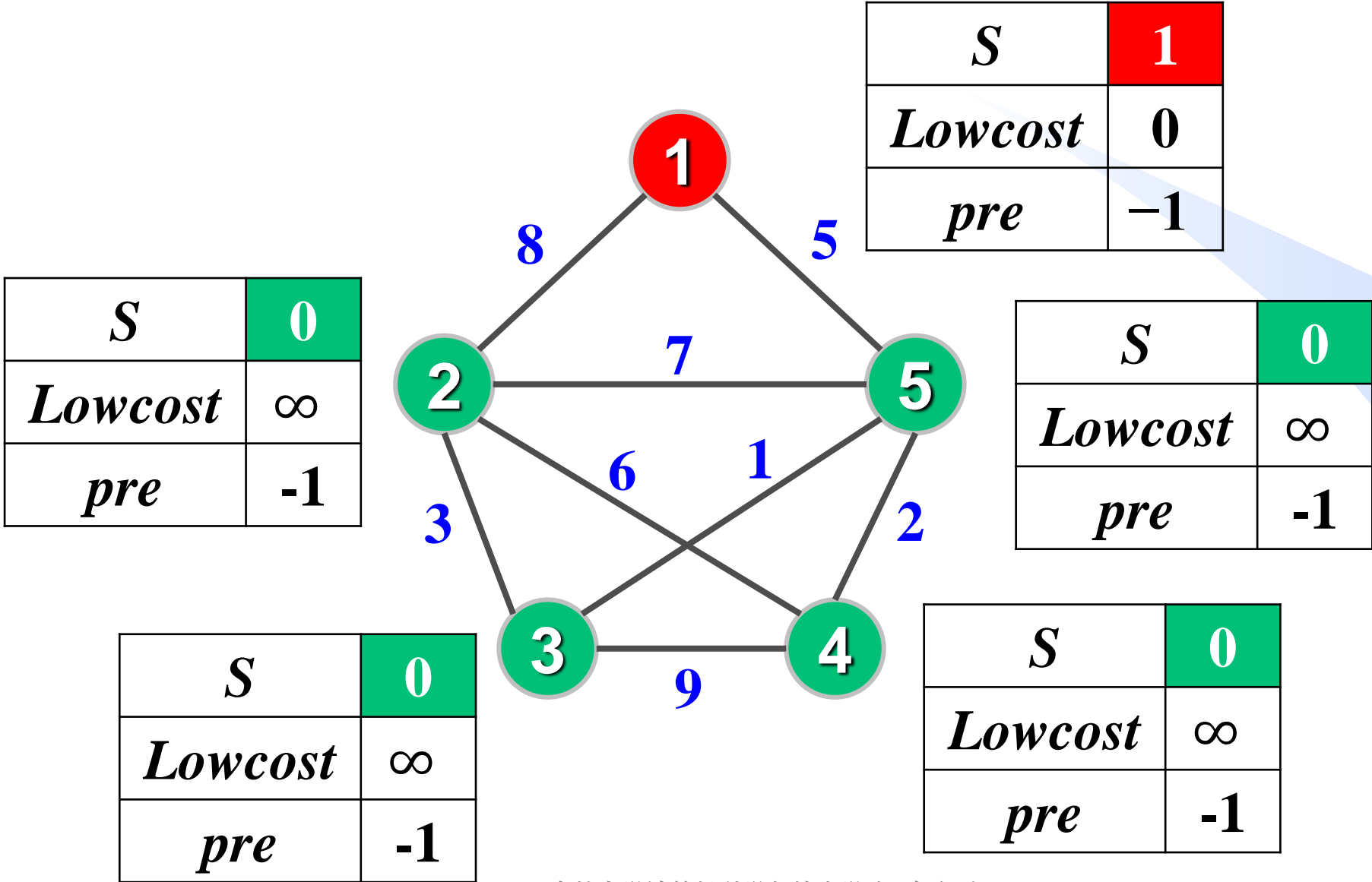
```
int Prim(int G[N][N],int n,int u,int Lowcost[],int pre[]){
    int S[N]={0}, sum=0;
    for(int i=1;i<=n;i++){Lowcost[i]=(i==u)?0:INF; pre[i]=-1;}
    for(int i=1;i<=n;i++){
        int v = FindMin(S, Lowcost, n);
        if(v==-1) return sum;
        S[v]=1; sum+=Lowcost[v];
        for(int w=1; w<=n; w++){
            if(S[w]==0 && G[v][w]<Lowcost[w]){
                Lowcost[w]=G[v][w]; pre[w]=v;
            }
        }
    }
    return sum;
}
```

仔细体会两段代码的细微不同，会加深你对这两个算法的理解

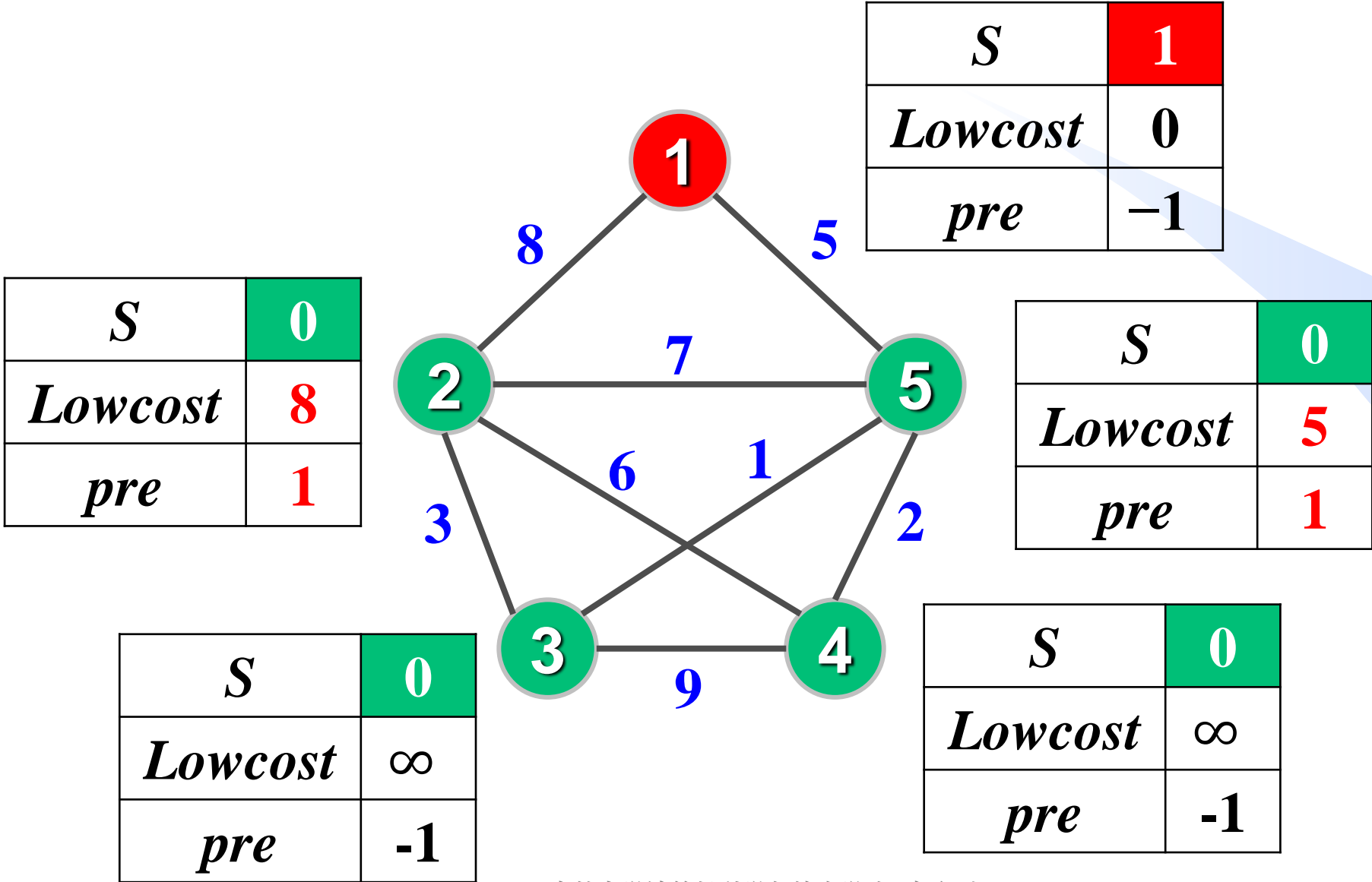
Prim算法举例



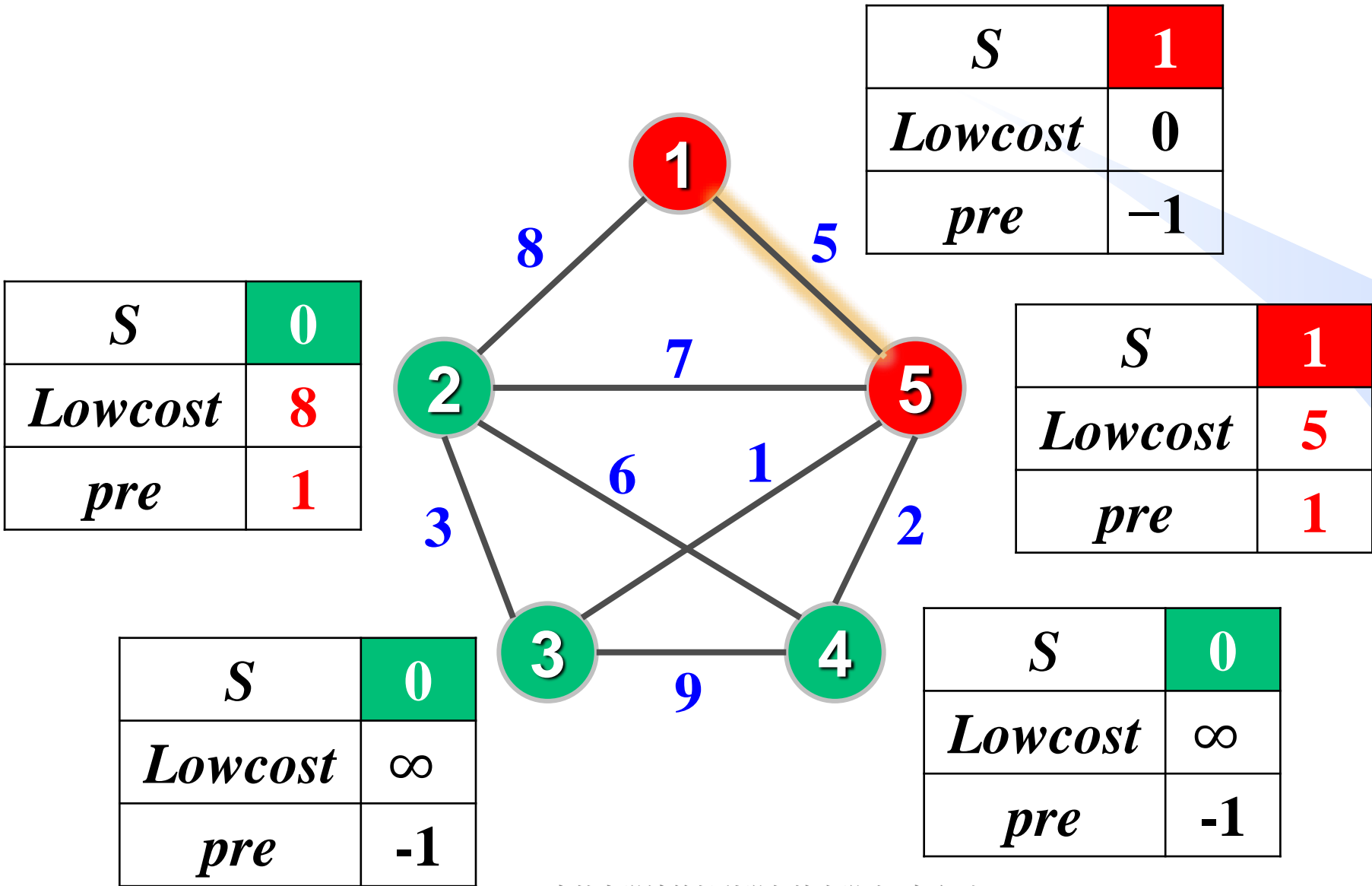
Prim算法举例



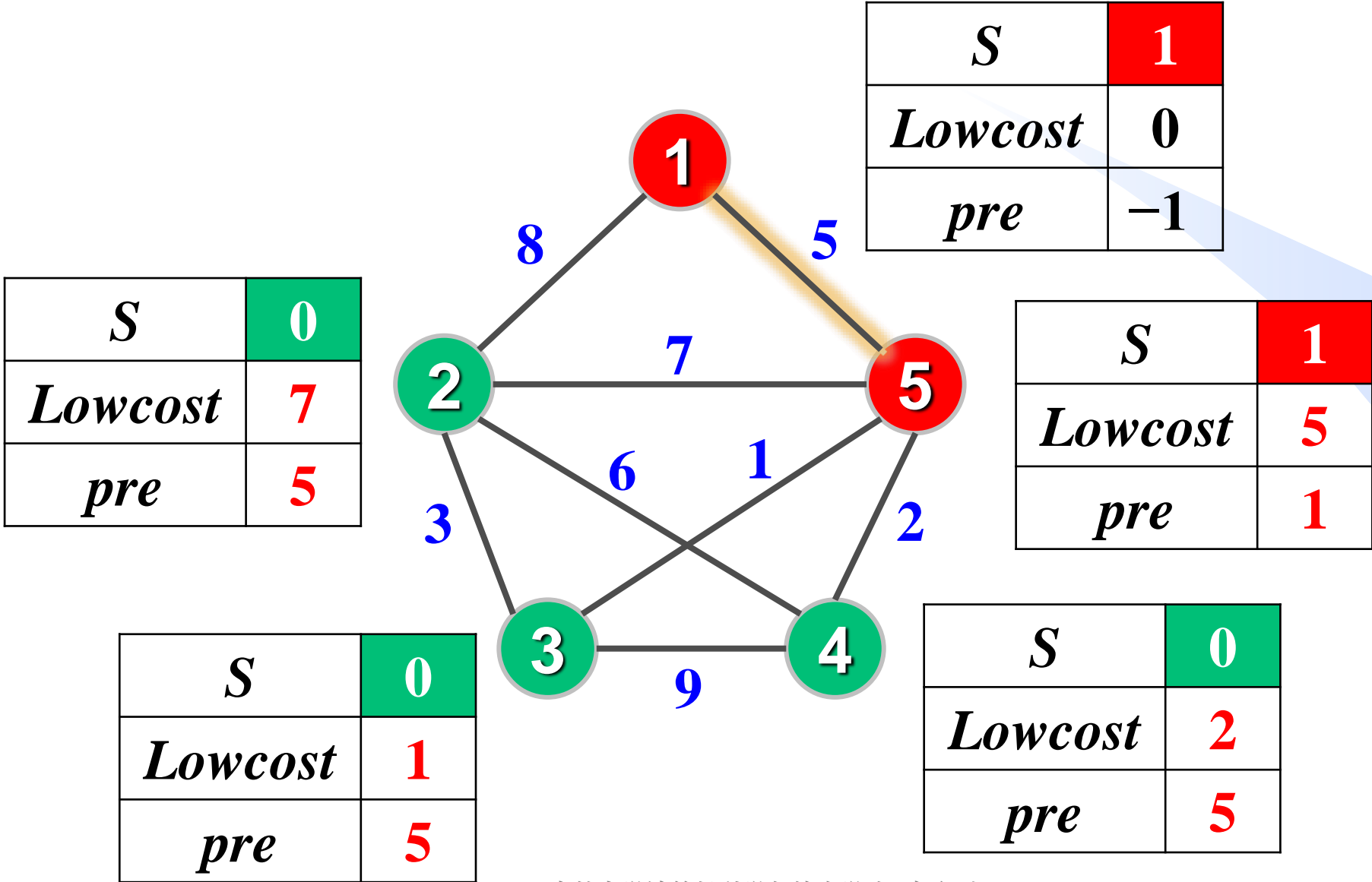
Prim算法举例



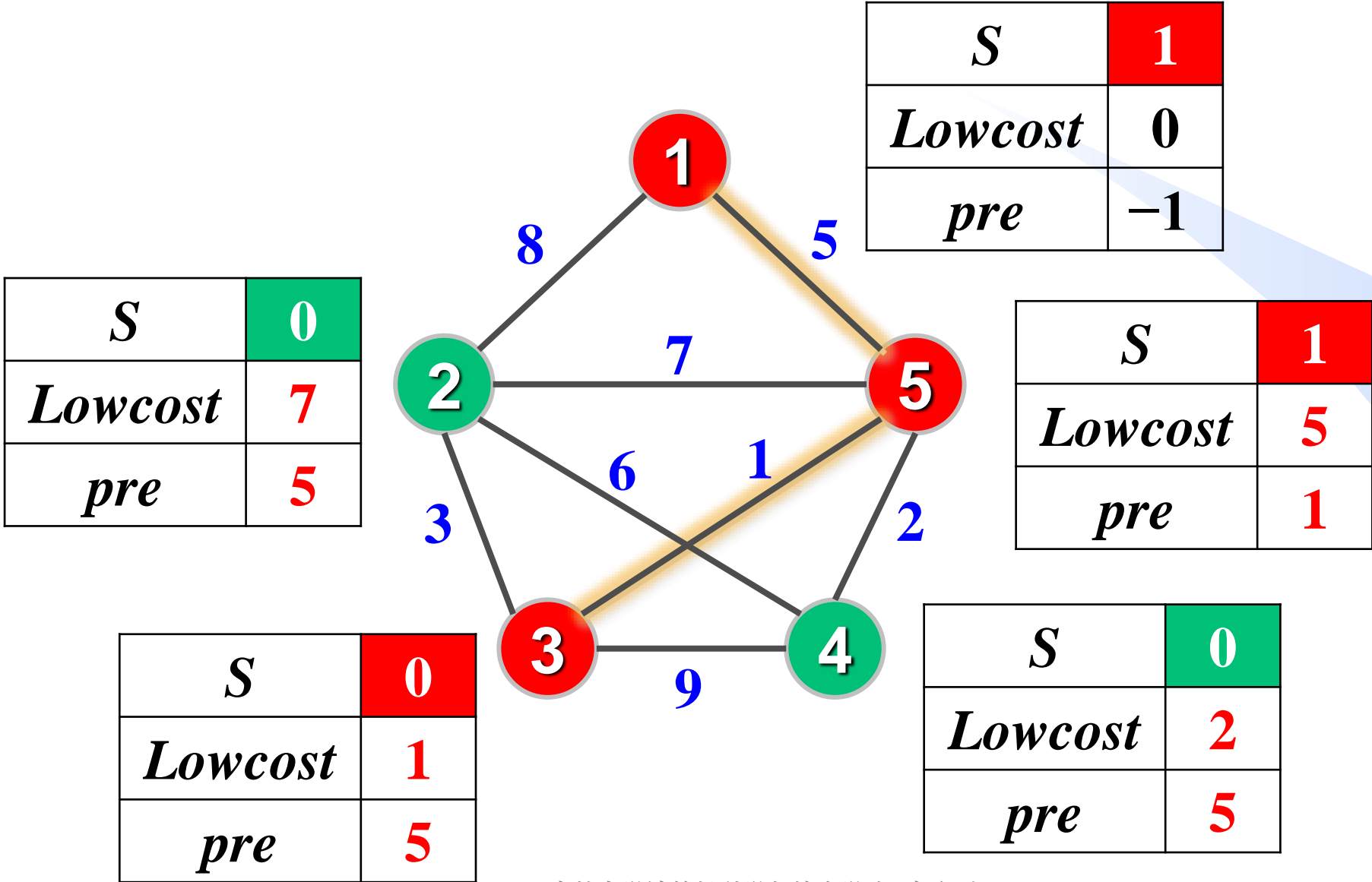
Prim算法举例



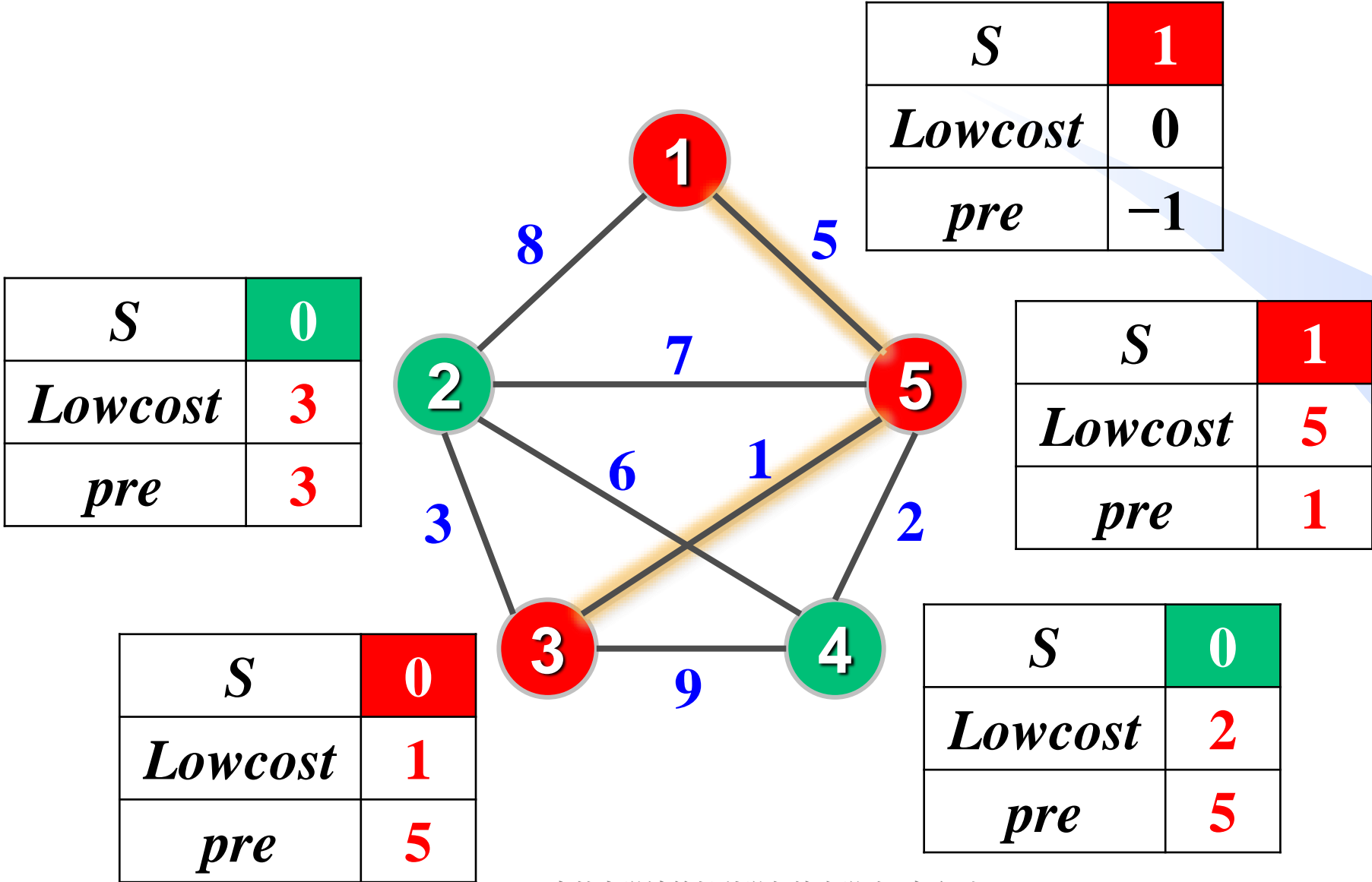
Prim算法举例



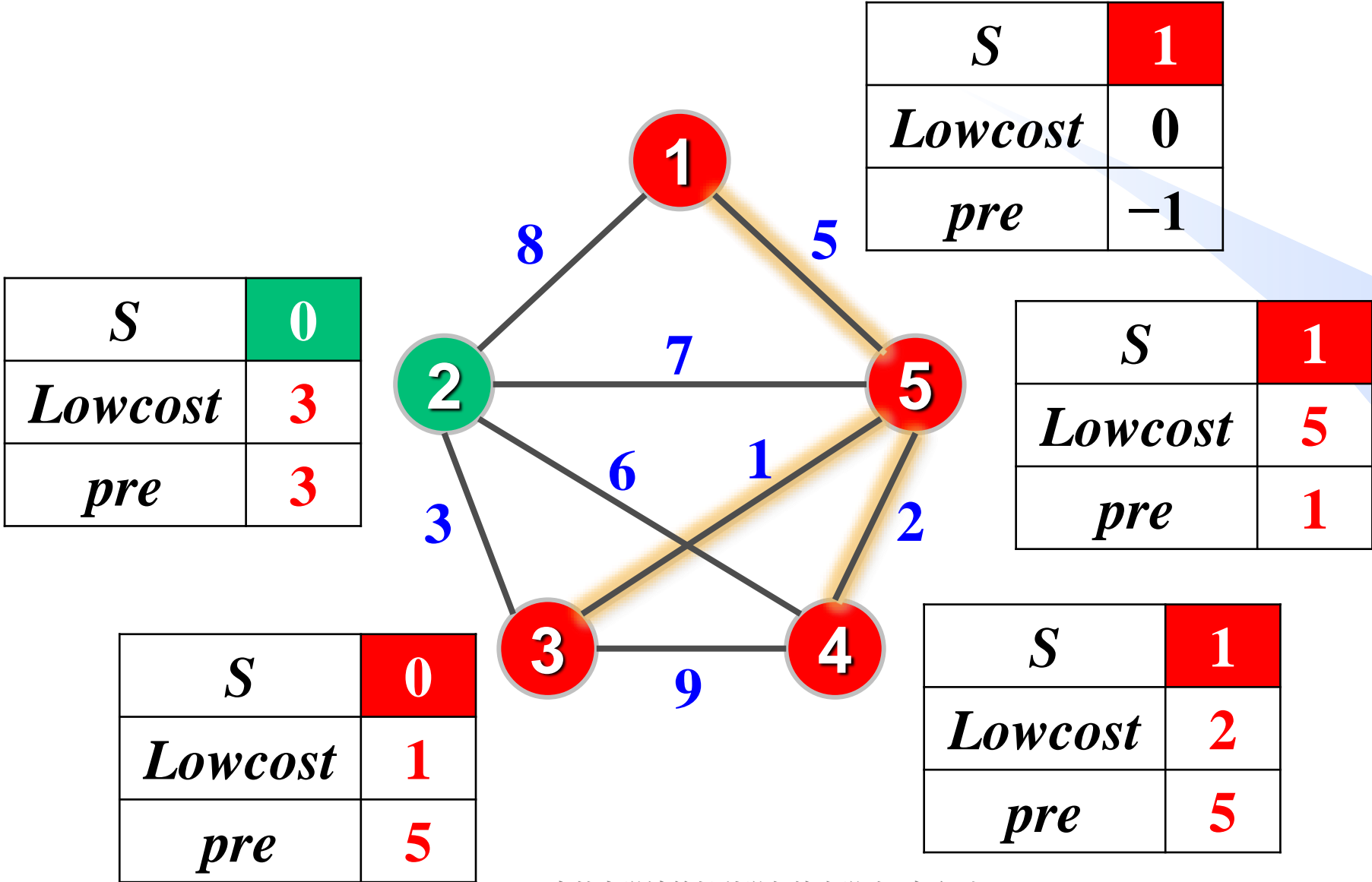
Prim算法举例



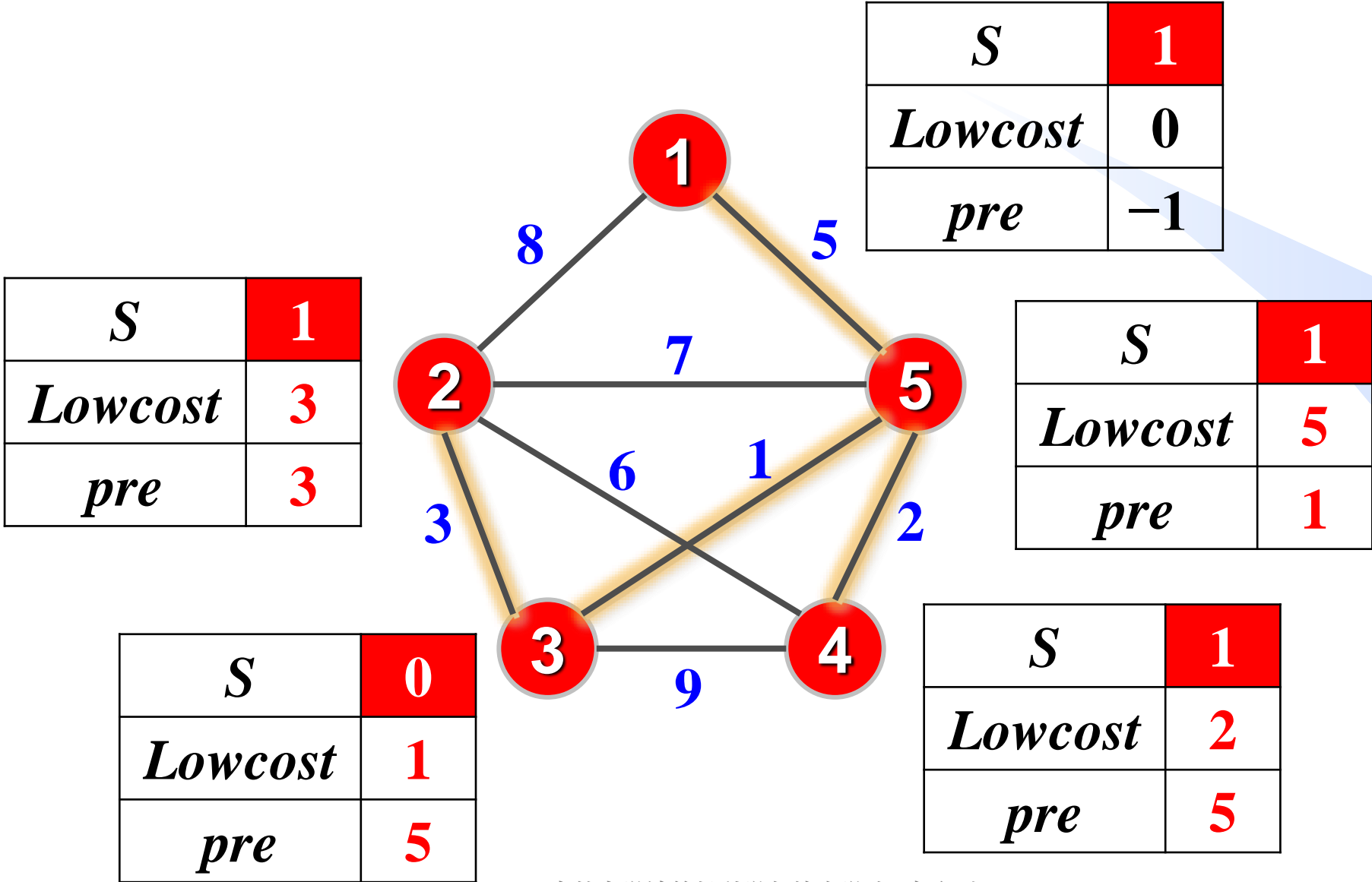
Prim算法举例

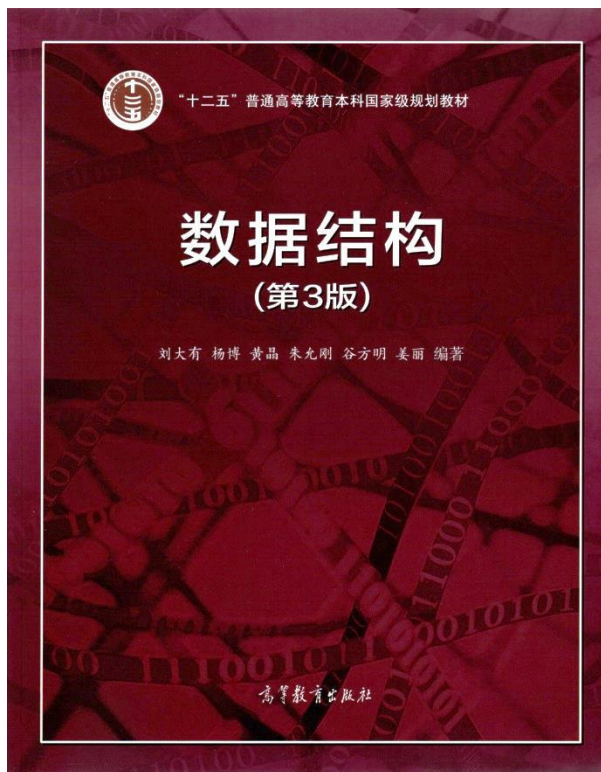


Prim算法举例



Prim算法举例





最小支撑树及图应用

- 最小支撑树的概念
- Prim算法
- **Kruskal算法**

数据之美
结构之美
算法之道

zhuyungang@jlu.edu.cn

Kruskal算法（逐边加入）



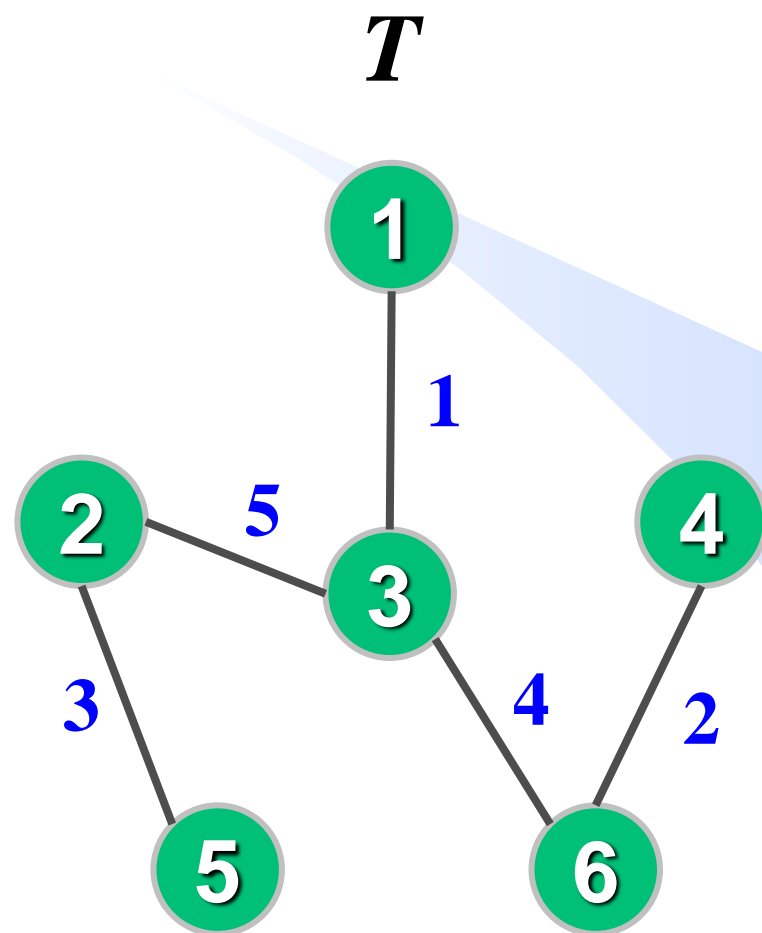
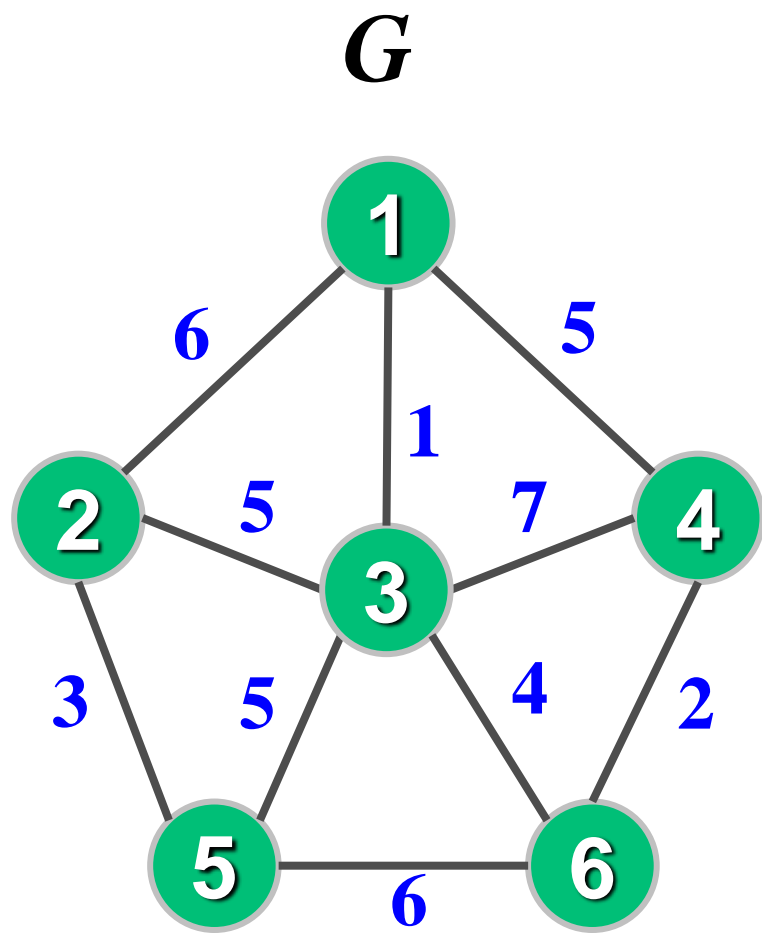
Joseph Kruskal
(1928-2010)

普林斯顿大学博士
贝尔实验室研究员
美国统计学会会士

Kruskal算法（逐边加入）

设连通图 $G=(V, E)$ ，令 T 为 G 的最小支撑树，初始时 T 中有 G 的 n 个顶点和0条边，可看成 n 个连通分量。

- ① 在 G 中选择权值最小的边，并将此边从 G 中删除；
- ② 若该边加入 T 后不产生环（即此边的两个端点在 T 的不同连通分量中），则将此边加入 T 中，从而使 T 减少一个连通分量，否则本步骤无操作。
- ③ 重复① ②直至 T 中仅剩一个连通分量。



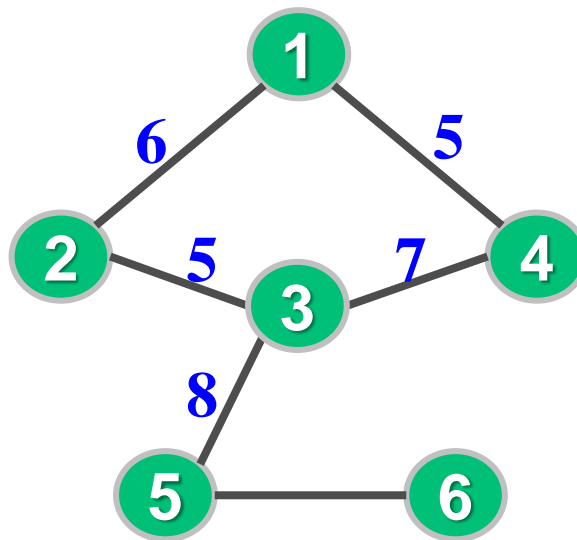
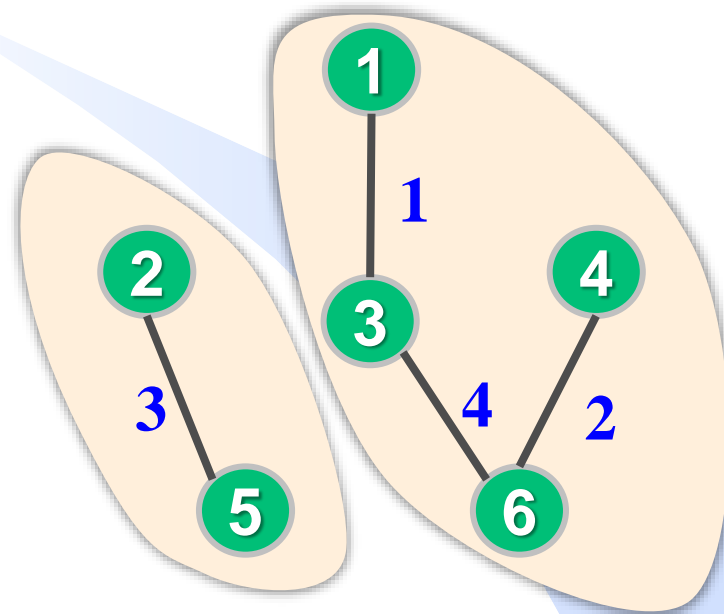
Kruskal算法的实现

- 选权值最小的边：先对所有边按权值递增排序。
- 判环：并查集（连通分量 \Leftrightarrow 集合）



顶点 u 和 v 不在同一连通分量，在MST中加入边 (u,v) 不会产生环，故边 (u,v) 是MST的边

```
if (Find(u) != Find(v)) {  
    将边 $(u,v)$ 加入最小支撑树  
    Union( $u, v$ ).  
}
```

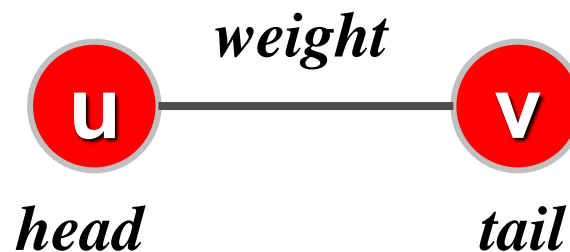
 G  T

合并 u 和 v 所在集合，使二者在同一连通分量里

Kruskal算法的实现

- 如何存图：只需存储图中边的信息，使用边集数组。

```
struct Edge{  
    int head;  
    int tail;  
    int weight;  
};  
Edge E[1010];
```



- 若图的最小支撑树存在，输出最小支撑树的边并返回边权之和，否则（图不连通）返回 ∞

Kruskal算法的实现

```

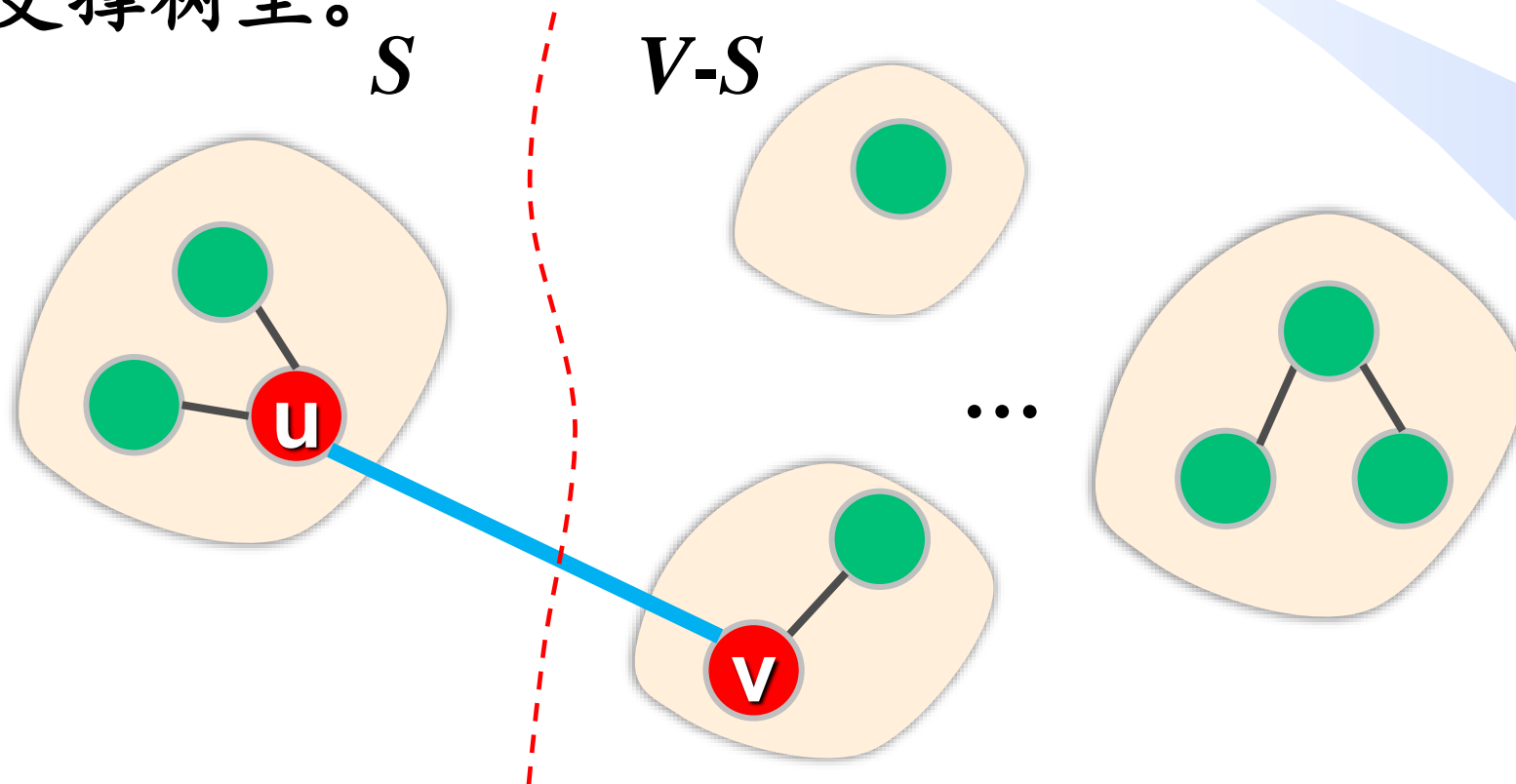
int Kruskal(Edge E[],int n,int e){ //E存图的边,n为顶点数,e为边数
    for(int i=1;i<=n;i++) Make_set(i); //初始化
    Sort(E,e); //对边按权值递增排序,时间复杂度O(eloge)
    int sum=0,k=0; //sum为mst边权和,k为已找到的mst边的条数
    for(int i=0;i<e;i++){ //从小到大依次扫描每条边
        int u=E[i].head, v=E[i].tail, w=E[i].weight;
        if(Find(u) != Find(v)){ //边(u,v)是最小生成树的一条边
            printf("%d-%d",u,v); k++; sum+=w;
            Union(u,v); //合并集合
        }
        if(k==n-1) return sum; //成功找到mst的全部n-1条边
    }
    return INF;
}

```

时间复杂度
 $O(e \log e)$

Kruskal算法的正确性依据

设Kruskal算法某次选出的最小边为 (u,v) ，其中 u 所在的连通分量的顶点集合为 S 。则边 (u,v) 是集合 S 和 $V-S$ 的最小跨边，故必在最小支撑树里。



Prim算法与Kruskal算法

贪心算法

算法	时间复杂度	适用场合
Prim	$O(n^2)$	稠密图
Kruskal	$O(e \log e)$	稀疏图

判断：相对于 Kruskal 算法， Prim 算法更适宜于稀疏图。
【2023年清华大学考研题】

主要研究进展举例

时间	时间复杂度	提出者
1975	$O(e \log \log n)$	姚期智
2002	$O(e \cdot \alpha(n, e))$	Pettie Ramachandran



姚期智 (1946-)

图灵奖获得者 (迄今唯一华人获奖者)
原Stanford, Princeton, UC Berkeley教授

现清华大学教授

中国科学院院士

美国科学院外籍院士

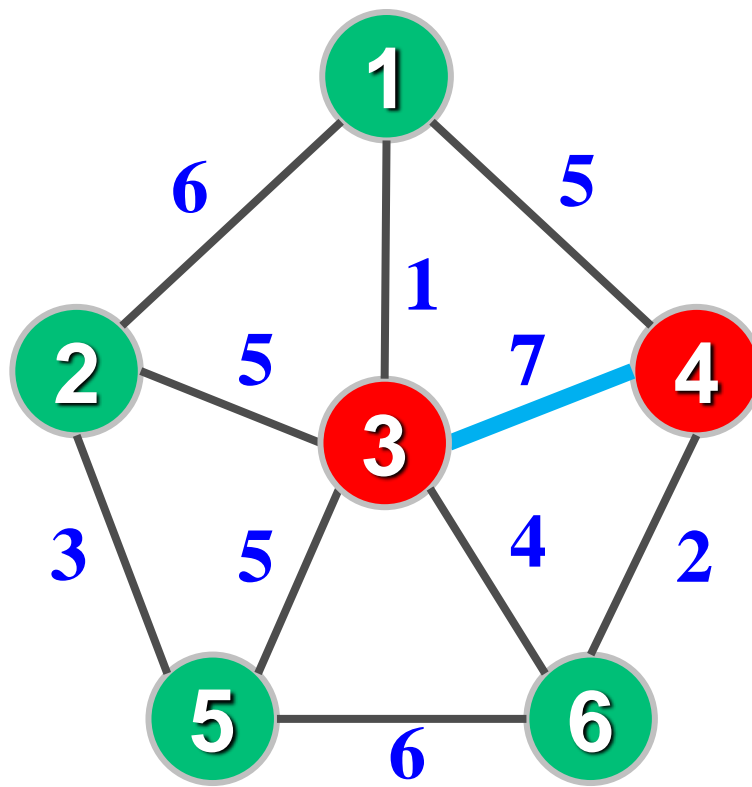
2016年放弃美国国籍成为中国公民

课下思考

- 图G的最小支撑树一定包含G中的最小边么？一定包含G中第2小的边么？一定包含G中第3小的边么？
- 若图中边权互异，其最小支撑树是否唯一？
- 若图中有权值相同的边，其最小支撑树一定不唯一么？

拓展问题

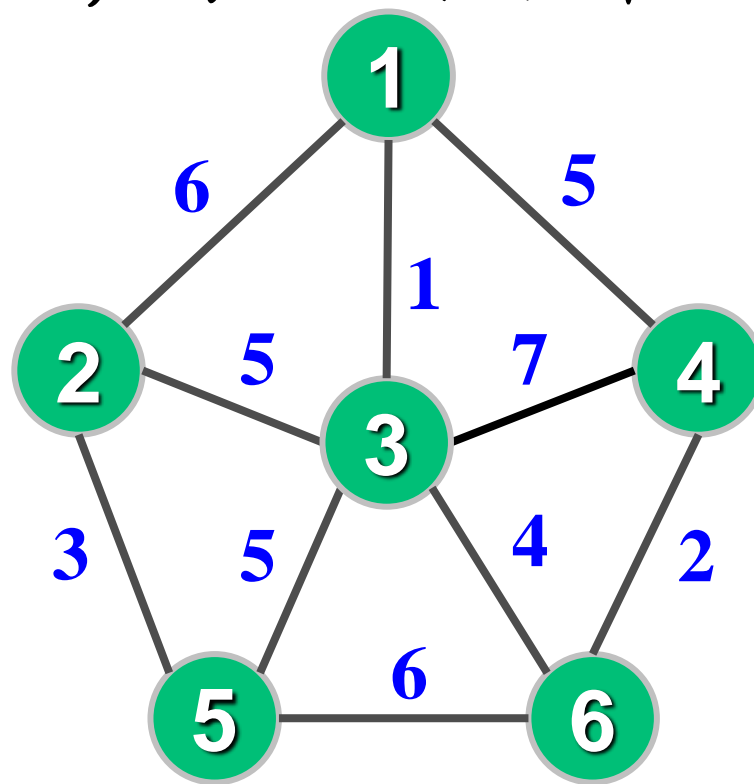
➤ 求图的必须包含某些边的最小支撑树。



拓展问题

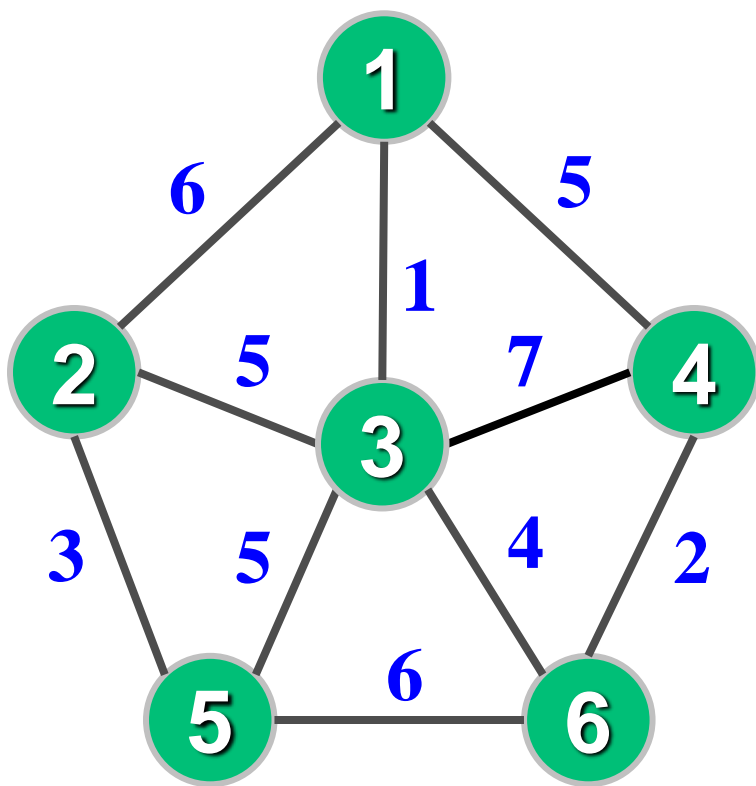
➤ 最大支撑树

- ✓ 边权值取反，求最小支撑树
- ✓ 修改Kruskal算法，每次选权值最大的边



拓展问题

- 最小乘积支撑树：给定正权图，求图的一棵支撑树，其边的权值乘积最小。



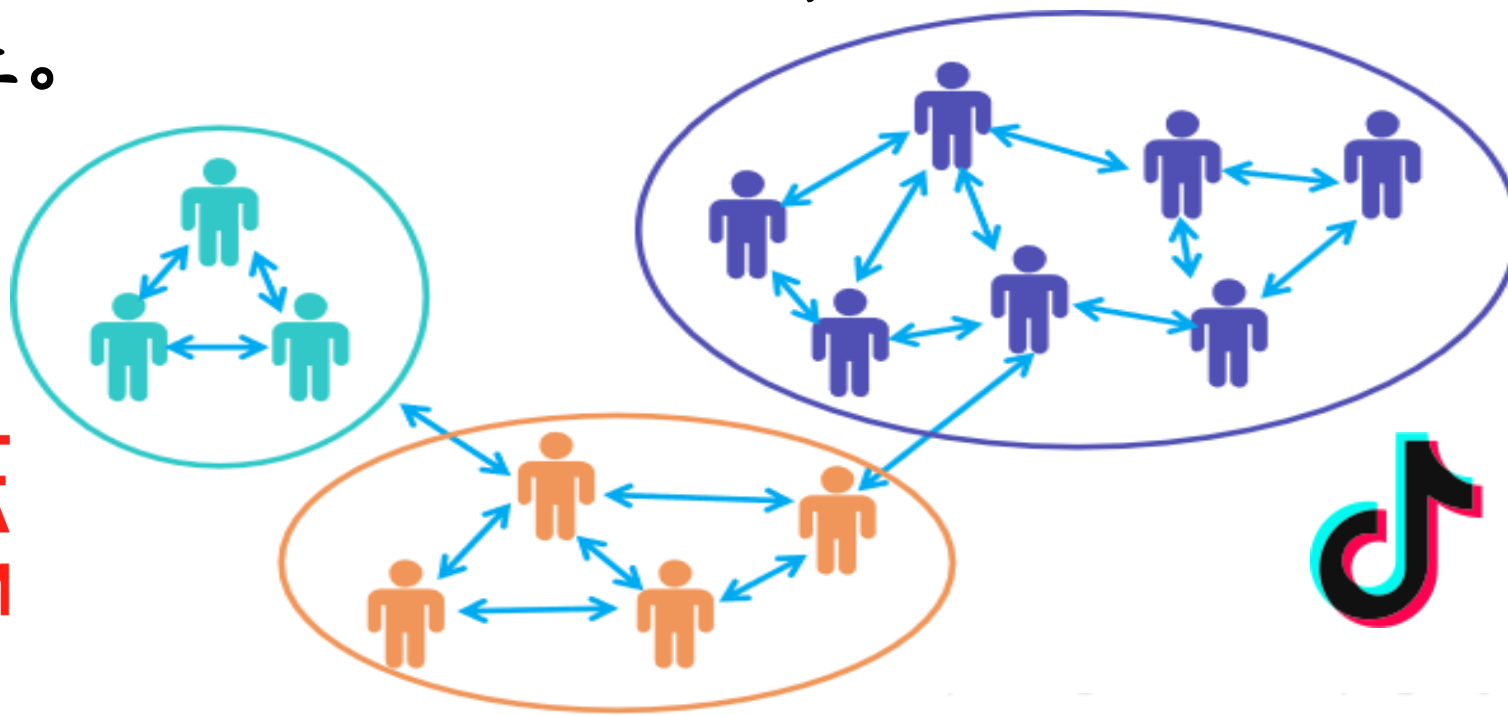
MST的其他应用举例——聚类

D

- 用户聚类：根据不同用户的兴趣，对所有用户聚类，相似的用户分到一组（簇），对不同组的用户推荐不同商品。
- 每个用户对应图中一个顶点，边权反映用户购买习惯的相似性。设置参数 k ，执行Kruskal算法直至剩下 k 个连通分量（ k 组用户）。或者设置一个阈值，选出的最小边权高于阈值算法就终止。

淘宝网
Taobao.com

京东
JD.COM



Tmall
理想生活上天猫

抖音

MST的其他应用举例——图像分割

- 每个像素对应一个顶点，边权表示两个像素颜色、亮度等的相似度，用Kruskal算法把颜色、亮度相似的像素分到一组（簇）。

