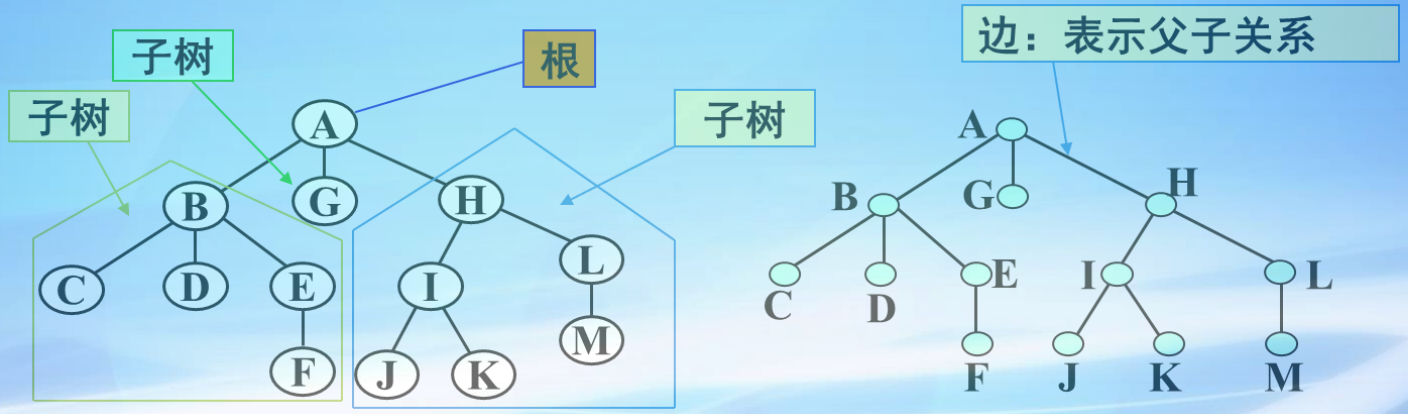
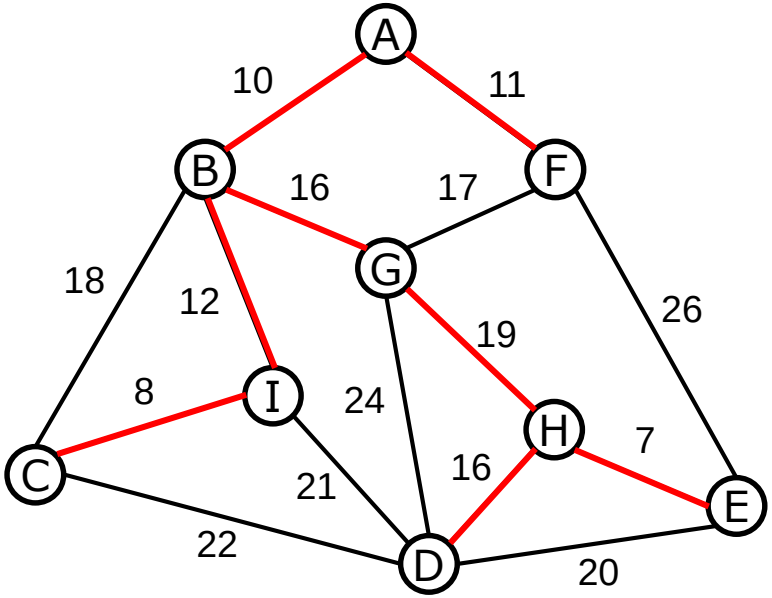
最小生成树：Kruskal算法

《数据结构》



课程回顾



无向图：所有边都是无向边的图

连通图：所有顶点都连通的图

加权图：所有边都带权重的图

生成树：包含图的全部n个顶点，但仅保留维持所有顶点连通的n-1条边

生成树的耗费：生成树所有边的权重的和

教学目标和要求

1.准确描述图的最小生成树的定义，知道最小生成树能够做什么

2.能够运用Kruskal算法求解图的最小生成树的边序列

3.能够计算机编程实现Kruskal算法

1、最小生成树的定义

什么是最小生成树？

设S=(V,T)是无向连通加权图G=(V,E)的一棵生成树，S的边长之和称为S的耗费，记为C(S)，

$$C(S)=\sum_{e\in T}C(e),$$

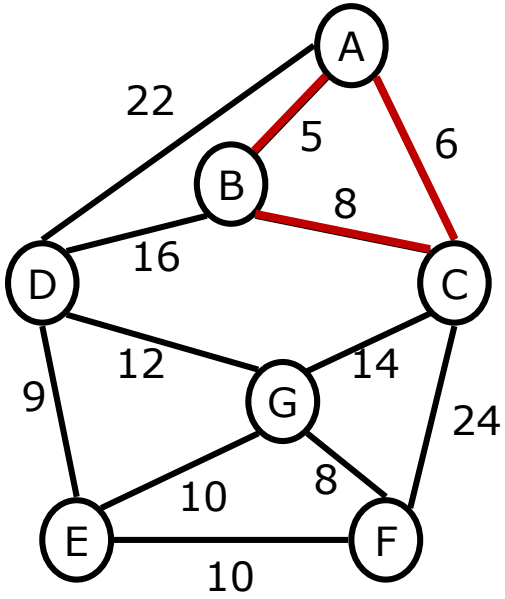
在G的所有生成树中，耗费最小的生成树S叫做最小生成树(Minimum Spanning Tree，MST)

若S是G的MST，对于G的任意生成树S'，C(S)≤C(S')

2、如何求解最小生成树？

穷举？

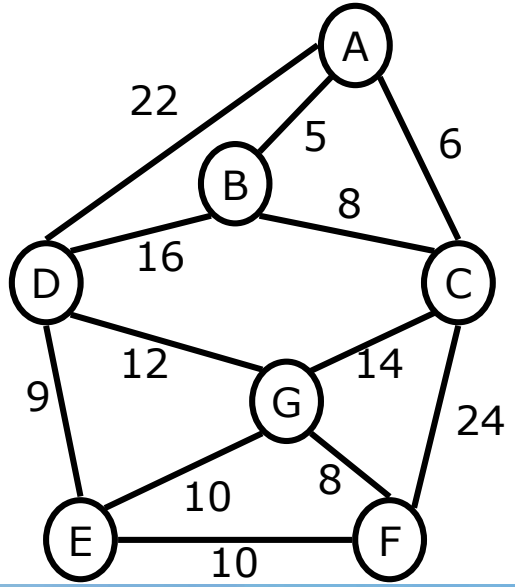
选短边？

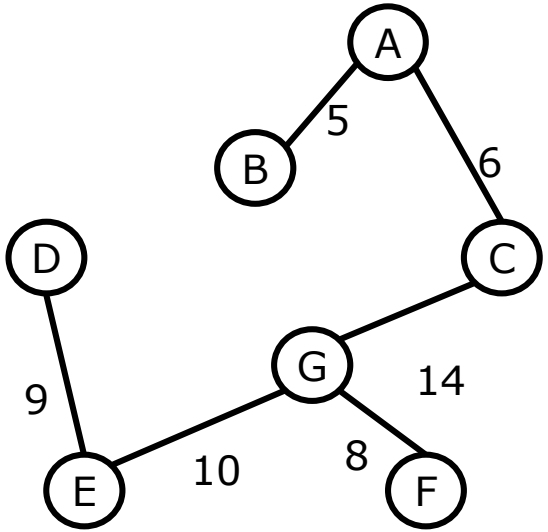


2.1 MST的求解：Kruskal算法

算法的基本思想

按长度从小到大的依次把最短边加进生成树的树边集
若添加某边后形成了回路，就舍弃这条边
反复如此，直到选出n-1条边，便得到最小生成树





耗费：58

2.2 Kruskal算法的描述

算法原始描述

```
void  Kruskal(*** )  //***表示参数
    {  int et=0;          //et记录边数
1.      置树边集T为空;
2.      while(et<n-1)        //n是顶点数
3.      { 从G中选出当前最短边（v，w）;
4.        if(添此边于T中，不产生回路)
5.          { 把（v，w）加进T;   et++; }
6.        else  舍弃（v，w）
        }
    }
```

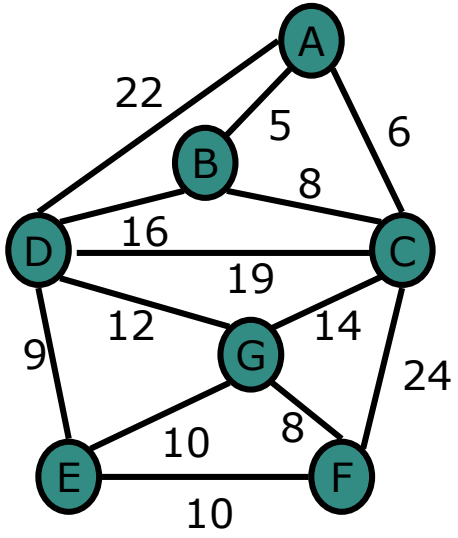
1、图怎么存？怎么选短边？

2、选出来的树边怎么存？

3、怎么样判回路？

3、Kruskal算法的实现

实现要点　1、图怎么存？怎么选短边？



采用三元数组保存图的边集

按照边的耗费从小到大排序，从左到右依次选取边

A	A	B	G	E	E	E	D	G	D	D	A	C
B	C	C	F	D	F	G	G	C	B	C	D	F
5	6	8	8	9	10	10	12	14	16	19	22	24
0	1	2	3	4	5	6	7	8	9	10	11	12

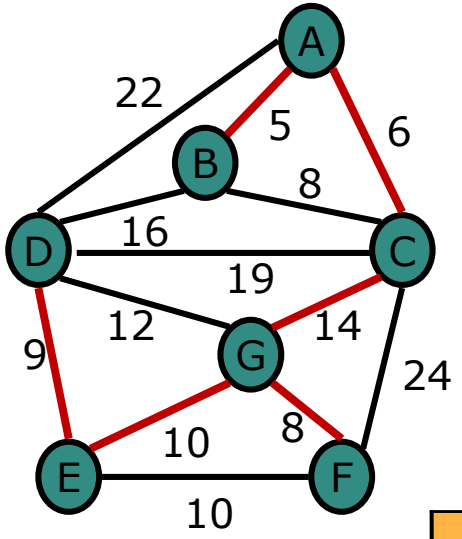
顶点1

顶点2

边的耗费

3、Kruskal算法的实现

实现要点　2、选中的树边怎么存？



复用图的边集数组

选中的边耗费保持不变，舍弃的边耗费改为-1

A	A	B	G	E	E	E	D	G	D	D	A	C
B	C	C	F	D	G	F	G	C	B	C	D	F
5	6	-1	8	9	10	-1	-1	14	-1	-1	-1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12

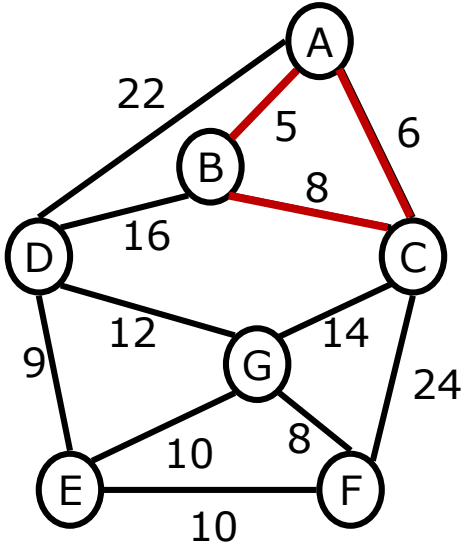
3、Kruskal算法的实现

实现要点 | 3、怎么判回路？

为什么会产生回路？

加边（V,W）以前，V和W已经通过以前选中的树边连通！

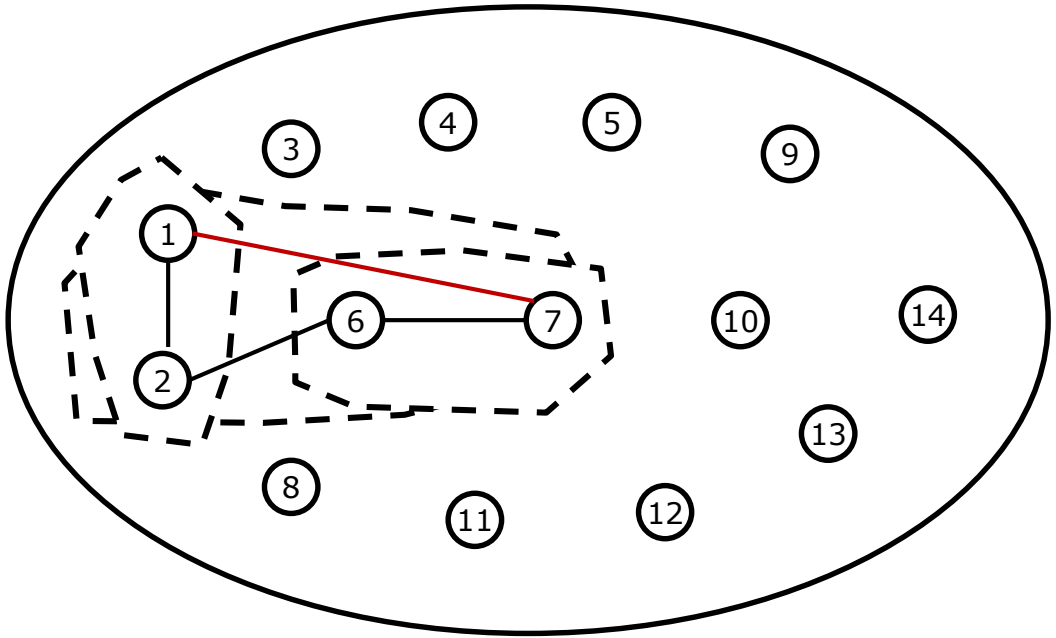
问题转变为，如何判断两个点是连通的！



A	A	B	G	D	G	E	D	G	D	C	A
B	C	C	F	E	E	F	G	C	B	F	D
5	6	8	8	9	10	10	12	14	16	24	22

3、Kruskal算法的实现

实现要点	4、如何判断顶点是不是连通？



3、Kruskal算法的描述

采用“集合合并法”描述Kruskal算法

```
void  Kruskal(***) //***表示函数要求的参数
 {   int et=0; 置树边集T为空;
1.  每个顶点自成一个集合;
2.   while(et<n-1)
3.   { 从G中选出当前最短边（v，w）;
4.     if(顶点v和顶点w不属于同一个集合)
5.      { 把（v，w）加进T; et++;
6.        将顶点v和顶点w所在的集合合并;
       }
 }
```

1、怎么表示顶点的集合？

2、集合怎么合并？

3、Kruskal算法的实现

实现要点

5、怎么表示顶点集合？

V_i={ B , E }

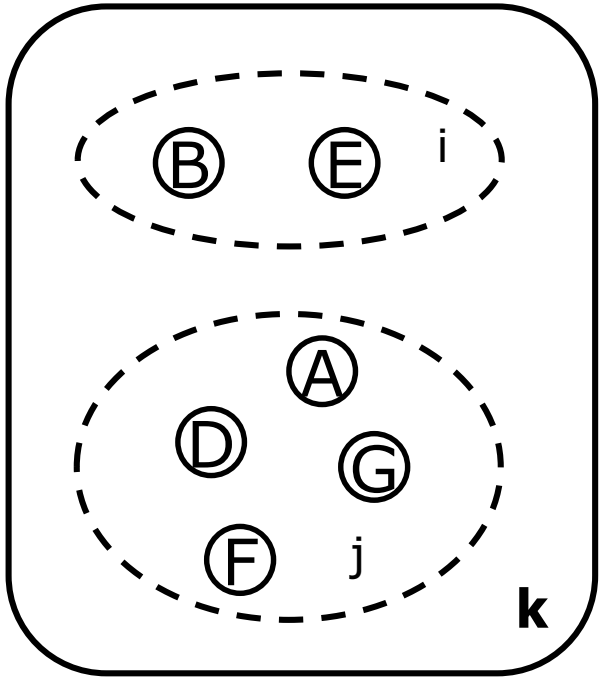
V_j={D , A , G , F}

V_m={ C }

顶点名	集合名
A	j
B	i
C	m
D	j
E	i
F	j
G	j

3、Kruskal算法的实现

实现要点　6、怎么对集合进行合并？



顶点名	集合名
A	j
B	k
C	m
D	j
E	k
F	j
G	j

3、Kruskal算法的实现

采用“集合合并法”描述Kruskal算法

```
void  Kruskal(***) //***表示函数要求的参数
 {   int et=0; 置树边集T为空;
1.  每个顶点自成一个集合，并指定集合名;
2.   while(et<n-1)
3.   { 从G中选出当前最短边（v，w）;
4.     找到v所在的集合名i，和w所在的集合名j;
       if(i!=j)
5.      { 把（v，w）加进T; et++;
6.        将集合i与集合j合并成一个集合k;    }
      }
```

复用三元数组

二元数组

三元数组

由判连通实现判回路

集合合并法

算法主要的时间耗费是边的排序，如果边数为m，那么算法的时间复杂度为O(mlogm)

The End, Thank You!