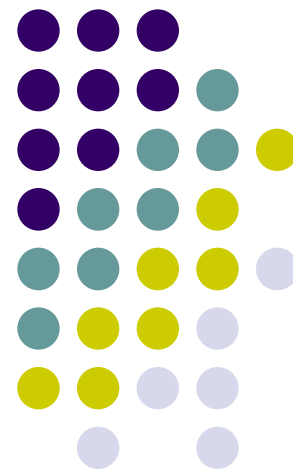


# 生成树

离散数学一树

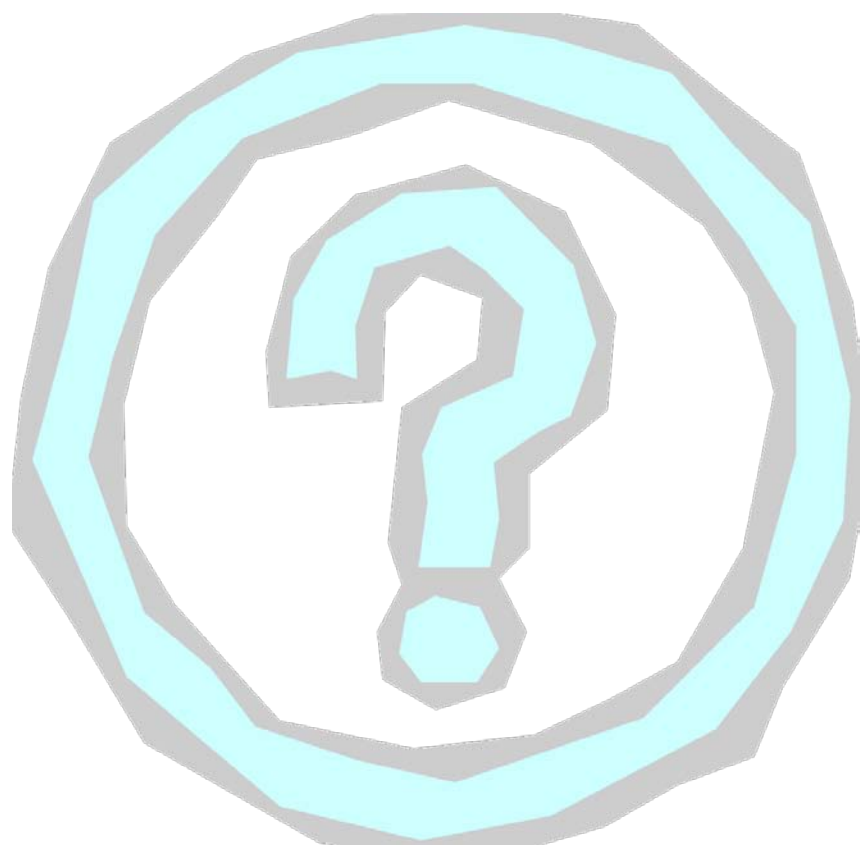
南京大学计算机科学与技术系

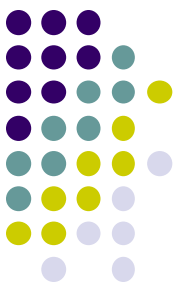




# 内容提要

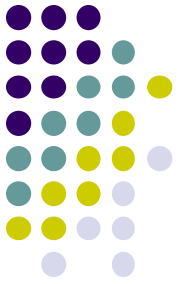
- 生成树
- 深度优先搜索
- 广度优先搜索
- 有向图的深度优先搜索
- 回溯
- 最小生成树算法



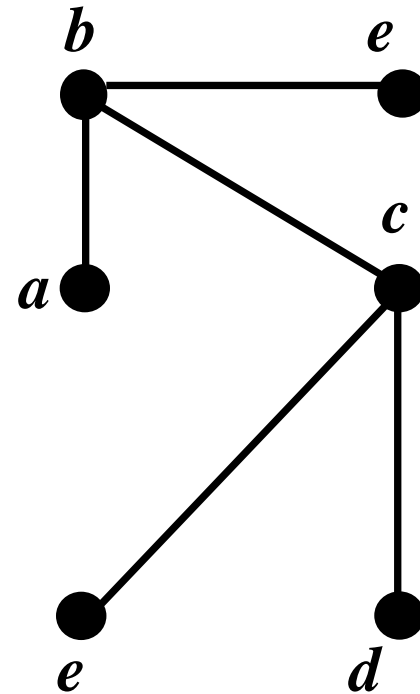
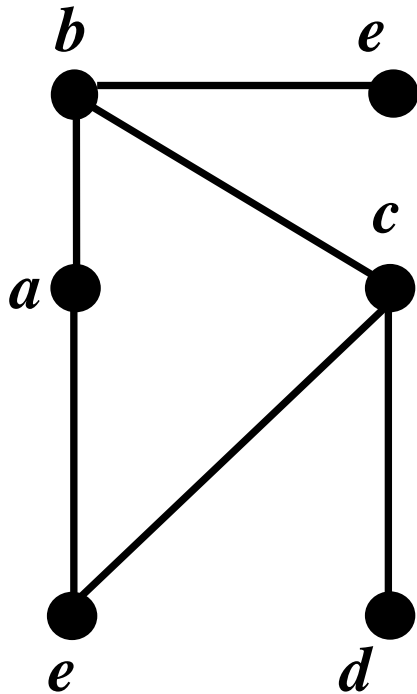


# 生成树

- 定义：若图 $G$ 的生成子图是树，则该子图称为 $G$ 的生成树。
- 无向图 $G$ 连通 当且仅当  $G$ 有生成树
  - 证明(充分性显然):
    - $\Rightarrow$  注意：若 $G$ 是有简单回路的连通图，删除回路上的一条边， $G$ 中的回路一定减少。(因此，用“破圈法”总可以构造连通图的生成树)
- 简单无向图 $G$ 是树 当且仅当  $G$ 有唯一的生成树。
  - 注意： $G$ 中任一回路至少有三条不同的边。



# 构造生成树：深度优先搜索





# 深度优先搜索算法

**Procedure DFS**( $G$ : 带顶点 $v_1, \dots, v_n$ 的连通图)

$T :=$ 只包含顶点 $v_1$ 的树;

**visit**( $v_1$ );

**Procedure visit**( $v$ :  $G$ 的顶点)

**for**  $v$ 每个邻居 $w$  {

**if**  $w$ 不在 $T$ 中 **then** {

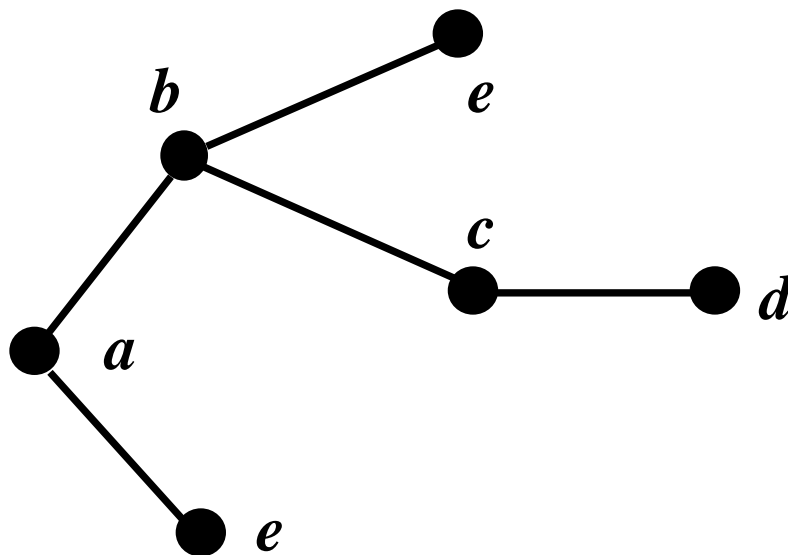
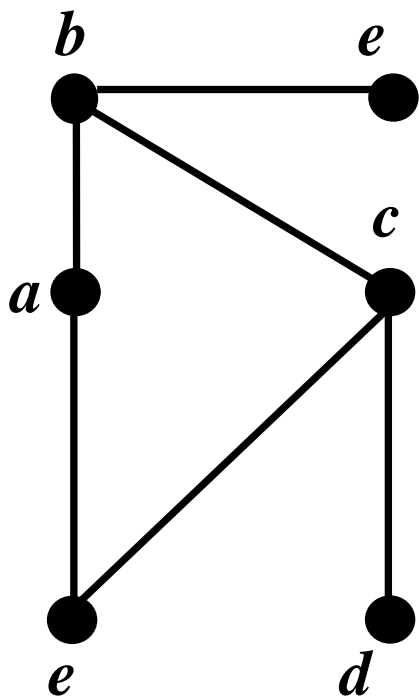
        加入顶点 $w$ 和边 $\{v, w\}$ 到 $T$ ;

**visit**( $w$ );

    }

}

# 构造生成树：广度优先搜索





# 广度优先搜索算法

**Procedure BFS**(G: 带顶点 $v_1, \dots, v_n$ 的连通图)

**T**:=只包含顶点 $v_1$ 的树; **L**:=空表; 把 $v_1$ 放入表**L**中

**While** **L**非空 {

    删除**L**中的第一个顶点 $v$ ;

**for**  $v$ 的每个邻居 $w$  {

**if**  $w$ 既不在**L**中也不在**T**中 **then** {

            加入 $w$ 到**L**的末尾;

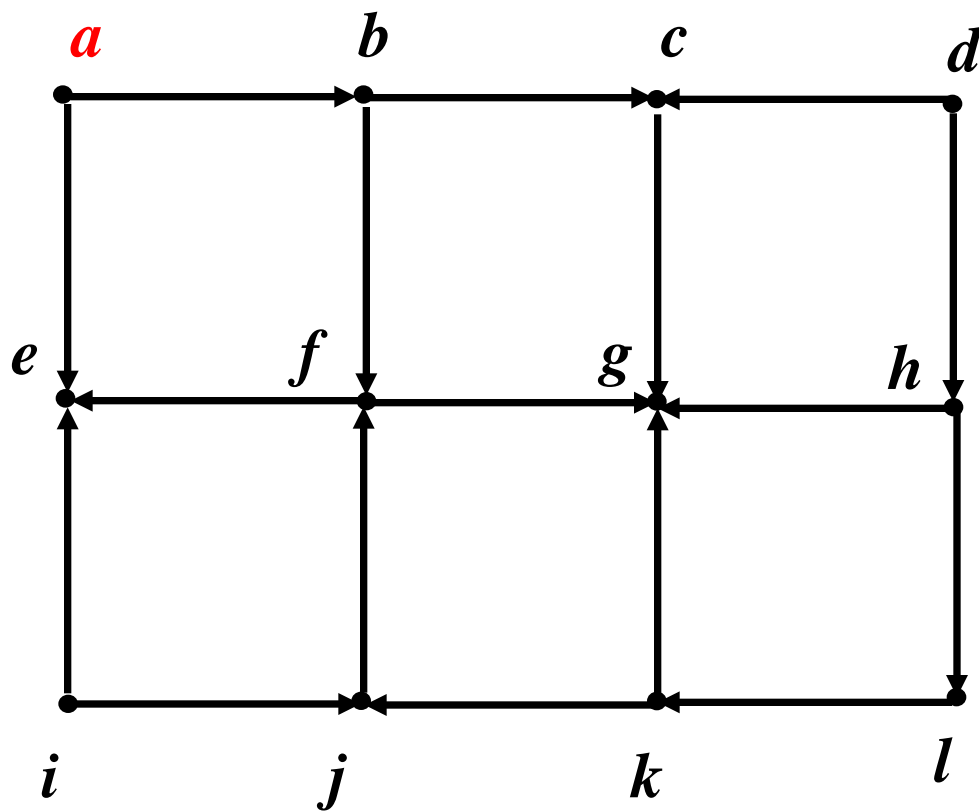
            加入顶点 $w$ 和边 $\{v, w\}$ 到**T**;

        }

    }

}

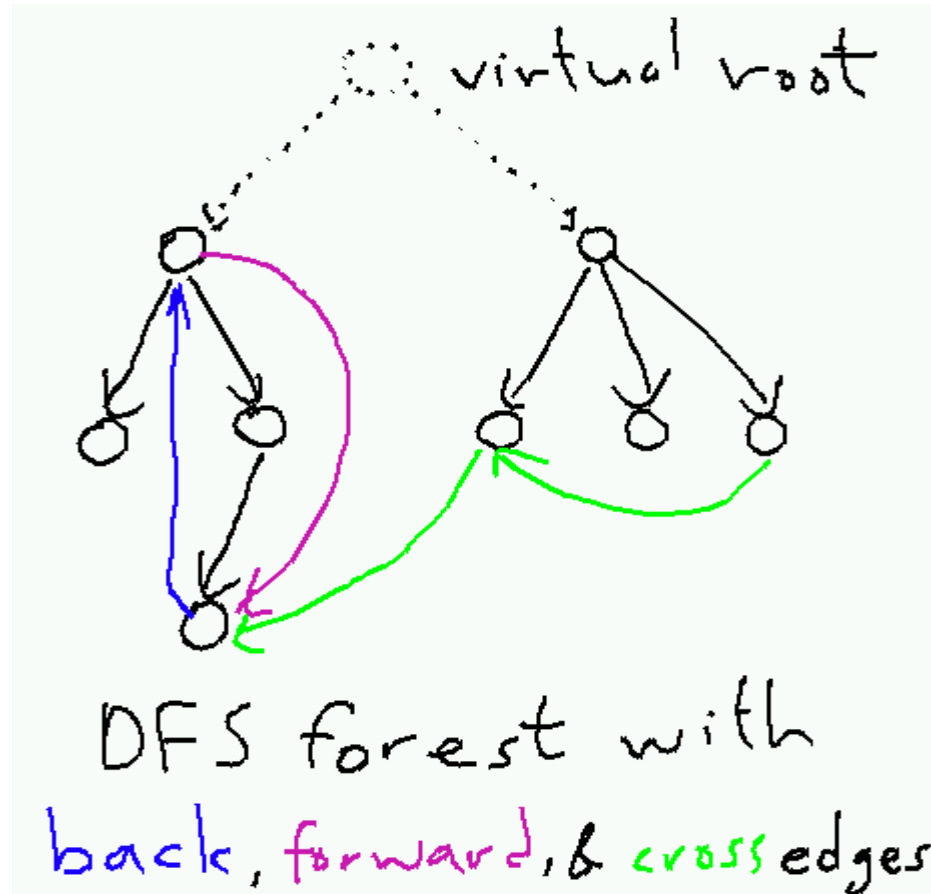
# 有向图的深度优先搜索







# 有向图的深度优先搜索





# 回溯（八皇后）

在 $n \times n$ 格的棋盘上放置彼此不受攻击的 $n$ 个皇后。

从空棋盘开始

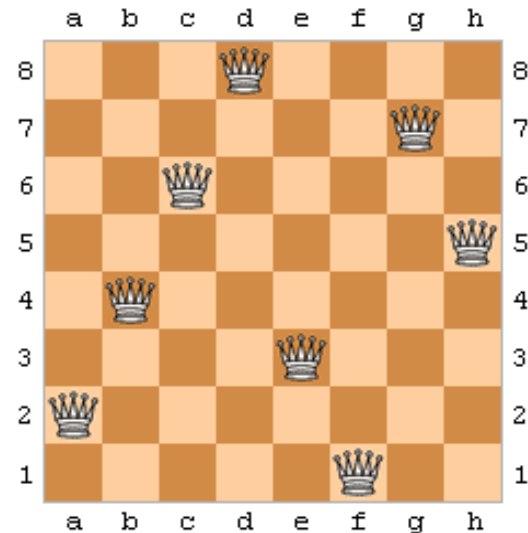
尝试第1列，第1行，... $n$ 行；

尝试第2列，第1行，... $n$ 行；

....

尝试第 $k+1$ 列，第1行，... $n$ 行；

...





# 回溯（子集和）

给定一组正整数 $x_1, \dots, x_n$ ，和为 $M$ 的一个子集？

从空子集开始

尝试添加一项，

和等于 $M$ ，结束；

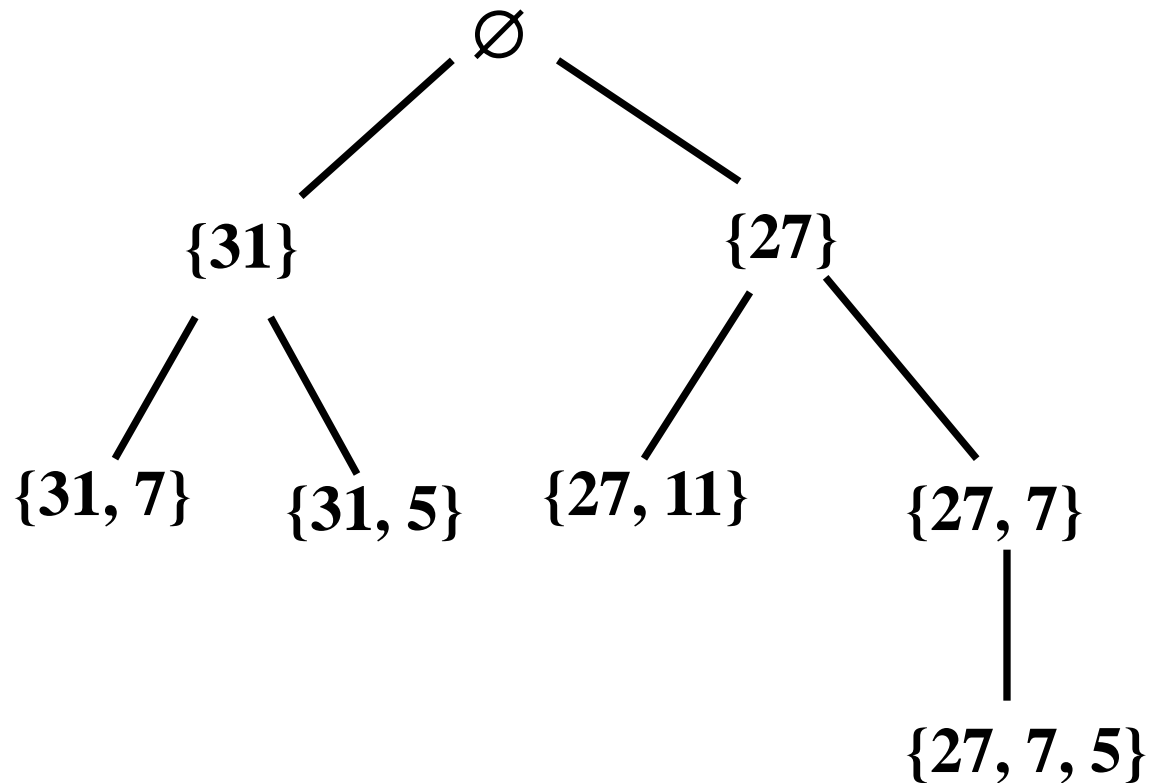
和不超过 $M$ ，子集包含它；

没有合适添加项，去掉和的最后一项，



# 回溯（子集和）

举例：{31, 27, 15, 11, 7, 5}, 和为39的子集？





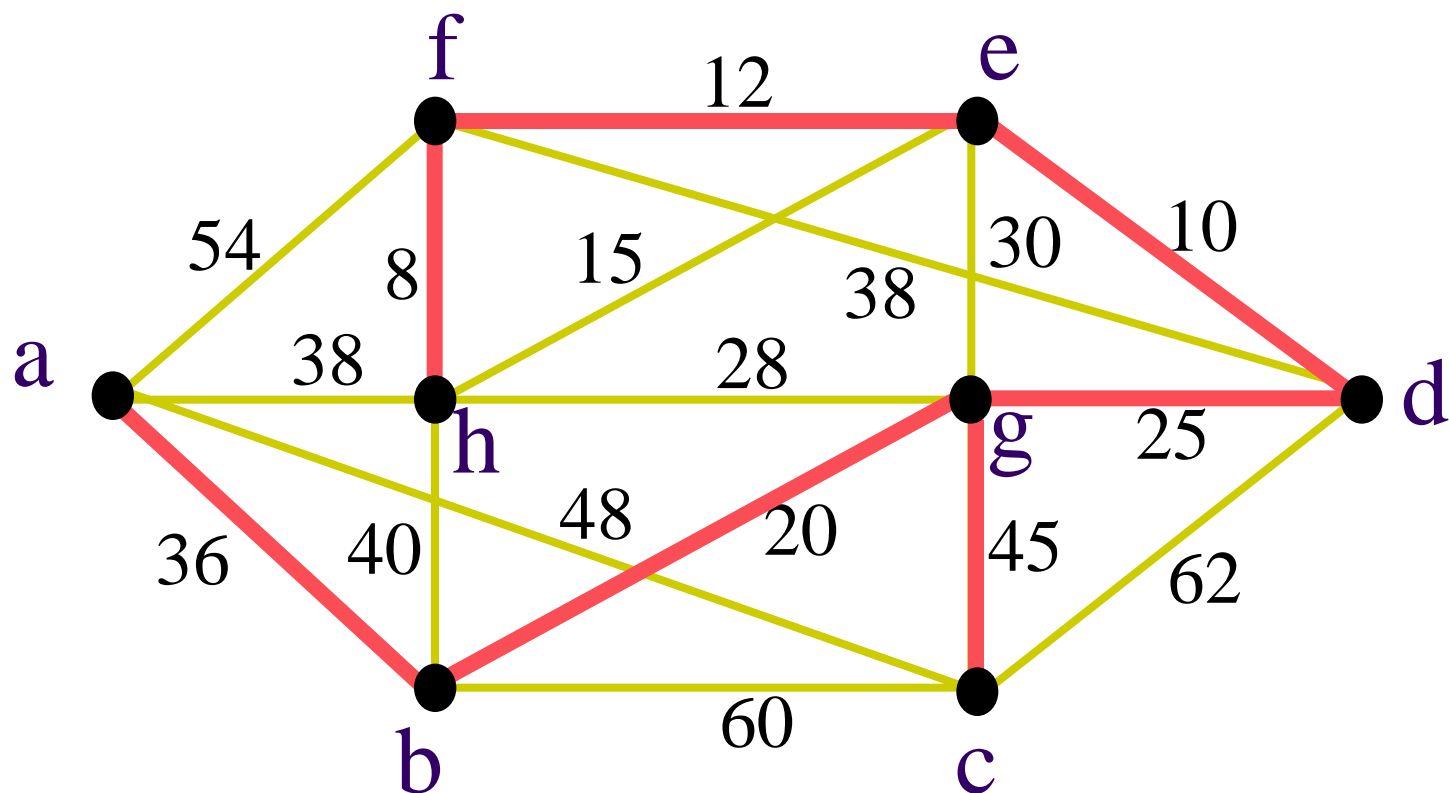
# Prim算法（求最小生成树）

- 1:  $E=\{e\}$ ,  $e$ 是权最小的边
  - 2: 从 $E$ 以外选择与 $E$ 里顶点关联, 又不会与 $E$ 中的边构成回路的权最小的边加入 $E$
  - 3: 重复第2步, 直到 $E$ 中包含 $n-1$ 条边
- 算法结束



# Prim算法（举例）

- 铺设一个连接各个城市的光纤通信网络（单位：万元）。





# Kruskal算法（求最小生成树）

1:  $E = \{ \}$

2: 从E以外选择不会与E中的边构成回路的权最小的边加入E

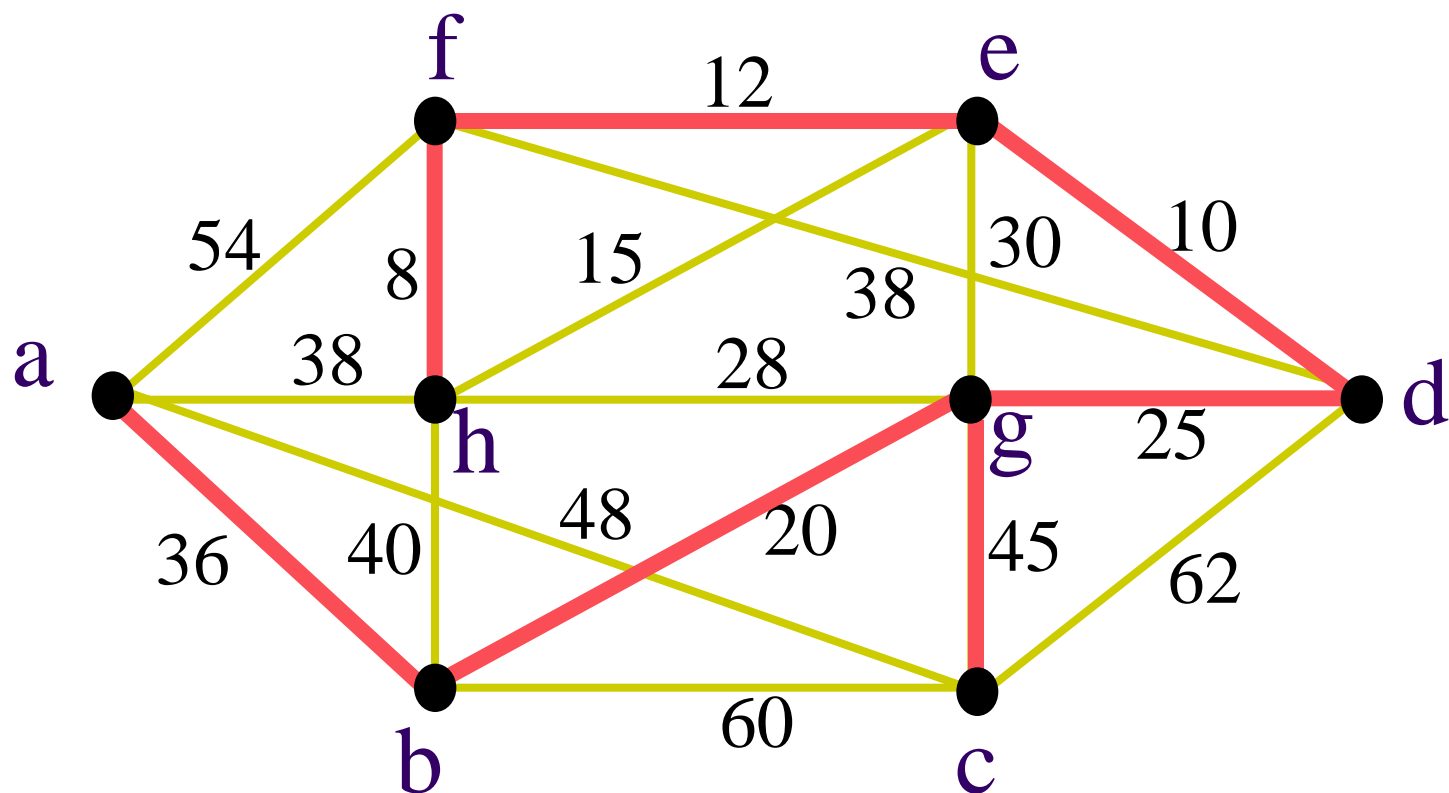
3: 重复第2步，直到E中包含n-1条边

算法结束



# Kruskal算法（举例）

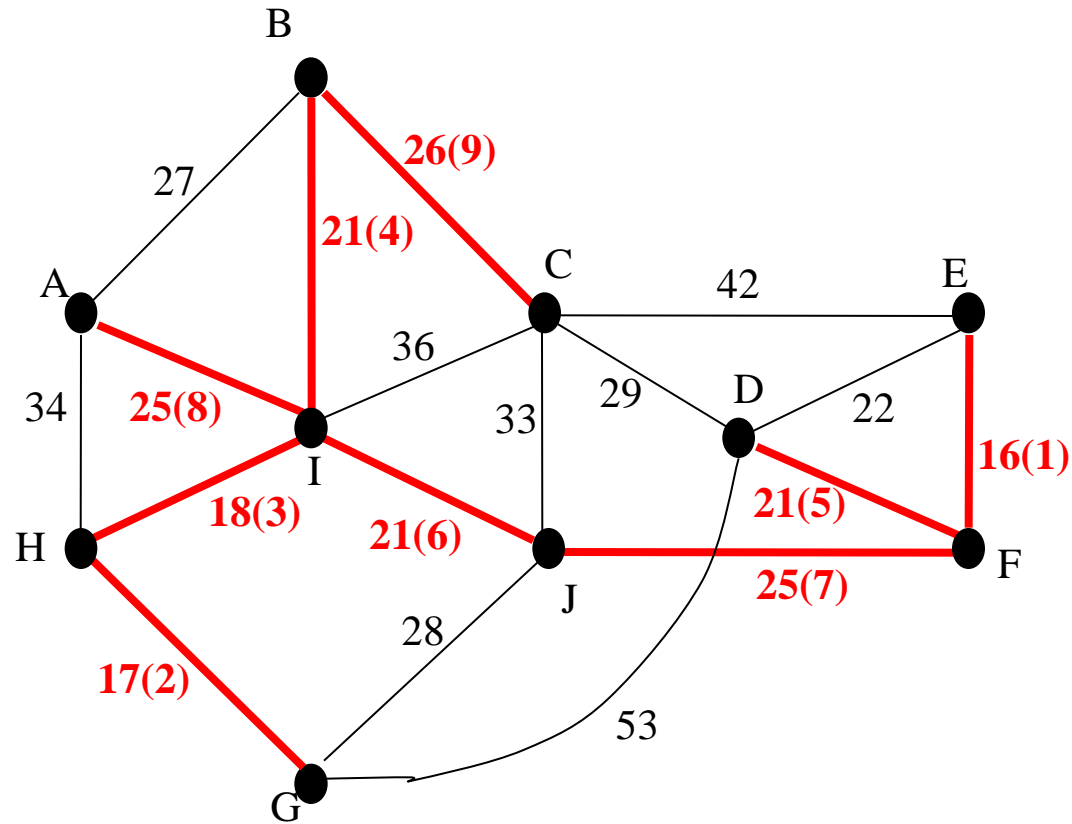
- 铺设一个连接各个城市的光纤通信网络（单位：万元）。



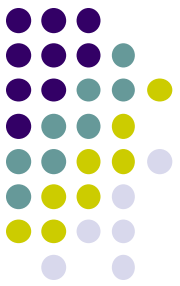




# Kruskal算法（举例）

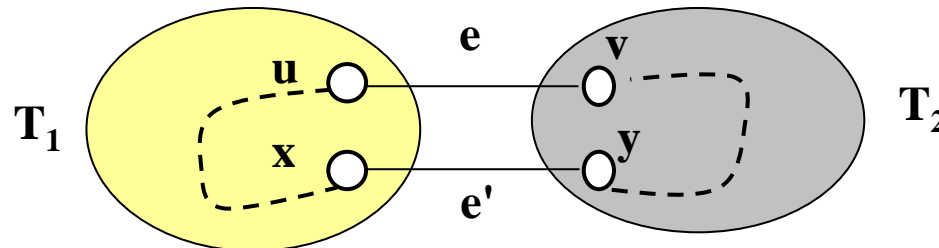


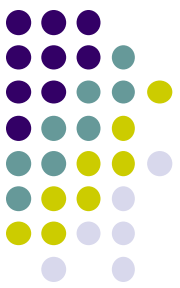
后面证明：Kruskal算法的正确性



## 引理（更换生成树的边）

- $T$ 与 $T'$ 均是图 $G$ 的生成树，若 $e \in E_T$ 且 $e \notin E_{T'}$ ，则必有 $e' \in E_{T'}$ ， $e' \notin E_T$ ，且 $T - \{e\} \cup \{e'\}$ 和 $T' - \{e'\} \cup \{e\}$ 均是 $G$ 的生成树。
- 设 $e = uv$ ， $T - \{e\}$ 必含两个连通分支，设为 $T_1$ ， $T_2$ 。因 $T'$ 是连通图， $T'$ 中有 $uv$ -通路，其中必有一边满足其两个端点 $x, y$ 分别在 $T_1, T_2$ 中，设其为 $e'$ ，显然 $T - \{e\} \cup \{e'\}$ 是生成树。  
而 $T' - \{e'\}$ 中 $x, y$ 分属两个不同的连通分支，但在 $T^* = T' - \{e'\} \cup \{e\}$ 中， **$xu$ -通路+ $e$ + $vy$ 通路**是一条 $xy$ -通路，因此 $T' - \{e'\} \cup \{e\}$ 连通，从而  $T' - \{e'\} \cup \{e\}$ 是生成树。





# Kruskal算法的正确性

- 显然 $T$ 是生成树。
- 按在算法中加边顺序,  $T$ 中边是 $e_1, e_2, \dots, e_{k-1}, e_k, \dots, e_{n-1}$ 。
- 假设 $T$ 不是最小生成树。对于任意给定的一棵最小生成树 $T'$ , 存在唯一的 $k$ , 使得 $e_k \notin E_{T'}$ , 且 $e_i \in E_{T'}$ , 使得 $(1 \leq i < k)$ . 设 $T'$ 是这样的一棵**最小生成树**, **使得上述的 $k$ 达到最大**。
- 根据前述引理,  $T'$ 中存在边 $e'$ ,  $e'$ 不属于 $T$ , 使得 $T^* = T' - \{e'\} \cup \{e_k\}$ 也是生成树。  $e' \in T'$ 与 $e_1, e_2, \dots, e_{k-1}$ 不会构成回路, 因此 $w(e') \geq w(e_k)$ . 所以 $w(T^*) \leq w(T')$ , 即 $T^*$ 也是最小生成树。但 $T^*$ 包含 $e_1, e_2, \dots, e_{k-1}, e_k$ , 矛盾。



# “避圈法”与“破圈法”

- 上述算法都是贪心地增加不构成回路的边，以求得最优树，通常称为“避圈法”；
- 从另一个角度来考虑最优树问题，在原连通带权图 $G$ 中逐步删除构成回路中权最大的边，最后剩下的无回路的子图为最优树。我们把这种方法称为“破圈法”。



# 作业

- 教材（第六版）[10.4, 10.5]
  - p.573: 5, 14, 24, 29(c), 39
  - p.580: 4, 7, 13, 21
- 教材（第七版）[11.4, 11.5]
  - p.675: 5, 14, 27(c), 34
  - p.682: 4, 7, 13, 21