生成树

离散数学–树

南京大学计算机科学与技术系

回顾

- 表达式的（逆）波兰记法
- 二叉搜索树
- 决策树
- 前缀码
- Huffman编码（算法）

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

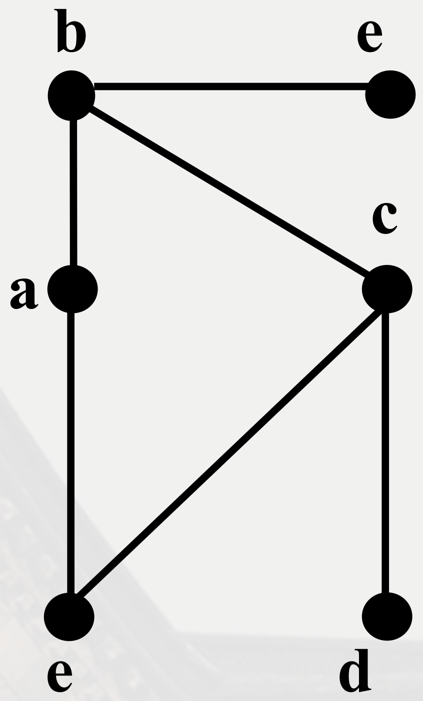
- 定义：若图G的生成子图是树，则该子图称为G的**生成树**。

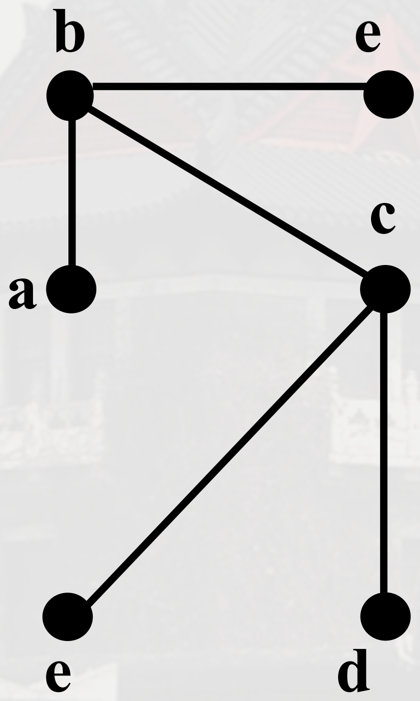
- 无向图G连通 当且仅当 G有生成树
  - 证明(充分性显然)：

⇒ 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中任一简单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, …,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 {

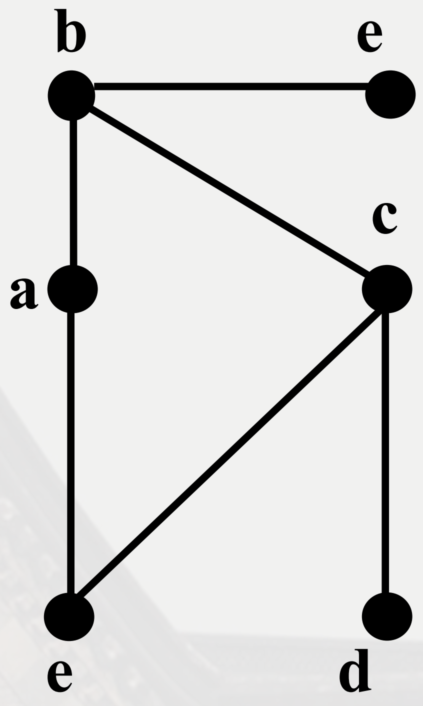
加入顶点w和边{v, w}到T;

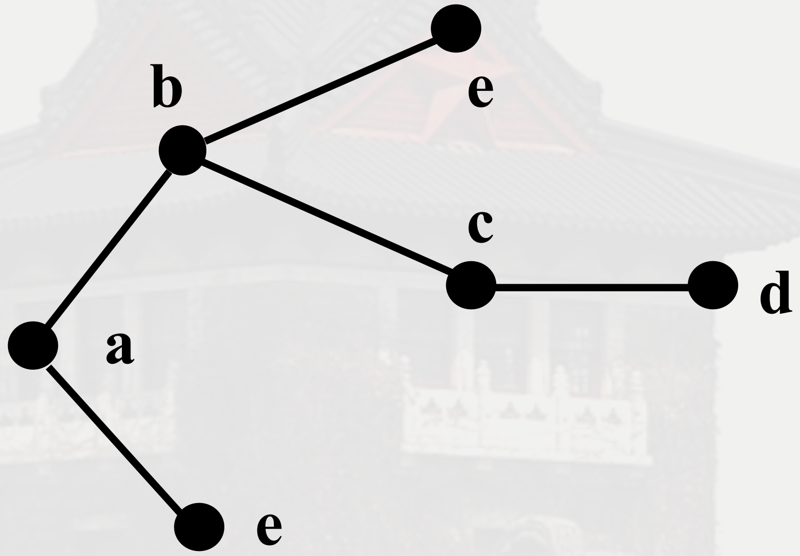
visit(w);

}

}

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v1, …,vn的连通图)
T:=只包含顶点v1的树;  L:=空表; 把v1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;
        }
    }
}
```

回溯（八皇后）

- 在n×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

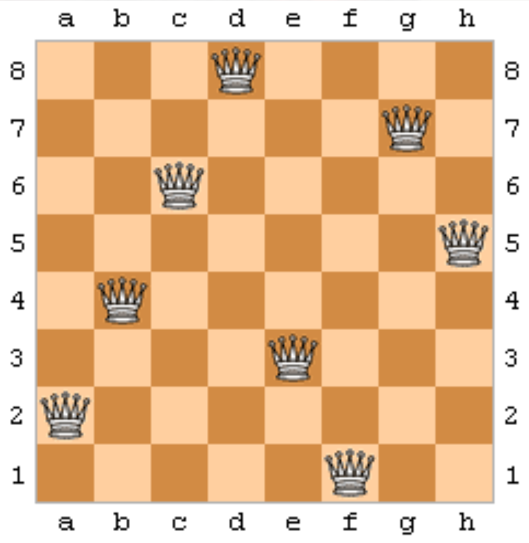
尝试第1列，第1行，…n行；

尝试第2列，第1行，…n行；

….

尝试第k+1列，第1行，…n行；

…



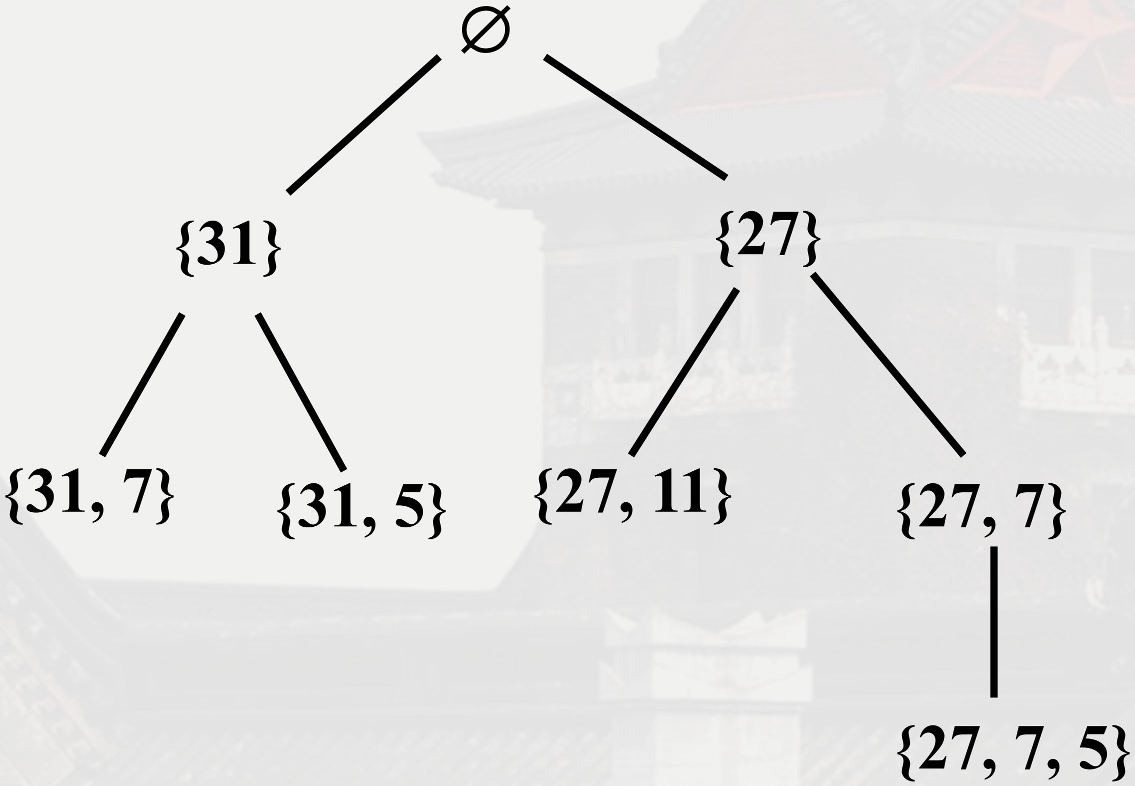
回溯（子集和）

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

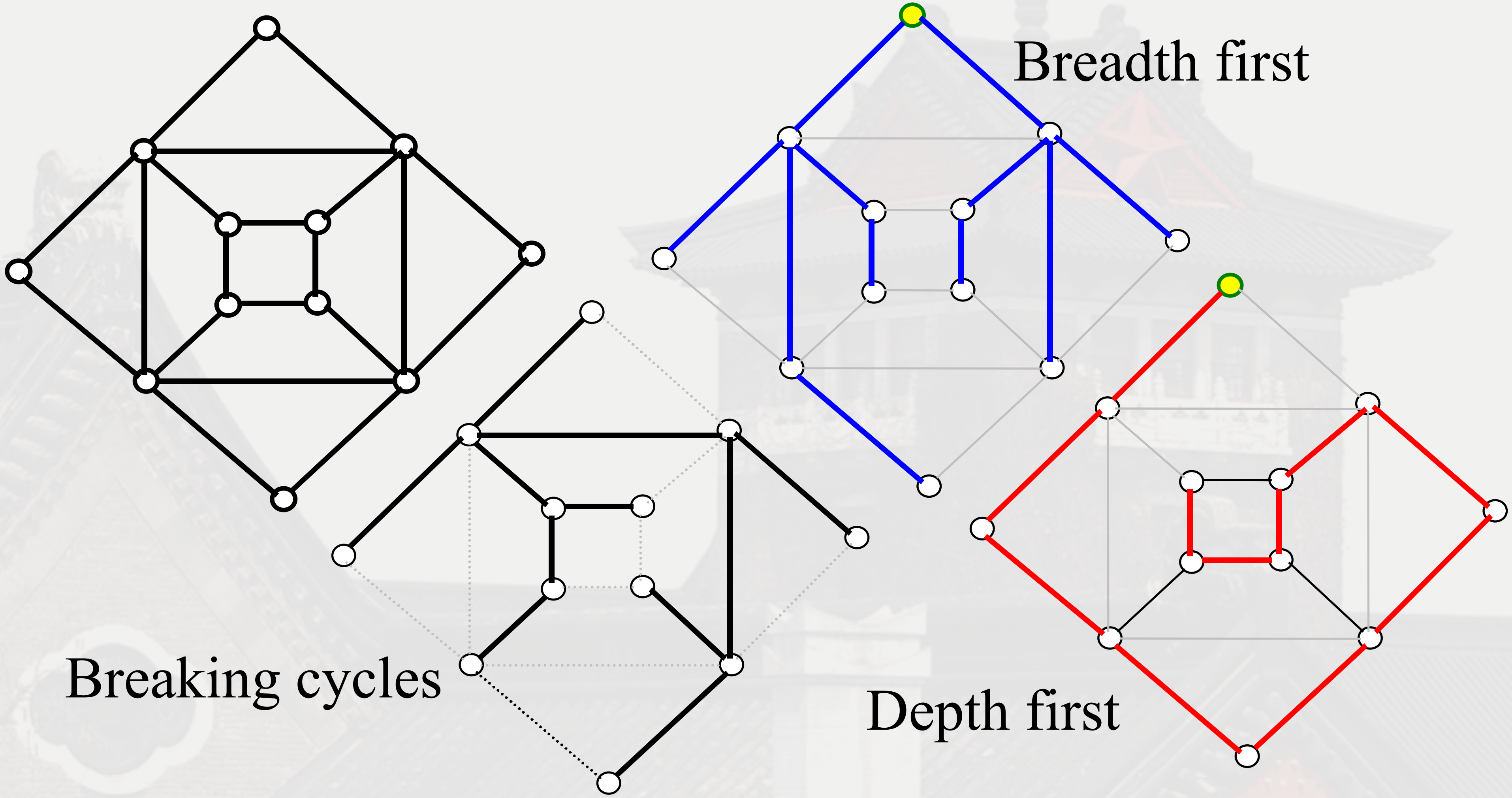
从空子集开始
尝试添加一项，
　和等于M，结束；
　和不超过M，子集包含它；
没有合适添加项，去掉和的最后一项，

回溯（子集和）

- 举例：{31, 27, 15, 11, 7, 5}, 和为39的子集？



生成树：举例



最小生成树 MST （Minimum Spanning Tree）

- 考虑边有权重的连通无向图。其生成树可能不唯一。定义生成树的权重为其所含各边之和。一个带权连通图的**最小生成树**是其权重最小的生成树。
  - 注意，这里的最小并不意味着唯一(Minimum)。

- 最小生成树有广泛的应用。

Prim算法（求最小生成树）

1: E={e}, e是权最小的边

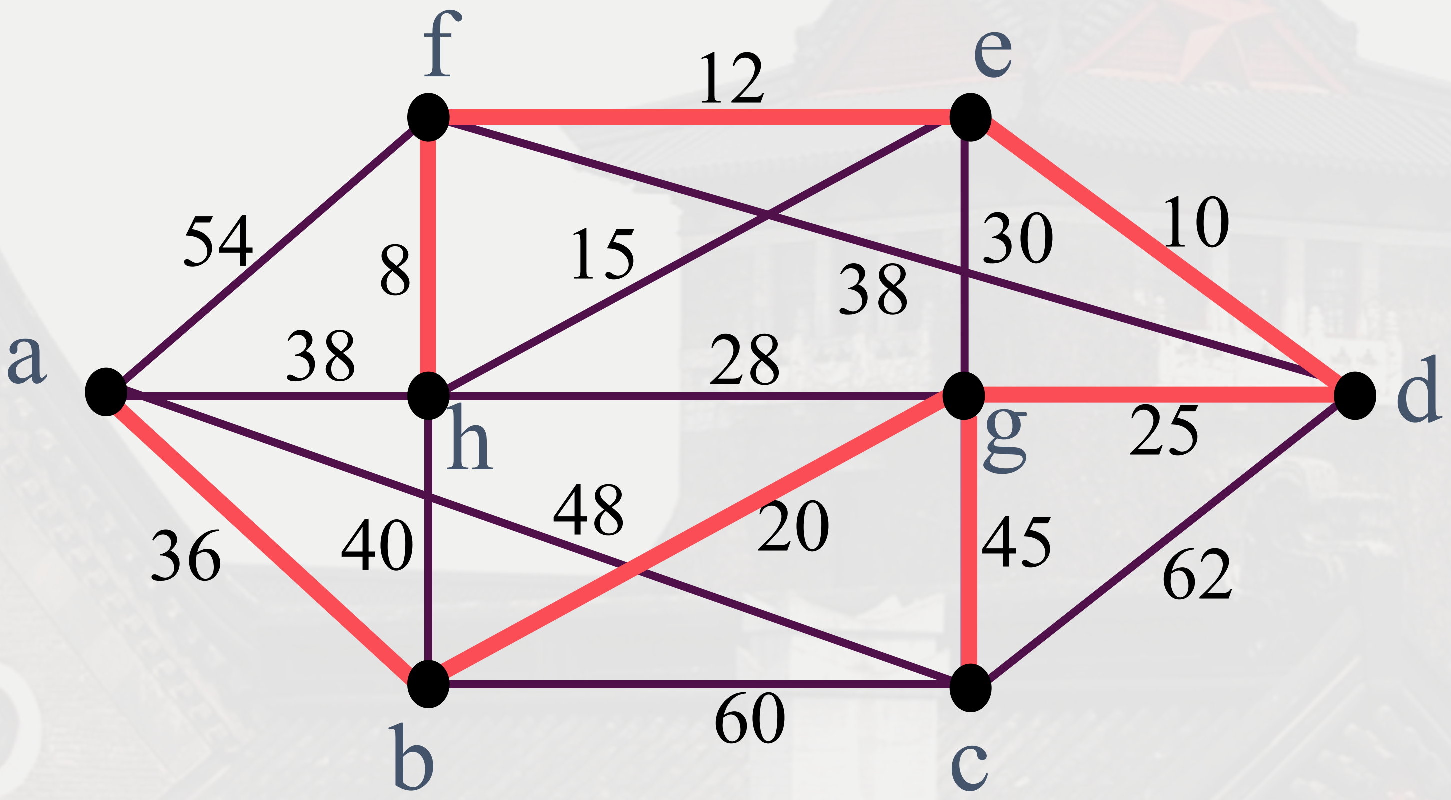
2: 从E以外选择与E里顶点关联，又不会与E中的边构成回路的权最小的边加入E

3: 重复第2步，直到E中包含n-1条边

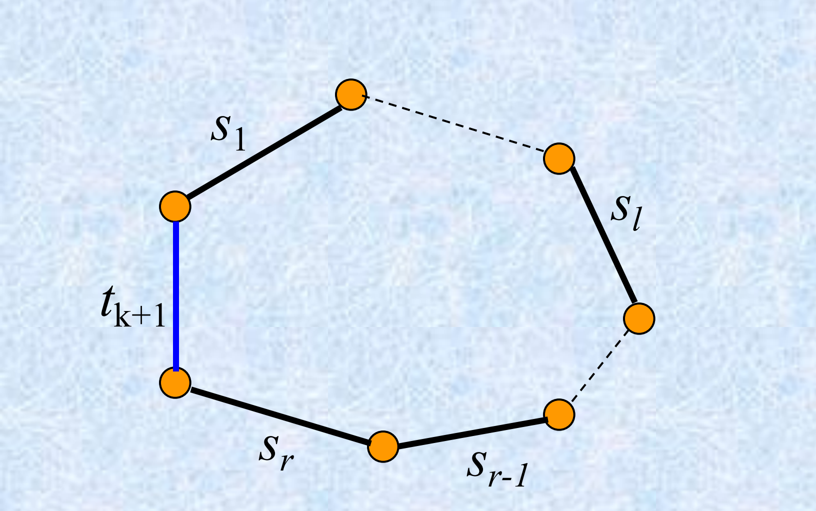
算法结束

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）。



Prim 算法的正确性



Let T be the output of Prim's alg

edges $t_1, t_2, \ldots, t_{n-1}$, as the order the

for $1 \le i \le n-1$, and $T_0 = \phi$.

It can be proved that each T_i is co

Assume that T_k is contained in a MST T', then $\{t_1, t_2, \ldots, t_k\} \subseteq T'$.

If $t_{k+1} \notin T'$, then $T' \cup \{t_{k+1}\}$ contains a cycle, which cannot wholly be in T_k.

Let s_l be the edge with smallest index l that is not in T_k. Exactly one of the vertices of s_l must be in T_k, which means that when t_{k+1} was chosen, s_l available as well. So, t_{k+1} has no larger weight than s_l. So, $(T' - \{s_l\}) \cup \{t_{k+1}\}$ is a MST containing T_{k+1}.

Kruskal算法（求最小生成树）

1: E={ }

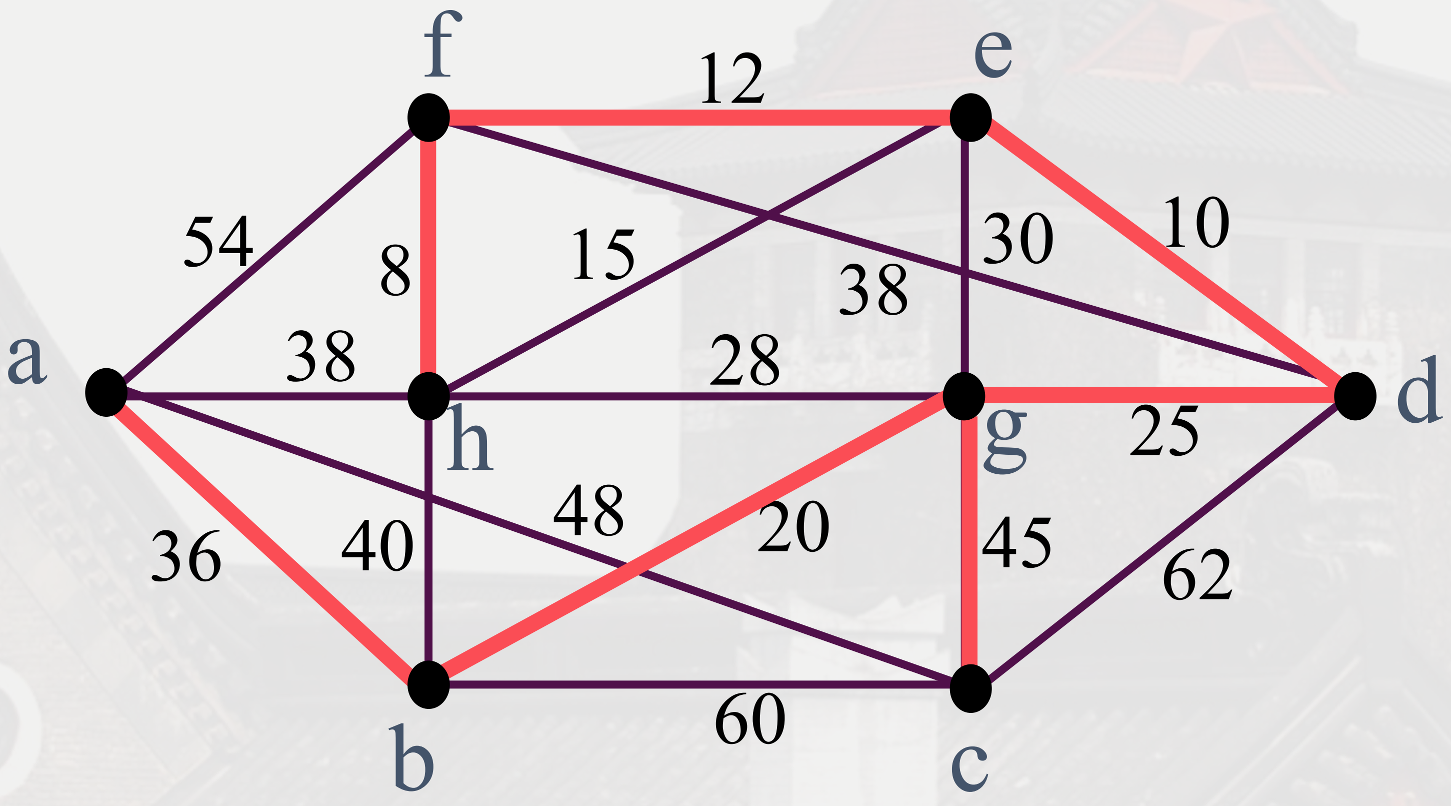
2: 从E以外选择不会与E中的边构成回路的权最小的边加入E

3: 重复第2步，直到E中包含n-1条边

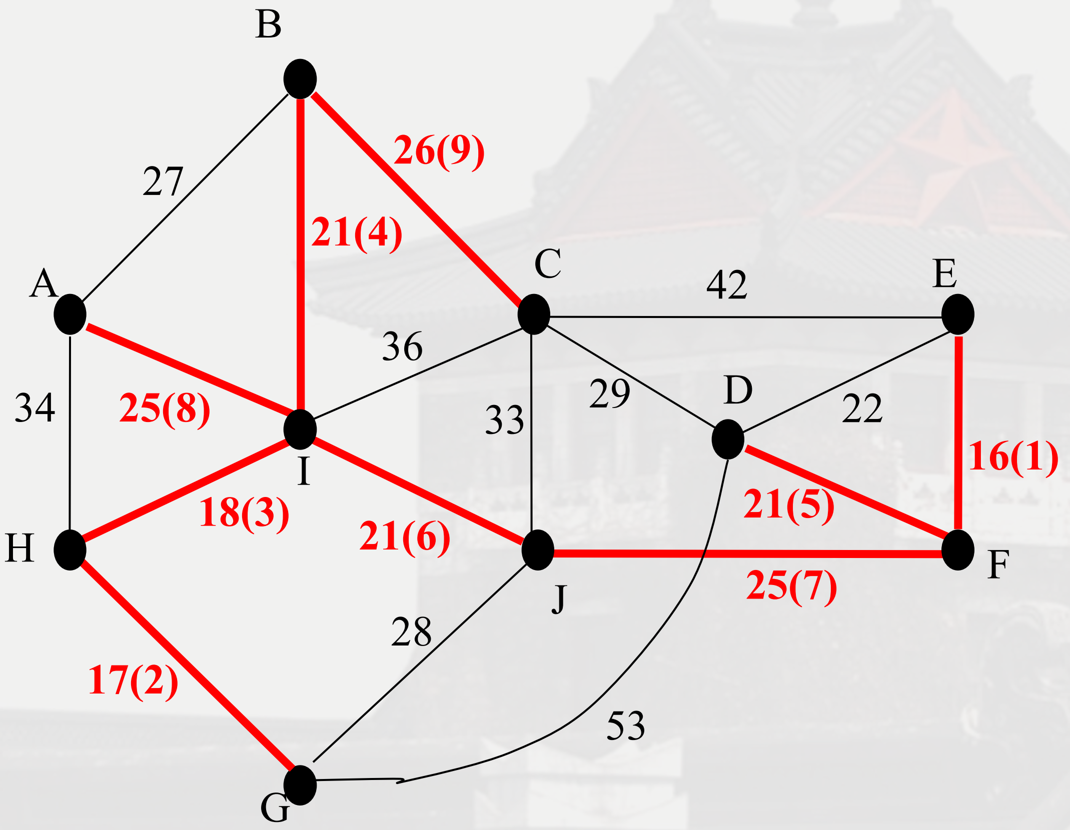
算法结束

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）

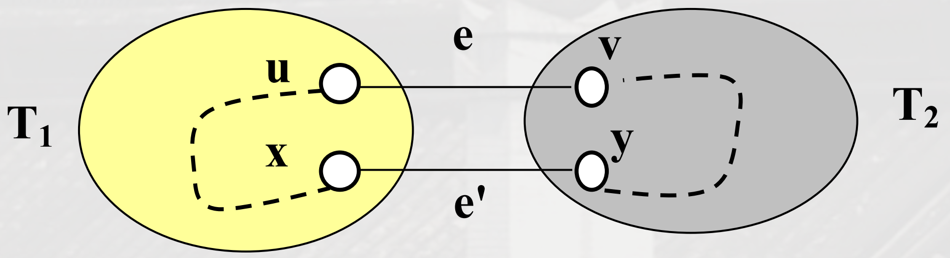


后面证明：**Kruskal算法的正确性**

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$e_1, e_2, \ldots e_{k-1}$, $e_k, \ldots e_{n-1}$。
- 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$e_i \in E_{T'}$ $(1 \le i < k)$. 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$e_1, e_2, \ldots e_{k-1}$不会构成回路，因此$w(e') \ge w(e_k)$. 所以$w(T^*) \le w(T')$, 即T*也是最小生成树。但T*包含$e_1, e_2, \ldots e_{k-1}, e_k$，矛盾。

Generic Algorithm for MST Problem

Input: G: a connected, undirected graph

w: a function from V_G to the set of real number

Generic-MST(G,w)
1 $A \leftarrow 0$
2 **while** A does not form a spanning tree
3 **do** find an edge (u,v) that is safe for A
4 $A \leftarrow A \cup \{(u,v)\}$
5 **return** A

Output: a minimal spanning tree of G

“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”。

小结

- 生成树
- 图上的搜索与回溯
- 求最小生成树的算法
  - Prim算法
  - Kruskal算法