# 最小生成树

王丽杰

Email: ljwang@uestc.edu.cn

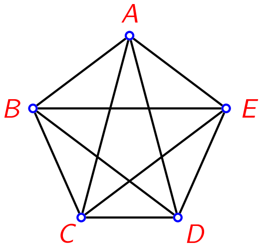
电子科技大学 计算机学院

2016-

# 引子



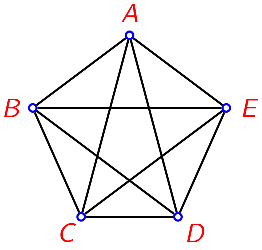
回顾生成树的学习中提到的构建一个包含 5 个信息中心 A,B,C,D,E 的通信系统的问题，如下左图所示。通常情况下，各中心之间的光纤连接长度并不相同，这会影响总体费用。所以我们建立一个带权图 (以百公里为单位，如右图所示)，希望能从这个图中找出一棵生成树，而且总权值最小。

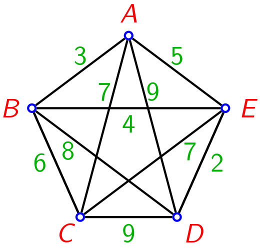


# 引子



回顾生成树的学习中提到的构建一个包含 5 个信息中心 A,B,C,D,E 的通信系统的问题，如下左图所示。通常情况下，各中心之间的光纤连接长度并不相同，这会影响总体费用。所以我们建立一个带权图 (以百公里为单位，如右图所示)，希望能从这个图中找出一棵生成树，而且总权值最小。



# 无向树



**Definition**

设 $G=<V,E>$ 是连通的赋权图，$T$ 是 $G$ 的一棵生成树，$T$ 的每个树枝所赋权值之和称为 $T$ 的权，记为 $w(T)$。$G$ 中具有最小权的生成树称为 $G$ 的最小生成树(minimal spanning tree)。

# 无向树

### Definition

设 $G = < V, E >$ 是连通的赋权图，$T$ 是 $G$ 的一棵生成树，$T$ 的每个树枝所赋权值之和称为 $T$ 的权，记为 $w(T)$。$G$ 中具有最小权的生成树称为 $G$ 的最小生成树(minimal spanning tree)。

一个无向图的生成树不是惟一的，同样地，一个赋权图的最小生成树也不一定是惟一的。求赋权图的最小生成树的方法很多，这里主要介绍 Kruskal 算法和 Prim 算法。

# Kruskal 算法

Kruskal 算法是克鲁斯克尔 (Kruskal) 于 1956 年将构造生成树的避圈法推广到求最小生成树，其要点是，在与已选取的边不构成回路的边中选取最小者。

**Kruskal 算法**

1. 在 $G$ 中选取最小权边 $e_1$，置 $i = 1$，$E_T = \{e_1\}$。

# Kruskal 算法



Kruskal 算法是克鲁斯克尔 (Kruskal) 于 1956 年将构造生成树的避圈法推广到求最小生成树，其要点是，在与已选取的边不构成回路的边中选取最小者。

**Kruskal 算法**

1. 在 $G$ 中选取最小权边 $e_1$，置 $i=1$，$E_T=\{e_1\}$。
2. 当 $i=n-1$ 时，结束，否则转 (3)。

# Kruskal 算法

Kruskal 算法是克鲁斯克尔 (Kruskal) 于 1956 年将构造生成树的避圈法推广到求最小生成树，其要点是，在与已选取的边不构成回路的边中选取最小者。

**Kruskal 算法**

1. 在 $G$ 中选取最小权边 $e_1$，置 $i = 1$，$E_T = \{e_1\}$。
2. 当 $i = n - 1$ 时，结束，否则转 (3)。
3. 在 $G$ 中选取不在 $E_T$ 中的边 $e_{i+1}$，使 $E_T \cup \{e_{i+1}\}$ 中无回路且 $e_{i+1}$ 是满足此条件的最小权边。

# Kruskal 算法

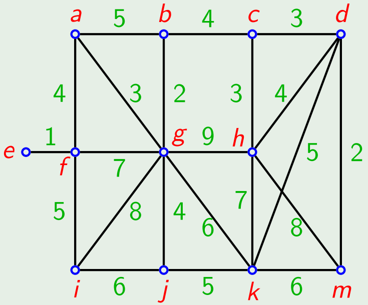
Kruskal 算法是克鲁斯克尔 (Kruskal) 于 1956 年将构造生成树的避圈法推广到求最小生成树，其要点是，在与已选取的边不构成回路的边中选取最小者。

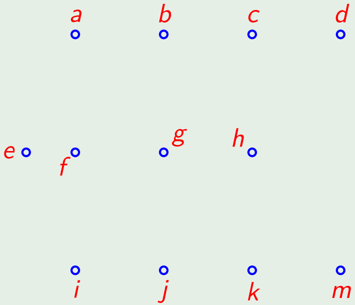
**Kruskal 算法**

1. 在 $G$ 中选取最小权边 $e_1$，置 $i=1$，$E_T=\{e_1\}$。
2. 当 $i=n-1$ 时，结束，否则转 (3)。
3. 在 $G$ 中选取不在 $E_T$ 中的边 $e_{i+1}$，使 $E_T\cup\{e_{i+1}\}$ 中无回路且 $e_{i+1}$ 是满足此条件的最小权边。
4. 置 $i=i+1$，$E_T=E_T\cup\{e_{i+1}\}$，转 (2)。

# Kruskal 算法

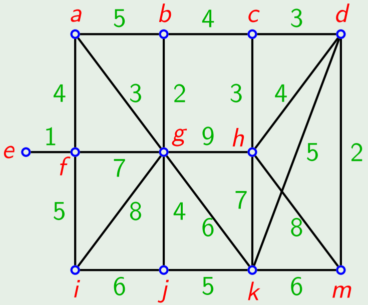
## Example

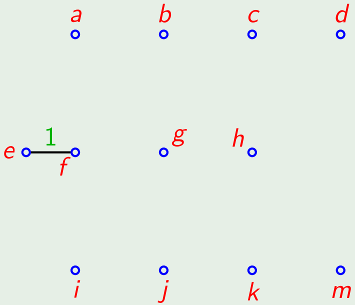


# Kruskal 算法

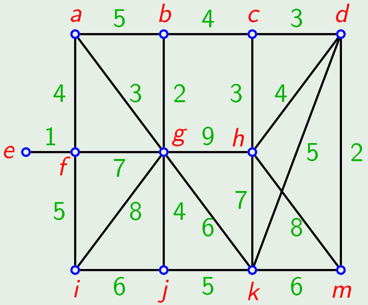
## Example

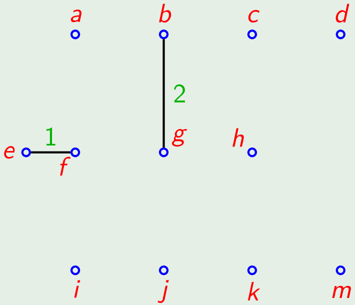


# Kruskal 算法

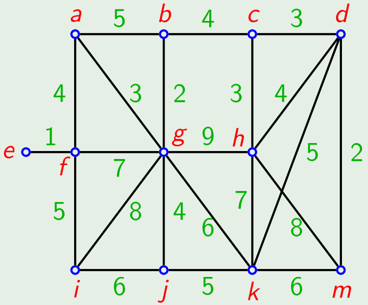
## Example

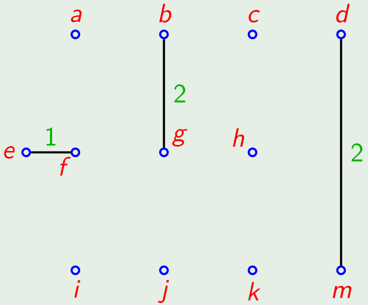


# Kruskal 算法

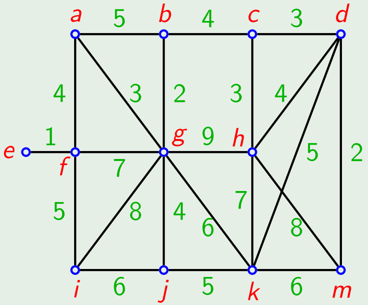
## Example

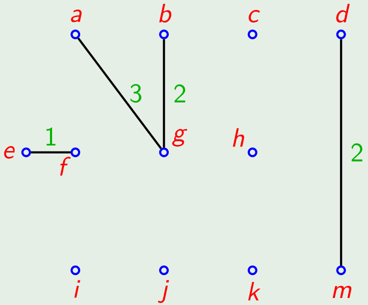


# Kruskal 算法

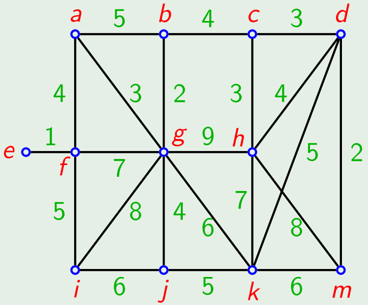
## Example

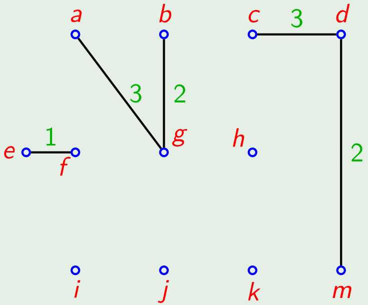


# Kruskal 算法

## Example

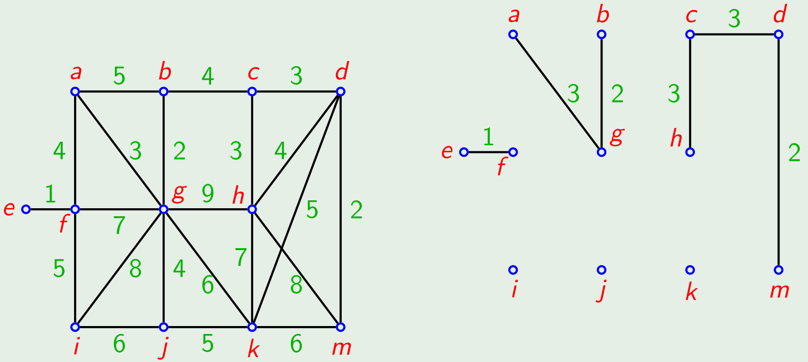


# Kruskal 算法



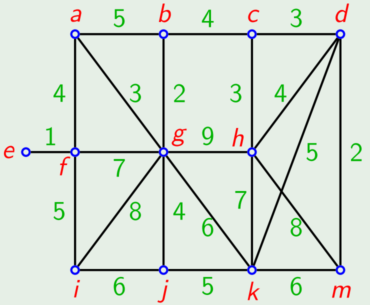
## Example

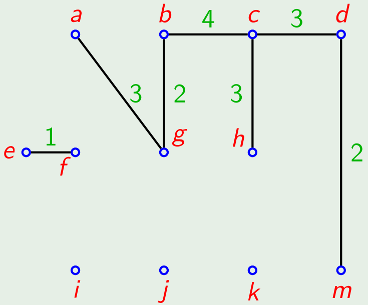


# Kruskal 算法



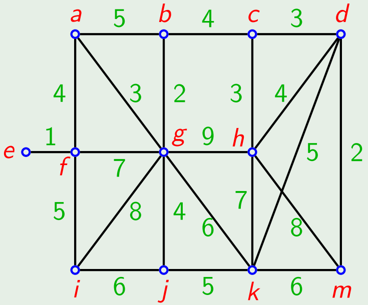
## Example

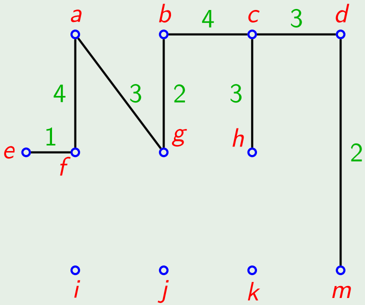




# Kruskal 算法

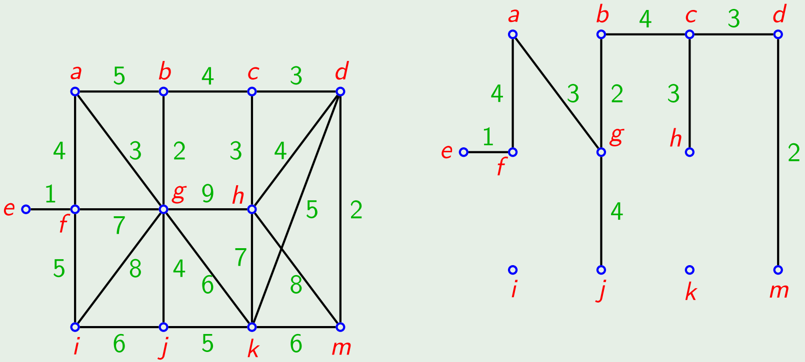
## Example



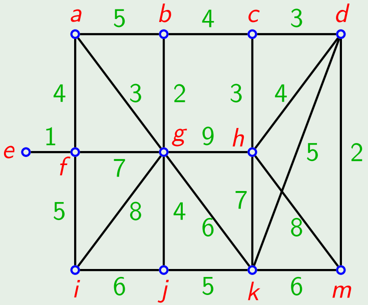
# Kruskal 算法

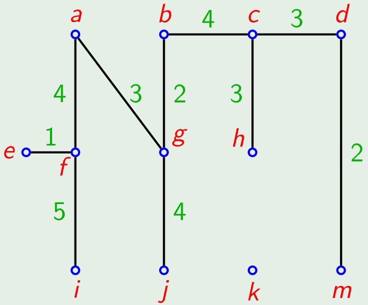
## Example



# Kruskal 算法

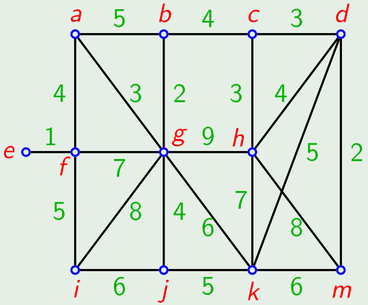
## Example

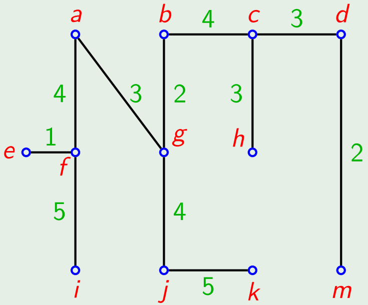


# Kruskal 算法

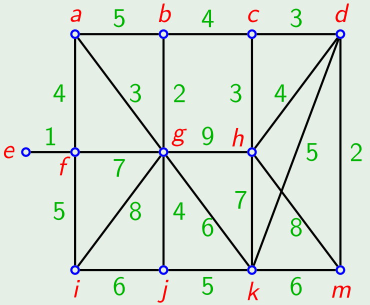
## Example

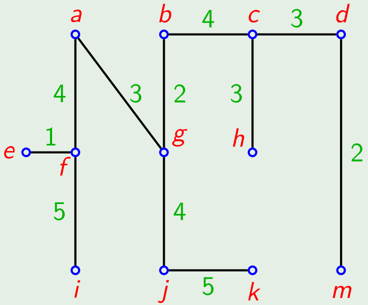


# Kruskal 算法

## Example




$w(T) = 36$

# Prim 算法

Prim 算法的要点是，从任意结点开始，每次增加一条最小权边构成一棵新树。

**Prim 算法**

1. 在 $G$ 中任意选取一个结点 $v_1$，置 $V_T = \{v_1\}, E_T = \varnothing$，$k = 1$；

# Prim 算法

Prim 算法的要点是，从任意结点开始，每次增加一条最小权边构成一棵新树。

**Prim 算法**

1. 在 $G$ 中任意选取一个结点 $v_1$，置 $V_T = \{v_1\}, E_T = \varnothing$，$k = 1$；
2. 在 $V - V_T$ 中选取与某个 $v_i \in V_T$ 邻接的结点 $v_j$，使得边 $(v_i, v_j)$ 的权最小，置 $V_T = V_T \cup \{v_j\}, E_T = E_T \cup \{(v_i, v_j)\}$，$k = k + 1$；

# Prim 算法

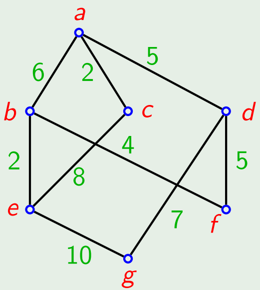
Prim 算法的要点是，从任意结点开始，每次增加一条最小权边构成一棵新树。

**Prim 算法**

1. 在 $G$ 中任意选取一个结点 $v_1$，置 $V_T = \{v_1\}$, $E_T = \varnothing$，$k = 1$；
2. 在 $V - V_T$ 中选取与某个 $v_i \in V_T$ 邻接的结点 $v_j$，使得边 $(v_i, v_j)$ 的权最小，置 $V_T = V_T \cup \{v_j\}$, $E_T = E_T \cup \{(v_i, v_j)\}$，$k = k + 1$；
3. 重复步骤 2，直到 $k = |V|$。

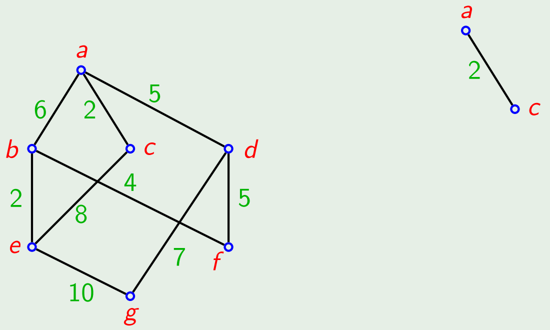
# Prim 算法

## Example



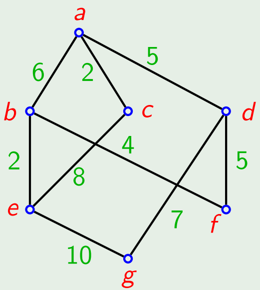

# Prim 算法

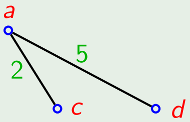
## Example



# Prim 算法

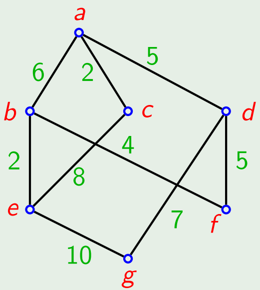
## Example

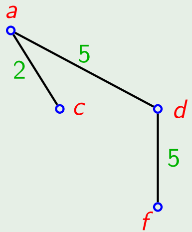


# Prim 算法

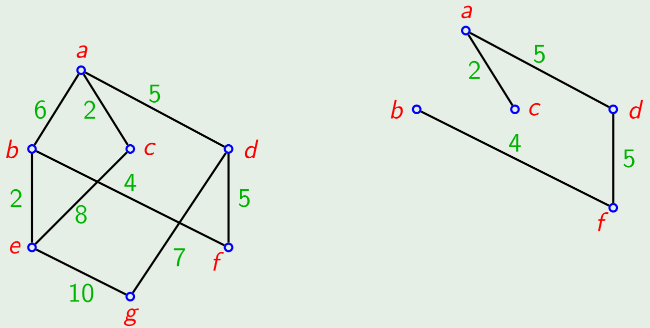
## Example



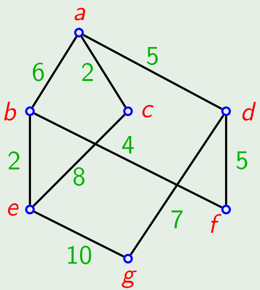
# Prim 算法

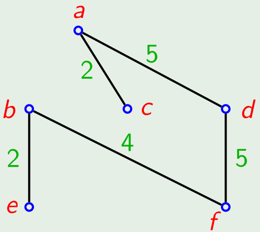
## Example



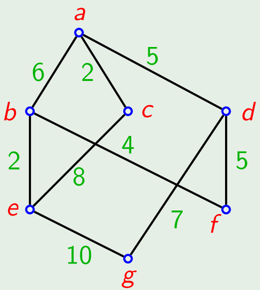
# Prim 算法

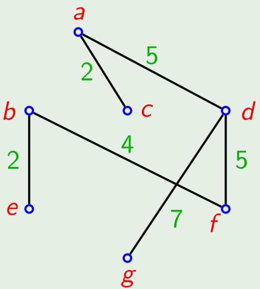
## Example



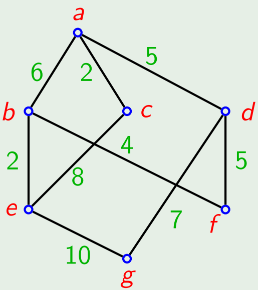
## Example

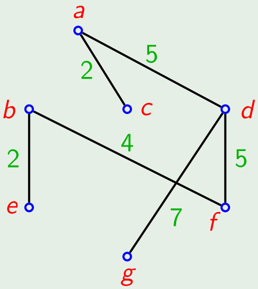


# Prim 算法

## Example




$w(T) = 25$

THE END, THANKS!