计算无向图中的最小生成树

章乐

网络空间安全系
北京电子科技学院

目录

最小生成树问题的描述

给定一个*连通的*无向图 ***G=(V,E)***，其的边权值函数为 ***w***，即 ***w(e)*** 是边 ***e*** 的权值。

考虑 ***G*** 的子图 ***T*** **(** 其的边集是 ***E*** 的子集 **)**，其需要满足：

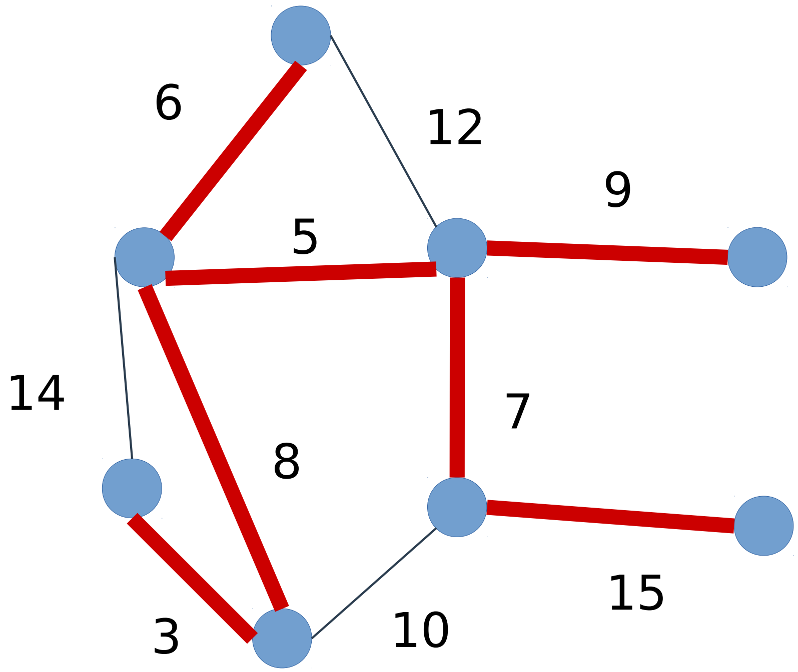
1. ***T*** 是一棵*树*；

2. ***T*** 是所有这样的树中权值*最小的*。

（树的权值是其的各条边的权值的总和。）

3. **“** 生成”，即 **span** 所有的顶点。

最小生成树， ***Minimum Spanning Tree***，也就是 ***MST*** 问题。



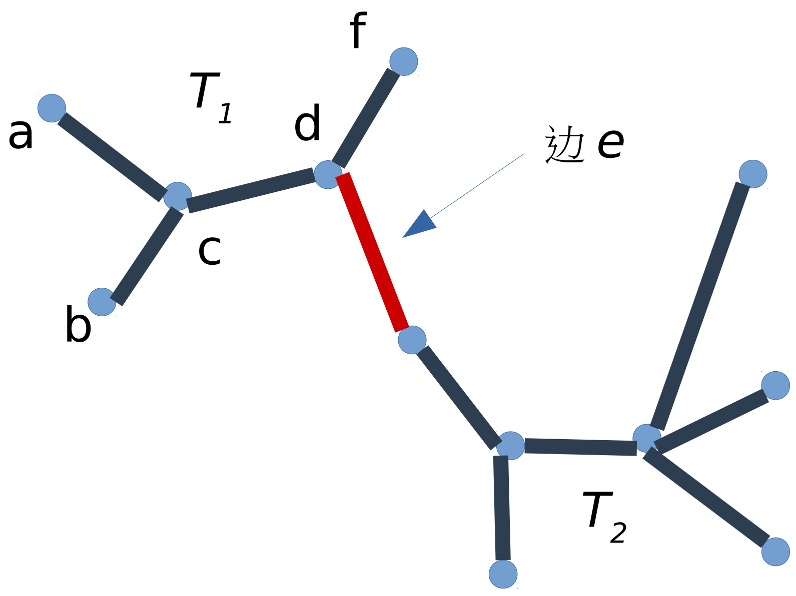
最小生成树问题的描述

对 **MST** 问题的观察：

1. 假设我们知道 **e** 在 **T** 中，

那么， T_1, T_2 分别是相应的*子图*的 **MST.**

$$w(T)=w(T_1)+w(T_2)+w(e)$$

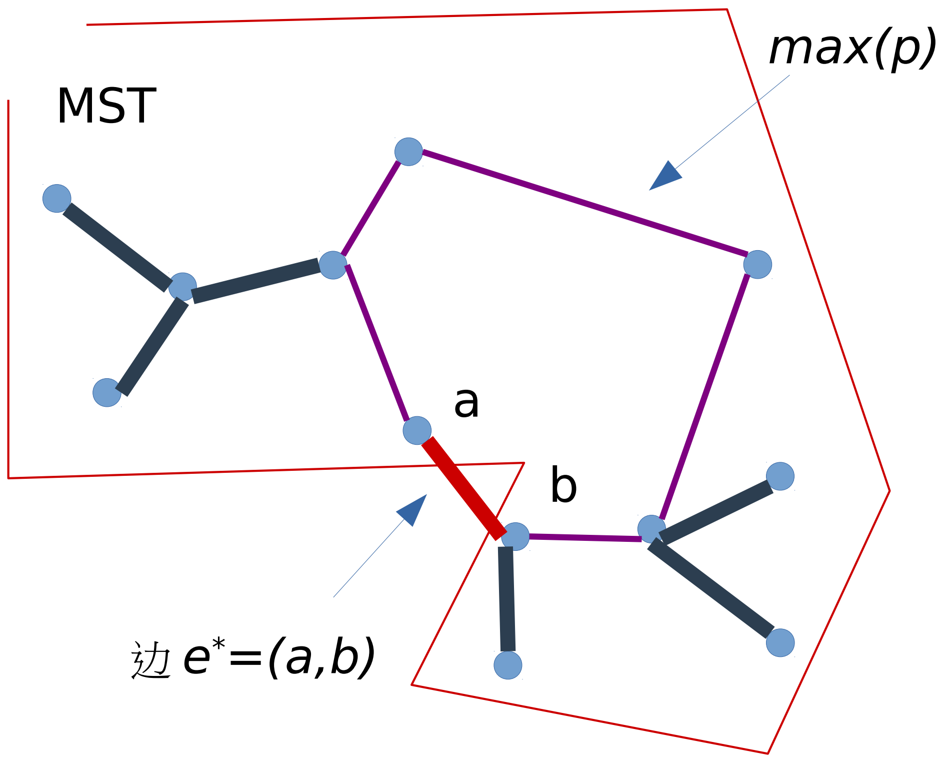


2. 怎样确定边 ***e*?**

我们先来考虑 ***G*** 中权值最小的边 **e***.

最小生成树问题的描述

考虑图 ***G*** 中权值最小的边 ***e****，
这条边一定在 ***MST*** 中吗？



1. *MST* 中有从 *a* 到 *b* 的路径 *p*.

2. *p* 中权值最大的边为 *max(p)*.

3. 构造新的 *MST'=T+{e*}-max(p)*.

观察结论： ***w(MST')*** *不大于* ***w(MST)*.**

所以， ***e**** 一定在 ***MST*** 中。

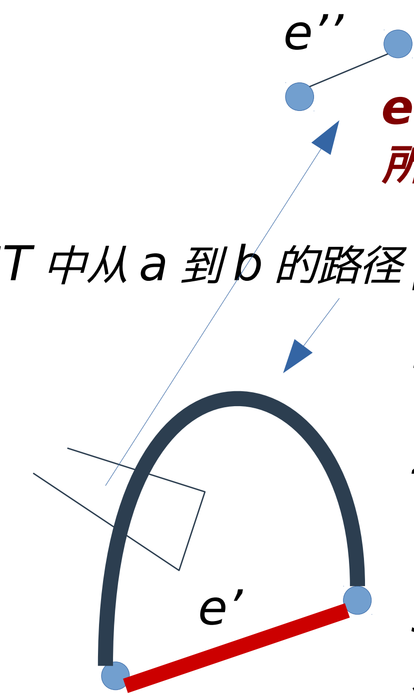
Kruskal 算法

我们确定 ***e**** 在 ***MST*** 中，那么，***MST*** 中的权值第 **2** 小的边是哪一条呢？

更一般的，如果我们确定了 ***MST*** 的边集的子集 ***A*，**那么，如何扩大 ***A*** 直到 ***A*** 包括 ***MST*** 中所有的边？

Kruskal 算法：

选择 ***E-A*** 中，权值最小的，且加入 ***A*** 后*不会带来回路*的边 ***e'*.**



e'' 来自 E-A,
所以， w(e') 不大于 w(e'').

MST 中从 a 到 b 的路径 p

1. 路径 p 上有些边来自 A，有些边来自 E-A.
2. 因为 e' 加入 A 后不带来回路，所以， p 上的边不可能都来自 A.
3. 考虑 p 上的来自 E-A 的边的集合，我们选择其中任何一条 e''，构造 MST'=MST-{e''}+{e'}.

Kruskal 算法

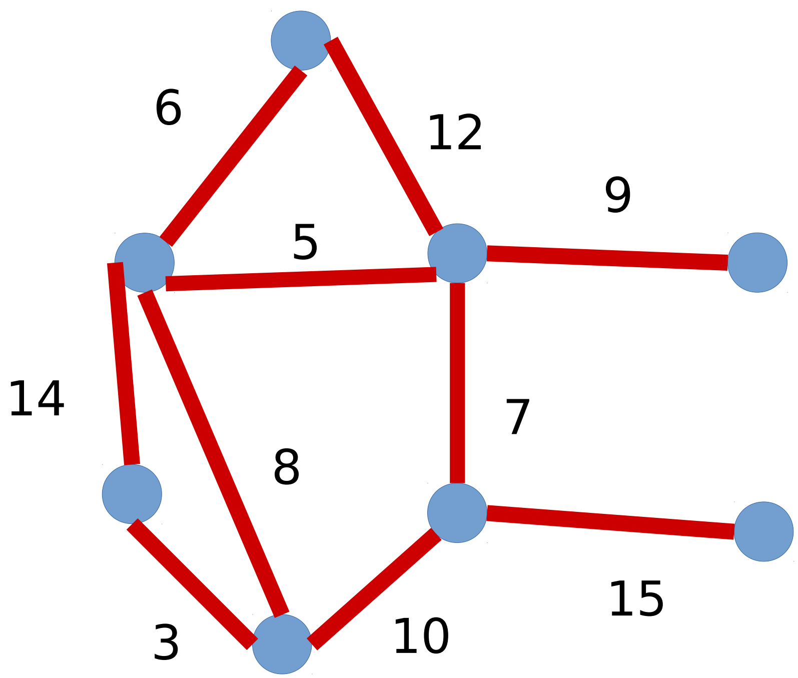
MST-Kruskal

***1. A** 初始化为空集。*

***2.** 对于 **G** 中所有的边，按权值递增顺序考虑：*

如果 A+{e} 没有回路，那么 A=A+{e}.

3.** 返回 **A.



*时间复杂度：**O(m log n)**，主要来自对所有的边的排序。*

~~*其中，判断是否有回路，可以利用并查集，一次判断的时间为 **alpha(n)=O(log n).***~~

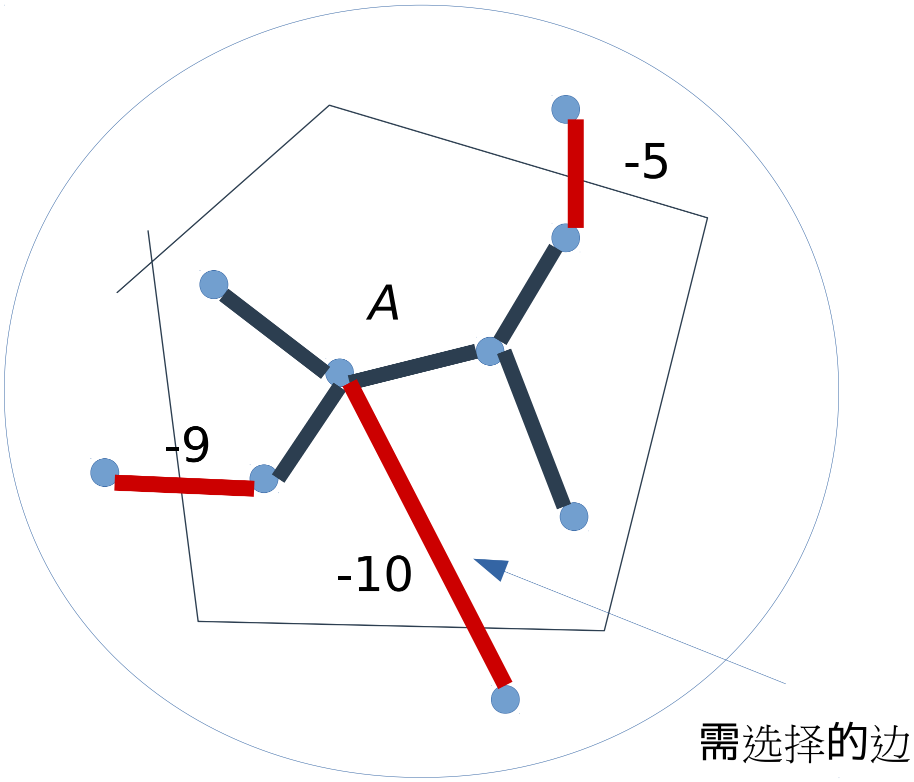
Prim 算法

之前的维护的，不断扩大的边集 ***A*** 带有一些任意性。

我们考虑，让 ***A*** 始终是 ***MST*** 的一个*连通的*子树。

Prim 算法：

选择将 ***A*** 所构成的子树和子树外的顶点相连的，权值最小的边。



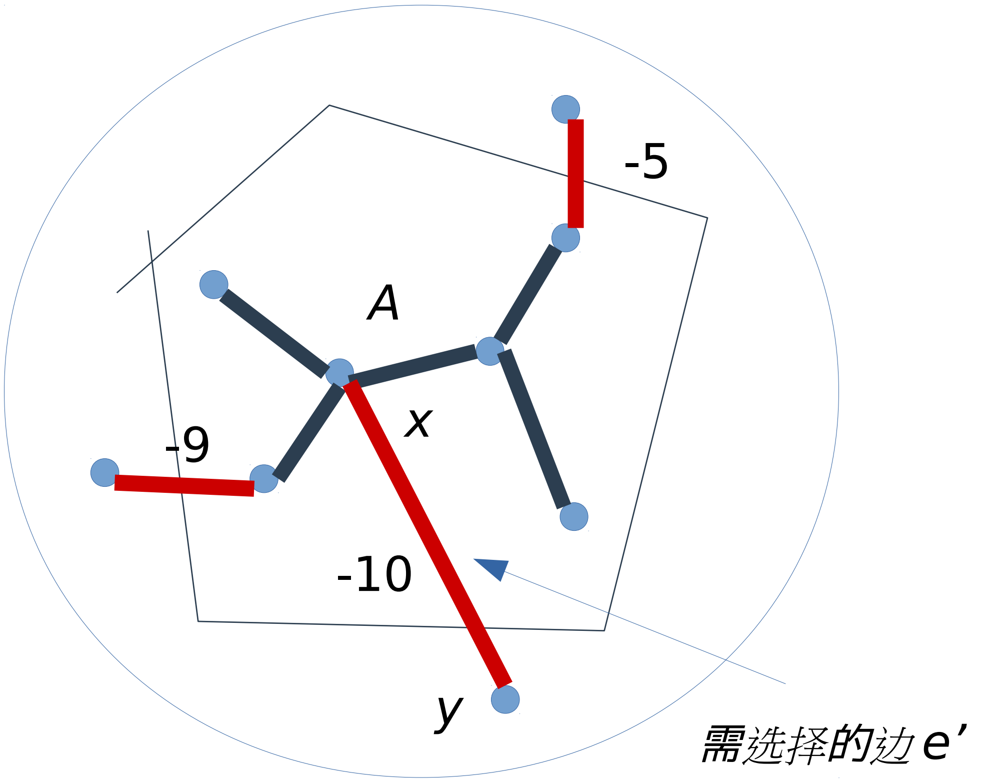
Prim 算法

正确性的证明：

如果 Prim 算法选择的边 e' 不在 MST 中，那么因为 e' 的两端分别在 A 和 MST-A 中，考虑 MST 中从 x 到 y 的路径 p.

路径 p 上第一条跨越 A 和 MST-A 的边 e. 我们有 w(e') 的权值不大于 w(e).

w(MST')=w(MST)-w(e)+w(e').



Prim 算法

MST-Prim

1. A *初始化为空集。*

2. *当* ***A*** *尚未连接所有顶点时：*

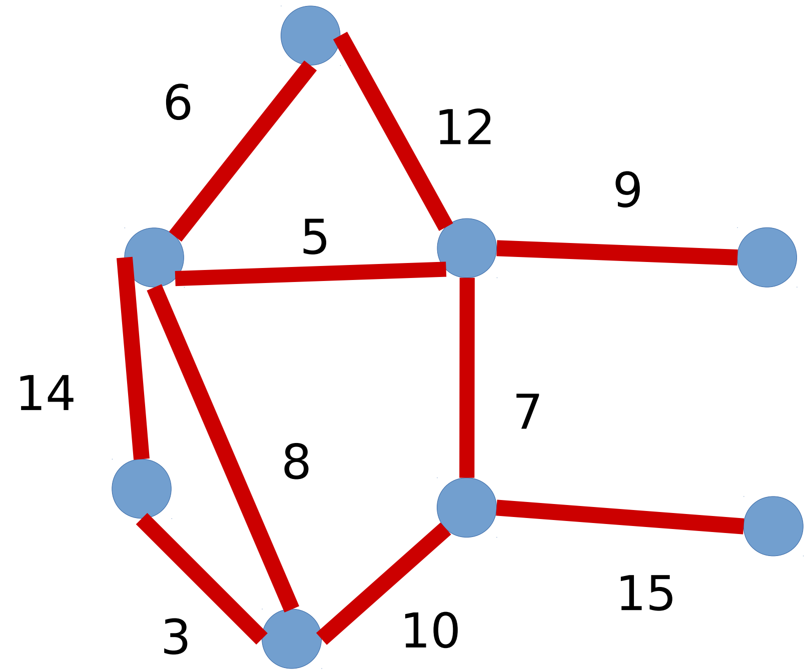
选择连接A 和V-A 的边中，权值最小的边e’，执行A=A+{e’}.

（A 理解为因为 A 而连通的顶点集，另外，A+{e’} 后A 仍然保持为连通的。）

3. *返回* ***A.***

时间复杂度： ***O(m+nlog n).***

每~~次寻找~~ ***~~e’~~*** *~~的平摊~~时间为* ***O(log n).***



Any question?

Applications:

Electronic circuit designs often need to make the *pins* of several components *electrically equivalent* by wiring them together.