

本节内容

最小生成树

王道考研/CSKAOYAN.COM

知识总览



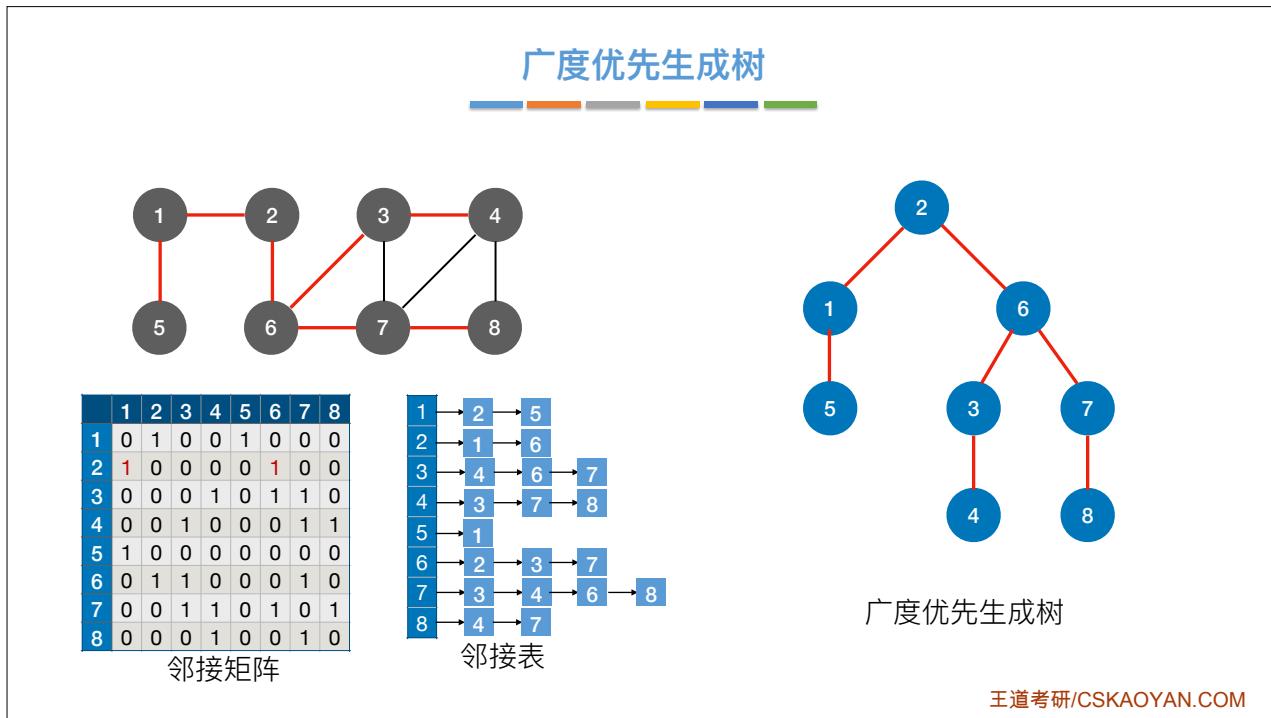
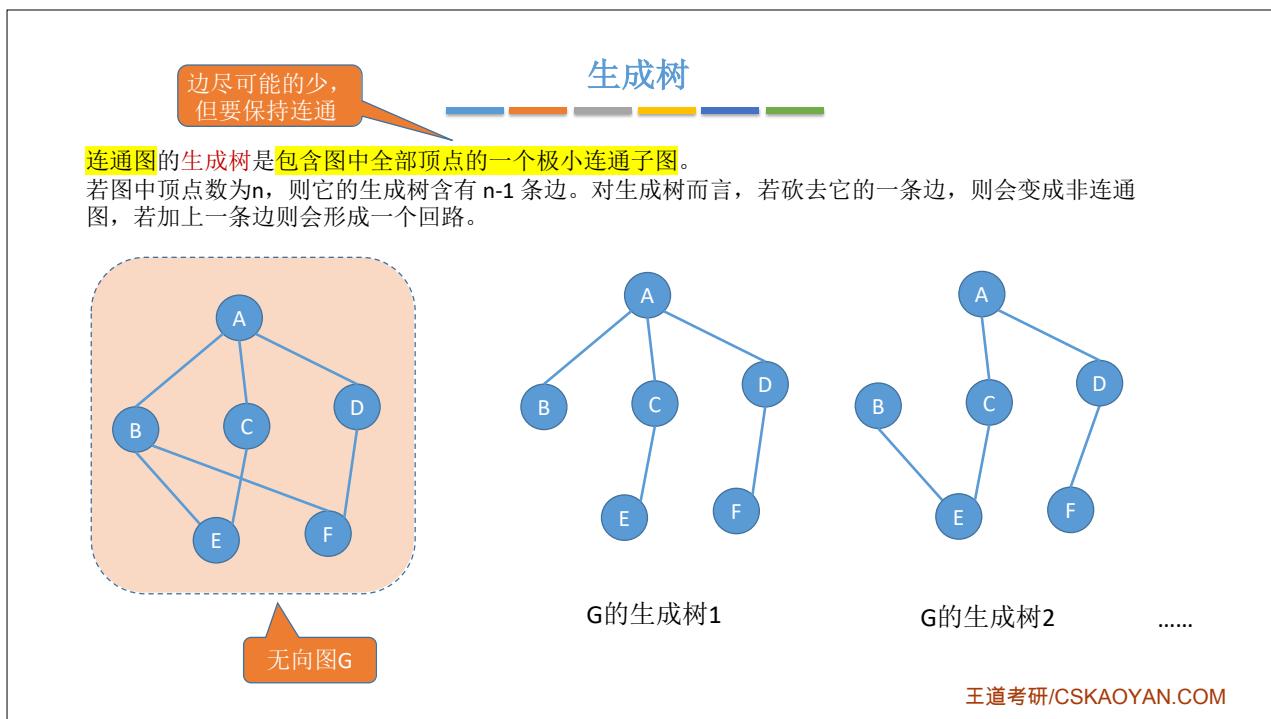
最小生成树

最小生成树的概念

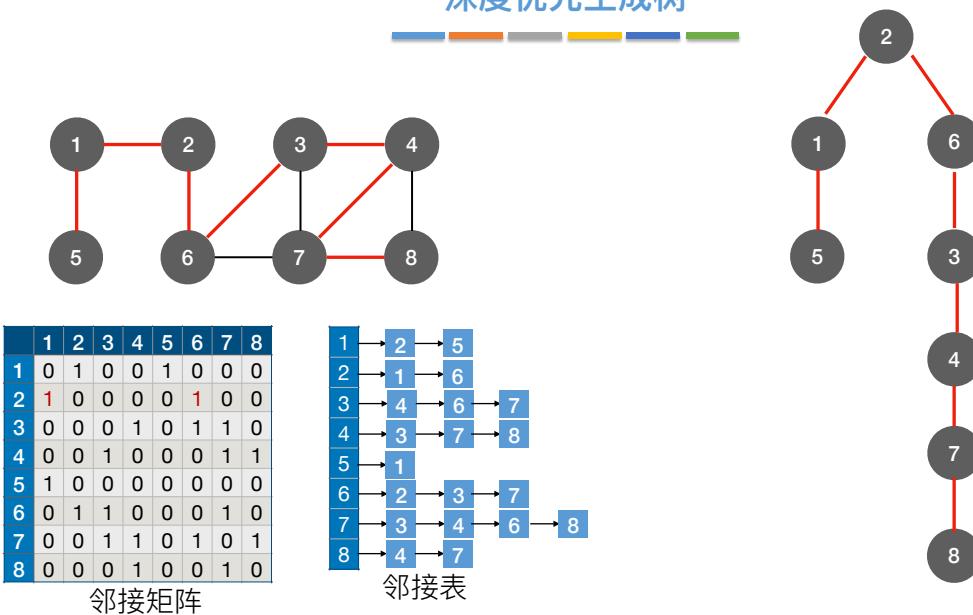
Prim 算法

Kruskal 算法

王道考研/CSKAOYAN.COM

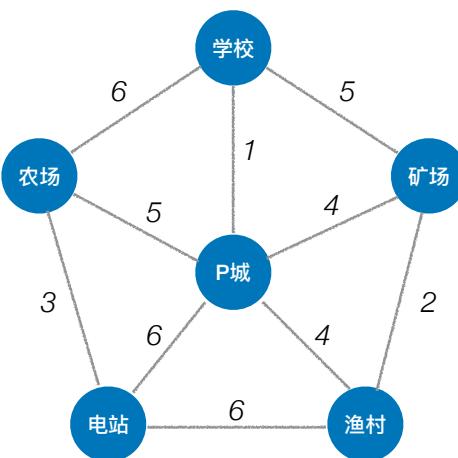


深度优先生成树



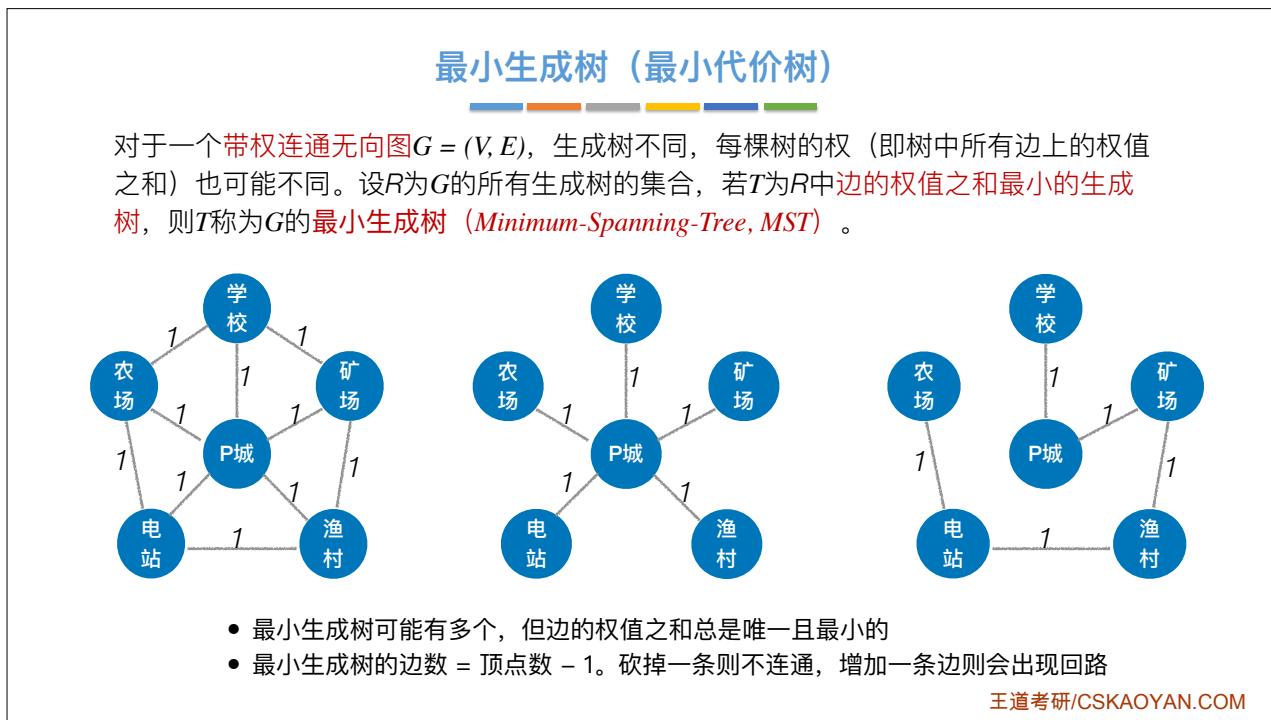
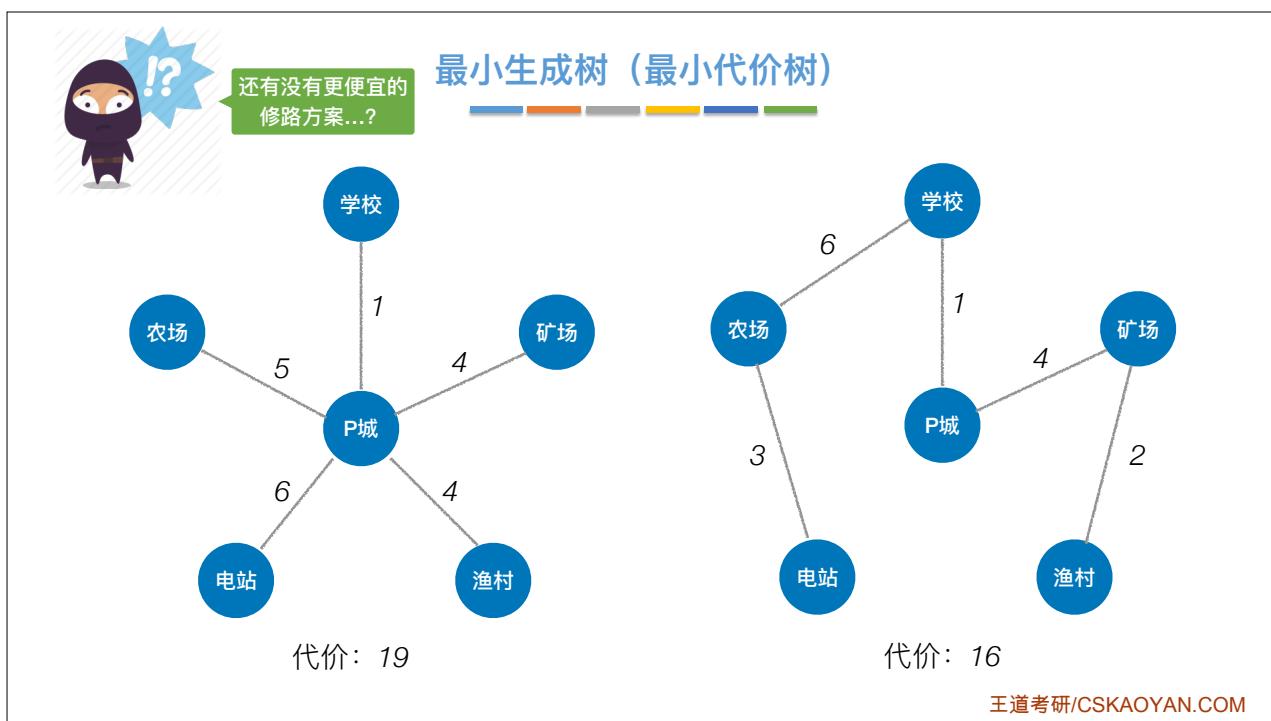
王道考研/CSKAOYAN.COM

最小生成树（最小代价树）



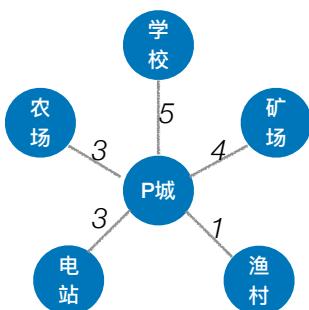
道路规划要求：所有地方都连通，且成本尽可能的低

王道考研/CSKAOYAN.COM

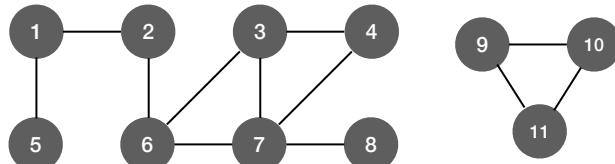


最小生成树（最小代价树）

对于一个带权连通无向图 $G = (V, E)$ ，生成树不同，每棵树的权（即树中所有边上的权值之和）也可能不同。设 R 为 G 的所有生成树的集合，若 T 为 R 中边的权值之和最小的生成树，则 T 称为 G 的最小生成树（Minimum-Spanning-Tree, MST）。



- 如果一个连通图本身就是一棵树，则其最小生成树就是它本身
- 只有连通图才有生成树，非连通图只有生成森林



王道考研/CSKAOYAN.COM

最小生成树（最小代价树）

对于一个带权连通无向图 $G = (V, E)$ ，生成树不同，每棵树的权（即树中所有边上的权值之和）也可能不同。设 R 为 G 的所有生成树的集合，若 T 为 R 中边的权值之和最小的生成树，则 T 称为 G 的最小生成树（Minimum-Spanning-Tree, MST）。

- 最小生成树可能有多个，但边的权值之和总是唯一且最小的
- 最小生成树的边数 = 顶点数 - 1。砍掉一条则不连通，增加一条边则会出现回路
- 如果一个连通图本身就是一棵树，则其最小生成树就是它本身
- 只有连通图才有生成树，非连通图只有生成森林



求最小生成树

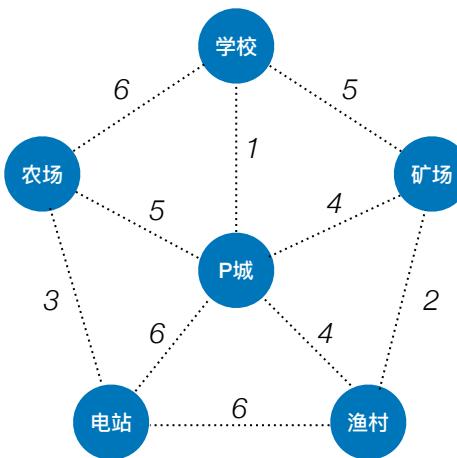
Prim 算法

Kruskal 算法

王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

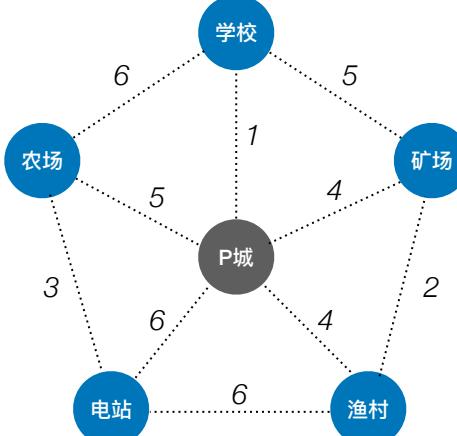
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

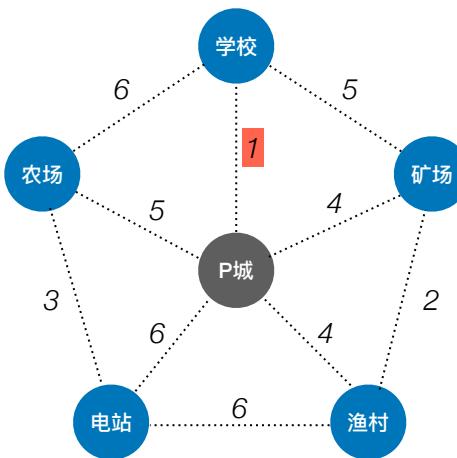
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

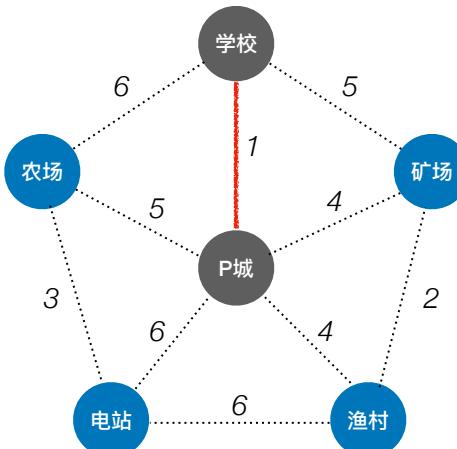
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

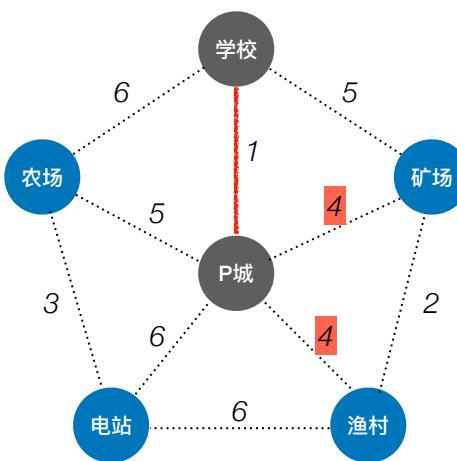
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

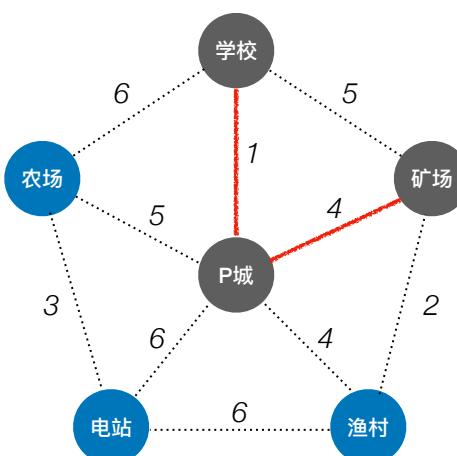
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

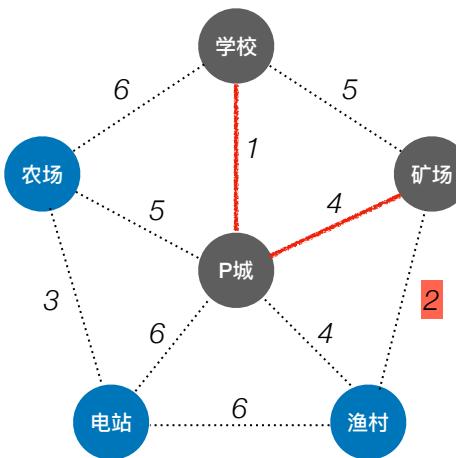
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

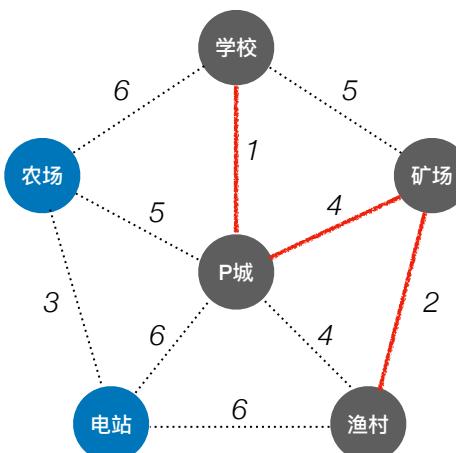
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

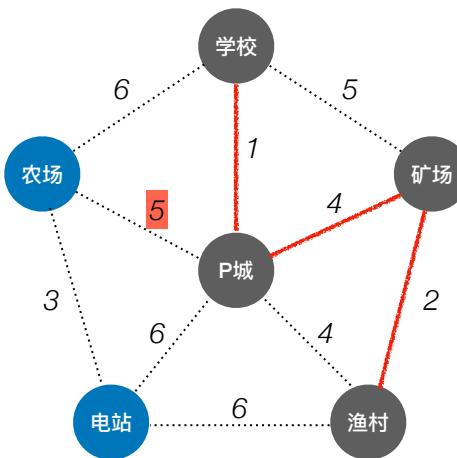
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

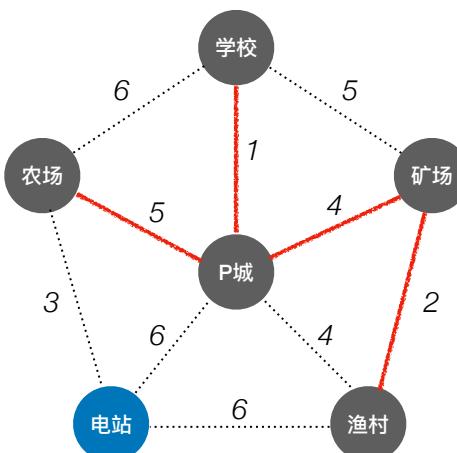
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

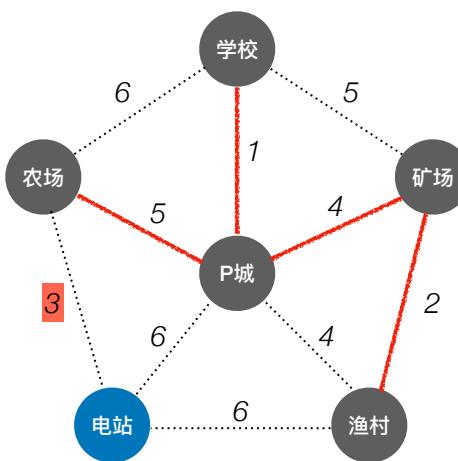
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

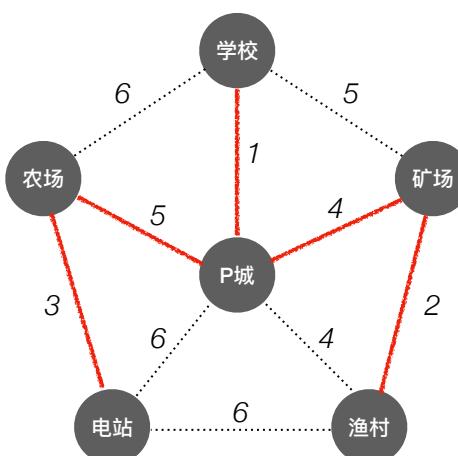
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

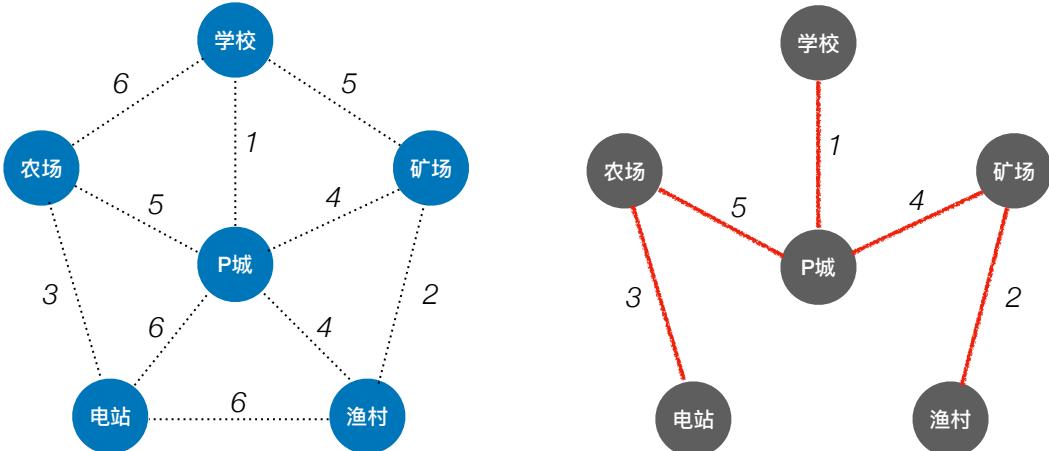
Prim 算法（普里姆）

Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

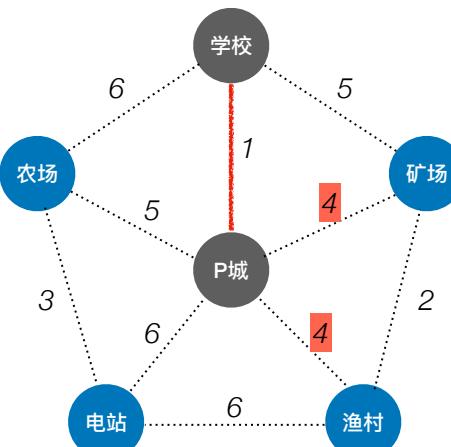
Prim 算法（普里姆）



最小代价：15

王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

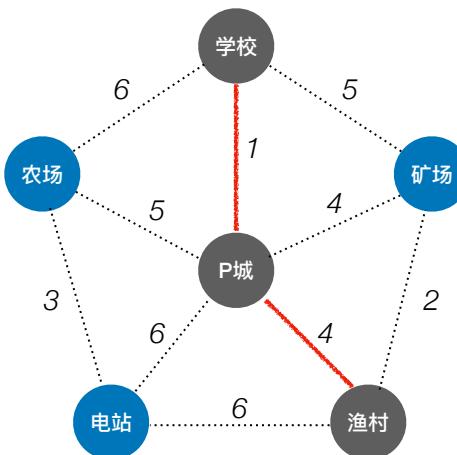


Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成树，直到所有顶点都纳入为止。

王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

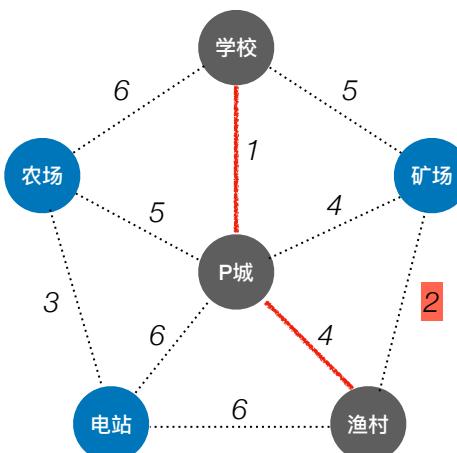
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

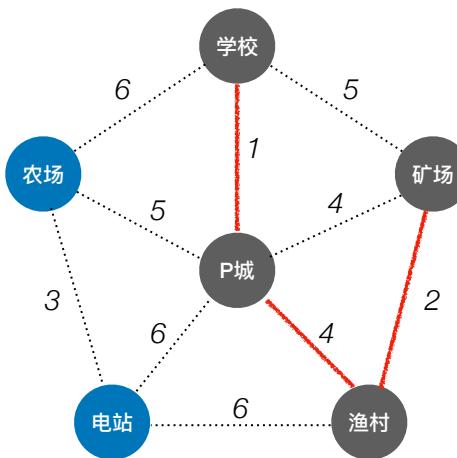
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

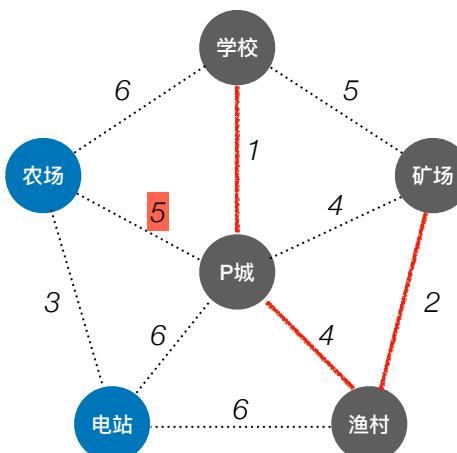
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

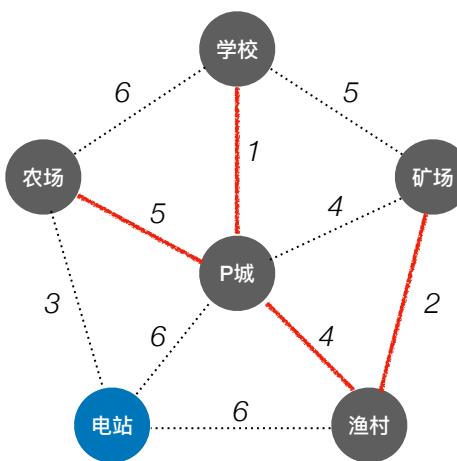
Prim 算法（普里姆）

Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

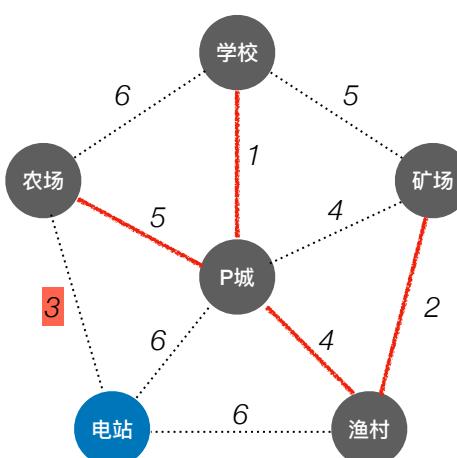
Prim 算法（普里姆）



Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。

王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

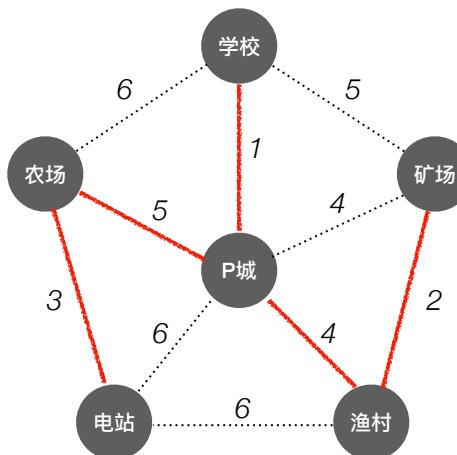


Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。

王道考研/CSKAOYAN.COM

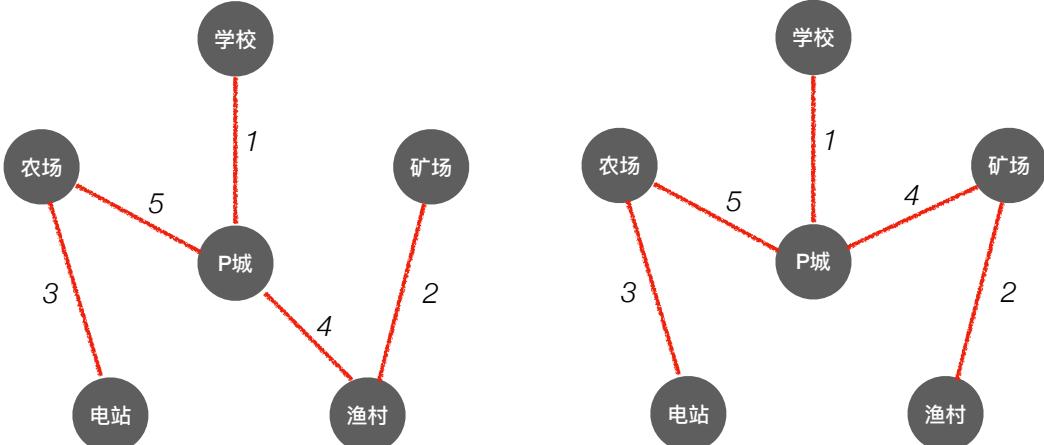
Prim 算法（普里姆）

Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

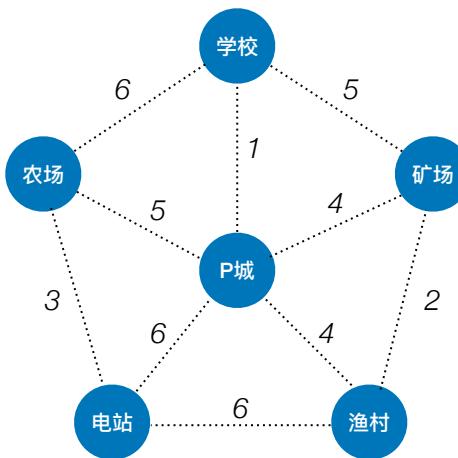


最小代价: 15

王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

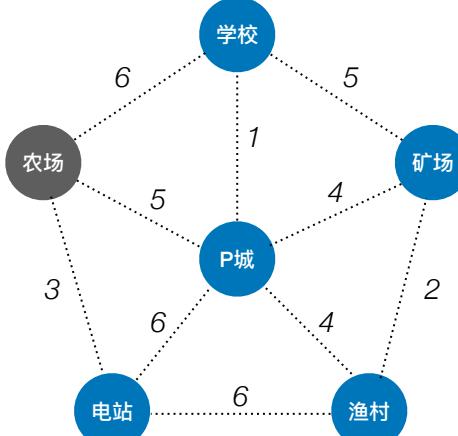
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

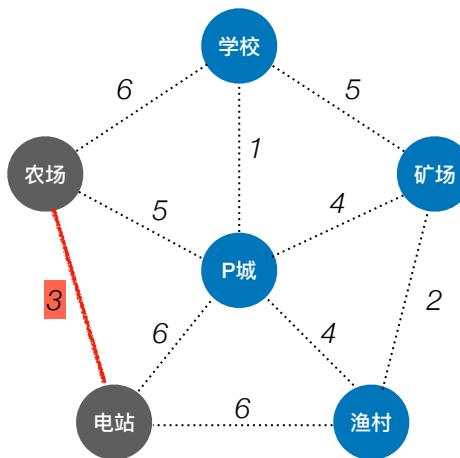
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

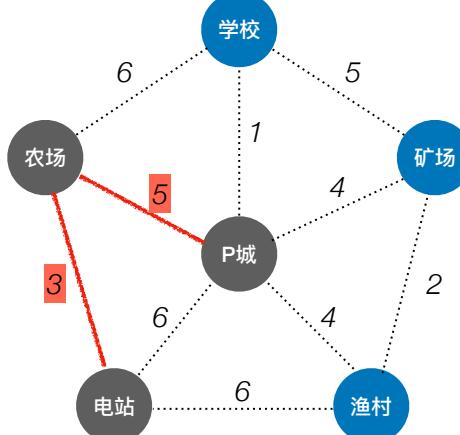
Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

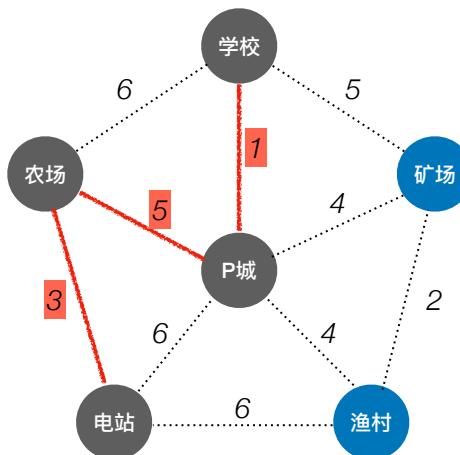
Prim 算法（普里姆）

Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。



王道考研/CSKAOYAN.COM

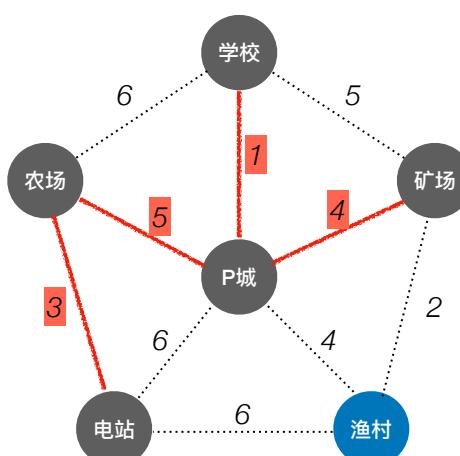
Prim 算法（普里姆）



Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。

王道考研/CSKAOYAN.COM

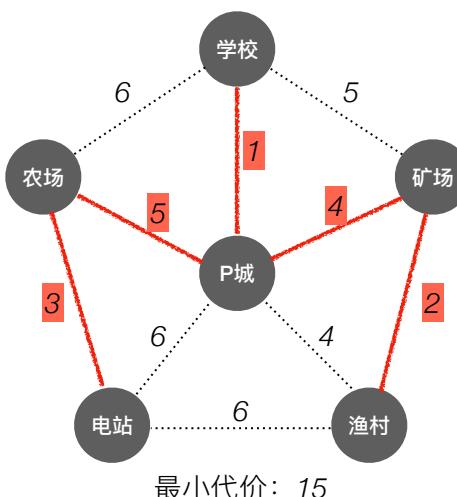
Prim 算法（普里姆）



Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。

王道考研/CSKAOYAN.COM

Prim 算法（普里姆）

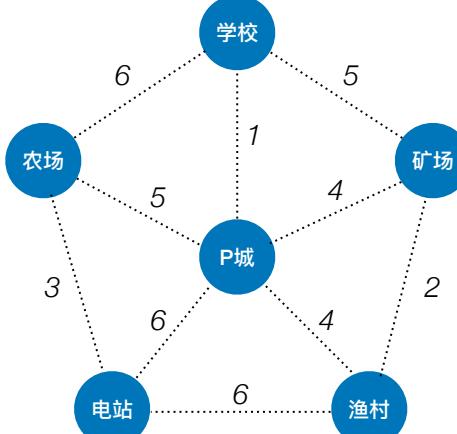


Prim 算法（普里姆）：
从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成
树，直到所有顶点都纳入为止。

最小代价：15

王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）



Kruskal 算法（克鲁斯卡尔）：
每次选择一条权值最小的边，使这
条边的两头连通（原本已经连通的
就不选）
直到所有结点都连通

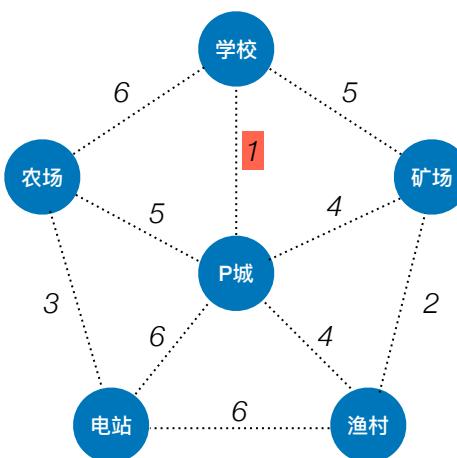
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



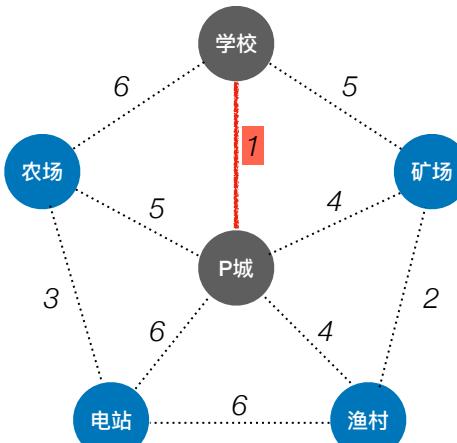
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



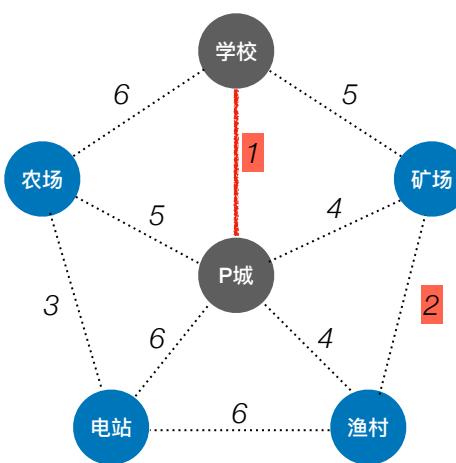
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



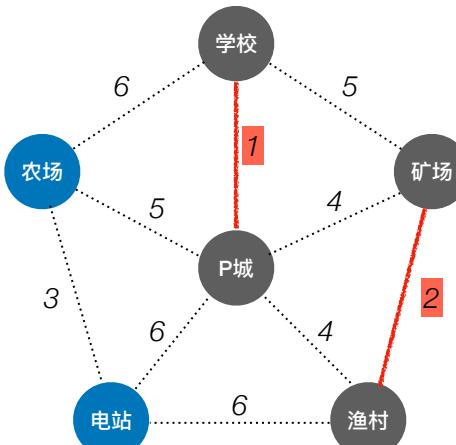
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



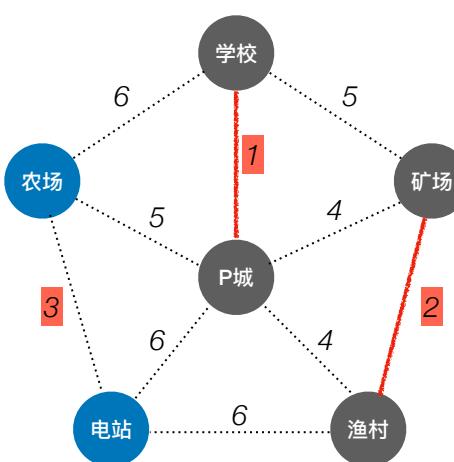
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



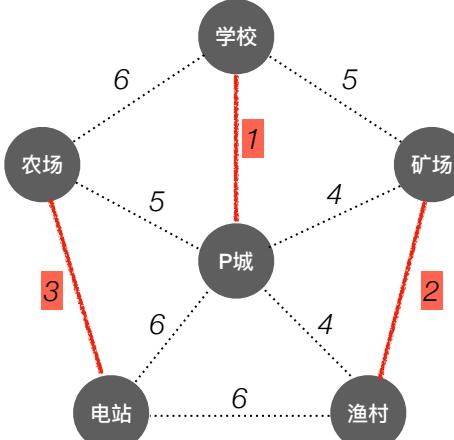
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



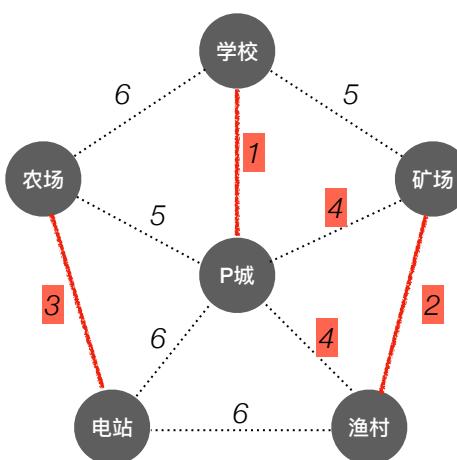
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



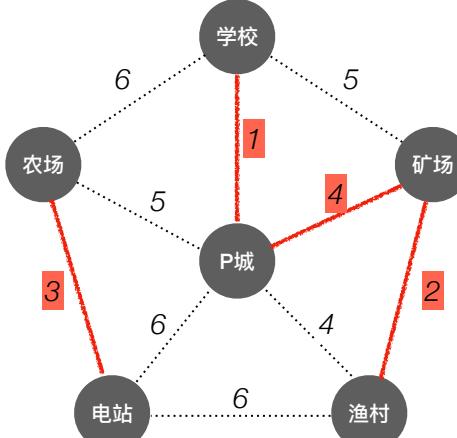
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



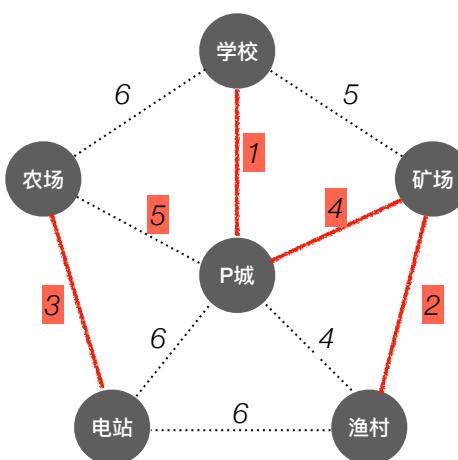
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



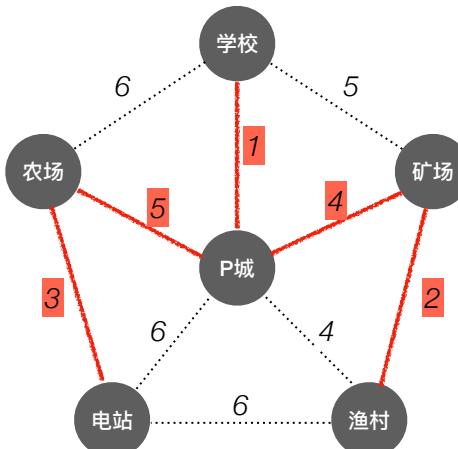
王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）

Kruskal 算法（克鲁斯卡尔）：

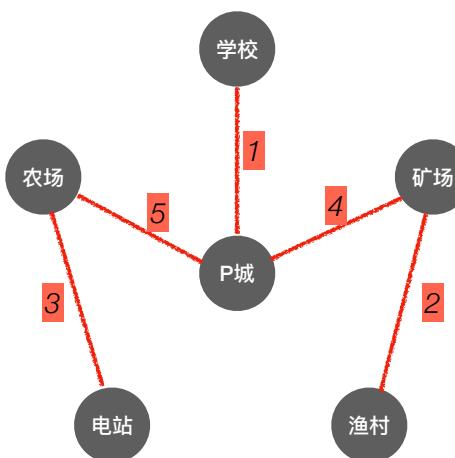
每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通



王道考研/CSKAOYAN.COM

Kruskal 算法（克鲁斯卡尔）



Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）

直到所有结点都连通

最小代价：15

王道考研/CSKAOYAN.COM

Prim 算法 v.s. Kruskal 算法

Prim 算法（普里姆）：

从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成树，直到所有顶点都纳入为止。

Kruskal 算法（克鲁斯卡尔）：

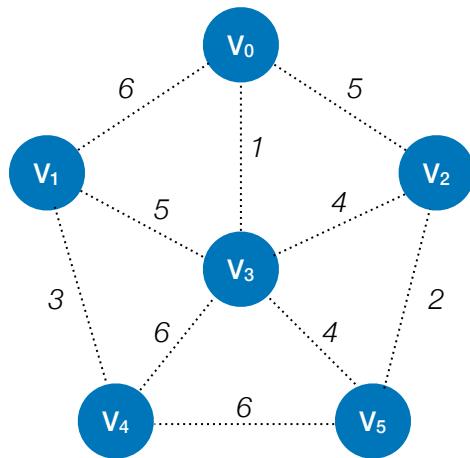
每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）
直到所有结点都连通

时间复杂度： $O(|V|^2)$
适合用于边稠密图

时间复杂度： $O(|E|\log_2|E|)$
适合用于边稀疏图

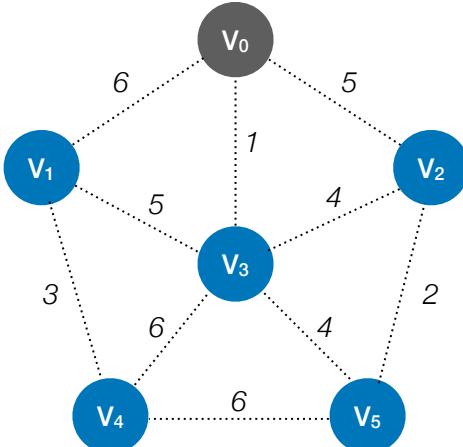
王道考研/CSKAOYAN.COM

Prim 算法的实现思想



王道考研/CSKAOYAN.COM

Prim 算法的实现思想



初始：从V₀开始

标记各节点是否已加入树

isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✗	✗	✗	✗

lowCost[6]

各节点加入树的最低代价

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第1轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✗	✗	✗	✗

isJoin[6]

各节点加入树的最低代价
lowCost[6]

0	6	5	1	∞	∞
---	---	---	---	----------	----------

lowCost[6]

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第1轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✗	✓	✗	✗

isJoin[6]

各节点加入树的最低代价
lowCost[6]

0	6	5	1	∞	∞
---	---	---	---	----------	----------

lowCost[6]

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

标记各节点是否已加入树

$isJoin[6]$

V0	V1	V2	V3	V4	V5
✓	✗	✗	✓	✗	✗

lowCost[6]

各节点加入树的最低代价

第1轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

再次循环遍历，更新还没加入的各个顶点的lowCost值

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

标记各节点是否已加入树

$isJoin[6]$

V0	V1	V2	V3	V4	V5
✓	✗	✗	✓	✗	✗

lowCost[6]

各节点加入树的最低代价

第1轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

再次循环遍历，更新还没加入的各个顶点的lowCost值

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第2轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✗	✓	✗	✗

lowCost[6] 各节点加入树的最低代价

0	5	4	1	6	4
---	---	---	---	---	---

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第2轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✗	✓	✗	✗

lowCost[6] 各节点加入树的最低代价

0	5	4	1	6	4
---	---	---	---	---	---

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第2轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✓	✓	✗	✗

lowCost[6]

0	5	4	1	6	4
---	---	---	---	---	---

各节点加入树的最低代价

再次循环遍历，更新还没加入的各个顶点的lowCost值

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第2轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✓	✓	✗	✗

lowCost[6]

0	5	4	1	6	2
---	---	---	---	---	---

各节点加入树的最低代价

再次循环遍历，更新还没加入的各个顶点的lowCost值

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第3轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✓	✓	✗	✗

各节点加入树的最低代价
lowCost[6]

0	5	4	1	6	2
---	---	---	---	---	---

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第3轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✓	✓	✗	✗

各节点加入树的最低代价
lowCost[6]

0	5	4	1	6	2
---	---	---	---	---	---

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第3轮：循环遍历所有个结点，找到 $lowCost$ 最低的，且还没加入树的顶点

标记各节点是否已加入树
 $isJoin[6]$

V0	V1	V2	V3	V4	V5
✓	✗	✓	✓	✗	✓

$lowCost[6]$

0	5	4	1	6	2
---	---	---	---	---	---

各节点加入树的最低代价

再次循环遍历，更新还没加入的各个顶点的 $lowCost$ 值

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第4轮：循环遍历所有个结点，找到 $lowCost$ 最低的，且还没加入树的顶点

标记各节点是否已加入树
 $isJoin[6]$

V0	V1	V2	V3	V4	V5
✓	✗	✓	✓	✗	✓

$lowCost[6]$

0	5	4	1	6	2
---	---	---	---	---	---

各节点加入树的最低代价

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第4轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✗	✓	✓	✗	✓

lowCost[6]

0	5	4	1	6	2
---	---	---	---	---	---

各节点加入树的最低代价

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第4轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✓	✓	✓	✗	✓

lowCost[6]

0	5	4	1	6	2
---	---	---	---	---	---

各节点加入树的最低代价

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第4轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✓	✓	✓	✗	✓

lowCost[6]

0	5	4	1	6	2
---	---	---	---	---	---

各节点加入树的最低代价

再次循环遍历，更新还没加入的各个顶点的lowCost值

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第4轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✓	✓	✓	✗	✓

lowCost[6]

0	5	4	1	3	2
---	---	---	---	---	---

各节点加入树的最低代价

再次循环遍历，更新还没加入的各个顶点的lowCost值

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第5轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

V0	V1	V2	V3	V4	V5
✓	✓	✓	✓	✗	✓

各节点加入树的最低代价
lowCost[6]

0	5	4	1	3	2
---	---	---	---	---	---

再次循环遍历，更新还没加入的各个顶点的lowCost值

王道考研/CSKAOYAN.COM

Prim 算法的实现思想

第5轮：循环遍历所有个结点，找到lowCost最低的，且还没加入树的顶点

标记各节点是否已加入树
isJoin[6]

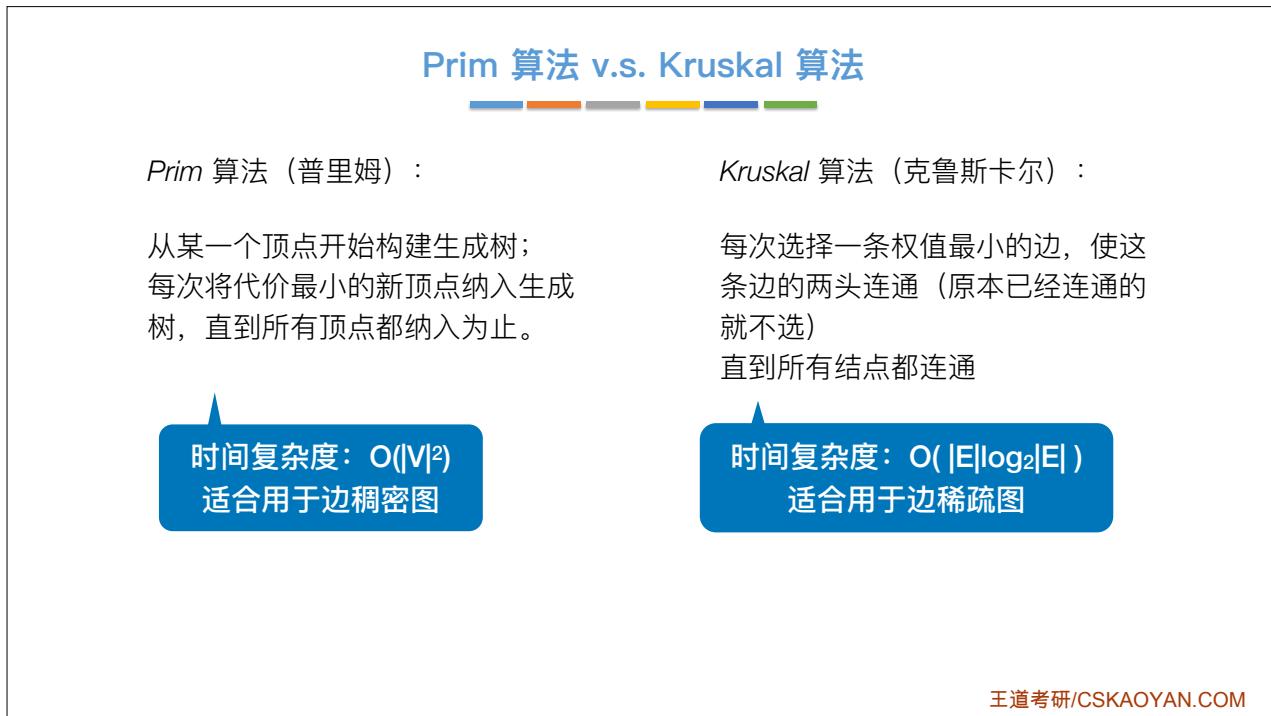
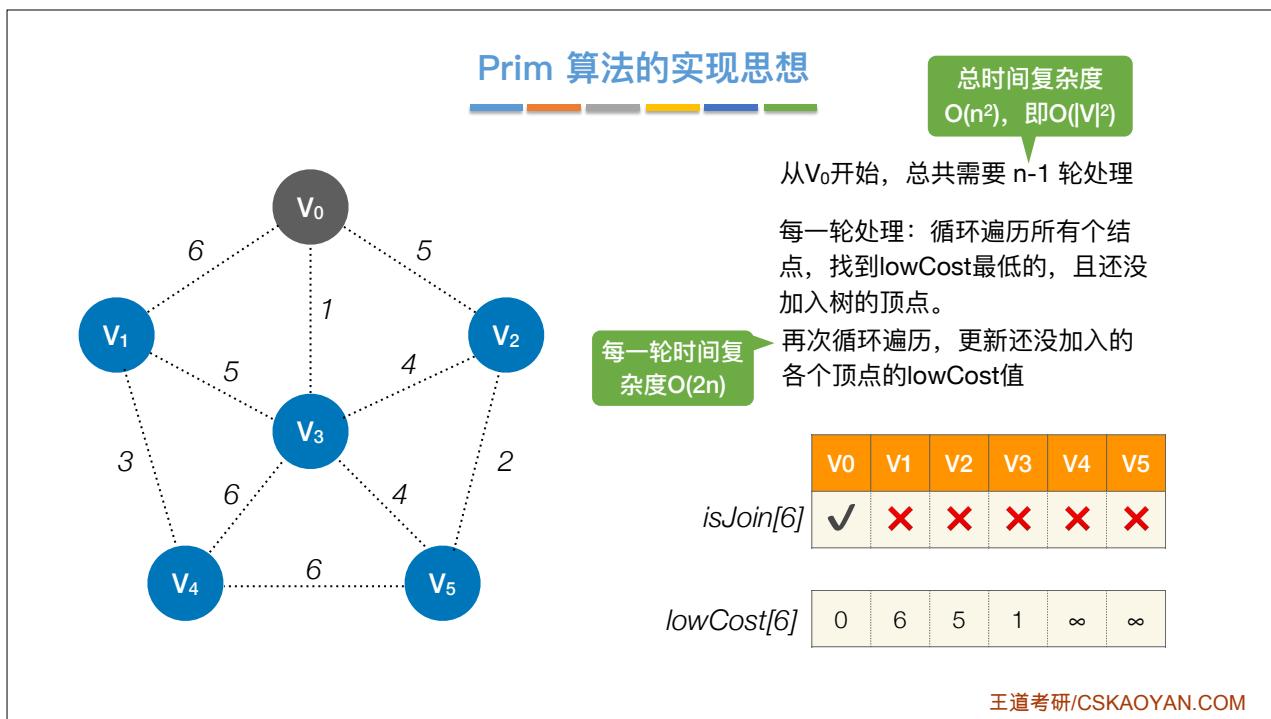
V0	V1	V2	V3	V4	V5
✓	✓	✓	✓	✓	✓

各节点加入树的最低代价
lowCost[6]

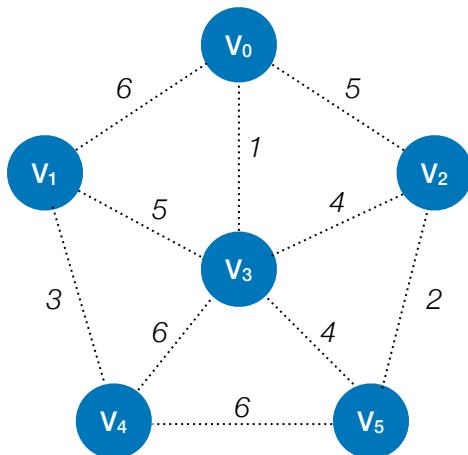
0	5	4	1	3	2
---	---	---	---	---	---

再次循环遍历，更新还没加入的各个顶点的lowCost值

王道考研/CSKAOYAN.COM



Kruskal 算法的实现思想

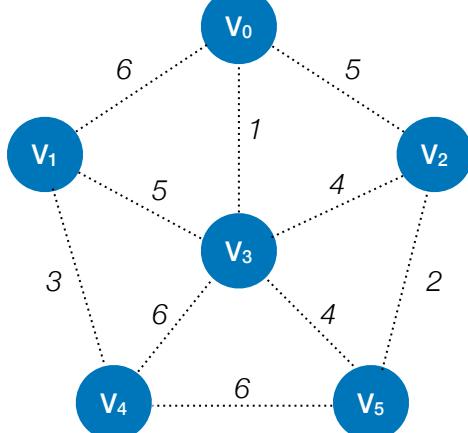


王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想



初始：将各条边按权值排序

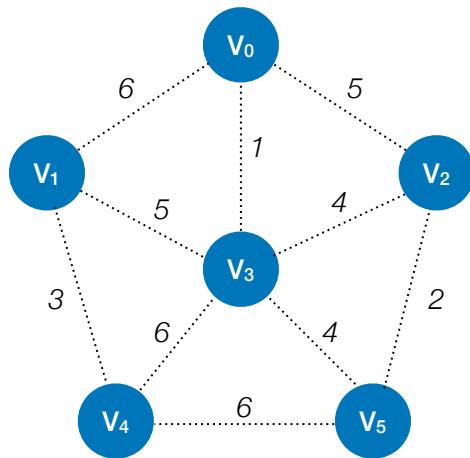


weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅

王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第1轮：检查第1条边的两个顶点是否连通（是否属于同一个集合）

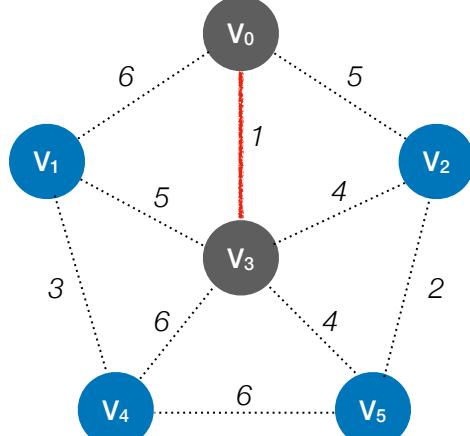


weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅

王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第1轮：检查第1条边的两个顶点是否连通（是否属于同一个集合）



weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅

不连通，连起来！

王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第2轮：检查第2条边的两个顶点是否连通（是否属于同一个集合）

weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅

王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第2轮：检查第2条边的两个顶点是否连通（是否属于同一个集合）

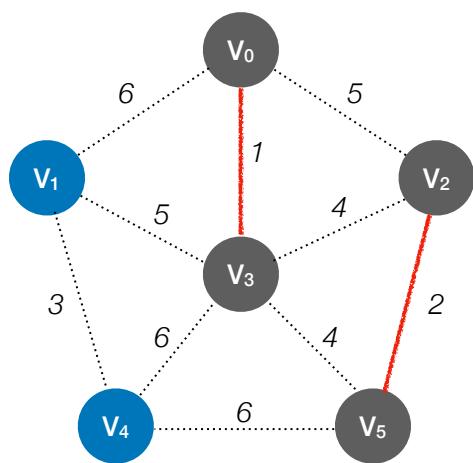
weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅

不连通，连起来！

王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第3轮：检查第3条边的两个顶点是否连通（是否属于同一个集合）

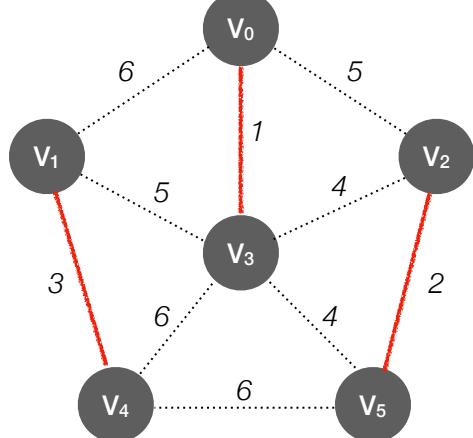


weight	Vertex1	Vertex2
1	V0	V3
2	V2	V5
3	V1	V4
4	V2	V3
4	V3	V5
5	V0	V2
5	V1	V3
6	V0	V1
6	V3	V4
6	V4	V5

王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第3轮：检查第3条边的两个顶点是否连通（是否属于同一个集合）



weight	Vertex1	Vertex2
1	V0	V3
2	V2	V5
3	V1	V4
4	V2	V3
4	V3	V5
5	V0	V2
5	V1	V3
6	V0	V1
6	V3	V4
6	V4	V5

不连通，连起来！

王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第4轮：检查第4条边的两个顶点是否连通（是否属于同一个集合）

weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅

不连通，连起来！

王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第5轮：检查第5条边的两个顶点是否连通（是否属于同一个集合）

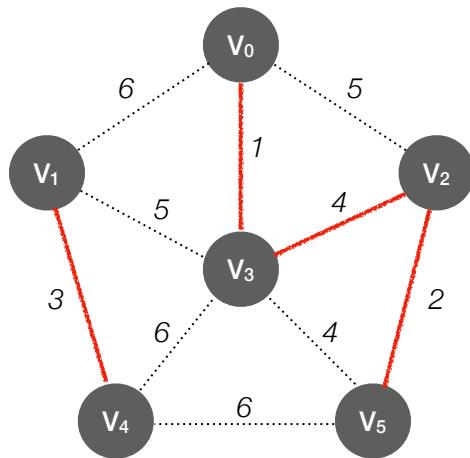
weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅

已连通，跳过！

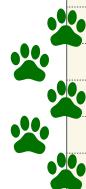
王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

第6轮：检查第6条边的两个顶点是否连通（是否属于同一个集合）



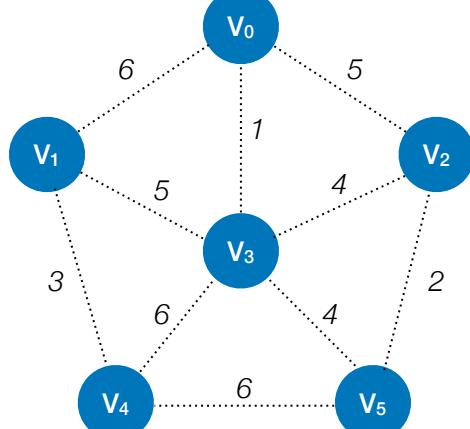
weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅



王道考研/CSKAOYAN.COM

Kruskal 算法的实现思想

初始：将各条边按权值排序



weight	Vertex1	Vertex2
1	V ₀	V ₃
2	V ₂	V ₅
3	V ₁	V ₄
4	V ₂	V ₃
4	V ₃	V ₅
5	V ₀	V ₂
5	V ₁	V ₃
6	V ₀	V ₁
6	V ₃	V ₄
6	V ₄	V ₅

共执行 e 轮，每轮判断两个顶点是否属于同一集合，需要 $O(\log_2 e)$

总时间复杂度 $O(e \log_2 e)$

王道考研/CSKAOYAN.COM

知识回顾与重要考点



对于一个带权连通无向图 $G = (V, E)$ ，生成树不同，每棵树的权（即树中所有边上的权值之和）也可能不同。设 R 为 G 的所有生成树的集合，若 T 为 R 中边的权值之和最小的生成树，则 T 称为 G 的最小生成树（Minimum-Spanning-Tree, MST）。

Prim 算法（普里姆）：

从某一个顶点开始构建生成树；
每次将代价最小的新顶点纳入生成树，直到所有顶点都纳入为止。

Kruskal 算法（克鲁斯卡尔）：

每次选择一条权值最小的边，使这条边的两头连通（原本已经连通的就不选）
直到所有结点都连通

时间复杂度： $O(|V|^2)$
适合用于边稠密图

时间复杂度： $O(|E| \log_2 |E|)$
适合用于边稀疏图