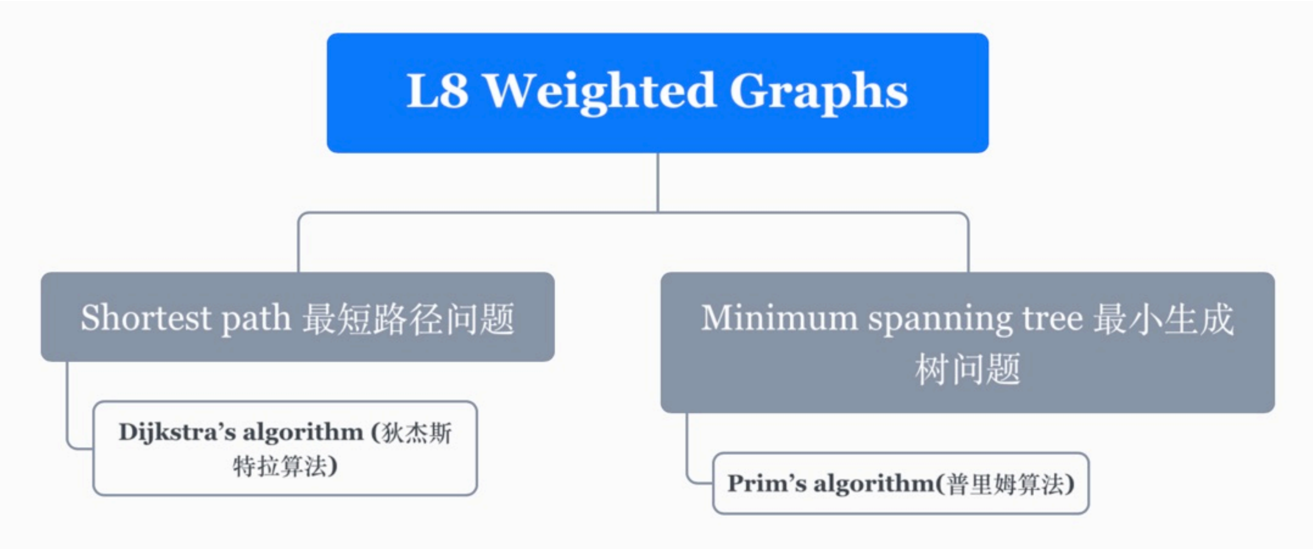
CS211FZ Note 5-2｜Algorithms & Data Structures｜DSA2

> 前言： A wise man changes his mind, a fool never.

Key Points 5-2

- Shortest path 最短路径问题
    - **Dijkstra's algorithm (狄杰斯特拉算法)**
- Minimum spanning tree 最小生成树问题
    - **Prim's algorithm**



L8 Weighted Graphs

- Shortest path 最短路径问题
    - **Dijkstra's algorithm (狄杰斯特拉算法)**
- Minimum spanning tree 最小生成树问题
    - **Prim's algorithm(普里姆算法)**

8.1 Dijkstra's algorithm

典型的最短路径算法，用于计算一个结点到其他节点的最短路径。

主要特别是以起始点为中心向外层层扩展（BFS），知道拓展到终点为止。

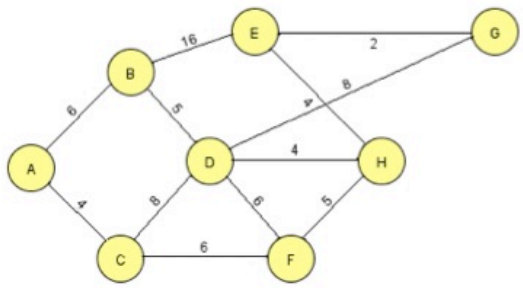
迪杰斯特拉(Dijkstra)算法过程

设置出发顶点为v，顶点集合V{v1,v2,vi...}，v到V中各顶点的距离构成距离集合Dis，Dis{d1,d2,di...}，Dis集合记录着v到图中各顶点的距离(到自身可以看作0，v到vi距离对应为di)

1) 从Dis中选择值最小的di并移出Dis集合，同时移出V集合中对应的顶点vi，此时的v到vi即为最短路径
2) 更新Dis集合，更新规则为：比较v到V集合中顶点的距离值，与v通过vi到V集合中顶点的距离值，保留值较小的一个(同时也应该更新顶点的前驱节点为vi，表明是通过vi到达的)
3) 重复执行两步骤，直到最短路径顶点为目标顶点即可结束

Dijkstra算法思想：采用贪心法思想，进行n-1次查找（PS:n为加权连通图的顶点总个数，除去起点，则剩下n-1个顶点），第一次进行查找，找出距离起点最近的一个顶点，标记为已遍历；下一次进行查找时，从未被遍历中的顶点寻找距离起点最近的一个顶点， 标记为已遍历；直到n-1次查找完毕，结束查找，返回最终结果。

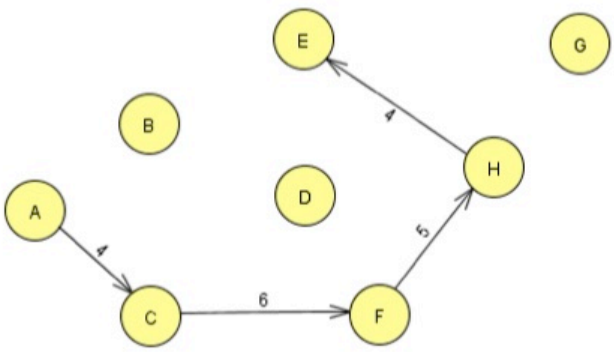
Worked Solution



Vertices Visited	A	B	C	D	E	F	G	H
A	0	6(A)	4(A)	-	-	-	-	-
A C	0	6(A)	4(A)	12(C)	-	10(C)	-	-
A C B	0	6(A)	4(A)	11(B)	22(B)	10(C)	-	-
A C B F	0	6(A)	4(A)	11(B)	22(B)	10(C)	-	15(F)
A C B F D	0	6(A)	4(A)	11(B)	22(B)	10(C)	19(D)	15(F)
A C B F D H	0	6(A)	4(A)	11(B)	19(H)	10(C)	19(D)	15(F)
A C B F D H E	0	6(A)	4(A)	11(B)	19(H)	10(C)	19(D)	15(F)

- Keep picking the shortest path to an unvisited vertex (red shows visited)
- When you move to a new vertex update the shortest paths

Solution

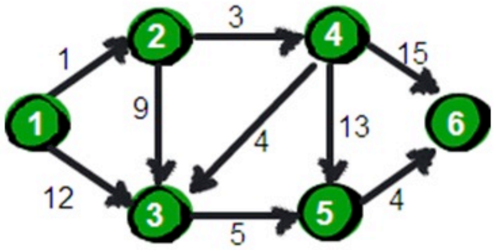


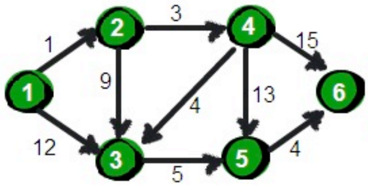
- The final row in the Shortest-Path Array represents the shortest paths to all of the vertices when starting at A

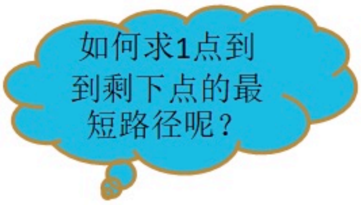
Vertices Visited	B	C	D	E	F	G	H
A C B F D H E G	6 (A)	4 (A)	11 (B)	19 (H)	10 (C)	19 (D)	15 (F)

Dijkstra复杂度是O(N^2)

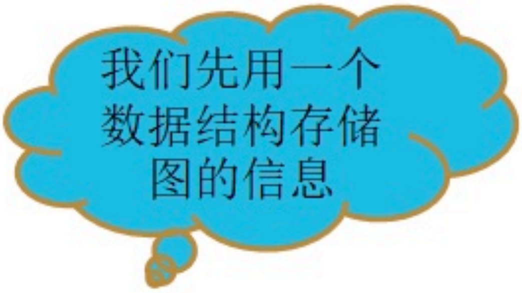
指定一个点（源点）到其余各个顶点的最短路径，也叫做“单源最短路径”。例如求下图中的1号顶点到2、3、4、5、6号顶点的最短路径。



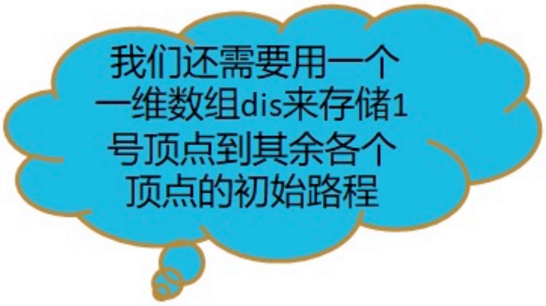




1 -> 2 -> 4 -> 3 -> 5 -> 6



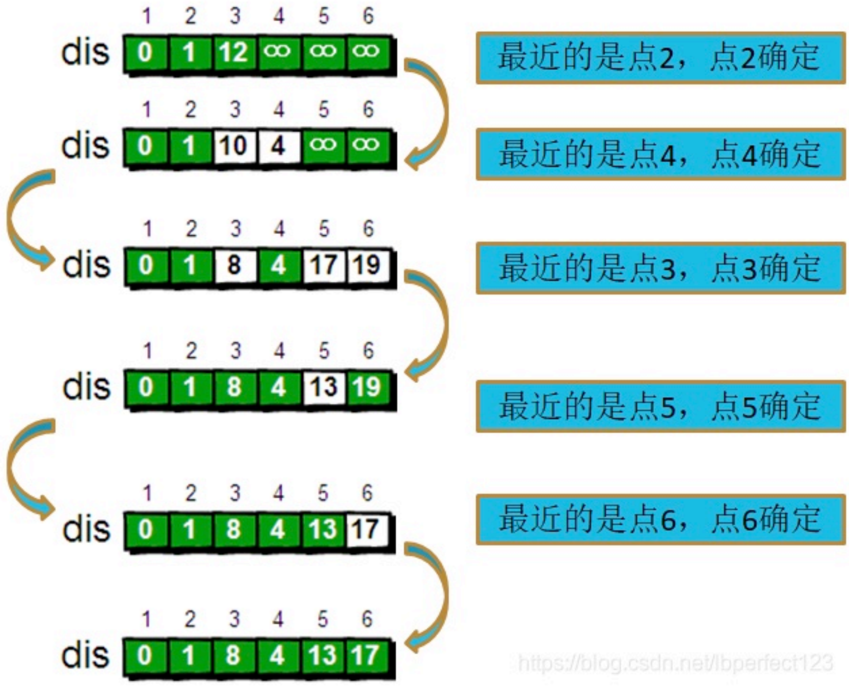
e	1	2	3	4	5	6
1	0	1	12	∞	∞	∞
2	∞	0	9	3	∞	∞
3	∞	∞	0	∞	5	∞
4	∞	∞	4	0	13	15
5	∞	∞	∞	∞	0	4
6	∞	∞	∞	∞	∞	0



	1	2	3	4	5	6
dis	0	1	12	∞	∞	∞

下面我们来模拟一下：

e	1	2	3	4	5	6
1	0	1	12	∞	∞	∞
2	∞	0	9	3	∞	∞
3	∞	∞	0	∞	5	∞
4	∞	∞	4	0	13	15
5	∞	∞	∞	∞	0	4
6	∞	∞	∞	∞	∞	0



访问次序： 1 -> 2 -> 4 -> 3 -> 5 -> 6

参考：《Dijkstra算法图文详解》

http://t.csdn.cn/FHC4w

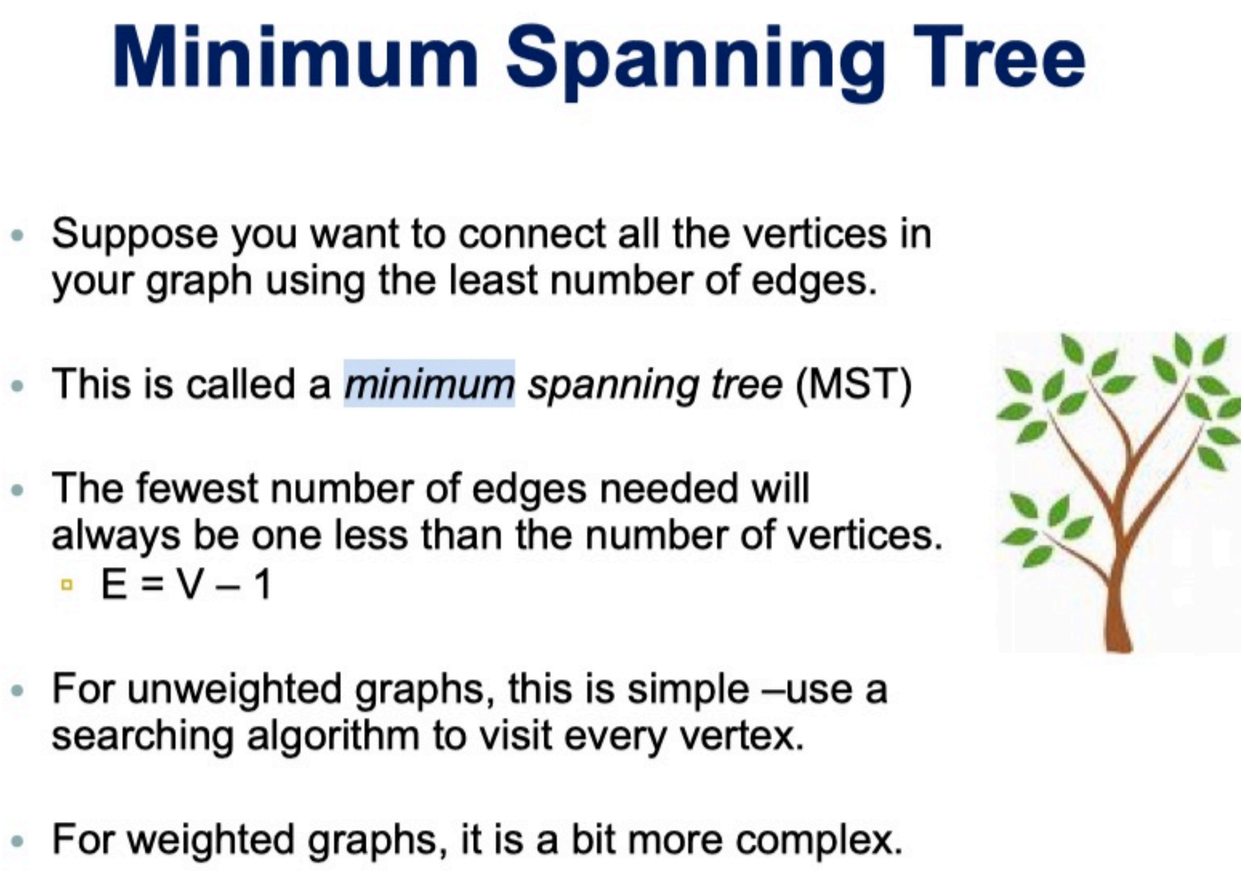
8.2 Minimum spanning tree 最小生成树

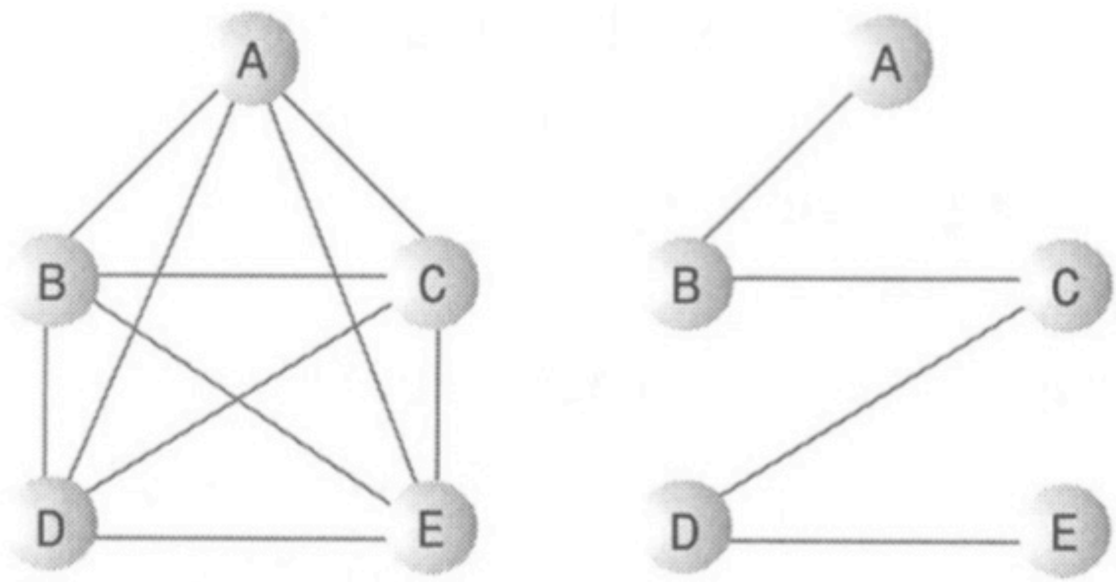
现在假设有一个实际问题：我们要在n个城市中建立一个通信网络，则连通这n个城市需要布置(n-1)一条通信线路，**这个时候我们需要考虑如何在成本最低的情况下建立这个通信网?**

于是我们就可以引入**连通图**来解决我们遇到的问题，n个城市就是图上的n个顶点，然后，边表示两个城市的通信线路，**每条边上的权重就是我们搭建这条线路所需要的成本**，所以现在我们有n个顶点的连通网可以建立不同的生成树，每一颗生成树都可以作为一个通信网，**当我们构造这个连通网所花的成本最小时，搭建该连通网的生成树，就称为最小生成树。**

构造最小生成树有很多算法，但是他们都是利用了最小生成树的同一种性质：MST性质，假设N=(V , {E})是一个连通网，U是顶点集V的一个非空子集，如果（u，v）是一条具有最小权值的边，其中u属于U，v属于V-U，则必定存在一颗包含边（u，v）的最小生成树）

两种使用MST性质生成最小生成树的算法：**普里姆算法(Prim's algorithm)**和*克鲁斯卡尔算法*





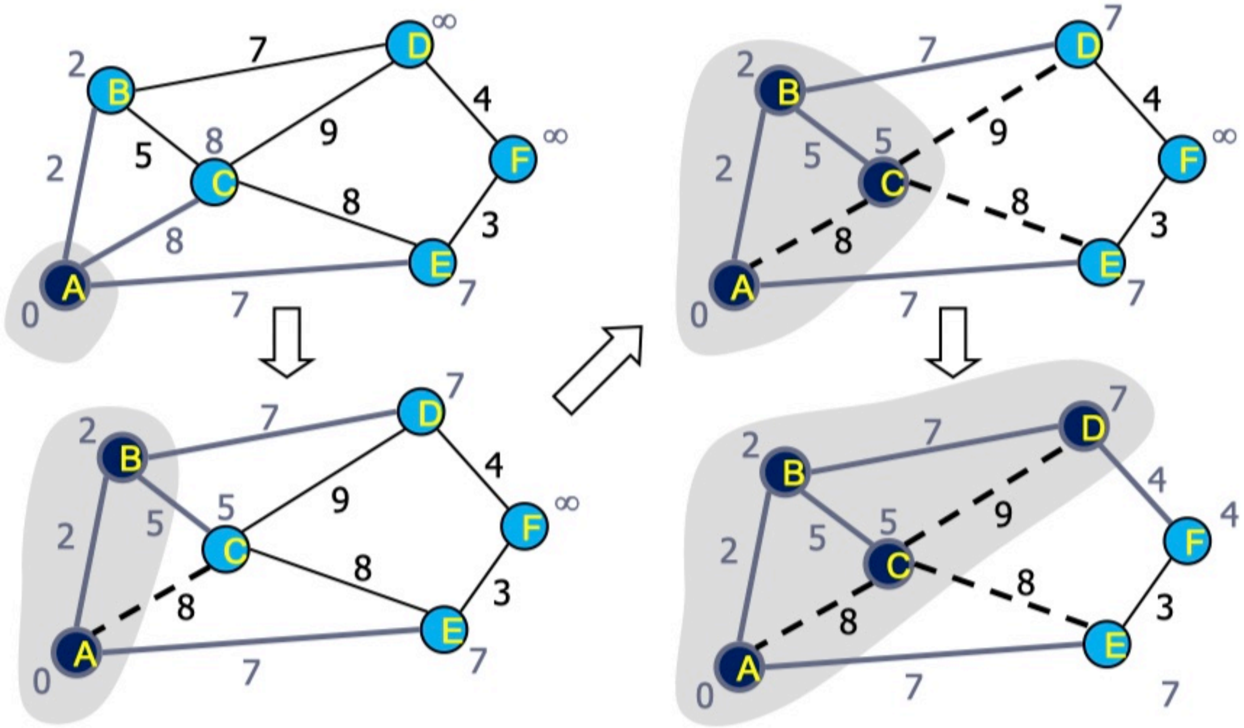
a) Extra Edges　　b) Minimum Number of Edges

Prim's algorithm(普里姆算法)

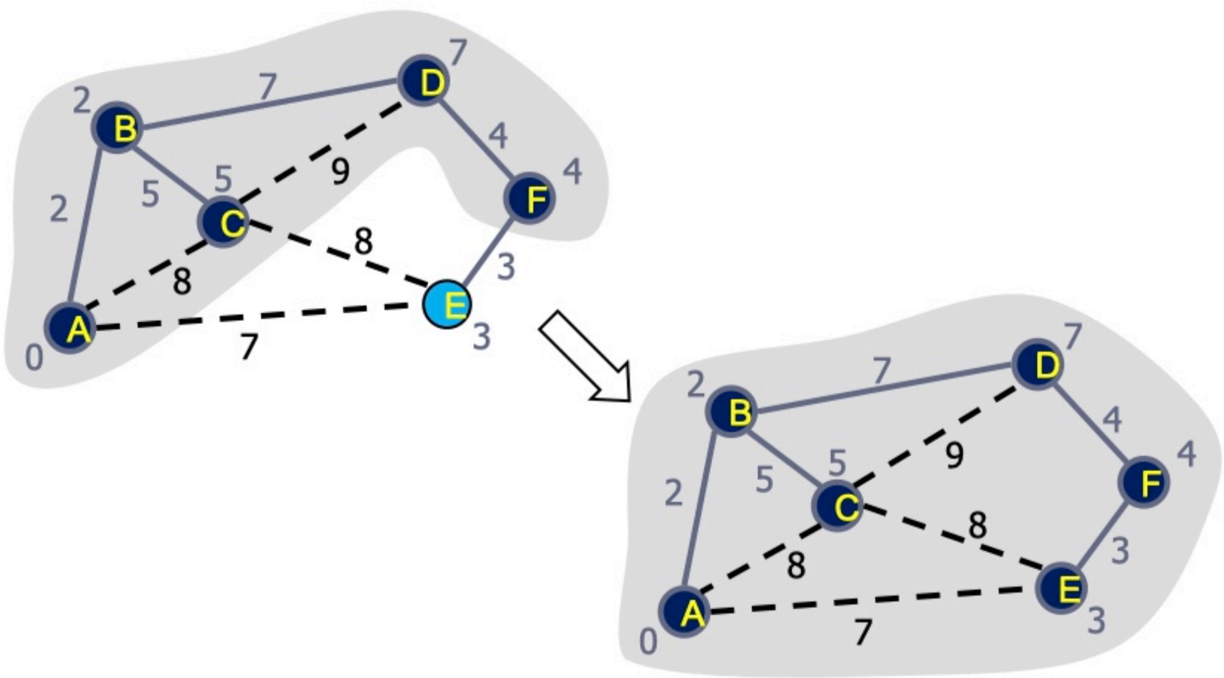
> 首先就是从图中的一个起点a开始，把a加入U集合，然后，寻找从与a有关联的边中，权重最小的那条边并且该边的终点b在顶点集合：（V-U）中，我们也把b加入到集合U中，并且输出边（a，b）的信息，这样我们的集合U就有：{a,b}，然后，我们寻找与a关联和b关联的边中，权重最小的那条边并且该边的终点在集合：（V-U）中，我们把c加入到集合U中，并且输出对应的那条边的信息，这样我们的集合U就有：{a,b,c}这三个元素了，一次类推，直到所有顶点都加入到了集合U。

Ex1.

Example



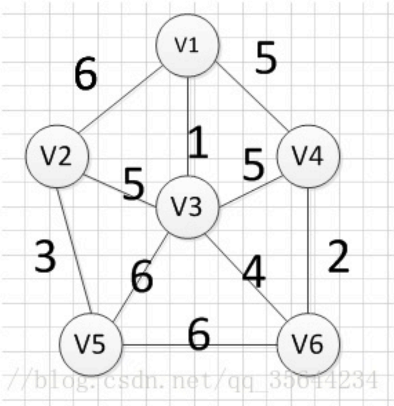
Example (contd.)



顺序：A B C D F E

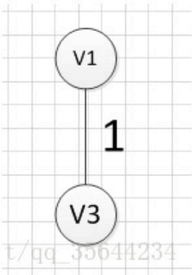
Ex2.

下面我们对下面这幅图求其最小生成树：

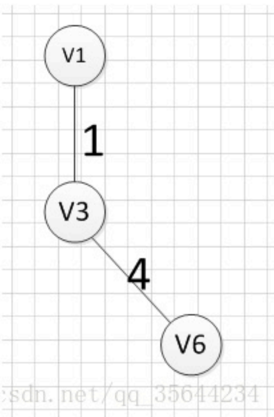


顺序：`v1 -> v3 -> v6 -> v4 -> v2 -> v5`

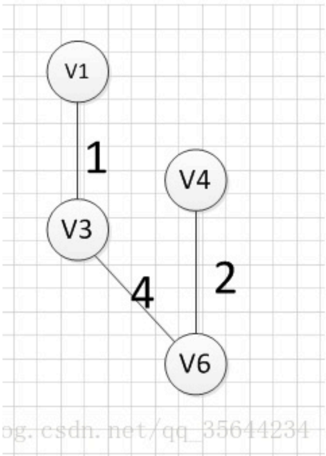
假设我们从顶点v1开始，所以我们可以发现（v1,v3）边的权重最小，所以第一个输出的边就是：v1—v3=1:



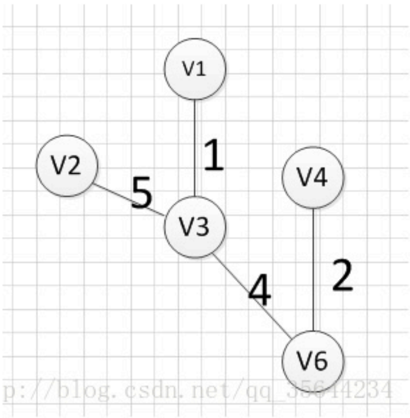
然后，我们要从v1和v3作为起点的边中寻找权重最小的边，首先了（v1,v3）已经访问过了，所以我们从其他边中寻找，发现(v3,v6)这条边最小，所以输出边就是：v3—-v6=4



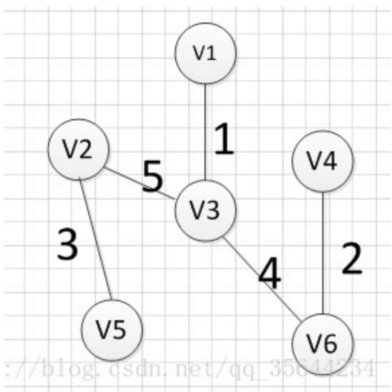
然后，我们要从v1、v3、v6这三个点相关联的边中寻找一条权重最小的边，我们可以发现边(v6,v4)权重最小，所以输出边就是：v6—-v4=2.



然后，我们就从v1、v3、v6、v4这四个顶点相关联的边中寻找权重最小的边，发现边（v3，v2）的权重最小，所以输出边：v3——v2=5



然后，我们就从v1、v3、v6、v4，v2这2五个顶点相关联的边中寻找权重最小的边，发现边（v2，v5）的权重最小，所以输出边：v2——v5=3



顺序：V1 -> V3 -> V6 -> V4 -> V2 -> V5

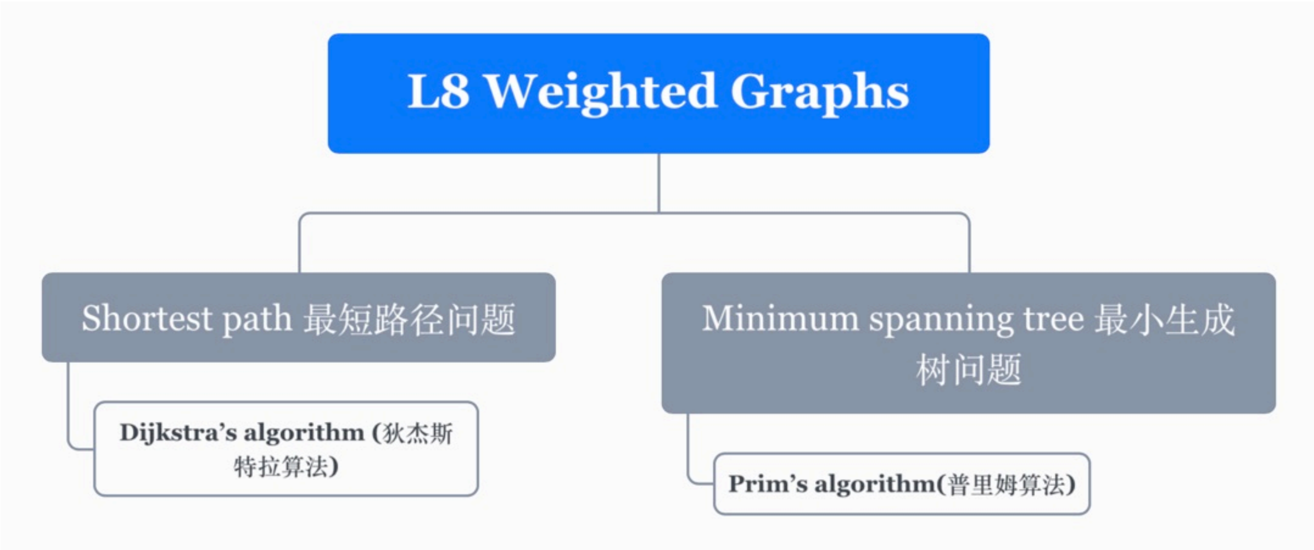
其中，建立邻接矩阵需要的时间复杂度为：O(n*n)

Prim函数中生成最小生成树的时间复杂度为：O(n*n).

参考：《数据结构--最小生成树详解》

http://t.csdn.cn/SMj22

Key Points 5-2 Review



CS211 Note-5-2

by Lance Cai

2022/07/05