数据结构与算法（Python版）

图的应用：最短路径问题

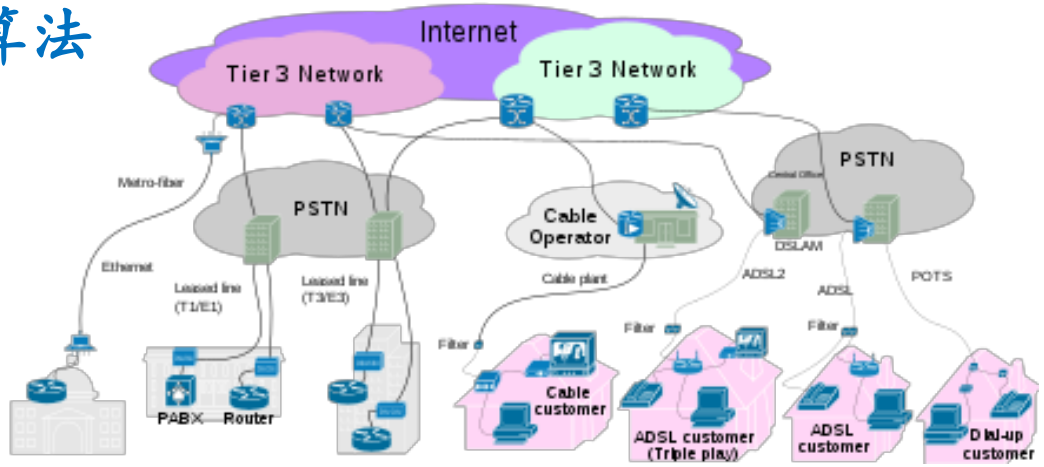
陈斌 北京大学 gischen@pku.edu.cn

最短路径问题：介绍

- 当我们通过网络浏览网页、发送电子邮件、QQ消息传输的时候，数据会在联网设备之间流动

  计算机网络专业领域会详尽地研究网络各层面上的技术细节

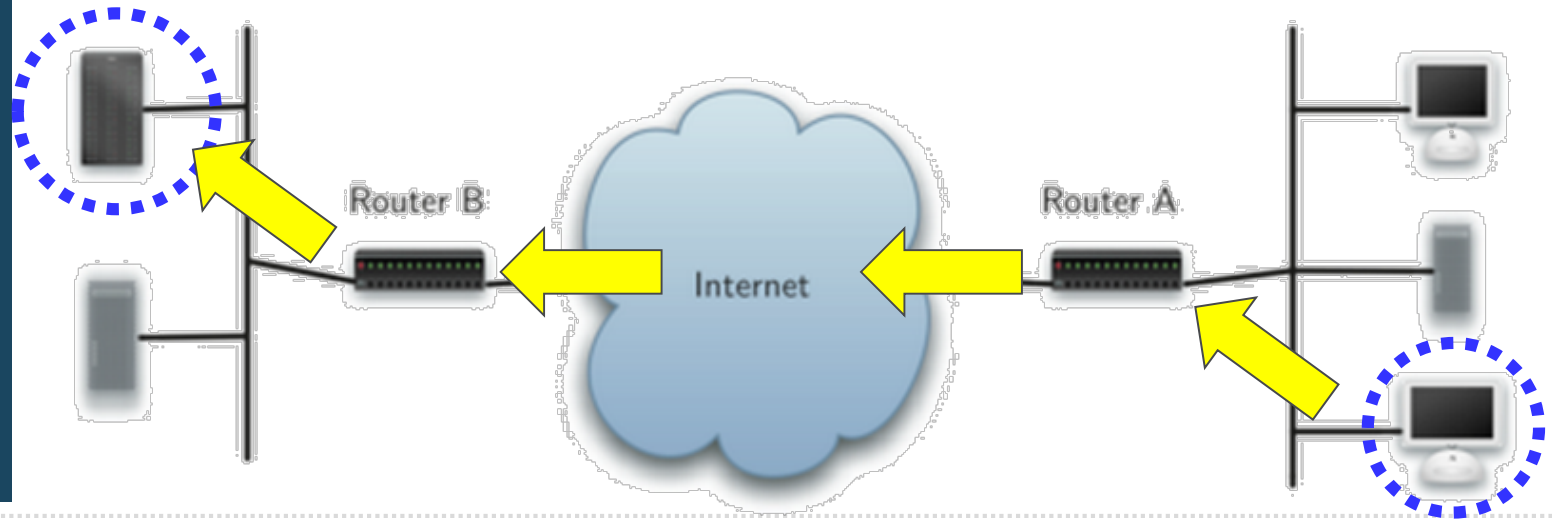
  我们对Internet工作方式感兴趣的主要是其中包含的图算法



最短路径问题：介绍

- 如图，当PC上的浏览器向服务器请求一个网页时，请求信息需要：

  先通过本地局域网，由路由器A发送到Internet，请求信息沿着Internet中的众多路由器传播，最后到达服务器本地局域网所属的路由器B，从而传给服务器。



最短路径问题：介绍

- 标注“Internet”的云状结构，实际上是一个由路由器连接成的网络
- 这些路由器各自独立而又协同工作，负责将信息从Internet的一端传送到另一端。
- 我们可以通过“traceroute”命令来跟踪信息传送的路径：

```
traceroute www.lib.pku.edu.cn
```

最短路径问题：介绍

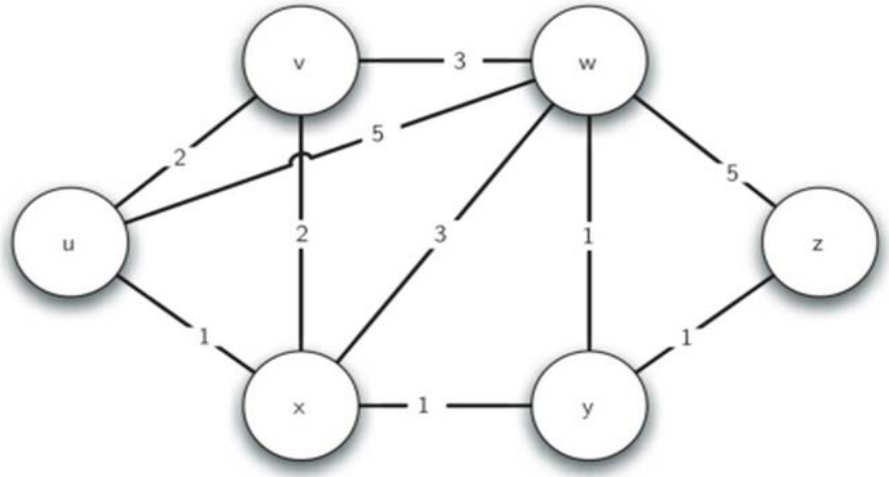
- 我们来看看从本机到北大图书馆服务器之间的一条路由器路径，包含了4个路由器

```
traceroute to www.lib.pku.edu.cn (162.105.138.158), 64 hops max,
 1  rt-ac54u.lan (192.168.123.1)  1.299 ms  1.042 ms  1.026 ms
 2  10.128.224.1 (10.128.224.1)  4.148 ms  1.311 ms  1.266 ms
 3  162.105.253.237 (162.105.253.237)  1.288 ms  1.207 ms  1.23
 4  162.105.253.94 (162.105.253.94)  1.575 ms  78.726 ms  284.6
 5  www.lib.pku.edu.cn (162.105.138.158)  1.861 ms  1.982 ms  1.
```

- 由于网络流量的状况会影响路径选择算法，在不同的时间，路径可能不同。

最短路径问题：介绍

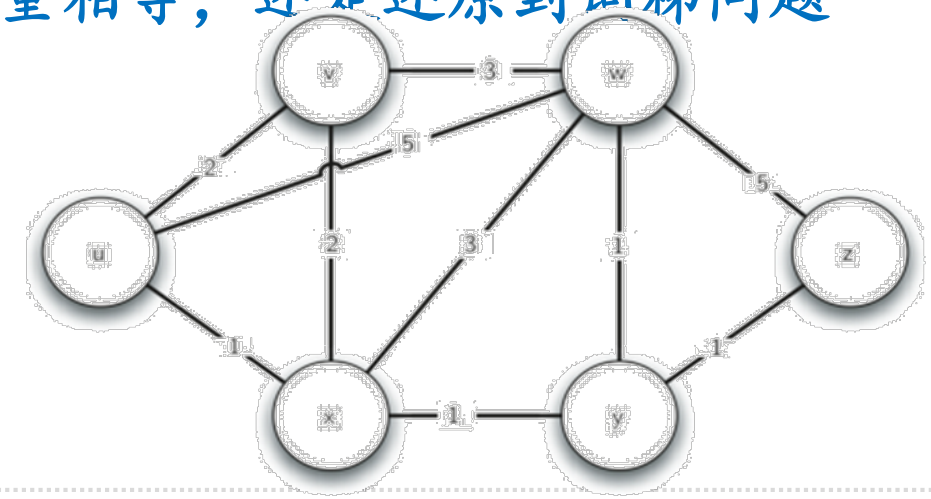
- 所以我们可以将互联网路由器体系表示为一个带权边的图
  - 路由器作为顶点，路由器之间网络连接作为边
  - 权重可以包括网络连接的速度、网络负载程度、分时段优先级等影响因素
  - 作为一个抽象，我们把所有影响因素合成为单一的权重



最短路径问题：介绍

- 解决信息在路由器网络中选择传播速度最快路径的问题，就转变为在**带权图上最短路径**的问题。
- 这个问题与广度优先搜索BFS算法解决的词梯问题相似，只是在边上增加了权重

如果所有权重相等，还是还原到词梯问题



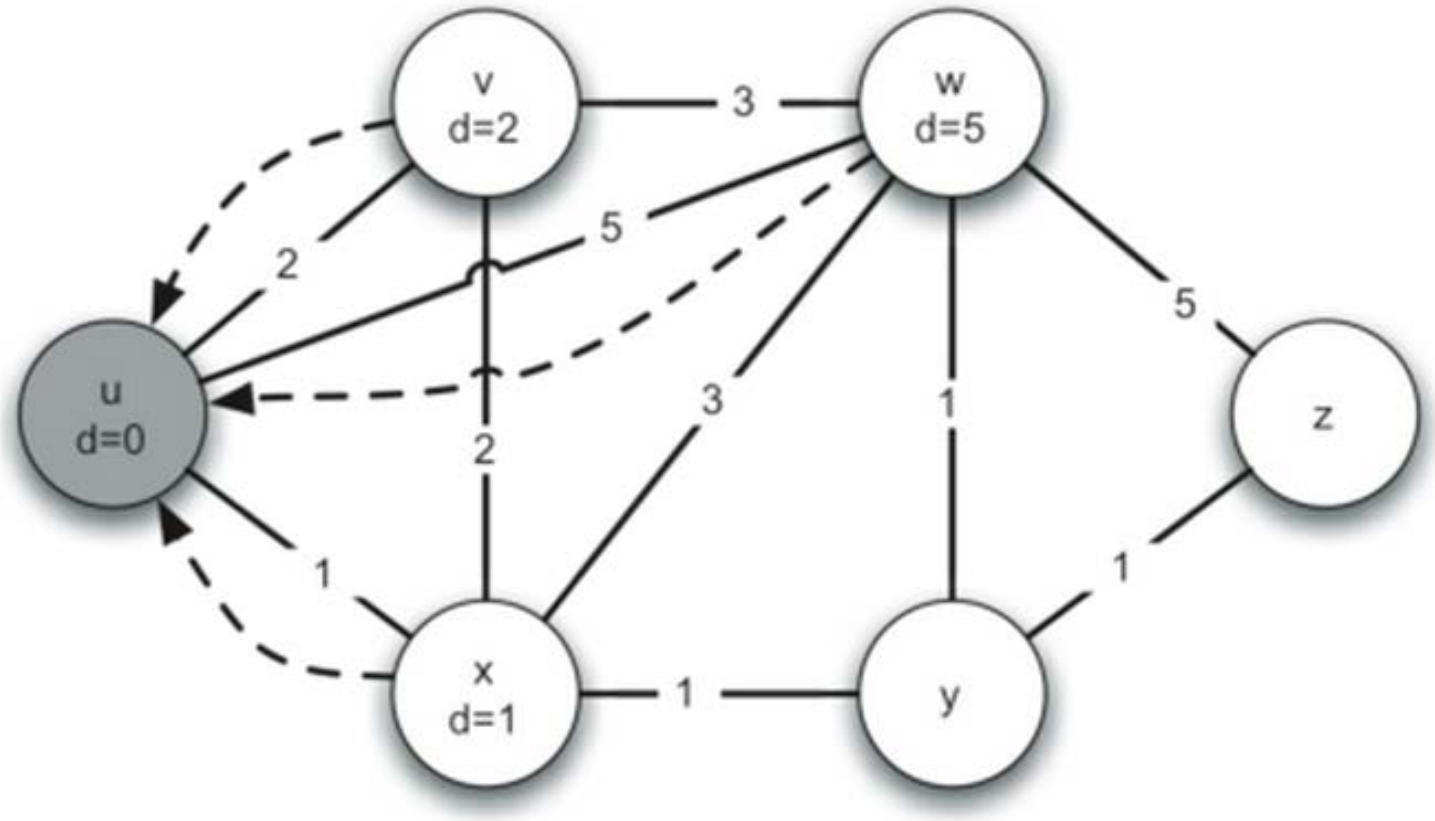
最短路径问题：Dijkstra算法

- 解决带权最短路径问题的经典算法是以发明者命名的“**Dijkstra算法**” /ˈdɛɪkstra/
- 这是一个迭代算法，得出从一个顶点到其余所有顶点的最短路径，很接近于广度优先搜索算法BFS的结果
- 具体实现上，在顶点Vertex类中的成员**dist**用于记录从开始顶点到本顶点的最短带权路径长度（**权重之和**），算法对图中的每个顶点迭代一次

最短路径问题：Dijkstra算法

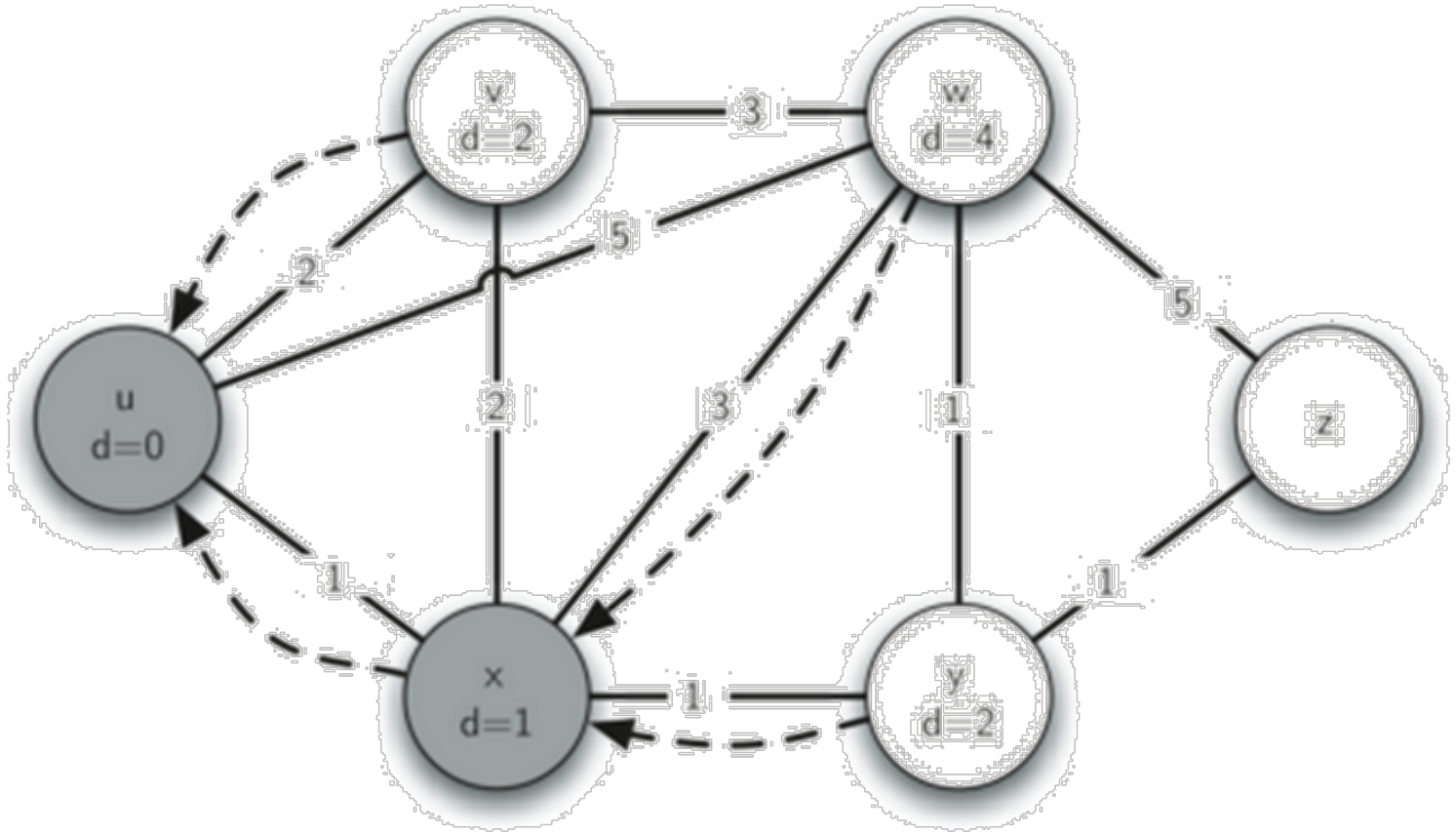
- 顶点的访问次序由一个**优先队列**来控制，队列中作为优先级的是顶点的dist属性。
- 最初，只有**开始顶点**dist设为0，而其他所有顶点dist设为sys.maxsize（最大整数），全部加入优先队列。
- 随着队列中每个**最低dist**顶点率先出队
- 并计算它与邻接顶点的权重，会引起其它顶点dist的减小和修改，引起堆重排
- 并据更新后的dist优先级再依次出队

最短路径问题：Dijkstra算法实例

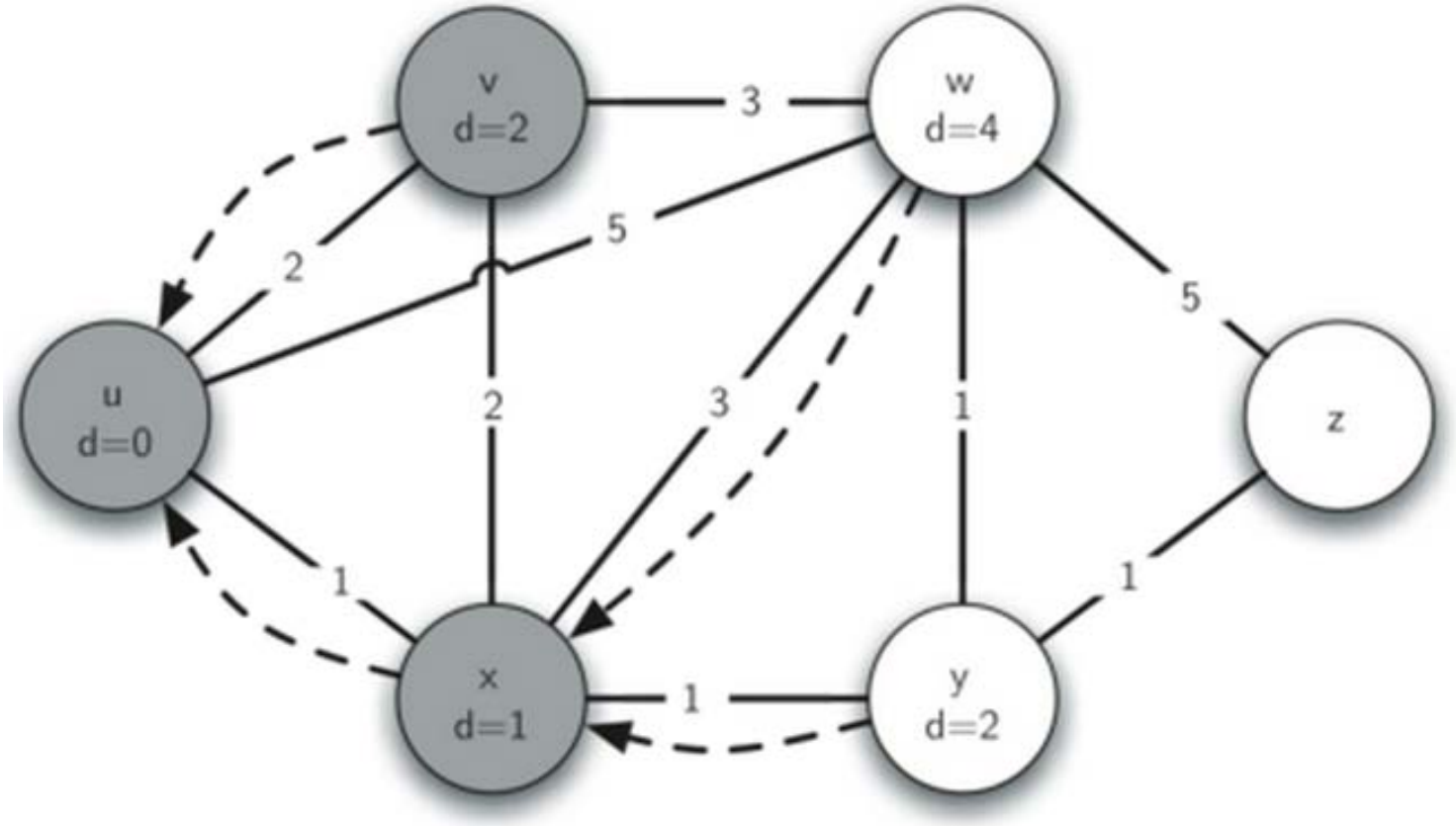


PQ = x,v,w

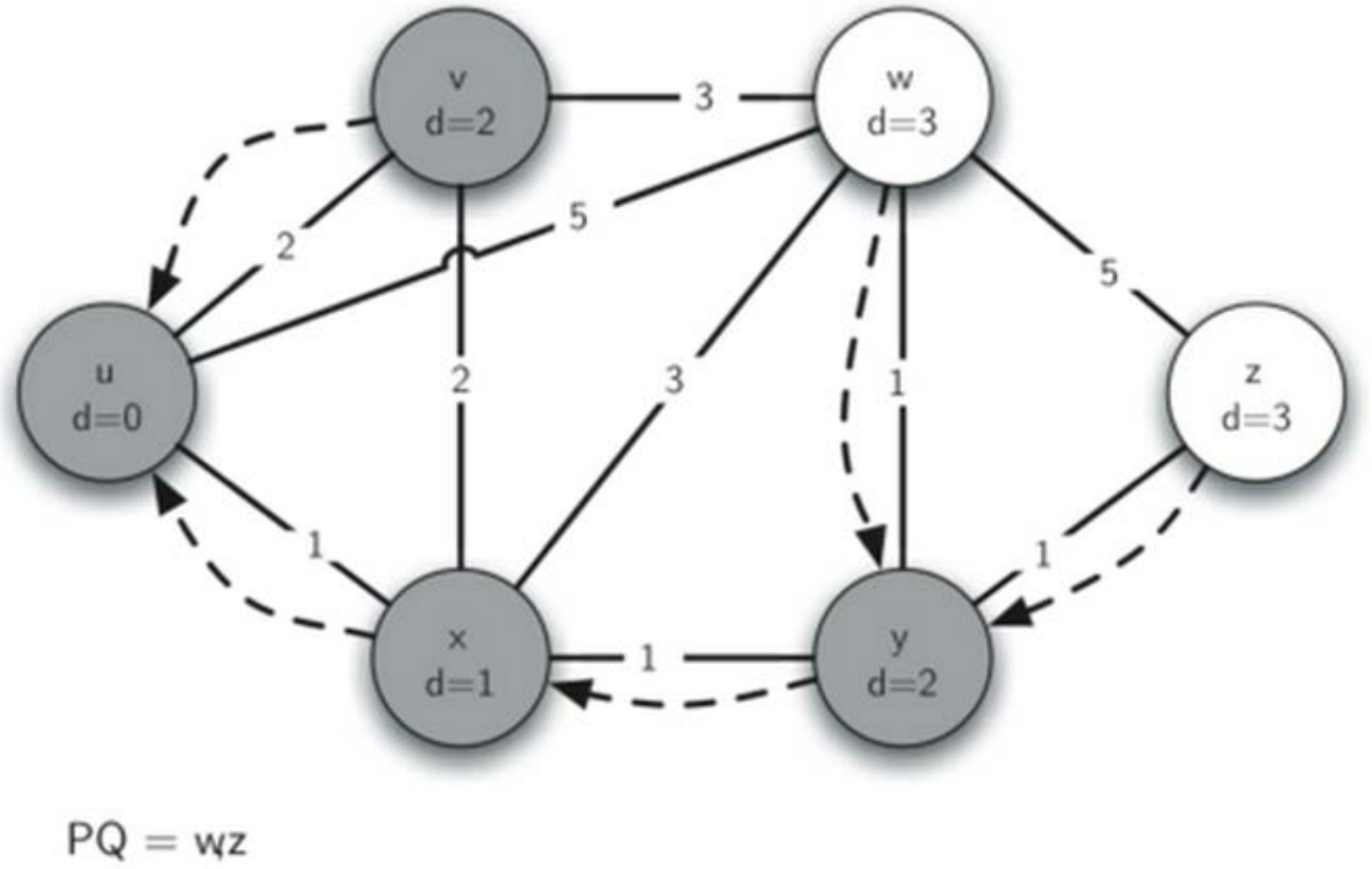
最短路径问题：Dijkstra算法实例



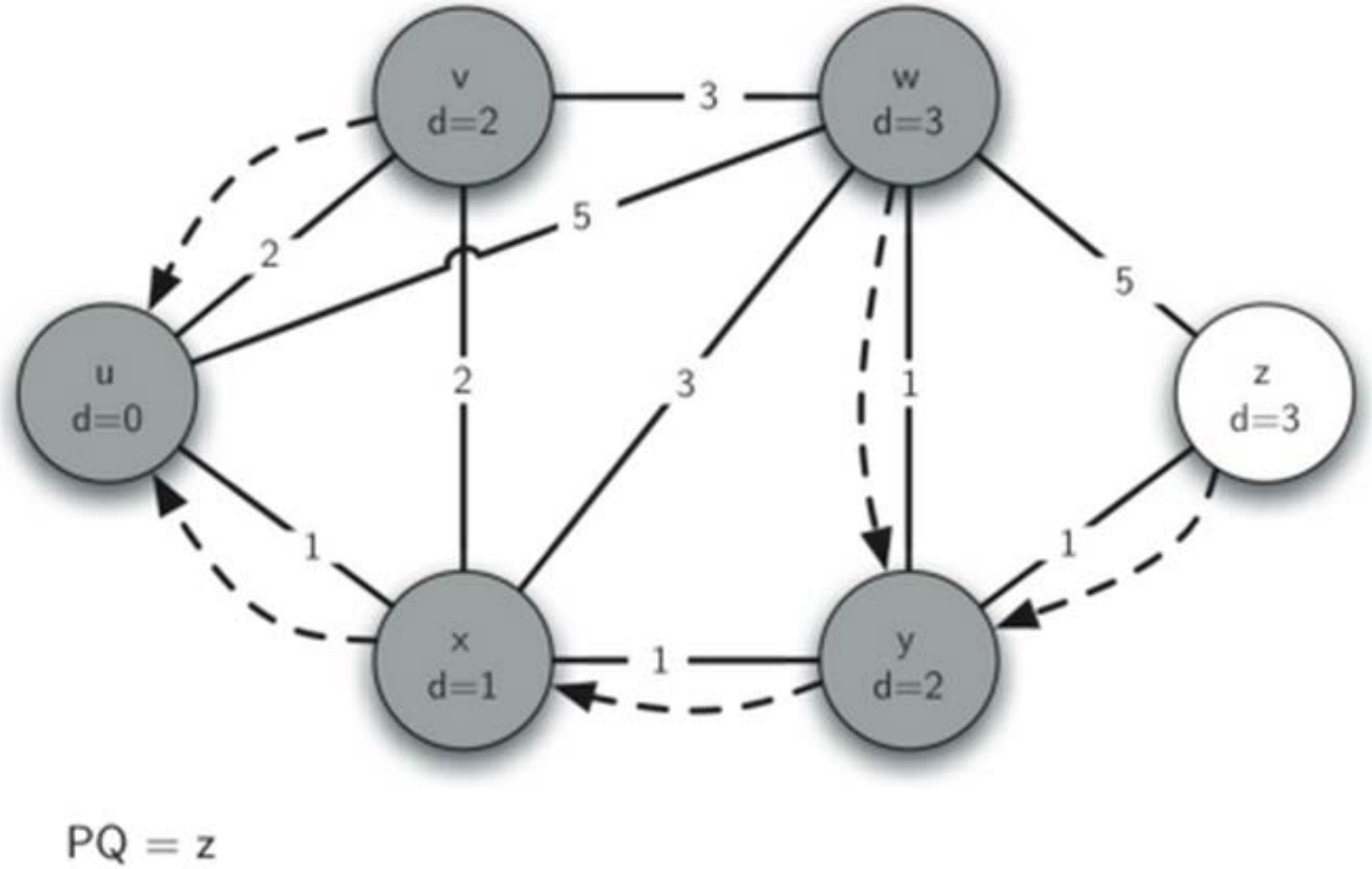
最短路径问题：Dijkstra算法实例



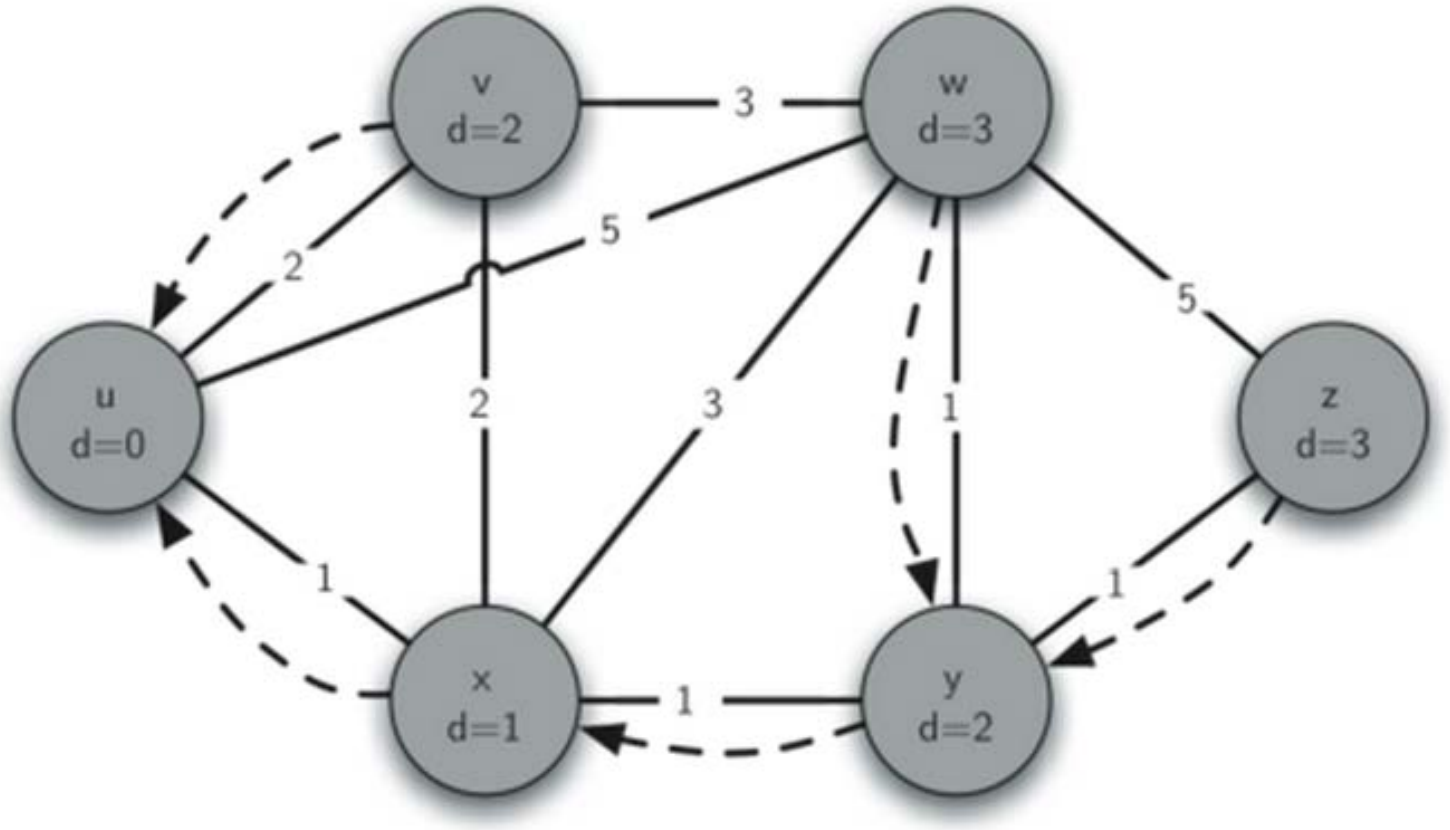
最短路径问题：Dijkstra算法实例



最短路径问题：Dijkstra算法实例



最短路径问题：Dijkstra算法实例



最短路径问题：Dijkstra算法代码

对所有顶点建堆，形成优先队列

优先队列出队

修改出队顶点所邻接顶点的dist，并逐个重排队列

```
from pythonds.graphs import PriorityQueue, Graph, Vertex
def dijkstra(aGraph,start):
    pq = PriorityQueue()
    start.setDistance(0)
    pq.buildHeap([(v.getDistance(),v) for v in aGraph])
    while not pq.isEmpty():
        currentVert = pq.delMin()
        for nextVert in currentVert.getConnections():
            newDist = currentVert.getDistance() \
                    + currentVert.getWeight(nextVert)
            if newDist < nextVert.getDistance():
                nextVert.setDistance( newDist )
                nextVert.setPred(currentVert)
                pq.decreaseKey(nextVert,newDist)
```

最短路径问题：Dijkstra算法

- 需要注意的是，Dijkstra算法只能处理大于0的权重

  如果图中出现负数权重，则算法会陷入无限循环

- 虽然Dijkstra算法完美解决了带权图的最短路径问题，但实际上Internet的路由器中采用的是其它算法（☺）

最短路径问题：Dijkstra算法

- 其中最重要的原因是，Dijkstra算法需要具备整个图的数据，但对于Internet的路由器来说，显然无法将整个Internet所有路由器及其连接信息保存在本地

  这不仅是数据量的问题，Internet动态变化的特性也使得保存全图缺乏现实性。

- 路由器的选径算法（或“路由算法”）对于互联网极其重要，有兴趣可以进一步参考“距离向量路由算法“

  https://baike.baidu.com/item/距离向量路由算法

最短路径问题：Dijkstra算法分析

- 首先，将所有顶点加入优先队列并**建堆**，时间复杂度为O(|V|)
- 其次，每个顶点仅出队1次，每次**delMin**花费O(log|V|)，一共就是O(|V|log|V|)
- 另外，每个边关联到的顶点会做一次**decreaseKey**操作（O(log|V|)），一共是O(|E|log|V|)
- 上面三个加在一起，数量级就是O((|V|+|E|)log|V|)