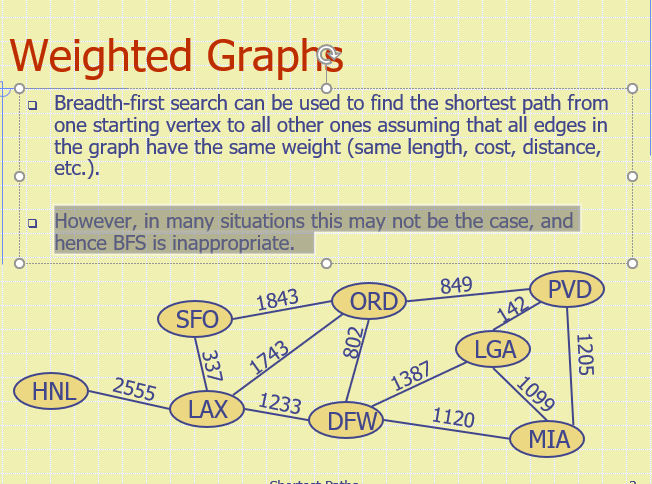
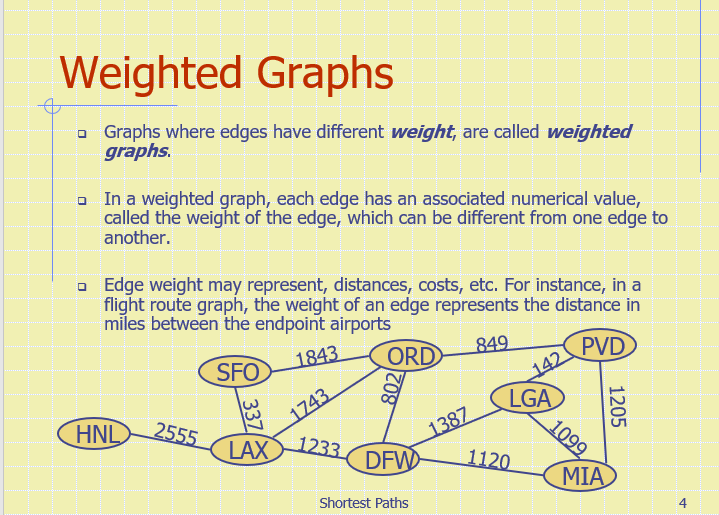
Weighted Graph

BFS 可以用来寻找从起始vertex到别的之间的shortest path，如果每个edge有着相同的weight权值（长度相同，cost,distance...）就edge没有本质区别

然而事实上，很多时候我们做不到这一点，所以用BFS是不合适的





那些edge有着不同weight权值的graph称作weighted graph，加权图

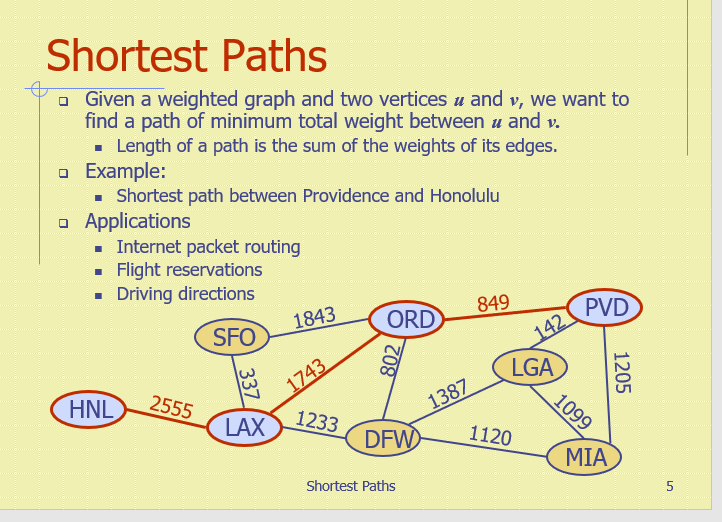
在weighted graph中，每个edge都有一个对应的数值，叫做the weight of the edge，每个edge的weight不同

weight可以代表距离,消费，等等

shortest path

给你weighted graph中的两个vertice u和v，最短路径指的是u v之间最少的总weight

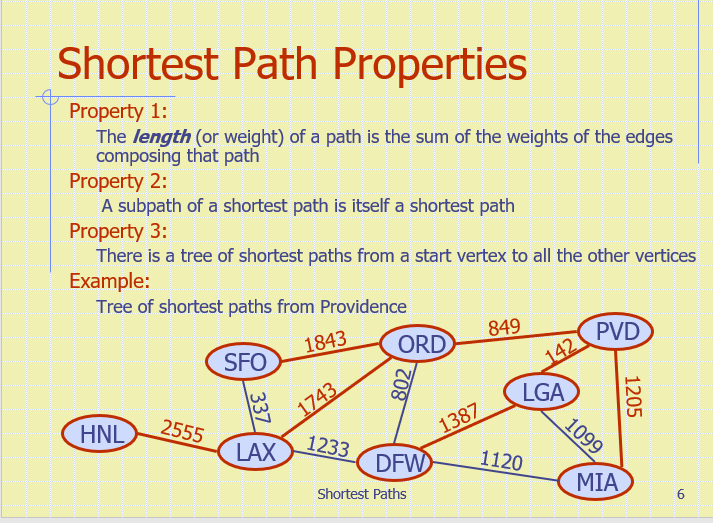
path的Length等于他的edge的weight相加



性质1，path的length/weight 是他所有的edge的weight和

性质2.shortest path的一部分对自己来说本身就是shortest path

性质3.以一个vertice为start，对每个vertice求shortest path，最终会构成一个tree



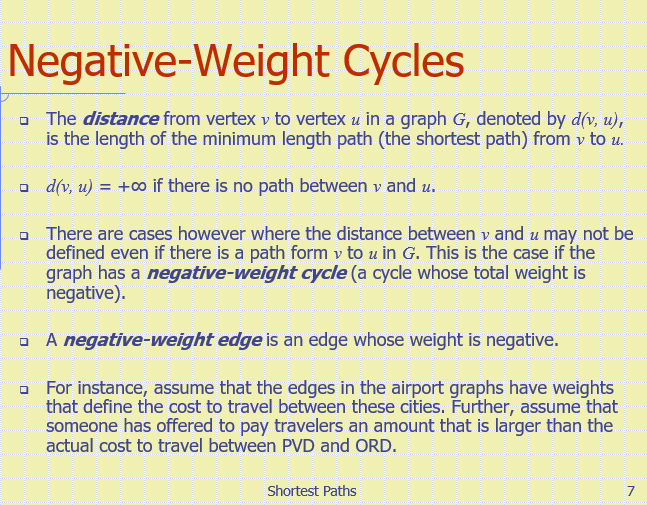
Negative-Weight Cycles

vertex v与vertex u之间的distance记作 d（v,u）,指的是v到u之间最短路径的长度

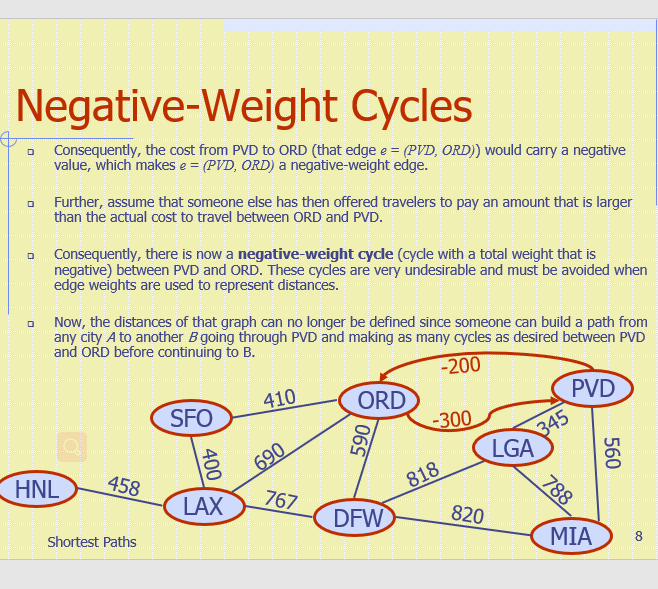
d(v,u)等于正无穷如果vu之间没有path

然而，在某些情况下，u v之间的distance无法定义，即使 有一条path从v到u，这种情况就是graph有一个negative-weight cycle

negative-weight cycle：一个循环，总weight是负数  
negative-weight edge:一个edge，他的weight是负数



例如，假设edge代表着两个城市飞行所花的钱，假设有一个人已经花了超过实际消费的钱当他在PVD与ORD之间travel的时候



那么ORD与PVD之间就会有一个negative value，让e成为了negative-weight edge

那么这就形成了一个negative-weight cycle

现在整张图的distance都没法再定义，因为你从A到B，完全可以经过negative-weight cycle，无限循环无限减少distance

Dijkstra's 算法

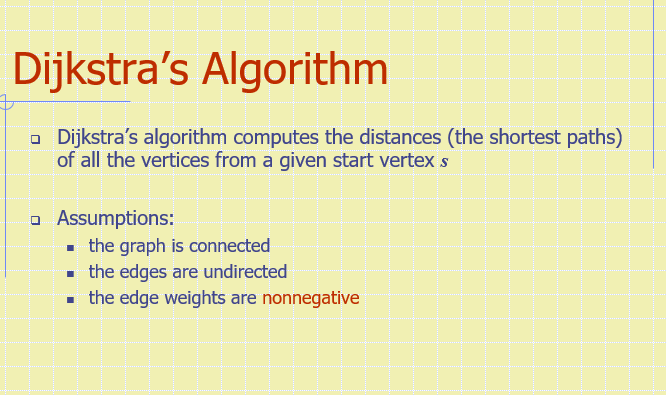
这个算法能用来计算任意终点的shortest path当你给了起始点x

假设：

这个graph是 connected

这个 edge是没有方向的

edge的weight>=0



Edge Relaxation

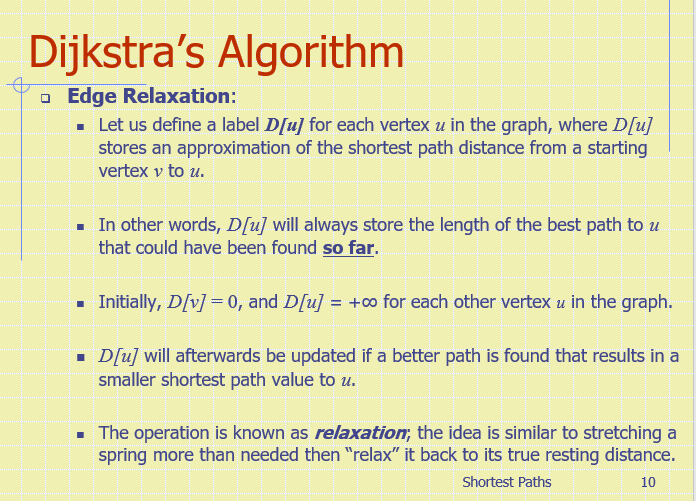
描述一个D[u]对graph中的每个vertex u，D[u]里面存放着v到u里distance值的估算

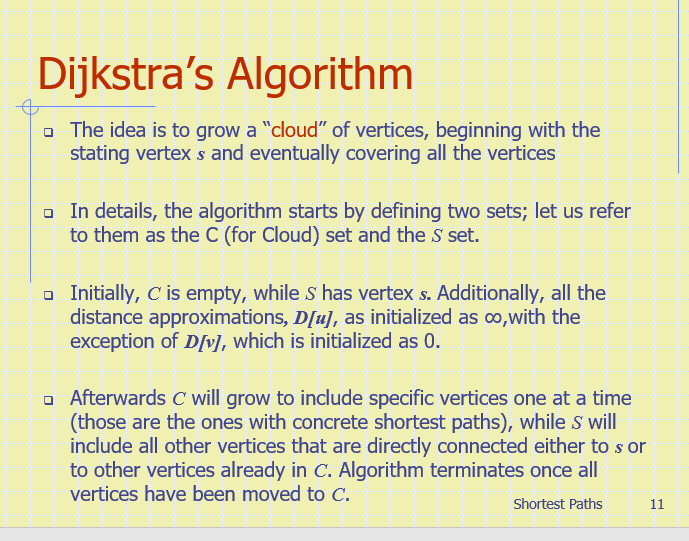
换句话说，D[u]将会存储至今为止到u的distance最优解

一开始D[V]=0//初始，其他的D[u]都等于正无穷

D[u]会被改变当找到了一个更好的path，

这个操作叫做relaxation松弛，这个想法类似于将弹簧拉伸到超出需要的长度，然后“放松”到它真正的静止距离。





另一个重要思想：培养一个cloud，

他由状态vertex s开始，最终cover所有vertice

这个算法有两个set，第一个叫做C(cloud)，第二个叫做S set

一开始，C是empty的，S有着vertex x,另外，所有distance被设置成正无穷，D[V]设置成0

接着，C每次增长一个特殊vertice，（可以创建最短路径的vertice），同时S将包含所有其他与s或者C里的vertice直接相连的vertice，算法结束当所有的vertice都移到C里

这个算法把S的元素移到C以以下方式：

1.把一个vertex 从S移到C，假设这个vertex是X，那么这个X必须满足

（1）.他必须直接与v或者于已存在于C中的vertex直接相连

（2）：与v所有相连的点中，他经过所有潜在路径，对V的distance最近  
2.当X从S移到C时，他的distance就是到v的最小distance

3.