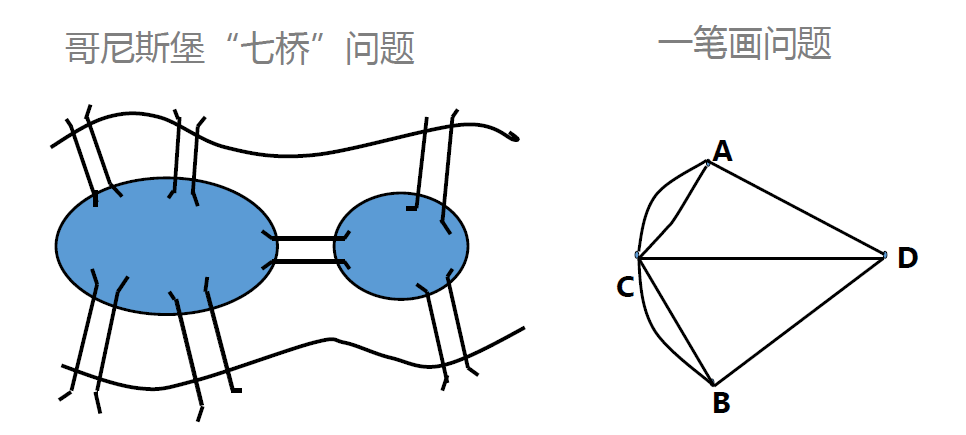
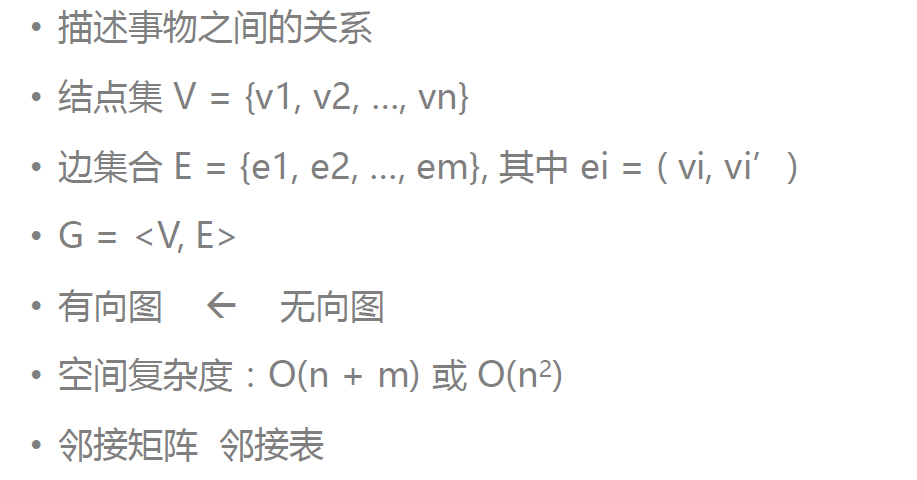
# 一笔画问题

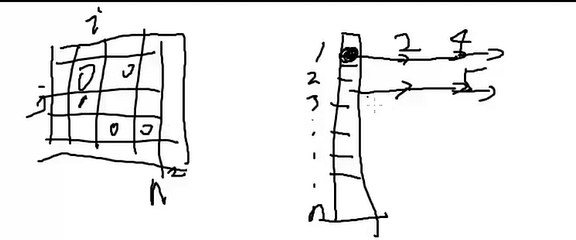


奇数点:路径是奇数

图中全是奇数点，但是如果你想一笔画完。对于起点和终点之外的点，你必须一进一出，所以必须是偶数点，所以此图是无解的

# 图的定义





N代表点的数量，m代表边的数量

空间复杂度:邻阶矩阵:n的平方，邻阶表:n+m.

邻阶矩阵:i和j之间有边相连就是1

邻阶表:数组中的每一个元素都是一个链表，存储了与i相连的点

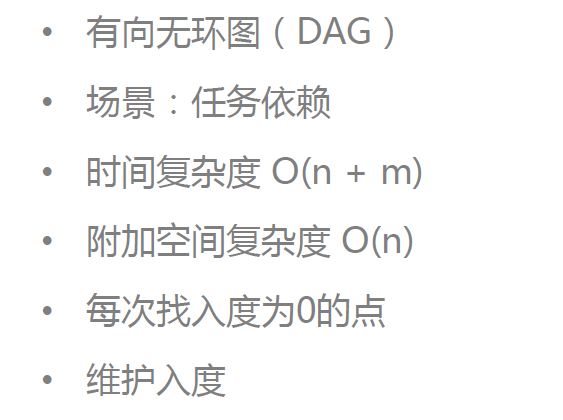
当边很稀疏的时候,邻阶矩阵会有大量的0,浪费空间

当边很多的时候,用左边的就更好

时间复杂度: 邻阶矩阵为n的平方，邻阶表:n+m，当m为n的平方时，两者的复杂度是一样的

# 拓扑排序

无环:不能从一个点回到自身



入度:有几个点依赖它

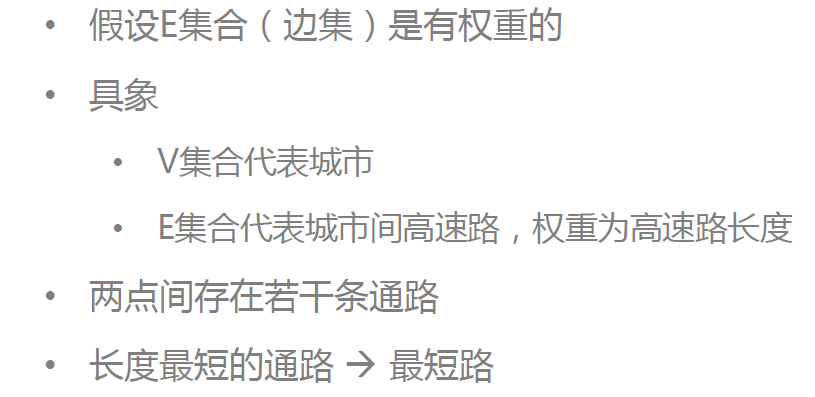
均摊:一共只有n个点会被加入到队列，每个点只会加入一次。入度为0的点只会出现在被维护的点上，无需遍历所有点

过程:从入度为0的点开始，将入度为0的加入队列，维护剩下的点的入度，将为0的点继续加入到队列里，继续维护剩下的点

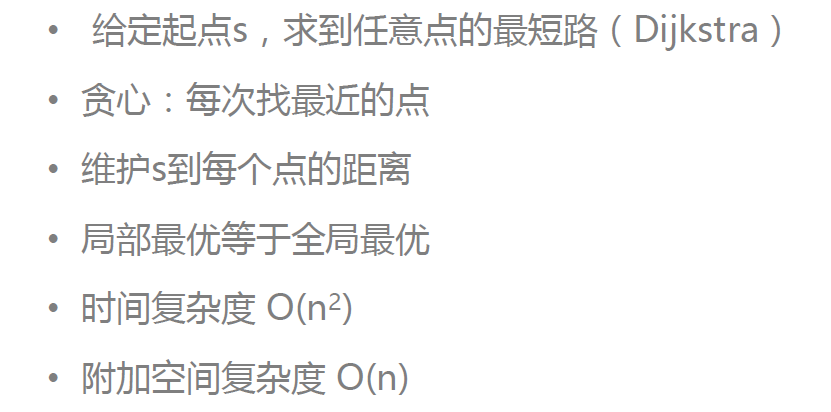
问题:假设无限并行，最少要多少时间

当出现同时有多个节点入度为0的时候，分别让不同的人去执行(正常情况是随机选一个)，计算各自的完成时间，选一个最小的

# 最短路



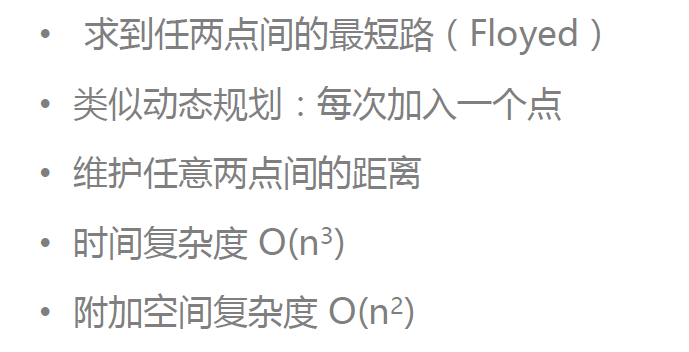
单源最短路:迪杰斯特算法



如果当前的决策不会影响你未来的决策，那么局部最优等于全局最优

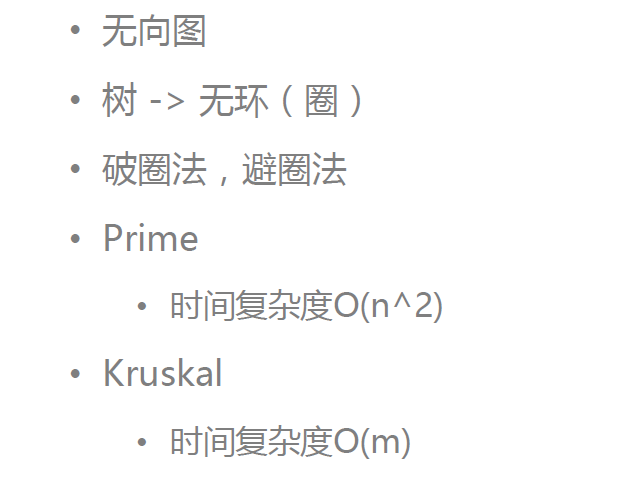
Dijkstra算法的核心:已经算过的两个点可以合并

任意两点最短路:Floyed算法



原来是一张很少边的图，进过不停的连接之后，我们连了一张很复杂的图，每次迭代没有损失信息(因为原本没有边的两点间的距离是无穷大)，所以在将来不会因为损失信息而损失最优解

# 最小生成树



# 10.知识图谱

知识图谱本质上是语义网络，是一种基于图的[**数据结构**](http://lib.csdn.net/base/datastructure)，由节点(Point)和边(Edge)组成。在知识图谱里，每个节点表示现实世界中存在的“实体”，每条边为实体与实体之间的“关系”。知识图谱是关系的最有效的表示方式。通俗地讲，知识图谱就是把所有不同种类的信息（Heterogeneous Information）连接在一起而得到的一个关系网络。知识图谱提供了从“关系”的角度去分析问题的能力。

应用:

1.**金融反欺诈**，需要利用[**机器学习**](http://lib.csdn.net/base/machinelearning)、[**自然语言**](http://lib.csdn.net/base/nlp)处理技术把这些数据变成结构化的数据。

2.**查询理解**

**3.自动问答**

**4.文档表示**

**需要的技术:**

**1.** **实体链指:** **实体链指的主要任务有两个，实体识别（Entity Recognition）与实体消歧（Entity Disambiguation），都是自然语言处理领域的经典问题**

2. 关系抽取: 构建知识图谱的重要来源之一是从互联网网页文本中抽取实体关系。关系抽取是一种典型的信息抽取任务。我们可以利用[**机器学习**](http://lib.csdn.net/base/machinelearning)分类模型（如最大熵分类器、SVM等）构建信息抽取系统

3. **知识推理：**知识推理可以用于发现实体间新的关系

4. **知识表示**