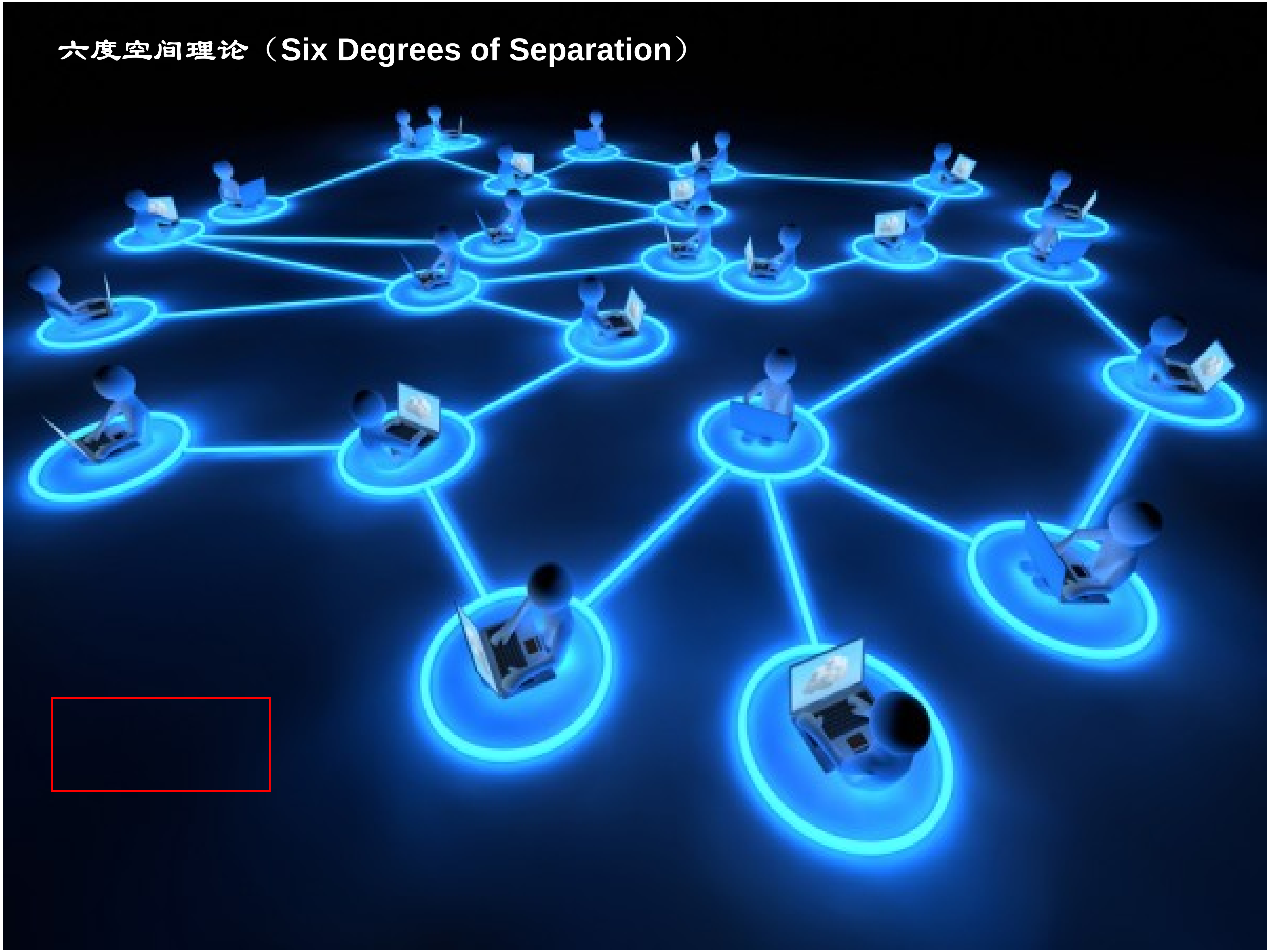
第六讲 图（上）

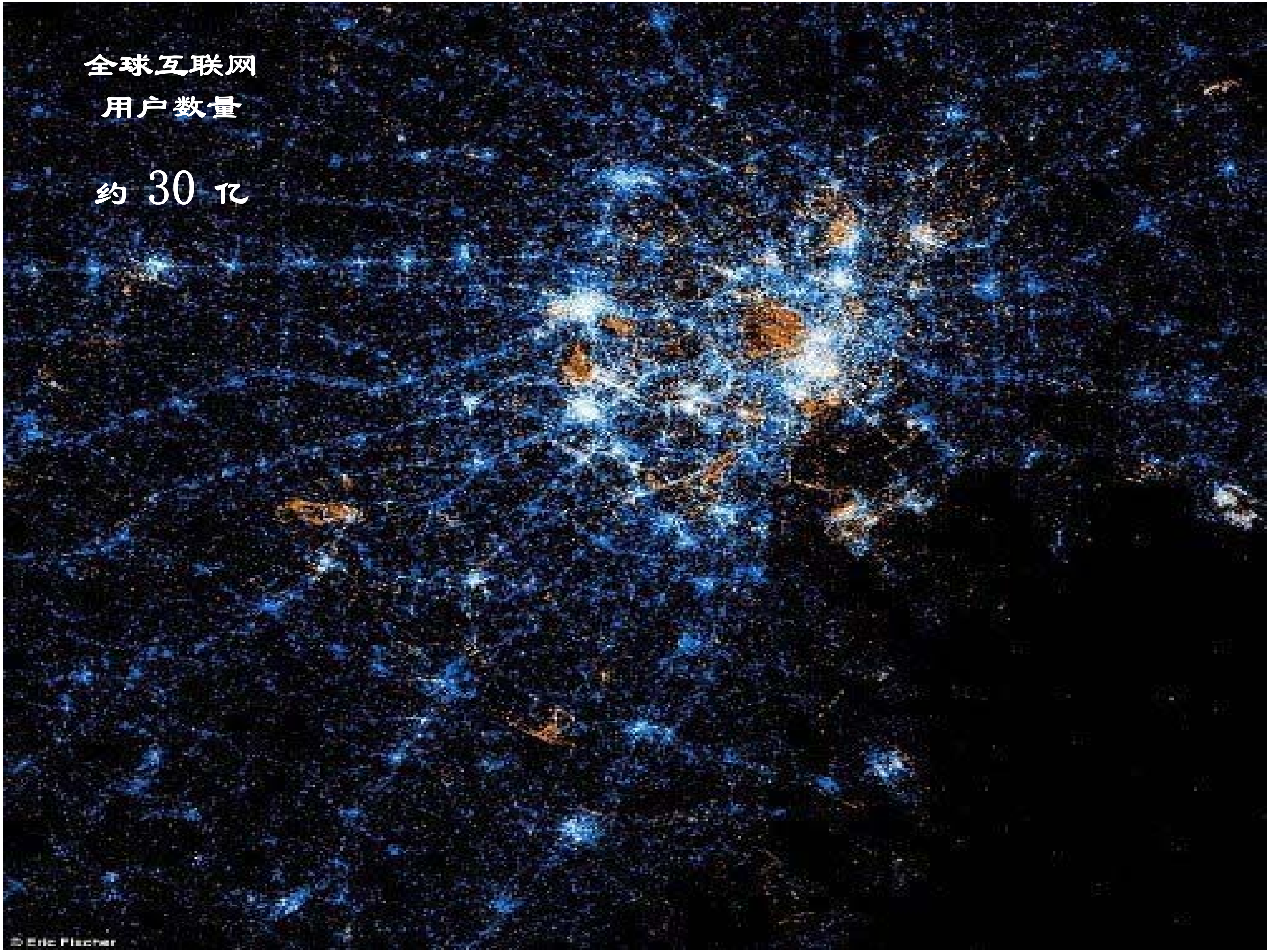
多对多的关系

浙江大学 陈 越

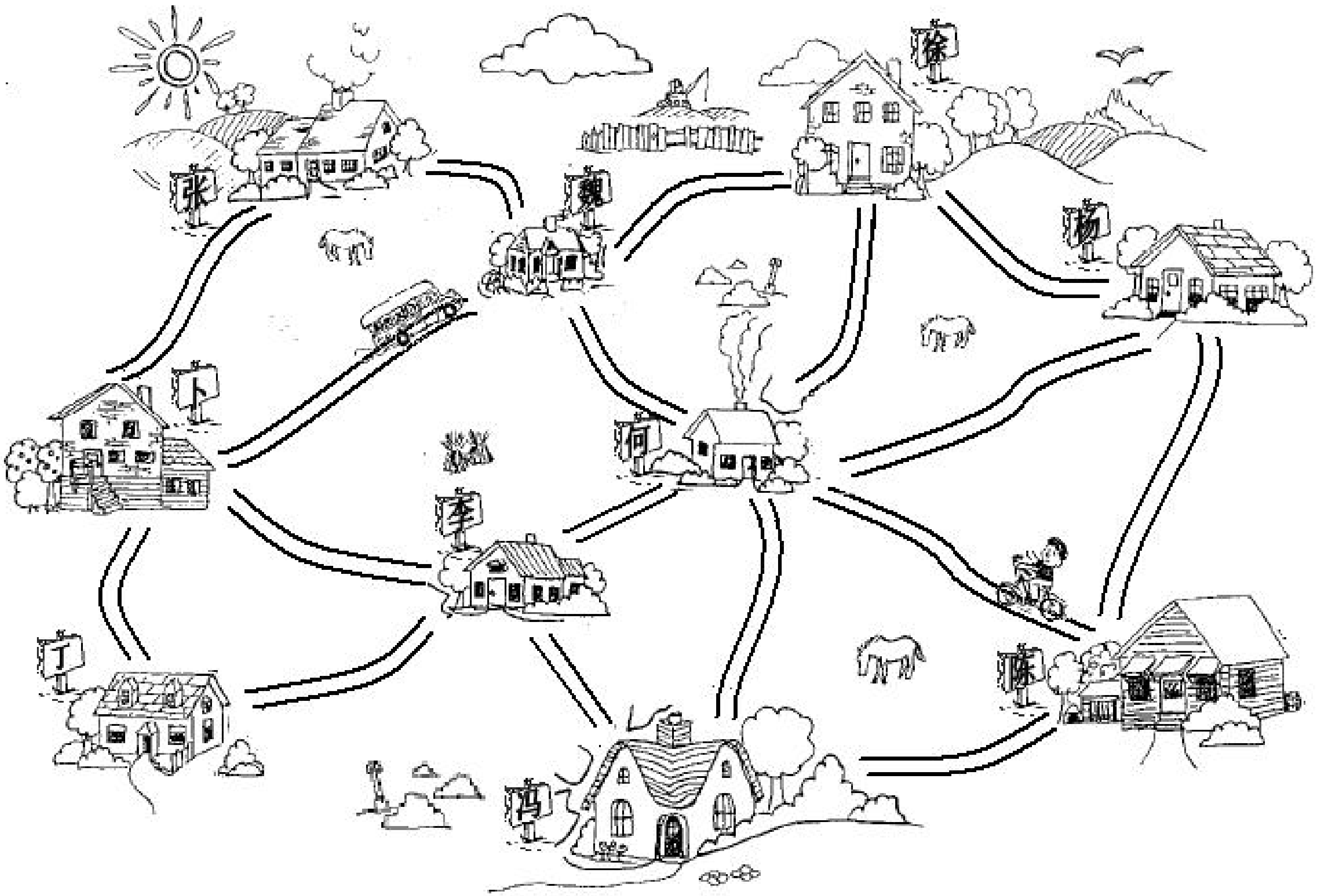
6.1 什么是图



不超过六个人就认识了



写程序解决六度空间的问题



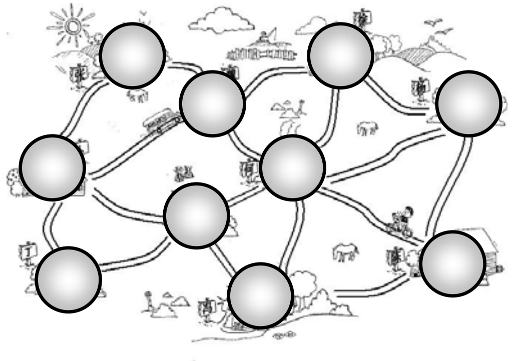
最短路径问题

最小生成树问题

从陈家庄到张家村，怎么走最快呢？
怎么修公路使得村村通的花费最少呢？

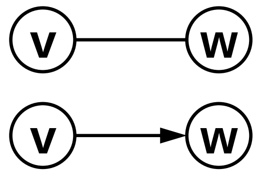
瞬间告诉我一个站到另一个站怎么走最省事

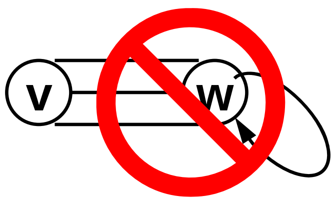
什么是“图”（Graph）



线性表和树都包含进去了
线性表和树是特殊的图

- 表示“多对多”的关系
- 包含
  - 一组顶点：通常用 V (Vertex) 表示顶点集合
  - 一组边：通常用 E (Edge) 表示边的集合
    - 边是顶点对：(v, w) ∈ E，其中 v, w ∈ V
    - 有向边 < v, w> 表示从v指向w的边（单行线）
    - 不考虑重边和自回路





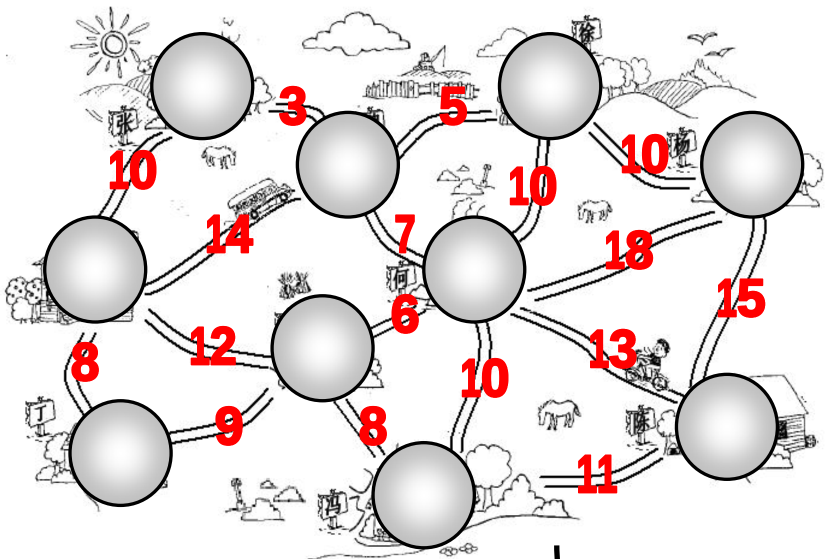
不可能指向自己

抽象数据类型定义

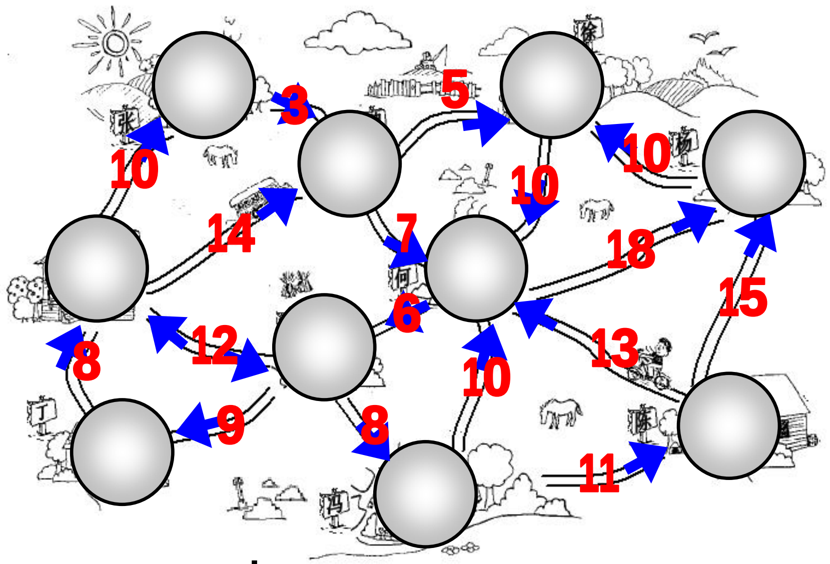
必须至少有一个顶点

- 类型名称：图（**Graph**）
- 数据对象集：**G(V,E)**由一个非空的有限顶点集合**V**和一个有限边集合**E**组成。
- 操作集：对于任意图 **G** ∈ **Graph**，以及 **v** ∈ **V**，**e** ∈ **E**
  - **Graph Create()**：建立并返回空图；
  - **Graph InsertVertex(Graph G, Vertex v)**：将**v**插入**G**；
  - **Graph InsertEdge(Graph G, Edge e)**：将**e**插入**G**；
  - **void DFS(Graph G, Vertex v)**：从顶点**v**出发深度优先遍历图**G**；
  - **void BFS(Graph G, Vertex v)**：从顶点**v**出发宽度优先遍历图**G**；
  - **void ShortestPath(Graph G, Vertex v, int Dist[])**：计算图**G**中顶点**v**到任意其他顶点的最短距离；
  - **void MST(Graph G)**：计算图**G**的最小生成树；
  - ……

常见术语



无 向 图



有 向 图

网 络

权重有各种各样的意义

关于图论的术语有很多

用到在介绍

还有好多，用到再说……

怎么在程序中表示一个图

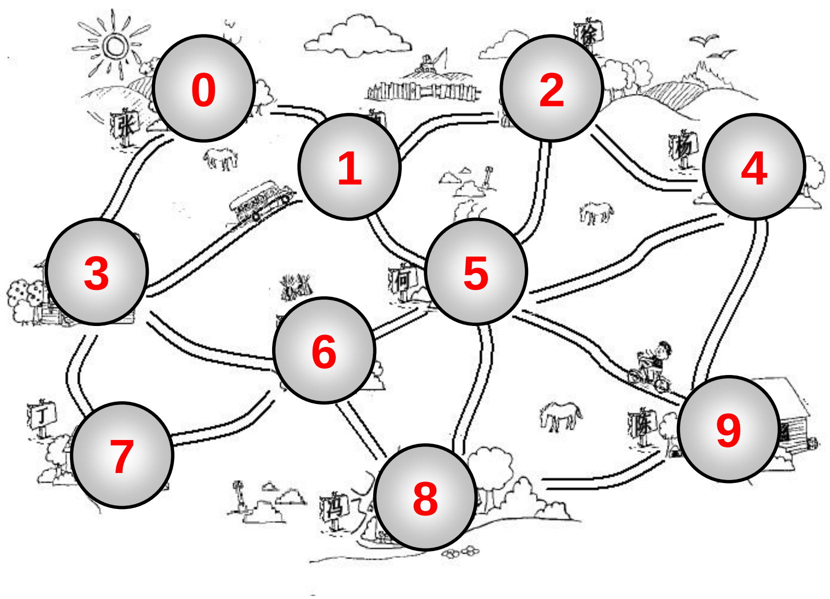
两个表示方法

二维的数组，邻接矩阵

- 邻接矩阵G[N][N]——N个顶点从0到N-1编号

$$G[i][j] = \begin{cases} 1 & \text{若} \langle v_i, v_j \rangle \text{是} G \text{中的边} \\ 0 & \text{否则} \end{cases}$$

这是有向边



$$\begin{array}{c|cccccccccc} & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 \\ \hline v_0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ v_2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ v_3 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ v_4 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ v_5 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ v_6 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ v_7 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ v_8 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ v_9 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{array}$$

不允许有自回路 因此对角线上均为0

矩阵是对称的
一条边存了两次

怎么在程序中表示一个图

- 邻接矩阵
  - 问题：对于无向图的存储，怎样可以省一半空间？

$$
\begin{array}{c}
\\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 \end{array} \\
\left(\begin{array}{cccccccccc}
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & \bullet & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0
\end{array}\right)
\end{array}
$$

用一个长度为N(N+1)/2的1维数组A存储 $\{G_{00}, G_{10}, G_{11}, \ldots\ldots, G_{n-1\ 0}, \ldots, G_{n-1\ n-1}\}$，则 G_{ij} 在A中对应的下标是：

$$(i*(i+1)/2 + j)$$

对于网络，只要把G[i][j]的值定义为边 $<v_i, v_j>$ 的权重即可。

问题：v_i 和 v_j 之间若没有边该怎么表示？

在这里是设置值为0

怎么在程序中表示一个图

- 邻接矩阵 —— 有什么好处？
  - ☑ 直观、简单、好理解
  - ☑ 方便检查任意一对顶点间是否存在边
  - ☑ 方便找任一顶点的所有“邻接点”（有边直接相连的顶点）有向无向均可
  - ☑ 方便计算任一顶点的“度”（从该点发出的边数为“出度”，指向该点的边数为“入度”）
    - 无向图：对应行（或列）非0元素的个数
    - 有向图：对应行非0元素的个数是“出度”；对应列非0元素的个数是“入度”

怎么在程序中表示一个图

- 邻接矩阵 —— 有什么不好？
  - ☑ 浪费空间 —— 存稀疏图（点很多而边很少）有大量无效元素
    - 对稠密图（特别是完全图）还是很合算的
  - ☑ 浪费时间 —— 统计稀疏图中一共有多少条边

有一大堆零

完全图 任意两定点都能直达

把所有元素都扫一遍 但是却找到很少的边 不划算

怎么在程序中表示一个图

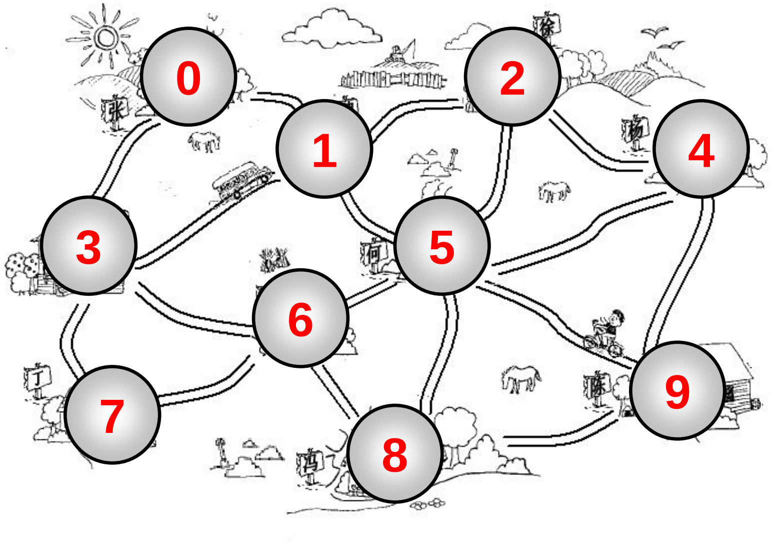
为了解决稀疏矩阵的问题 用邻接表

- 邻接表：**G[N]**为指针数组，对应矩阵每行一个链表，只存非**0**元素

对于网络，结构中要增加权重的域。

每一个节点开一个指针
几个节点以什么顺序出现是无所谓的

一条边必定存了两遍，不仅存了标号而且存了next，还有权重



G[0] → 1 → 3 → ●
G[1] → 5 → 3 → 0 → 2 → ●
G[2] → 1 → 5 → 4 → ●
G[3] → 7 → 1 → 0 → 6 → ●
G[4] → 2 → 5 → 9 → ●
G[5] → 2 → 1 → 4 → 6 → 8 → 9 → ●
G[6] → 5 → 8 → 7 → 3 → ●
G[7] → 6 → 3 → ●
G[8] → 9 → 5 → 6 → ●
G[9] → 4 → 5 → 8 → ●

一定要够稀疏才合算啊~~~~~

否则就很不合算

怎么在程序中表示一个图

- 邻接表
  - ☑ 方便找任一顶点的所有“邻接点”
  - ☑ 节约稀疏图的空间
    - 需要**N**个头指针 **+ 2E**个结点（每个结点至少**2**个域）
  - ☑ 方便计算任一顶点的“度”？
    - 对无向图：是的
    - 对有向图：只能计算“出度”；需要构造“逆邻接表”（存指向自己的边）来方便计算“入度”
  - ☑ 方便检查任意一对顶点间是否存在边？

    ☹ **No**

图的表示还有很多，取决于要解决的问题