

# 第五讲 图（上）

浙江大学 陈 越

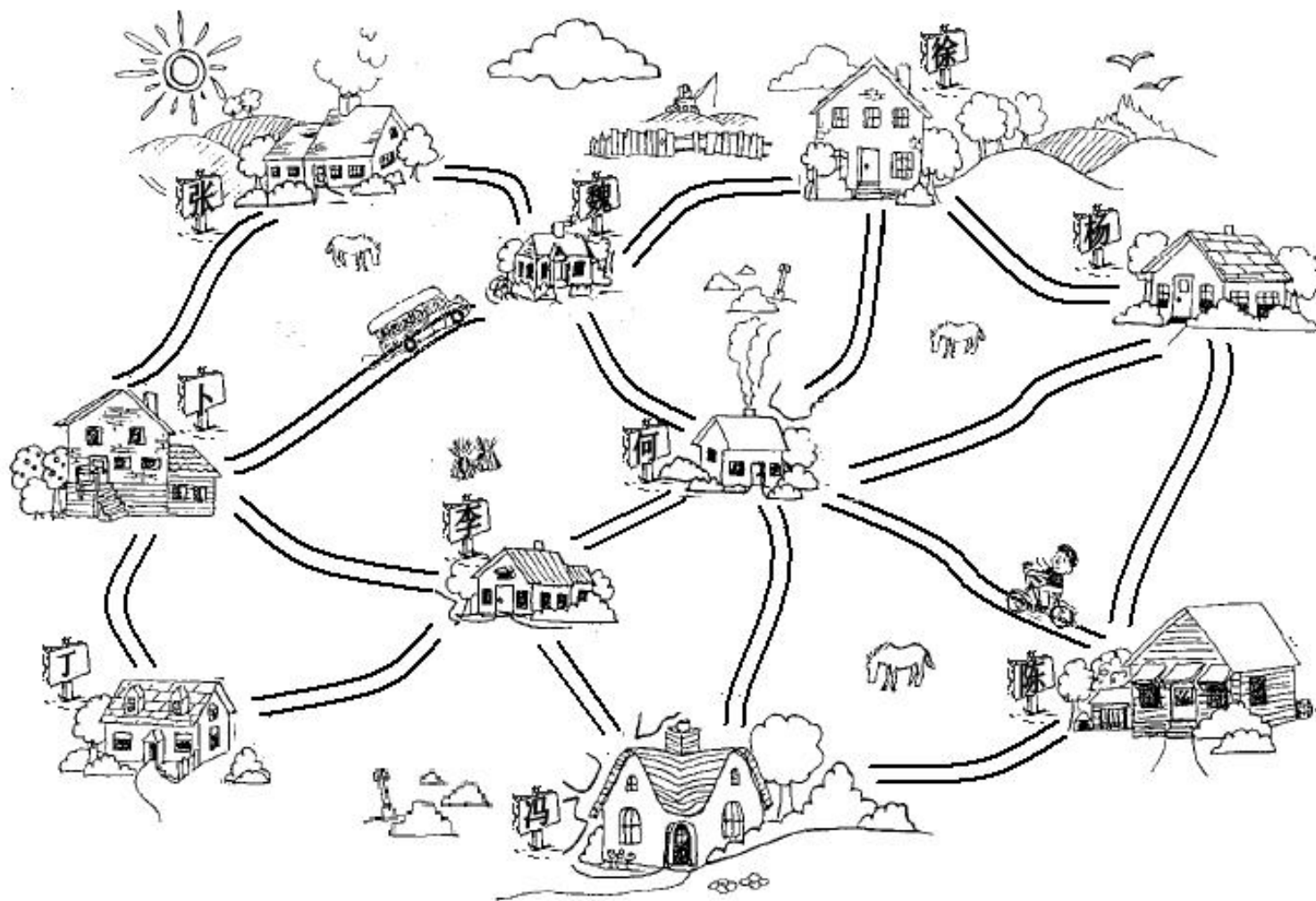
# 5.1 什么是图

## 六度空间理论 (Six Degrees of Separation)



全球互联网  
用户数量

约 30 亿

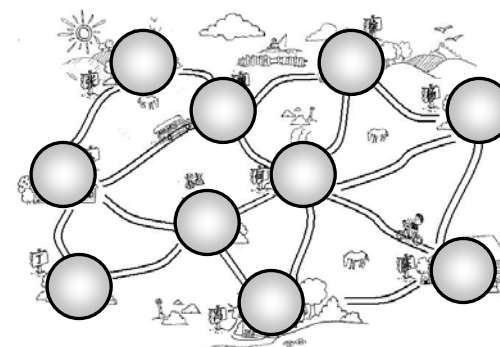


从陈家庄到张家村，怎么走最快呢？  
怎么修公路使得村村通的花费最少呢？



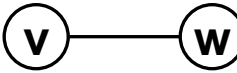
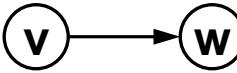


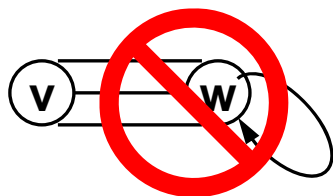
# 什么是“图” (Graph)



- 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：  $(v, w) \in E$ ，其中  $v, w \in V$  
- 有向边  $\langle v, w \rangle$  表示从  $v$  指向  $w$  的边（单行线） 
- 不考虑重边和自回路

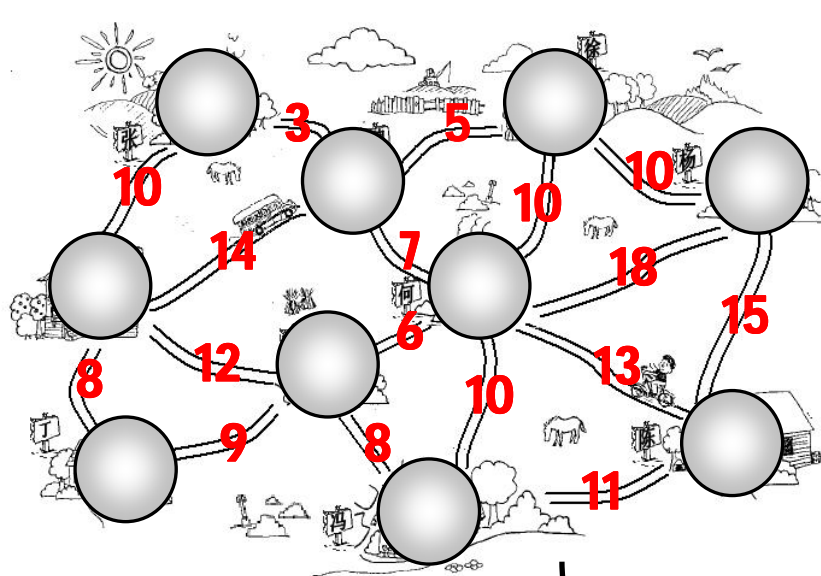


# 抽象数据类型定义

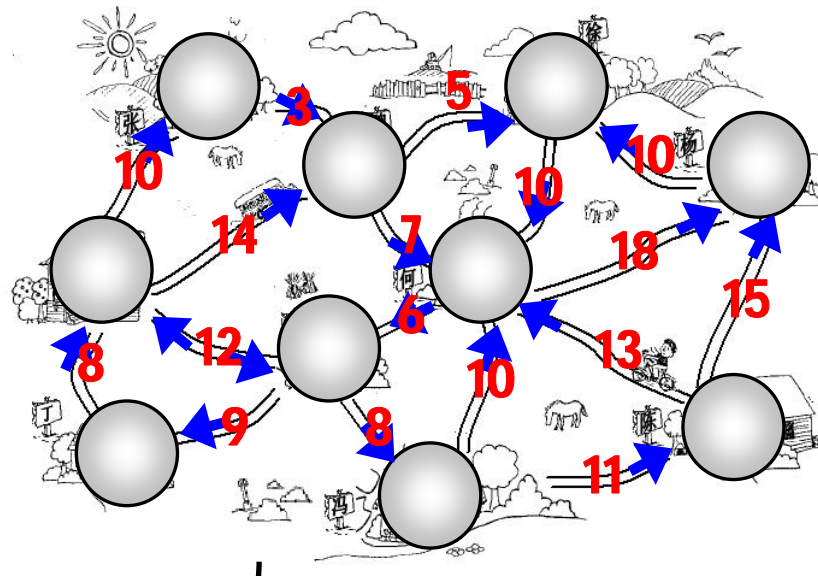
- 类型名称: 图 (**Graph**)
- 数据对象集:  $G(V, E)$  由一个非空的有限顶点集合  $V$  和一个有限边集合  $E$  组成。
- 操作集: 对于任意图  $G \in \text{Graph}$ , 以及  $v \in V, e \in E$ 
  - **Graph Create()**: 建立并返回空图;
  - **Graph InsertVertex(Graph G, Vertex v)**: 将  $v$  插入  $G$ ;
  - **Graph InsertEdge(Graph G, Edge e)**: 将  $e$  插入  $G$ ;
  - **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

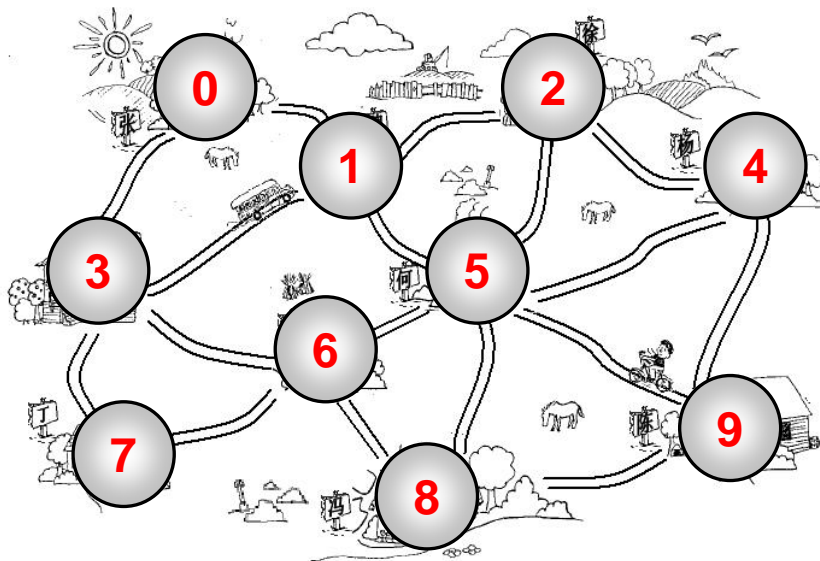
网络

还有好多，用到再说.....

# 怎么在程序中表示一个图

- 邻接矩阵 $G[N][N]$ —— $N$ 个顶点从0到 $N-1$ 编号

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



	$v_0$	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$	$v_7$	$v_8$	$v_9$
$v_0$	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

# 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？

	$v_0$	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$	$v_7$	$v_8$	$v_9$
$v_0$	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	0	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

用一个长度为 $N(N+1)/2$ 的1维数组A存储  
 $\{G_{00}, G_{10}, G_{11}, \dots, G_{n-1, 0}, \dots, G_{n-1, n-1}\}$ ,  
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  
 $\langle v_i, v_j \rangle$ 的权重即可。

问题： $v_i$ 和 $v_j$ 之间若没有边该怎么表示？

# 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）
- ☑ 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

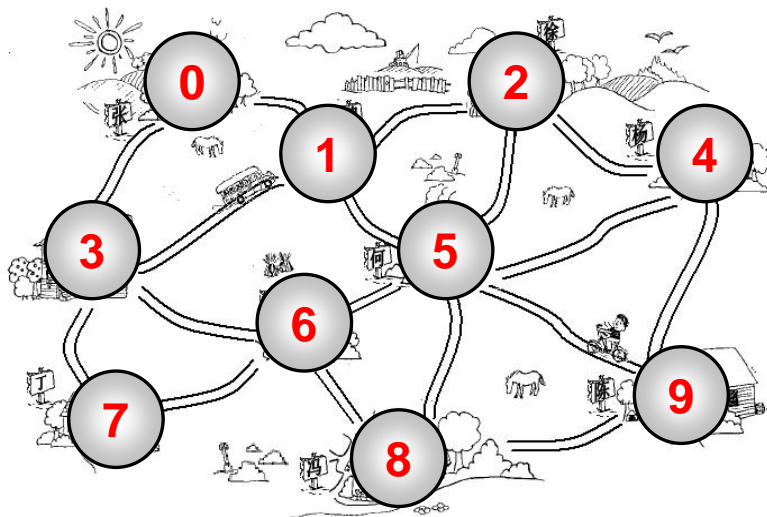
☑ 浪费时间 — 统计稀疏图中一共有多少条边



# 怎么在程序中表示一个图

- 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。



```
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~~~~~

# 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