# 离散数学

周建宇

2022 秋季学期

# 前言

你们最关心的事

- 期末考试改为大作业，占总成绩的 60%
- 平时成绩占 40%，为之前布置的作业，近期组织大家提交（拍照后提交电子版）
- 期末大作业的将会是报告或者小论文的形式，围绕这三周课程的内容，自由选题，不分小组，一人一份
- 大作业的具体要求与样例稍后发布
- 最后三周课程：津南校区线下 + 线上，泰达校区暂定线上（待风险区解封消息），线上采用飞书会议。

# 前言

我们这三周要做什么

**《离散数学》21 天补完计划**

# 前言

我们这三周要做什么

**《离散数学》21 天补完计划**


- 计算机科学的理论基石
- 一个学期、甚至一个学年都很难学完，更谈不上精通
- 无论如何，至少“不懂细节”“不会推导”“不会证明”好过“根本没听说过”
- 走马观花，留下印象：这三周只会讲是什么，至于为什么、怎样做则通过大作业对某个知识深入学习
- 包含：数理逻辑、集合论、信息论、数论、**组合数学**、**图论**、**抽象代数** ……

# 图　论

# 图论

**图论 (graph theory)**：刻画二元关系的强大模型

- 基本概念
- 特殊的图
- 连通性问题
- 生成树问题
- 路径问题
- 网络流问题
- 匹配问题
- 着色问题

# 图论：基本概念

**图 (graph) 是什么**：表示物体之间存在某种关系的**数学模型**

- 点 (vertex)：数学抽象后的“物体”
- 边 (edge)：数学抽象后的“某种关系”

——图的强大之处在于其建模能力，难点是如何将问题抽象为图

# 图论：基本概念

**图 (graph) 是什么**：表示物体之间存在某种关系的**数学模型**

- 点 (vertex)：数学抽象后的“物体”
- 边 (edge)：数学抽象后的“某种关系”

——图的强大之处在于其建模能力，难点是如何将问题抽象为图

**数学形式化表示**：图一般表示为二元组 $G = (V, E)$，其中 $V = \{v_1, v_2, \cdots, v_n\}$ 表示点的集合，$E$ 则表示边的集合。
注意边集有两种不同的表示，因此产生了两类图：

- **无向图** (undirected graph)
  $E = \{\{u, v\} | u, v \in V\}$，$\{x, y\}$ 是集合，与 $\{y, x\}$ 等价
- **有向图** (directed graph、digraph)
  $E = \{(u, v) | u, v \in V\}$，$(x, y)$ 是有序对，与 $(y, x)$ 不同

——两种图可以刻画不同的关系，关注的重点也不同，一定在学习和应用时先确认是什么图（PS：其实无向图可以视作双向图）

**赋权图 (weighted graph)** （建模能力强化）
在点或边上添加**权重** (weight)，用来补完物体或关系的特殊意义

广义的形式化表示就是在图上添加一个**映射** $w$，**该映射可以将点集** $V$ **或边集** $E$ **映射到另一个集合**，故权重会有两种定义：

- **点权**：$w : V \to S$，或者对于点 $v \in V$，点权表示为 $w(v)$
- **边权**：$w : E \to S$，或者对于边 $(u, v) \in E$，边权为 $w(u, v)$

注意：目标集合 $S$ 由问题定义，只不过 $\mathbb{N}$、$\mathbb{R}$ 作为 $S$ 比较常见

注意：**无权图** (unweighted graph) 也可以视作权值都为 1 的图。

——有了权值，图的建模能力进一步增强。比如：

- 点权例子：数据结构中的排序树、优先队列、字典树
- 边权例子：路径问题定义了费用，网络流问题定义了容量

**更多基础概念** （关于大小的讨论）

- 点集的大小通常用 $|V|$ 表示，有些地方称之为**阶 (order)**
- 边集的大小通常用 $|E|$ 表示，可以发现对于无向图而言，

$$0 \leq |E| \leq \binom{|V|}{2} = \frac{|V|\,(|V|-1)}{2}$$

- **稀疏图** (sparse graph)、**稠密图** (dense graph)：无严格定义，通常稀疏图边数 $|E|$ 在 $O(|V|)$ 量级，稠密图在 $O(|V|^2)$ 量级

——两种模型会导致不同复杂度的求解和实现算法，一般来说，稀疏图通常用邻接表实现，稠密图用邻接矩阵实现，相应时间复杂度大不相同

**更多基础概念** （两种“令人讨厌”的边）

- **自环** (loop)：点指向自身的边：$(v, v) \in E$, where $v \in V$
- **重边** (multiple edges): 同一对点间的多条边（边集为多重集）
- **简单图** (simple graph)：不包含自环与重边的图

——在建模过程中，自环与重边并无大碍，甚至体现了图的强大；但是，图论问题的诸多算法都是基于简单图的，自环和重边通常是会导致算法失效的罪魁祸首。

——建议在求解问题的过程中，注意讨论是否为简单图，注意讨论自环和重边带来的 *corner cases*。

**更多基础概念** （关于点的一些性质）

- **邻居** (neighborhood)：**无向图中，点** $v$ **的邻居** $N(v)$ **是一个点集，表示为** $N(v) = \{u|\{v,u\} \in E\}$
- **度** (degree)：**与点相连的边的数量**
  - **无向图中度的定义为点** $v$ **的邻居集合大小**：$\deg(v) = |N(v)|$
  - **有向图中分为入度** (in-degree) **和出度** (out-degree)。

    **入度为指向点** $v$ **的边数**：$\deg_{in}(v) = |\{(u,v)|(u,v) \in E\}|$

    **出度为点** $v$ **引出的边数**：$\deg_{out}(v) = |\{(v,u)|(v,u) \in E\}|$

——有一个应用场景对于度展开了深入讨论：社交网络 (social network)。对于社交关系用图建模后，度就拥有了特殊含义。社交网络中，所有点度的分布服从幂律分布 (power-law distribution)，这个图也被称作无标度网络(scale-free network)。

**更多基础概念**（开始在图上进行“游走”了）

- **链** (chain、walk)：点和边交替出现的序列。一个长度为 $m$ 的链表示为（其中 $e_{i_k} = (v_{i_{k-1}}, v_{i_k}) \in E$，无向图同理）：

$$W = v_{i_0} e_{i_1} v_{i_1} e_{i_2} v_{i_2} \cdots e_{i_m} v_{i_m}$$

- **轨迹、迹** (trail)：所有边互不相同的链
- **路径、路** (path)：所有点互不相同的链
- **环、闭链** (cycle)：两个端点 $v_{i_0}$ 和 $v_{i_m}$ 相同的链
- **回路、闭迹** (circuit)：两个端点 $v_{i_0}$ 和 $v_{i_m}$ 相同的迹

——至于链的英文为什么会是 *walk*，这其实又是图论研究的一个分支：随机游走 (random walk)。概率论知识在这里有非常巧妙的应用。

**更多基础概念** （“游走”了发现走不到，图就分裂了）

- **连通性** (connectivity)：对于图中两个点 $u$ 和 $v$，如果存在 $u$ 到 $v$ 的路径，那么 $u$ 和 $v$ 就是**连通**的
- **连通图** (connected graph)：分两种情况讨论
  - 无向图中，若任意两点都是连通的，那么就是一个**连通图**
  - 有向图则分三类：
    - 若将所有的边变成无向边，生成的无向图是连通的，那么该有向图就是**弱连通的** (weakly connected)
    - 若对于任意两点 $u, v$，$u$ 到 $v$ 或者 $v$ 到 $u$ 存在路径，那么该有向图就是**单连通的** (unilaterally connected)
    - 若对于任意两点 $u, v$，$u$ 到 $v$ 并且 $v$ 到 $u$ 存在路径，那么该有向图就是**强连通的** (strongly connected)
- **连通分量** (connected component)：在一个图中，满足上述不同连通性质，相对应的极大子图，就是连通分量

**更多基础概念**（如何比较两个图之间的关系？）

- **子图** (subgraph)：对于图 $G$ 和 $G'$，如果 $V(G') \subseteq V(G)$ 且 $E(G') \subseteq E(G)$，则称 $G'$ 是 $G$ 的子图
- **生成子图** (spanning subgraph)：如果图 $G$ 的子图 $G'$ 满足 $V(G') = V(G)$，那么 $G'$ 是 $G$ 的生成子图
- **图的同构** (graph isomorphism)：如果两个简单图 $G$ 和 $H$ 是同构的，当且仅当存在一个一一映射 $f : V(G) \to V(H)$，使得：对于 $u, v \in V(G)$，如果 $(u, v) \in E(G)$，当且仅当 $(f(u), f(v)) \in E(H)$。记作 $G \simeq H$。

——经典难题：判定两个图是否同构，还不能确定是否可在多项式复杂度内求解；更难的**子图同构问题**（给定图 $G$ 和 $H$，$G$ 的阶小于 $H$，判断是否存在 $H$ 的子图与 $G$ 同构），已被证明是 *NP* 完全问题

——不过对于特殊的图，比如树、平面图，图同构问题可以快速求解

**图论 (graph theory)**：刻画二元关系的强大模型

- 基本概念
- 特殊的图
- 连通性问题
- 生成树问题
- 路径问题
- 网络流问题
- 匹配问题
- 着色问题

# 图论：特殊的图

**树** (tree)：满足以下条件之一 **无向简单图** $G$

- $G$ 是**没有回路**的**连通**图
- $G$ 没有回路，但是**添加**任意一条边，就会形成一个回路
- $G$ 是连通图，但是**删除**任意一条边，就不再连通
- $G$ 内任意两个点，只能被**唯一路径**连通

**树的三个重要性质**：（当图退化为树后，很多难题都很简单了）

1. 没有回路　　2. 连通　　3. $n$ 个点，$n-1$ 条边

——注意以上三个性质，对于一个无向图而言，知道其中两个就可以确定是一棵树，并且这三个性质是“知二推三”的。

# 图论：特殊的图

**森林**(forest)：即每个连通分量都是树的图（一棵树也是森林）

**生成树**(spanning tree)：连通无向图的树状生成子图。

——生成子图定义是？生成树等价于找 $n-1$ 条边连起所有点

**根** (root)：树中可以任意指定一个点，作为根。因此一般树称作**无根树** (unrooted tree)，指定树根就是**有根树** (rooted tree)。
根为树带来了层级意义，于是定义：（求解树问题一般都要先指定根）

- **父亲** (parent)：对除根以外的点，该点到根的路径上的第二个点即为父亲。根节点没有父亲。
- **祖先** (ancestor)：在点到根的路径上，除了自身的所有点的集合。根节点的祖先为空集。
- **孩子** (child)：若 $u$ 是 $v$ 的父亲，那么 $v$ 是 $u$ 的孩子。
  ——孩子一般不区分顺序，后面讲的二叉树才会定义顺序
- **后代** (descendant)：孩子和孩子的后代构成的点集。若 $u$ 是 $v$ 的祖先，那么 $v$ 是 $u$ 的后代。

# 图论：特殊的图

**森林**(forest)：即每个连通分量都是树的图（一棵树也是森林）

**生成树**(spanning tree)：连通无向图的树状生成子图。

——生成子图定义是？生成树等价于找 $n-1$ 条边连起所有点

**根** (root)：树中可以任意指定一个点，作为根。因此一般树称作**无根树** (unrooted tree)，指定树根就是**有根树** (rooted tree)。
根为树带来了层级意义，于是定义：（求解树问题一般都要先指定根）

- **叶子** (leaf)：没有孩子的点。
- **兄弟** (sibling)：同一个父亲的多个孩子互为兄弟。
- **子树** (subtree)：一般称做以某个点为根的子树，即删掉该点与其父亲的边后，该点所在的子图。
- **深度** (depth)：形容一个点，即点到根的路径中边的数量。
- **高度** (height)：形容一棵树，所有点的深度最大值。

**特殊的树：链、星** (但是，树里面也有两种“令人讨厌的情况”)

- **链** (chain/path graph)：**每个点的度数都不超过 2 的树**
- **星** (star)：**存在一个点 $u$，除它以外的点都与之相连。**

——以上也被称作两种退化的树，即对 $n$ 个点的树

- 链可以使树高达到最大（取端点为根时高度为 $n-1$），最大孩子数最少 (为 1)
- 星可以使树高达到最小（为 1），最大孩子数最多 (为 $n-1$)

因此，链和星通常会作为某个问题或某个解法的最坏情况来考虑

# 图论：特殊的图

**特殊的树：二叉树** (广泛应用在数据结构的**离散数学模型**)

- **二叉树** (binary tree)：每个点最多只有两个孩子的有根树。
  通常对两个孩子的顺序加以区分：左孩子、右孩子。
- **完整二叉树** (full/proper binary tree)：每个点只有 0 或 2 个孩子。
- **完全二叉树** (complete binary tree)：只有最下面两层结点的度数可以小于 2，且最下层的结点都集中在该层最左边连续位置。
- **完美二叉树** (perfect binary tree)：所有叶结点的深度均相同。

——这里值得讨论的事情就很多了，比如：

- 二叉树如何定义了哪些数据结构？究竟是对点赋予了特殊含义的权值，还是对边赋予了特殊含义？要利用树实现什么效果？
- 关于二叉树的组合数学问题：$n$ 个点的二叉树一共有几种？加强难度的话，$n$ 个点的树有几种？（这里牵扯着同构问题，即同构的树只能被计数一次；还有组合数学基本问题：点可不可区分）

**树上的问题：树的直径和重心** (如何找出“最合适”的根)

- **树的直径** (diameter)：树上任意两个点之间的最长路径长度

一些重要性质：

1. 最长路径有多条（即有多条直径），但这些直径的中点重合
2. 以直径的中点作为根，得到的有根树高度最小
3. 如果对树赋予边权，如果边权均为正，性质 1 仍成立，若存在负边权，性质 1 失效

——如果使有根树的高度最小，很多问题的求解可以避开最坏情况。

——求树的直径一般有两种算法：1. 两次深度优先搜索；2. 树上动态规划。两种算法的思想和证明都非常巧妙，如果继续深入讨论的话，在出现性质 3 中提到的负边权时，算法 1 会失效，算法 2 仍旧正确。

**树上的问题：树的直径和重心** (如何找出“最合适”的根)

- **树的重心** (centroid)：对于一个点，若删掉该点后产生的多个树 (即连通分量) 中，使点数最多的树最少的点就是重心。

一些重要性质：

1. 以重心为根时，每个孩子的子树点数不会超过总数的一半
2. 树中所有点到某点的距离和中，到重心的距离和是最小的
3. 把两棵树通过一条边相连得到一棵新的树，那么新的树的重心在连接原来两棵树重心的路径上
4. 添加或删除一个叶子，重心最多只移动一条边

——上述性质的证明、求解算法、以及实际问题中的应用都非常巧妙

——与直径的中心不太相同，重心作为根的树是更加“平衡”的，是一种与分治算法中的主定理 $T(n) = aT(\frac{n}{b}) + O(n^d)$ 相恰的“平衡”

**树上的问题：最近公共祖先** (lowest common ancestor, LCA)

在多个点的公共祖先（即各自祖先的交集）中，离根最远的点。
形式化定义一个点集 $S = \{v_1, \cdots, v_k\}$ 的最近公共祖先，记作 $\mathrm{LCA}(\{v_1, \cdots, v_k\})$ 或 $\mathrm{LCA}(S)$

一些重要性质：

1. $\mathrm{LCA}(\{u\}) = u$
2. 若 $u$ 是 $v$ 的祖先，当且仅当 $\mathrm{LCA}(\{u, v\}) = u$
3. 如果 $u, v$ 互相不是对方的祖先，那么 $u, v$ 分别处于 $\mathrm{LCA}(\{u, v\})$ 的两棵不同子树中；
4.

# 图论：基本概念

正则图、完全图
二分图
平面图
循环图
仙人掌
弦图
佩特森图
有向无环图
树

# 连通性问题

无向图上的连通性问题
割点、桥边
双连通分量
有向图上的连通性问题
强连通分量
弱连通分量
半联通分量

# 路径问题

最短路径
DAG 的关键路径
欧拉路径、哈密顿路径
生成树问题
最小生成树
最小树形图
斯坦纳树

# 匹配问题

图的匹配
最大权匹配
覆盖问题
独立集问题
着色问题
团问题

# 流问题

流网络
费用流
最优传输理论

# 抽 象 代 数

重新定义最基础的运算

群环域

# Frame Title

阿贝尔群置换群模群多项式运算群
伽罗瓦域乘法逆元（费马小定理）

# Frame Title

可以解决什么问题？尺规作图
五次方程

# 谢　谢