离散数学

马汝辉 副教授 博导

上海交通大学 计算机科学与工程系

电信群楼3-229

ruhuima@sjtu.edu.cn

Tel: 34207874

课程内容

- 数理逻辑
- 图论
- 教材及下载地址
  - **数理逻辑与集合论（第二版）：石纯一，清华大学出版社**
  - **图论与代数结构：戴一奇，清华大学出版社**

助教 TA

- 赵鸿宇、周子昂
- 电院3号楼229，答疑时间：每周三14:00-16:00
- 课程网址： https://aisigsjtu.github.io/CS2501-05/

参考书籍

1. 离散数学：董晓蕾，曹珍富，机械工业出版社
2. 数理逻辑：莫绍揆，科学文献出版社
3. 数理逻辑教程：莫绍揆，华中工学院出版社
4. 集论与逻辑：沈恩绍，科学出版社
5. **A Mathematical Introduction to Logic, 2nd. Ed, H.Enderton, Academic Press**
6. **Logic for Applications, 2nd ed., A. Nerode, Springer**
7. 离散数学（第4版），**Richard Johnsonbaugh**，电子工业出版社

成绩评定

- 平时成绩（40%）＋期末考试成绩（40%）+图论测验（10%）+数理逻辑测验（10%）
- Canvas上提交作业
- 7-8次习题课、随堂练习
- 平时成绩
  - 签到出勤
  - 作业

微信班级群

群聊：2026 春离散数学周一周四草

该二维码7天内(3月7日前)有效，重新进入将更新

什么是离散数学?

- **离散数学**(*discrete mathematics*)是研究离散对象的数学分支.
  - 离散:由分离的元素组成.如自然数集.
    - 相对应的是连续对象. 如实数集.
      - 微积分就是研究连续函数的数学分支.
- 内容包括:
  - 集合,关系,函数
  - 数理逻辑
  - 图论
  - 抽象代数
  - 组合数学
  - 数论, ……

离散数学

- 研究对象：离散个体及其结构
- 离散数学是：
  - 现代数学的一个重要分支
  - 计算机科学与技术的理论基础
  - 是计算机应用必不可少的工具，所以又称为计算机数学
- 离散数学应用举例

在数字电路中的应用

交通信号灯控制

有红、黄、绿三灯，由3个开关分别控制

(1) 每次仅一种颜色灯亮

(2) 红灯及绿灯不会连续出现，即二者之间必须有黄灯

(3) 可加上延时：如红灯20秒，黄灯10秒，绿灯13秒

(4) 更复杂的情形：十字路口多信号灯、信号灯闪断等

本质上是数理逻辑

理发师悖论

在某个城市中有一位理发师，他的广告词是这样写的：

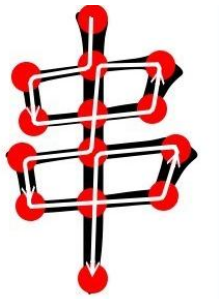
“本人的理发技艺十分高超，誉满全城。我将为本城所有不给自己刮脸的人刮脸，我也只给这些人刮脸。我对各位表示热诚欢迎！”

来找他刮脸的人络绎不绝，自然都是那些不给自己刮脸的人。可是，有一天，这位理发师从镜子里看见自己的胡子长了，他本能地抓起了剃刀，你们看他能不能给他自己刮脸呢？如果他不给自己刮脸，他就属于“不给自己刮脸的人”，他就要给自己刮脸，而如果他给自己刮脸呢？他又属于“给自己刮脸的人”，他就不该给自己刮脸。

本质上是集合论

一笔画及社交网络

- 一笔画：给定一个由顶点和边所组成的图。能否无重复的遍历该图的边？



- 社交网络：在社交网络环境中，依据后台数据库，自动得到一个成员之间彼此直接/间接认识的最大群体

本质上是图论

围棋策略学习

围棋“人机大战”

AlphaGo

AlphaGo，直译为**阿尔法围棋**，亦被音译为**阿尔法狗**等，是于2014年开始由英国伦敦Google DeepMind开发的人工智能围棋软件。

离散数学

- 事实上，从计算机产生到其发展每一步都离不开数学
  - 1936年，英国数学家图灵(A.M.Turing)发表了著名论文 “理想计算机”，从而给出了计算机的理论模型
  - 1946年在著名数学家冯·诺依曼(J.Von Neumann)的领导下，制造了世界上第一台现代意义的通用计算机EDVAC (Electronic Discrete Variable Automatic Computer)
  - . . .
- 为什么离散数学对计算机科学来说如此重要？
  - 数字电子计算机是一个离散结构，它只能处理离散的或离散化了的数量关系
  - 无论计算机科学本身，还是与计算机科学及其应用密切相关的现代科学研究领域，都面临着如何对离散结构建立相应的数学模型；又如何将已用连续数量关系建立起来的数学模型离散化，从而可由计算机加以处理

离散数学与计算机科学的关系

- 数理逻辑：人工智能、程序正确性证明、程序验证等
- 集合论：关系数据库模型等
- 图论：数据结构、数据库模型、网络模型等
- 代数结构：软件规范、形式语义、编译系统、编码理论、密码学、数据仓库等
- 组合数学：算法分析与设计、编码理论、容错等

离散数学课程说明

- 研究对象----离散个体及其结构
- 研究思想----以集合和映射为工具、体现公理化和结构的思想
- 研究内容----包含不同的数学分支，模块化结构
  - 数理逻辑：推理、形式化方法
  - 集合论：离散结构的表示、描述工具
  - 图论：离散结构的关系模型
  - 代数结构：离散结构的代数模型
  - 组合数学：离散结构的存在性、计数、枚举、优化、设计
  - 离散概率（概率统计课程）

图

一阶逻辑

On the Unusual Effectiveness of Logic in Computer Science*

Joseph Y. Halpern† Robert Harper‡ Neil Immerman§ Phokion G. Kolaitis¶
Moshe Y. Vardi‖ Victor Vianu**

January 2001

1 Introduction and Overview

In 1960, E.P. Wigner, a joint winner of the 1963 Nobel Prize for Physics, published a paper titled *On the Unreasonable Effectiveness of Mathematics in the Natural Sciences* [Wig60]. This paper can be construed as an examination and affirmation of Galileo's tenet that "The book of nature is written in the language of mathematics". To this effect, Wigner presented a large number of examples that demonstrate the effectiveness of mathematics in accurately describing physical phenomena. Wigner viewed these examples as illustrations of what he called *the empirical law of epistemology*, which asserts that the mathematical formulation of the laws of nature is both appropriate and accurate, and that mathematics is actually the *correct* language for formulating the laws of nature. At the same time, Wigner pointed out that the reasons for the success of mathematics in the natural sciences are not completely understood; in fact, he went as far as asserting that " ... the enormous usefulness of mathematics in the natural sciences is something bordering on the mysterious and there is no rational explanation for it."

In 1980, R.W. Hamming, winner of the 1968 ACM Turing Award for Computer Science, published a follow-up article, titled *The Unreasonable Effectiveness of Mathematics* [Ham80]. In this article, Hamming provided further examples manifesting the effectiveness of mathematics in the natural sciences. Moreover, he attempted to answer the "implied question" in Wigner's article: "Why is mathematics so unreasonably effective?" Although Hamming offered several partial explanations, at the end he concluded that on balance this question remains "essentially unanswered".

Since the time of the publication of Wigner's article, computer science has undergone a rapid, wide-ranging, and far-reaching development. Just as in the natural sciences, mathematics has been highly effective in computer science. In particular, several areas of mathematics, including linear algebra, number theory, probability theory, graph theory, and combinatorics, have been instrumental in the development of computer science. Unlike the natural sciences, however, computer science has also benefitted from an extensive and continuous interaction with logic. As a matter of fact, logic has turned out to be significantly more effective in computer science than it has been in mathematics. This is quite remarkable, especially since much of the impetus for the development of logic during the past one hundred years came from mathematics.

*This paper summarizes a symposium, by the same title, which was held at the 1999 Meeting of the American Association for the Advancement of Science. The authors wrote the following: Section 1 and 7 – Kolaitis, Section 2 – Immerman, Section 3 – Vianu, Section 4 - Harper, Section 5 - Halpern, and Section 6 – Vardi.

†Cornell University. Work partially supported by NSF Grant IRI-96-25901.
‡Carnegie-Mellon University Work partially supported by NSF Grant CCR-9502674 and DARPA Contract F19628-95-C-0050.
§University of Massachusetts, Amherst. Work partially supported by NSF grant CCR-9877078.
¶University of California, Santa Cruz. Work partially supported by NSF Grant CCR-9610257.
‖Rice University. Work partially supported by NSF Grants CCR-9700061, CCR-9988322, IIS-9978135, and CCR-9988322.
**University of California, San Diego. Work partially supported by NSF Grant IIS-9802288.

2. FO(LFP) = FO(PFP)
3. P = PSPACE

Descriptive complexity reveals a simple but elegant view of computation. Natural complexity classes and measures such as polynomial time, nondeterministic polynomial time, parallel time, and space have natural descriptive characterizations. Thus, logic has been an effective tool for answering some of the basic questions in complexity. [2]

3 Logic as a Database Query Language

The database area is an important area of computer science concerned with storing, querying and updating large amounts of data. Logic and databases have been intimately connected since the birth of database systems in the early 1970's. Their relationship is an unqualified success story. Indeed, first-order logic (FO) lies at the core of modern database systems, and the standard query languages such as *Structured Query Language* (SQL) and *Query-By-Example* (QBE) are syntactic variants of FO. More powerful query languages are based on extensions of FO with recursion, and are reminiscent of the well-known fixpoint queries studied in finite-model theory (see Section 2). The impact of logic on databases is one of the most striking examples of the effectiveness of logic in computer science.

This section discusses the question of why FO has turned out to be so successful as a query language. We will focus on three main reasons:

- FO has syntactic variants that are easy to use. These are used as basic building blocks in practical languages like SQL and QBE.
- FO can be efficiently implemented using *relational algebra*, which provides a set of simple operations on relations expressing all FO queries. Relational algebra as used in the context of databases was introduced by Ted Codd in [Cod70]. It is related to Tarski's Cylindric Algebras [HMT71]. The algebra turns out to yield a crucial advantage when large amounts of data are concerned. Indeed, the realization by Codd that the algebra can be used to efficiently implement FO queries gave the initial impetus to the birth of relational database systems[3].
- FO queries have the potential for "perfect scaling" to large databases. If massive parallelism is available, FO queries can *in principle* be evaluated in *constant time*, independent of the database size.

A relational database can be viewed as a finite relational structure. Its signature is much like a relational FO signature, with the minor difference that relations and their coordinates have names. The name of a coordinate is called an *attribute*, and the set of attributes of a relation R is denoted $att(R)$. For example, a "beer drinker's" database might consist of the following relations:

frequents	*drinker*	*bar*
	Joe	*King's*
	Joe	*Molly's*
	Sue	*Molly's*
	...	...

serves	*bar*	*beer*
	King's	*Bass*
	King's	*Bud*
	Molly's	*Bass*
	...	...

The main use of a database is to query its data, e.g., find the drinkers who frequent only bars serving Bass. It turns out that each query expressible in FO can be broken down into a sequence of simple subqueries.

[2]This section is based in part on the article [Imm95]. See also the books [EF95, Imm99] for much more information about descriptive complexity.

[3]Codd received the ACM Turing Award for his work leading to the development of relational systems.

学习目标

- 理解、掌握命题逻辑的基本概念及命题逻辑的等值和推理演算
- 理解、掌握谓词逻辑的基本概念及谓词逻辑的等值和推理演算
- 理解、掌握集合概念、集合运算等
- 理解、掌握关系、关系表示、闭包等
- 理解、掌握图的基本概念及代数表示
- 理解、掌握道路与回路、欧拉道路与回路及哈密顿道路与回路
- 理解、掌握树的基本概念、Huffman树及最短树
- 应用逻辑推理、形式化方法、离散结构的关系模型（如树等）来解决实际问题

主要内容

- 《图论与代数结构》
  - 第一章 基本概念

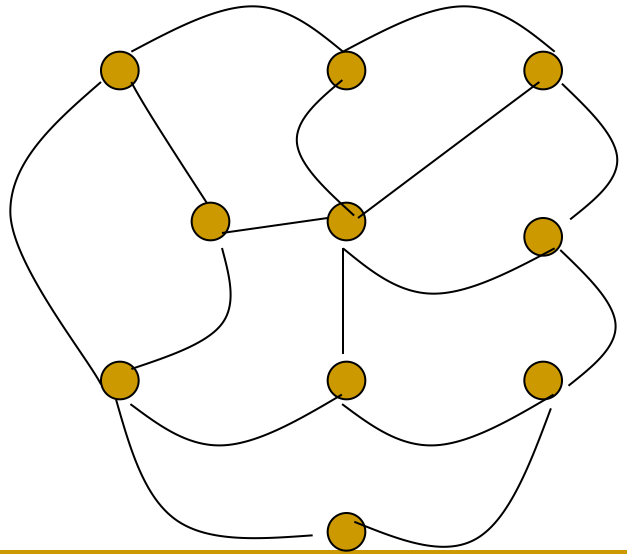
    1.1, 1.2(1,2,3,7)
  - 第二章 道路与回路

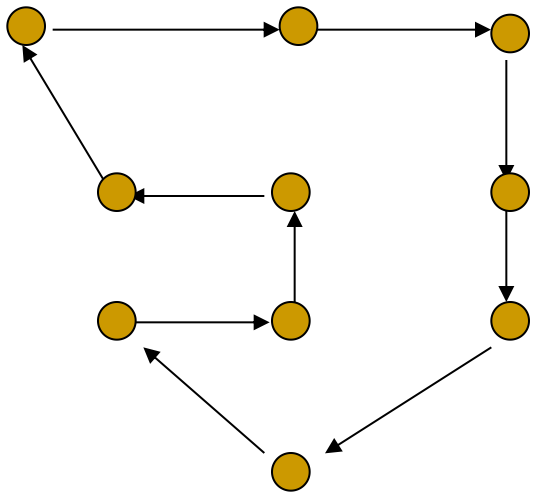
    2.1, 2.3, 2.4
  - 第三章 树

    3.1, 3.6, 3.7

实例

- 城市街道如图，市政府规定各街道只能单向行驶，问如何定向才能保证车辆能从一个地点到达任一个其他的地点。





图论

第一章　基本概念

1.1 图的概念

- 许多事物以及它们之间的联系可以用图形直观地表示
- 用结点表示事物

  用边表示它们之间的联系
- 图论的研究对象

  结点和边构成的图形

图的概念

- 定义1.1.1
  二元组(V(G),E(G))称为图。其中，
  V(G)是非空集合，称为结点集
  E(G)是V(G)诸结点之间边的集合
  常用G=(V,E)表示图
- 只讨论有限图, 即V和E都是有限集
  给定某个图G=(V, E), 如果不加特殊说明, $V=\{v_1, v_2, \ldots, v_n\}$, $E=\{e_1, e_2, \ldots, e_m\}$
  即结点数|V|=n, 边数|E|=m

边和图

- 用结点表示边

  $e_k=(v_i, v_j)$ (称v_i和v_j是相邻结点, 或者e_k分别与v_i,v_j相关联)
- 有向边(弧)，有向图

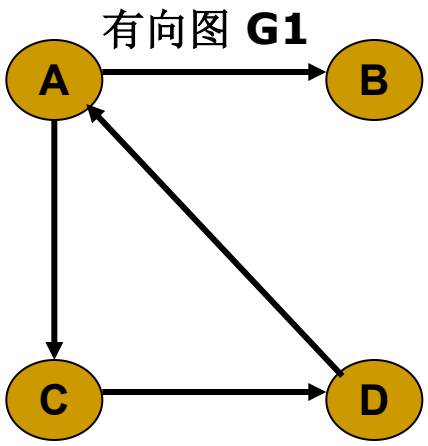
  v_i是e_k的始点，v_j是e_k的终点
- 无向边，无向图

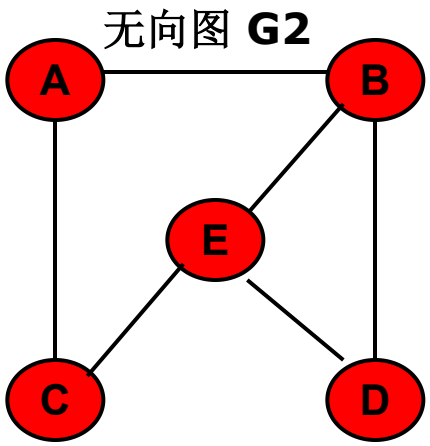
  v_i , v_j是e_k的端点
- 自环(圈)

  只与一个结点相关联的边
- 重边、多重图

  同一对结点之间存在多条边

有向图与无向图





G1 =（V1, E1）

V1 = {A, B, C, D}

E1 = {(A,B), (A,C), (C,D), (D,A)}

G2 =（V2, E2)

V2 = {A, B, C, D, E}

E2 = {(A,B), (A,C), (B,D), (B,E), (C,E), (D,E)}

图的概念

- 定义1.1.2

  G=(V,E)的某结点v所关联的边数称为该结点的度，用d(v)表示。如果v带有自环，则自环对d(v)的贡献为2

- 有向图

  出度(正度)$d^+(v)$: 以结点v为始点的边的数目

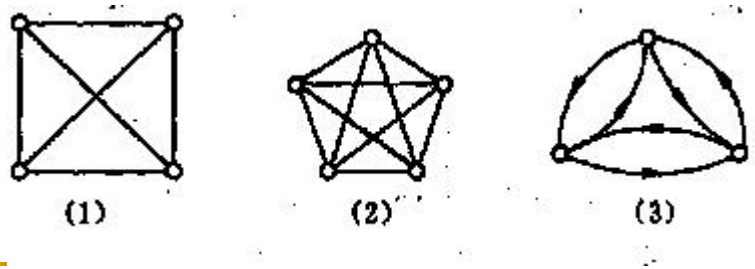
  入度(负度)$d^-(v)$: 以结点v为终点的边的数目

  $d(v) = d^+(v)+d^-(v)$

图的概念

- 定义1.1.3

  任意两结点间最多只有一条边，且不存在自环的无向图称为简单图
- 没有任何边的简单图(V, Ø)叫空图，用N_n表示. 若此时恰有|V|=1, 称为平凡图
- 任何两个结点间都有边的简单图称为完全图，用K_n表示. K_n中每个结点的度都是n-1



图的概念

- 性质1.1.1
  设G=(V, E)有n个结点，m条边，则

$$\sum_{v \in V(G)} d(v) = 2m$$

证明：由于每条边e=(u,v)对结点u和v度的贡献各为1，因此m条边对全部结点的总贡献为2m

图的概念

- 性质1.1.2

  G中度为奇数的结点必为偶数个.

证明：G中任一结点的度或为偶数或为奇数，设V_e是度为偶的结点集，V_o是度为奇的结点集，于是有

$$\sum_{v\in V_e} d(v) + \sum_{v\in V_0} d(v) = 2m$$

因此上式左边第二项也为偶数，也即度为奇数的结点必为偶数个

图的概念

- 性质1.1.3

  有向图G中正度之和等于负度之和.

这是因为每条边对结点的正、负度贡献各为1

- 性质1.1.4

  K_n的边数是n(n-1)/2.

证明: K_n中各结点的度都是(n-1), 由性质1.1.1就可以得到

图的概念

- 性质1.1.5

  非空简单图G中一定存在度相同的结点.

证明：设在G中不存在孤立结点，则对n个结点的简单图，每个结点度d(v)的取值范围是1~(n-1)，由抽屉(鸽巢)原理，一定存在两个度相同的结点.

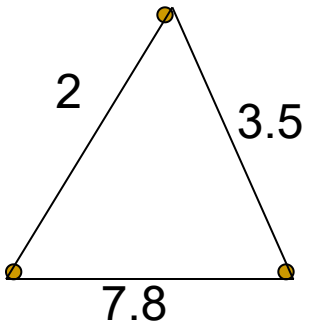
若存在一个孤立的结点，亦类似可证.

图的概念

- 定义1.1.4

  如果图G=(V,E)的每条边$e_k=(v_i, v_j)$ 都赋以一个实数w_k作为该边的权，则称G是赋权图．特别地，如果这些权都是正实数，就称G是正权图

- 权可以表示该边的长度、时间、费用或者容量等



图的概念

- 定义1.1.5

  给定G=(V, E), 如果存在另一个G’=(V’, E’), 满足V’⊆V, E’ ⊆ E, 则称G’是G的子图

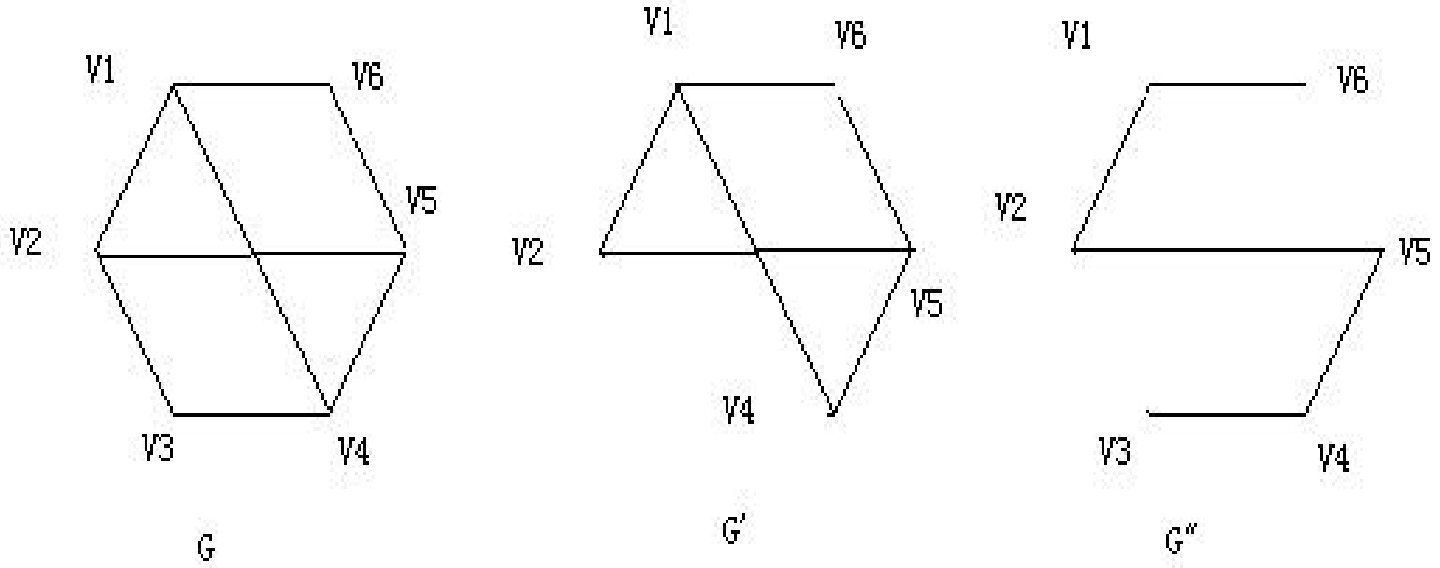
  特别地，如果V’=V, 就称G’是G的支撑子图或者生成子图

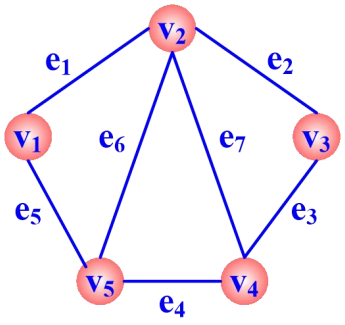
  如果V’⊆V, 且E’包含了G在结点子集V’之间的所有边, 则称G’是G的导出子图
- G自身既是支撑子图，又是导出子图；

  空图是G的支撑子图；

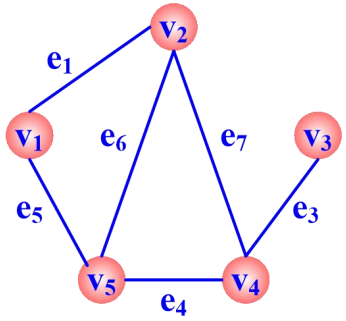
  G自身和空图称为平凡子图

- 在如下图中，给出了图G以及它的真子图G’和G’’，其中，G’是结点集为{v_1，v_2，v_4，v_5，v_6}的导出子图;
  G’’是支撑子图(或生成子图)。

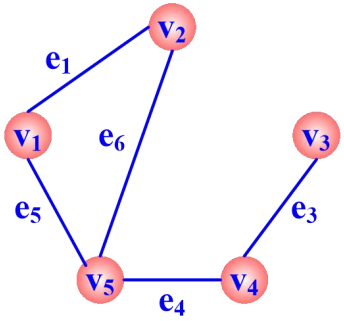




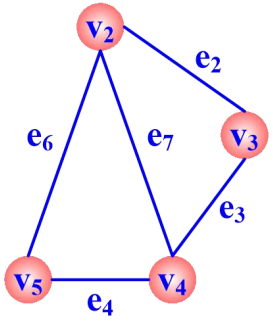
G



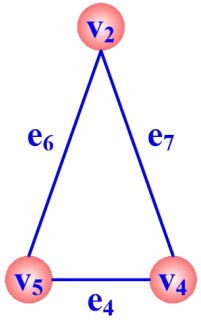
$G-e_2$



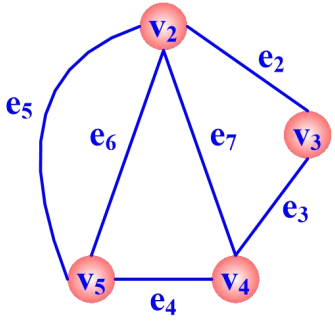
$G-\{e_2, e_7\}$



$G-v_1$



$G-\{v_1, v_3\}$



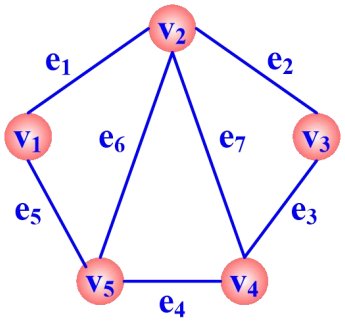
$G-v_1+e_{25}$

图的概念

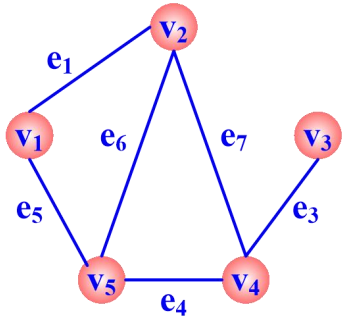
- 定义1.1.6
  给定两个图$G_1=(V_1,E_1),G_2=(V_2,E_2)$.令
  $G_1\cup G_2=(V,E)$,其中$V=V_1\cup V_2,E=E_1\cup E_2$;
  $G_1\cap G_2=(V,E)$,其中$V=V_1\cap V_2,E=E_1\cap E_2$;
  $G_1\oplus G_2=(V,E)$,其中$V=V_1\cup V_2,E=E_1\oplus E_2$;
  分别称为G_1和G_2的并，交和对称差

图的增删

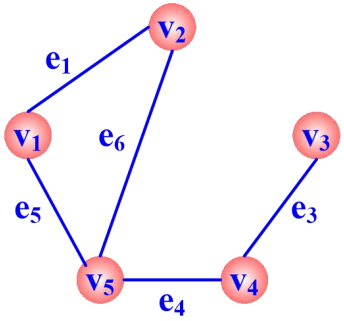
- G-H
  表示在G中删去一个子图H，即删掉H中的各条边(支撑子图)
  特别地，对于简单图G，称K_n-G为G的补图
- G-v
  表示从G中删去结点v及其关联的边(导出子图)
- G-e
  表示从G中删去边e(支撑子图)
- G+e_{ij}
  表示在G中增加某条边$e_k=(v_i, v_j)$



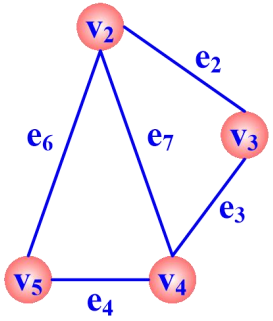
G



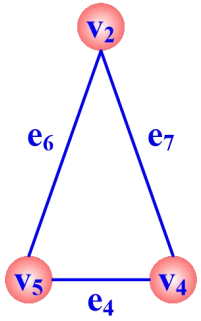
$G-e_2$



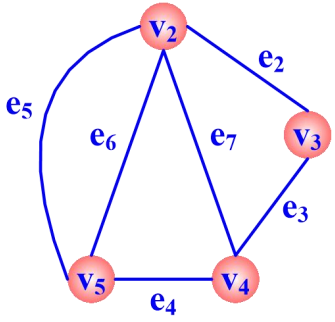
$G-\{e_2, e_7\}$



$G-v_1$



$G-\{v_1, v_3\}$



$G-v_1+e_{25}$

图的概念

- 无向图的邻点集

  $\Gamma(v)=\{u|(v, u)\in E\}$

- 定义1.1.7

  设v是有向图G的一个结点，则

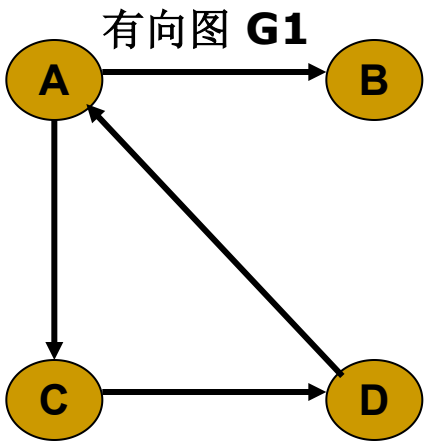
  $\Gamma^{+}(v)=\{u|(v, u)\in E\}$

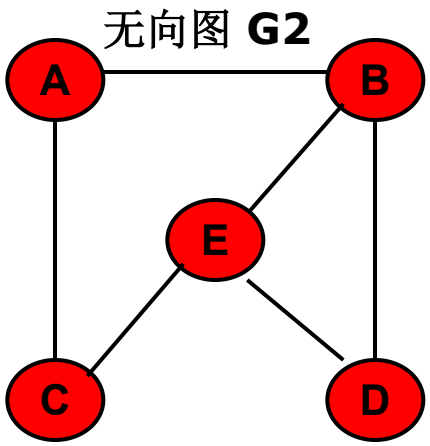
  称为v的直接后继集或者外邻集; 相应地

  $\Gamma^{-}(v)=\{u|(u, v)\in E\}$

  称为v的直接前趋集或者内邻集

有向图与无向图





G1 =（V1, E1）

V1 = {A, B, C, D}

E1 = {(A,B), (A,C), (C,D), (D,A)}

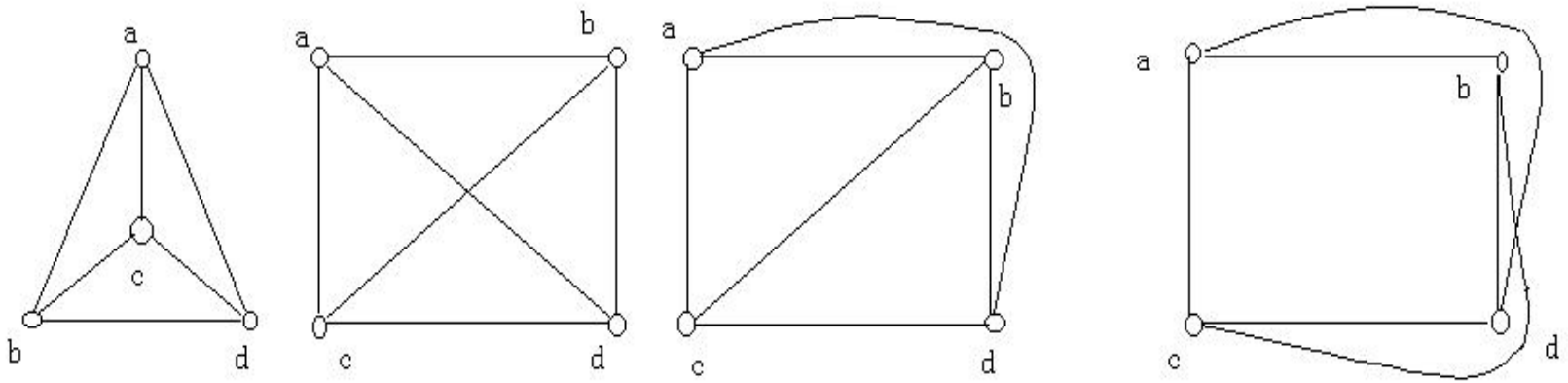
G2 =（V2, E2)

V2 = {A, B, C, D, E}

E2 = {(A,B), (A,C), (B,D), (B,E), (C,E), (D,E)}

图的同构

如下四张图所表示的图形实际上都是一样的



图的同构

- 定义1.1.8

  两个图$G_1=(V_1, E_1)$, $G_2=(V_2, E_2)$,如果V_1和V_2之间存在双射f, 而且$(u, v)\in E_1$, 当且仅当$(f(u), f(v))\in E_2$时, 称G_1和G_2同构

  记作$G_1\cong G_2$

- 如何判断两个图是否同构呢？

  答案：迄今为止还没有有效的算法。

图的同构

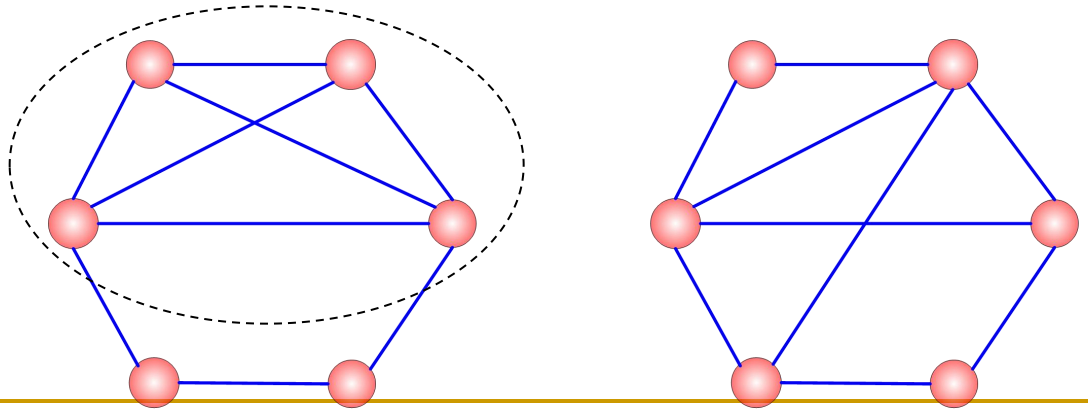
假如$G_1 \cong G_2$，则必须满足：

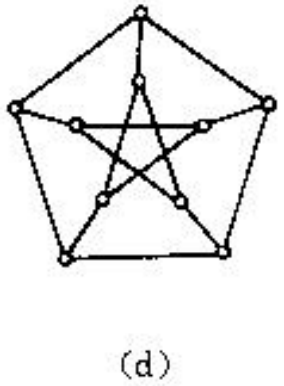
(1) $|V(G_1)|=|V(G_2)|$, $|E(G_1)|=|E(G_2)|$

(2) G_1和G_2结点度的非增序列相同

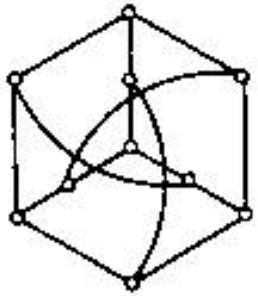
(3) 存在同构的导出子图(用来判断不同构)

上述是必要条件, 但不是充分条件，只能用来判断图的不同构。例如下面两个图是不同构的。

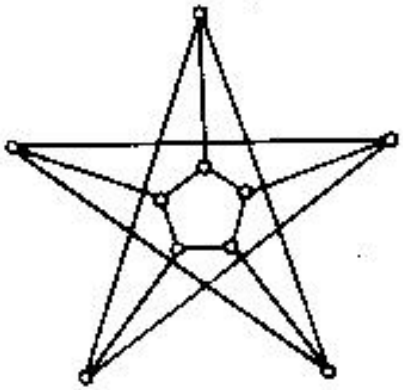




(d)



(e)



(f)

(d)≅（e)≅（f). (d)所示图称为彼德森图

1.2 图的代数表示

- 邻接矩阵
- 权矩阵
- 关联矩阵
- 邻接表

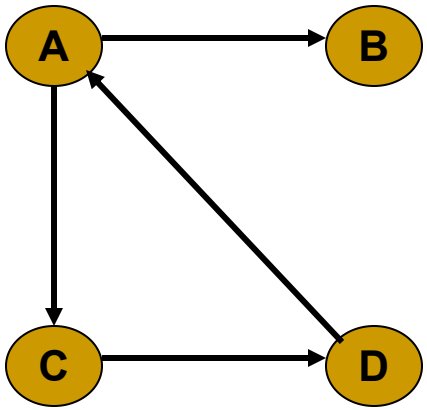
邻接矩阵

- 邻接矩阵表示结点之间的邻接关系
  - 无权值的有向图的邻接矩阵

    设有向图具有n个结点，则用n行n列的布尔矩阵A表示该有向图；并且A[i,j]=1, 如果i至j有一条有向边；A[i,j]=0, 如果i至j没有一条有向边

    v_i出度: 第i行之和; v_j入度: 第j列之和

    注: 可以表示自环，但无法表示重边



表示成右图矩阵

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

邻接矩阵

- 无权值的无向图的邻接矩阵(对称矩阵)

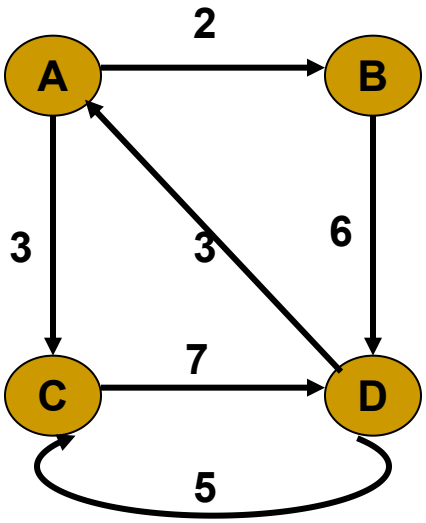
  设无向图具有n个结点，则用n行n列的布尔矩阵 A 表示该无向图；并且A[i,j]=1 , 如果i至j有一条无向边；A[i,j]=0, 如果i至j没有一条无向边

  v_i结点的度: 第i行或i列之和

权矩阵

- 用来表示赋权图
  - 例如：有向图的加权邻接矩阵

    设有向图具有n个结点，则用n行n列的矩阵 A 表示该有向图；并且 A[i,j]=w_{ij} , 如果i至j有一条有向边且它的权值为w_{ij}; A[i,j]=0, 如果i至j没有一条有向边



表示成右图矩阵

$$\begin{pmatrix} 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 7 \\ 3 & 0 & 5 & 0 \end{pmatrix}$$

关联矩阵

- 关联矩阵表示结点与边之间的关联关系
- 设G=(V, E), |V|=n, |E|=m. 则G的关联矩阵B是n×m的矩阵
- 有向图的关联矩阵

B[i,j]=1, 如果v_i是边$e_j=(v_i,v_k)$的始点

B[i,j]=-1, 如果v_i是边$e_j=(v_k,v_i)$的终点

B[i,j]=0, 其他(v_i既不是e_j的始点亦非终点)

关联矩阵

- 有向图关联矩阵的性质
  - 每列只有两个非零元：1和-1
  - 第i行非零元的数目恰是结点v_i的度，其中1元的数目是出度，-1元的数目是入度
  - 能够表示重边，但不能表示自环

关联矩阵

- 无向图的关联矩阵

  B[i,j]=1，　如果v_i是边e_j的端点

  B[i,j]=0，　如果v_i不是边e_j的端点

矩阵表示的特点

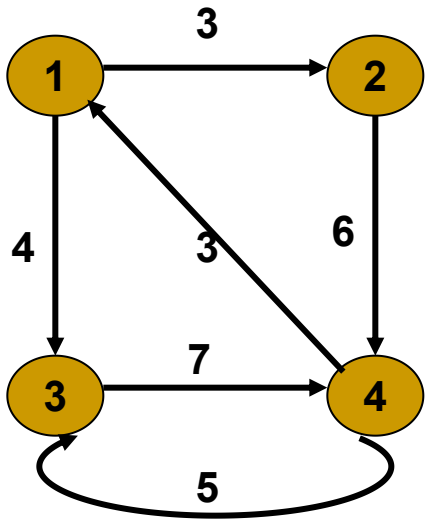
- 如果可以用邻接矩阵/关联矩阵表示某个图G,则表示是唯一的
- 邻接矩阵不能表示重边

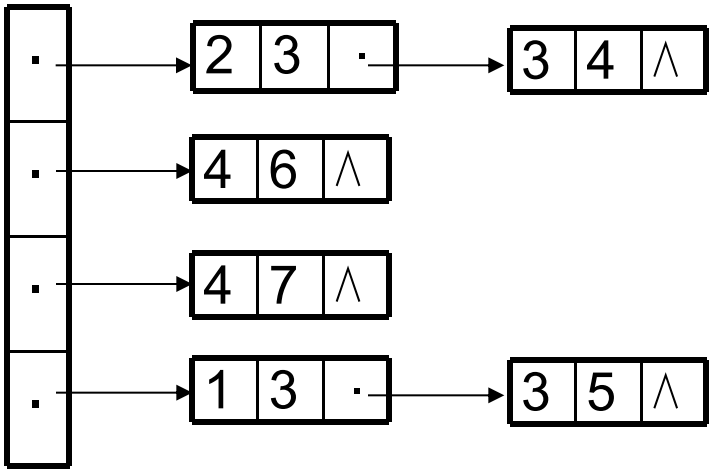
  关联矩阵不能表示自环
- 从数据结构和算法的角度来看，矩阵表示法占据的存储空间较大，可能增加计算复杂度

邻接表

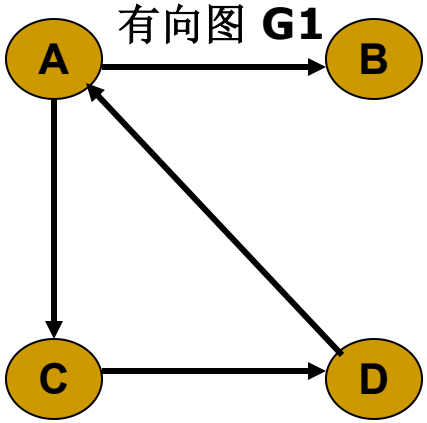
- 单链表
- 表结点的结构

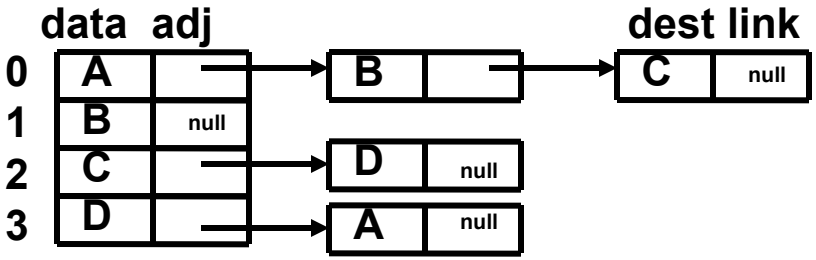
邻接点的编号	边权值	下一个结点的地址指针

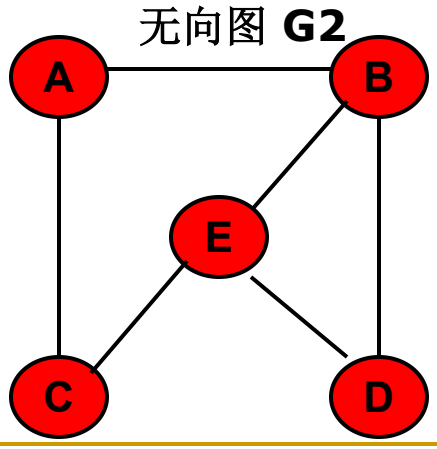


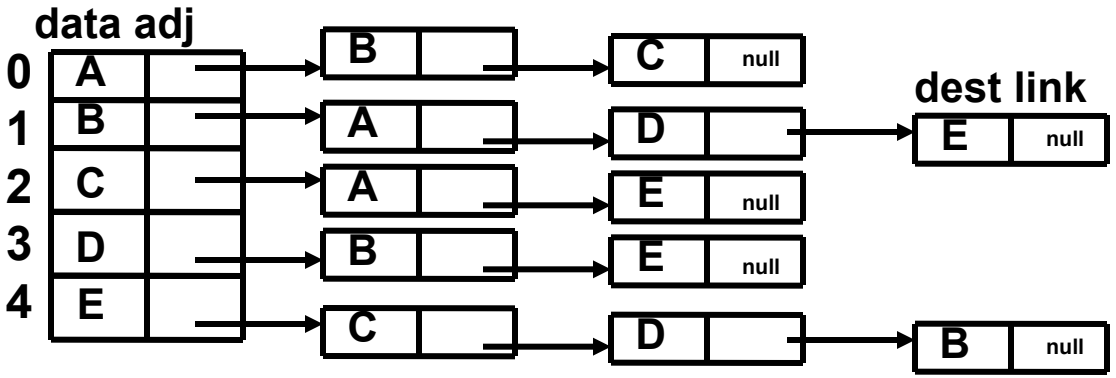


邻接表


有向图 G1
A
B
C
D


data adj
dest link
0
1
2
3
A
B
C
D
null
B
C
D
A
null


无向图 G2
A
B
E
C
D


data adj
dest link
0
1
2
3
4
A
B
C
D
E
B
C
null
A
D
null
E
null
B
E
null
C
D
null
E
null
B
null

邻接表

- 特点
  - 便于增加和删除边
  - 可以表示重边和自环
  - 可以唯一表示任意一个图
  - 所需存储空间较小

图的应用

- 三个量杯容量分别是8升，5升，3升，现8升的量杯装满了水，问怎样才能把水分成2个4升的。

初始状态(8,0,0)，终止状态(4,4,0)

状态转移图：状态看作点，状态间的转化看作边

答案: (8,0,0)→(5,0,3)→(5,3,0)→(2,3,3)
→(2,5,1)→(7,0,1)→(7,1,0)→(4,1,3)→(4,4,0)

图的应用

- 人狼羊菜过河

  有一个人带着一只狼，一只羊，一筐菜过河，当这个人在狼和羊身边时，狼不敢吃羊，羊也不敢吃菜，但是当人不在它们身边时，狼就可能把羊吃掉，羊也可能把菜吃掉，现在，渡船时只有一只船，能承载一个人及一件东西或物品，问怎样渡才能使人.狼.羊.菜安全过河？

使用有限状态机，初始状态(0,0,0,0)，终止状态(1,1,1,1)

答案：

1、人羊坐船过河
2、人单独回
3、人白菜过河
4、人羊一起回
5、人狼过河
6、人单独回
7、人羊过河

图论第一章作业

- 《图论与代数结构(第2版)》P13: 1,2,4,7,10,16(邻接矩阵、关联矩阵),17。