计算机专业复杂工程问题与在线教学实验基础的思考

马殿富

dfma@buaa. edu. cn

2017-7

西宁

主要内容

- 复杂工程问题
- 在线教学实验
- 在线教学实验基础

“复杂工程问题”特征

- （1）必须运用深入的工程原理，经过分析才可能得到解决；
- （2）涉及多方面的技术、工程和其它因素，并可能相互有一定冲突；
- （3）需要通过建立合适的抽象模型才能解决，在建模过程中需要体现出创造性；
- （4）不是仅靠常用方法就可以完全解决的；
- （5）问题中涉及的因素可能没有完全包含在专业工程实践的标准和规范中；
- （6）问题相关各方利益不完全一致；
- （7）具有较高的综合性，包含多个相互关联的子问题。

毕业要求（1/3）

- 1.工程知识：能够将数学、自然科学、工程基础和专业知识用于解决复杂工程问题。
- 2.问题分析：能够应用数学、自然科学和工程科学的基本原理，识别、表达、并通过文献研究分析复杂工程问题，以获得有效结论。
- 3.设计/开发解决方案：能够设计针对复杂工程问题的解决方案，设计满足特定需求的系统、单元（部件）或工艺流程，并能够在设计环节中体现创新意识，考虑社会、健康、安全、法律、文化以及环境等因素。
- 4.研究：能够基于科学原理并采用科学方法对复杂工程问题进行研究，包括设计实验、分析与解释数据、并通过信息综合得到合理有效的结论。

毕业要求（2/3）

- 5.使用现代工具：能够针对复杂工程问题，开发、选择与使用恰当的技术、资源、现代工程工具和信息技术工具，包括对复杂工程问题的预测与模拟，并能够理解其局限性。
- 6.工程与社会：能够基于工程相关背景知识进行合理分析，评价专业工程实践和复杂工程问题解决方案对社会、健康、安全、法律以及文化的影响，并理解应承担的责任。
- 7.环境和可持续发展：能够理解和评价针对复杂工程问题的专业工程实践对环境、社会可持续发展的影响。
- 8.职业规范：具有人文社会科学素养、社会责任感，能够在工程实践中理解并遵守工程职业道德和规范，履行责任。

毕业要求（3/3）

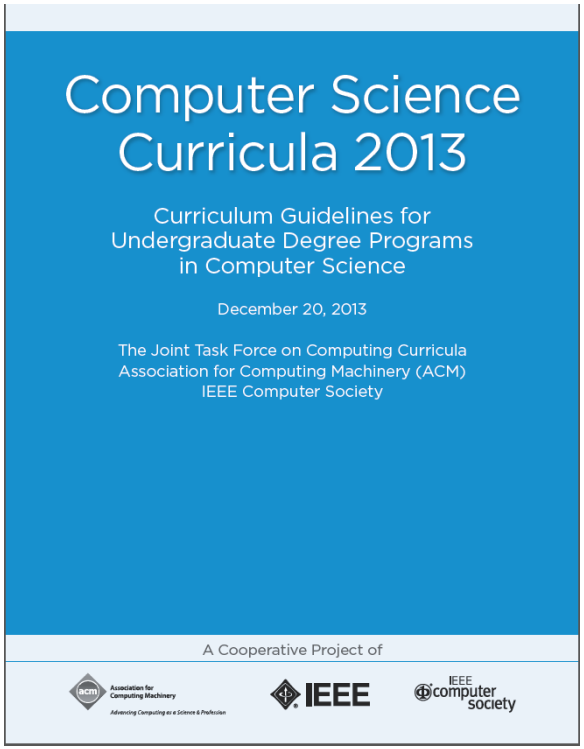
- 9.个人和团队：能够在多学科背景下的团队中承担个体、团队成员以及负责人的角色。
- 10.沟通：能够就复杂工程问题与业界同行及社会公众进行有效沟通和交流，包括撰写报告和设计文稿、陈述发言、清晰表达或回应指令。并具备一定的国际视野，能够在跨文化背景下进行沟通和交流。
- 11.项目管理：理解并掌握工程管理原理与经济决策方法，并能在多学科环境中应用。
- 12.终身学习：具有自主学习和终身学习的意识，有不断学习和适应发展的能力。

2013ACM/IEEE课程体系（18）

Knowledge Area	CS2013 Tier1	CS2013 Tier2	CS2008 Core	CC2001 Core
AL-Algorithms and Complexity	19	9	31	31
AR-Architecture and Organization	0	16	36	36
CN-Computational Science	1	0	0	0
DS-Discrete Structures	37	4	43	43
GV-Graphics and Visual Computing	2	1	3	3
HC-Human-Computer Interaction	4	4	8	8
IAS-Security and Information Assurance	2	6	--	--
IM-Information Management	1	9	11	10
IS-Intelligent Systems	0	10	10	10
NC-Networking and Communication	3	7	15	15
OS-Operating Systems	4	11	18	18
PBD-Platform-based Development	0	0	--	--
PD-Parallel and Distributed Computing	5	10	--	--
PL-Programming Languages	8	20	21	21
SDF-Software Development Fundamentals	42	0	47	38
SE-Software Engineering	6	21	31	31
SF-Systems Fundamentals	18	9	--	--
SP-Social and Professional Issues	11	5	16	16
Total Core Hours	**163**	**142**	**290**	**280**

计算机专业知识体系

- 知识体系按知识领域、知识单元和知识点组织。
  - 知识领域（18）、知识单元（163）、知识点（1105,1618）
- 对于知识点，按了解、理解、实现及应用进行划分。
- 教学方法
  - 了解：讲解历史
  - 理解：分析方法
  - 实现：综合方法
  - 应用：综合应用
- 能力培养
  - 系统设计和实现能力
  - 综合应用所学知识解决实际问题能力



计算机专业的复杂工程问题是什么？

- 运用深入的工程原理
- 包含多个相互关联的子问题
- 建立合适的抽象模型
- 不是仅靠常用方法就可以完全解决的
- 涉及多方面的技术、工程和其它因素，并可能相互有一定冲突
- 超出专业工程实践的标准和规范的因素
- 问题相关各方利益不完全一致

- 专业基础
- 离散结构
- 程序设计
- 数据结构
- 计算机组成
- 操作系统
- 计算机网络
- 信息管理

- 专业知识
- 数字电路
- 计算机系统结构
- 算法
- 软件工程
- 数据库系统
- 程序设计语言（编译系统）
- 智能技术

《计算机科学与技术专业标准》

解决“复杂工程问题”能力培养

- 计算机系统基础
  - 计算机组成（数字逻辑）
  - 操作系统
  - 编译系统
  - 数据库系统、计算机网络
- 解决复杂工程问题的能力
  - 运用离散数学的原理，解决计算机系统工程问题
  - 运用计算机系统的工程原理，解决专业工程问题
  - 运用计算机专业的工程原理，解决领域工程问题
- 运用计算机专业基础及工程原理，解决计算机系统的工程问题是系统能力培养的重要目标
  - 不能解决计算机系统工程问题，就难于解决专业工程问题，更难于解决领域工程问题

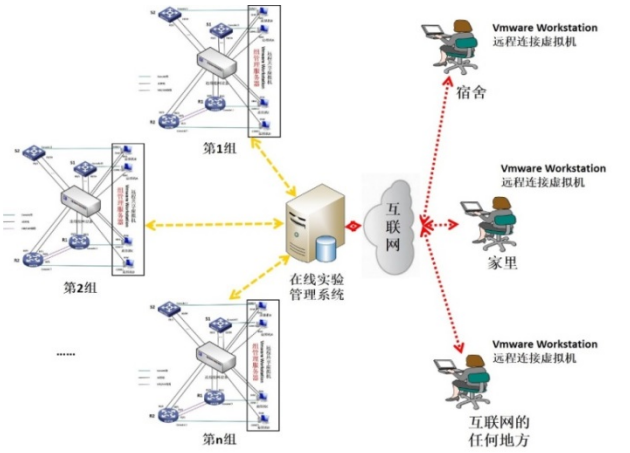
主要内容

- 复杂工程问题
- 在线教学实验
- 在线教学实验基础

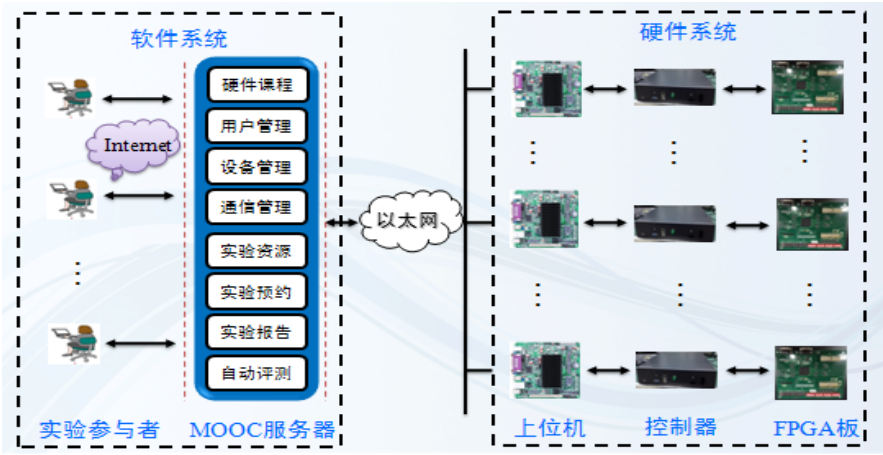
建成北航学堂MOOC平台

- 2012.7：计算机学院启动在线课程资源建设
- 2014.3：北航学堂MOOC平台正式上线

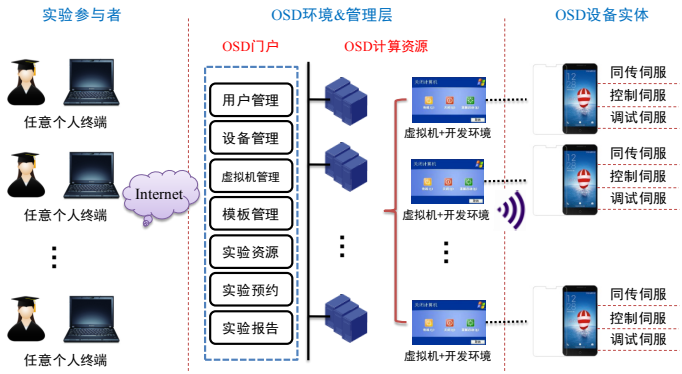
建成计算机硬件类在线实验平台



1、计算机网络在线实验子平台



2、基于FPGA的硬件类课程在线实验子平台

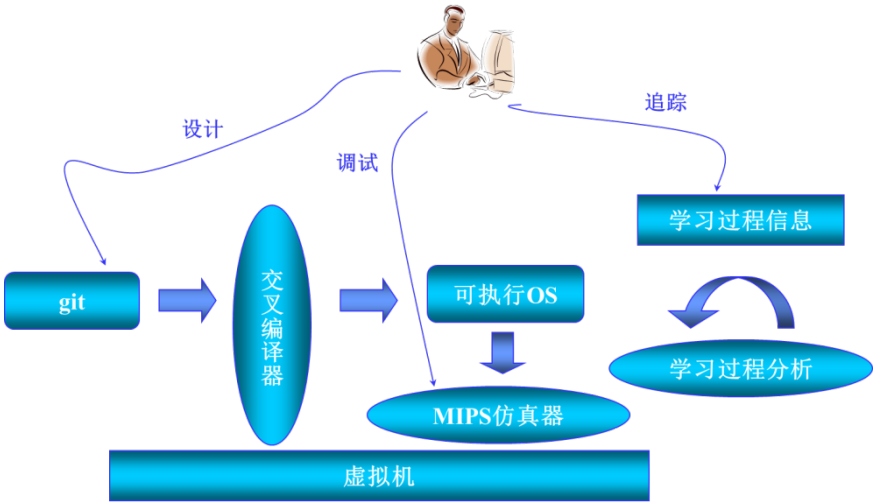


3、移动计算与嵌入式系统在线实验子平台

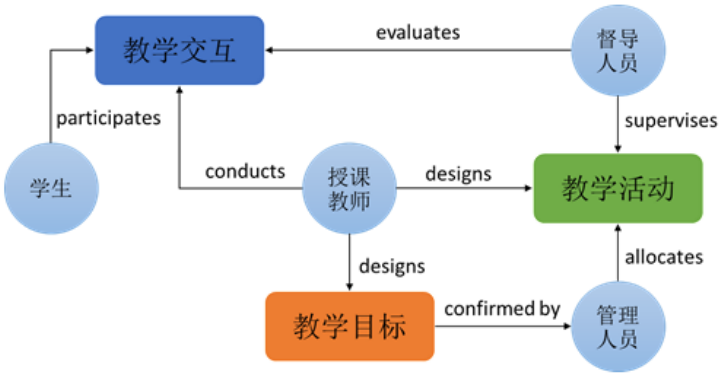
- 3个硬件类在线实验系统
- 已集成至MOOC平台

建成软件类在线实验平台

- 覆盖软件开发技术实验教学全过程
  - 发布：初始代码的在线发布
  - 开发：代码在线编写、在线编译、在线调试、在线部署
  - 管理：代码集中管理、精准查重
  - 评价：在线测试/考试、评价与反馈



软件在线实验集成环境



在线实验教学交互过程

建设在线实验课程

- 11门在线实验课程
  - 6门：有完整教学周期
  - 1门：即将有完整教学周期
  - 4门：正在建设中

分系统	可以支持的在线实验课程
基于FPGA的硬件在线实验系统	数字逻辑实验 计算机组成实验 计算机EDA设计 FPGA多核并行计算 数字系统设计
软件类在线实验集成环境	操作系统实验 面向对象构造与验证
移动计算与嵌入式在线实验系统	移动计算 嵌入式系统
计算机网络在线实验系统	计算机网络实验（本科生） 计算机网络与通信实验（研究生）

有完整检验周期

正在建设或教学

主要内容

- 复杂工程问题
- 在线教学实验
- 在线教学实验基础

问题提出—关系R的特征?

X=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]

R={(0,0),(0,4),(0,8),(0,12),(0,16),(0,20),(1,1),(1,5),(1,9),(1,13),(1,17),(1,21),(2,2),(2,6),(2,10),(2,14),(2,18),(2,22),(3,3),(3,7),(3,11),(3,15),(3,19),(3,23),(4,0),(4,4),(4,8),(4,12),(4,16),(4,20),(5,1),(5,5),(5,9),(5,13),(5,17),(5,21),(6,2),(6,6),(6,10),(6,14),(6,18),(6,22),(7,3),(7,7),(7,11),(7,15),(7,19),(7,23),(8,0),(8,4),(8,8),(8,12),(8,16),(8,20),(9,1),(9,5),(9,9),(9,13),(9,17),(9,21),(10,2),(10,6),(10,10),(10,14),(10,18),(10,22),(11,3),(11,7),(11,11),(11,15),(11,19),(11,23),(12,0),(12,4),(12,8),(12,12),(12,16),(12,20),(13,1),(13,5),(13,9),(13,13),(13,17),(13,21),(14,2),(14,6),(14,10),(14,14),(14,18),(14,22),(15,3),(15,7),(15,11),(15,15),(15,19),(15,23),(16,0),(16,4),(16,8),(16,12),(16,16),(16,20),(17,1),(17,5),(17,9),(17,13),(17,17),(17,21),(18,2),(18,6),(18,10),(18,14),(18,18),(18,22),(19,3),(19,7),(19,11),(19,15),(19,19),(19,23),(20,0),(20,4),(20,8),(20,12),(20,16),(20,20),(21,1),(21,5),(21,9),(21,13),(21,17),(21,21),(22,2),(22,6),(22,10),(22,14),(22,18),(22,22),(23,3),(23,7),(23,11),(23,15),(23,19),(23,23)}

```
isequivalence(X,R)
True
isequivalencerelation(X,R)
True
```

- 人脑难于认识
- 机器擅长处理

问题提出—极限概念

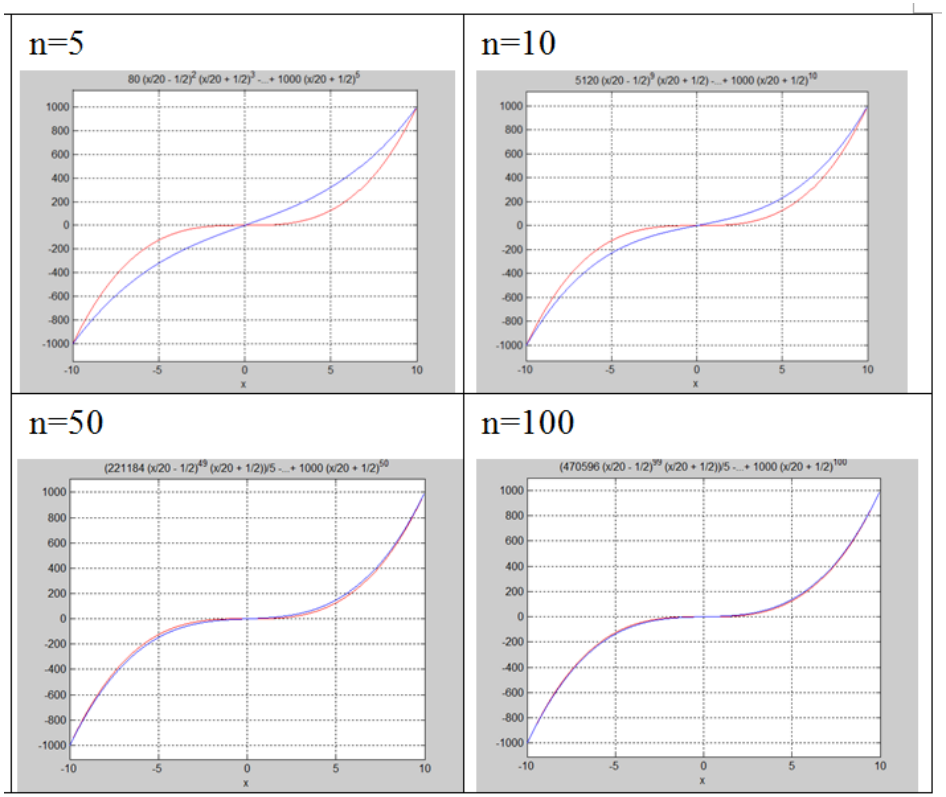
- 柯西：当属于一个变量的相继值无限地趋近某个固定值时，如果以这样一种方式告终，变量值同固定值之差小到我们希望的任意小，那么这个固定值就称为其他所有值的极限。
  - 《微积分的历程—从牛顿到勒贝格》[美]邓纳
- 定义（维尔斯特拉斯）：设函数$f(x)$在点x_0邻域有定义，对于任意$\varepsilon>0$，存在$\delta>0$，对于任意x，当$|x-x_0|<\delta$时，都有$|f(x)-A|<\varepsilon$，则称x趋于x_0时，函数$f(x)$的极限为A，记为$\lim_{x\to x_0} f(x)=A$。
- 数学语言描述
  - $\lim_{x\to x_0} f(x)=A \Leftrightarrow \forall\varepsilon>0$，$\exists\delta>0$，$\forall x(0<|x-x_0|<\delta)$：$|f(x)-A|<\varepsilon$
- 人脑易于记忆和理解
- 机器难于记忆和理解

问题提出—魏尔斯特拉斯多项式

- 定理（魏尔斯特拉斯）：设函数$f(x)$在[a,b]上连续，则$f(x)$可多项式逼近。
  - $h(x)=\Sigma_{k=0,n} f(a+(k/n)(b-a))\, C_{(n,k)}\, ((x-a)/(b-a))^k(1-(x-a)/(b-a))^{n-k}$ 是逼近多项式

```
function [h] = Weirstras_polynomials( sv,e,n,a,b )
eval(sv);
    f(x)=eval(e);
    g(x)=f(a+(k/n)*(b-a))*nchoosek(n,k)*((x-a)/(b-a)).^k*(1-(x-a)/(b-a)).^(n-k)
    h(x)=symsum(g(x),k,0,n)
    c=ezplot(f(x),[a,b]);
    set(c,'color','r');
    hold on
    ezplot(h(x),[a,b]);
    grid on
end
```

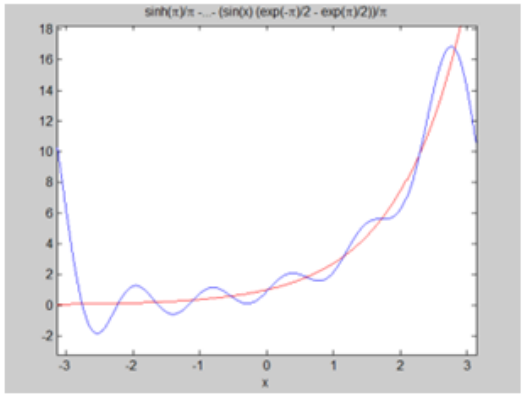
```
function [tv] = main(    )
syms x
    sv='syms x k';
    e='x.^3';
    a=-10;
    b=10;
    n=100;
    h=Weirstras_polynomials(sv,e,n,a,b);
end
```



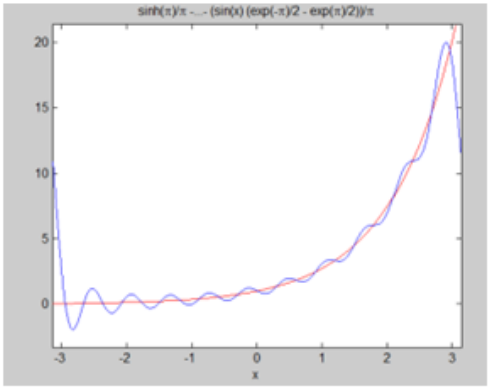
问题提出—傅里叶（Fourier）级数

```
function [ g ] = Fourier_series( sv,e,m )
eval(sv);
      f(x)=eval(e);
      c=ezplot(f,[-pi,pi]);
      set(c,'color','r');
      hold on
      a0=int(f,x,-pi,pi)/pi
      an=int(f*cos(n*x),x,-pi,pi)/pi
      bn=int(f*sin(n*x),x,-pi,pi)/pi
      w=an*cos(n*x)+bn*sin(n*x);
      g=a0/2+symsum(w,n,1,m);
      ezplot(g,[-pi,pi])
      hold on
end
```

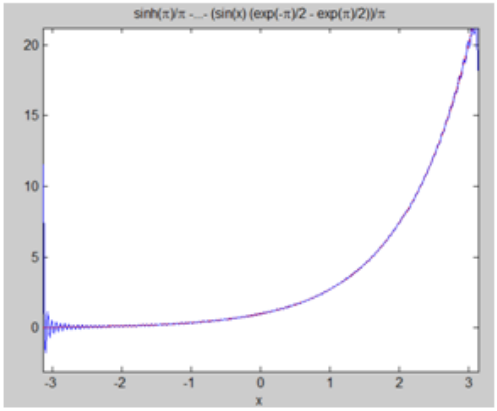
```
function [g] = main(    )
syms x
    sv='syms x n';
    e='exp(x)';
    m=100;
    g=Fourier_series( sv,e,m) ;
end
```



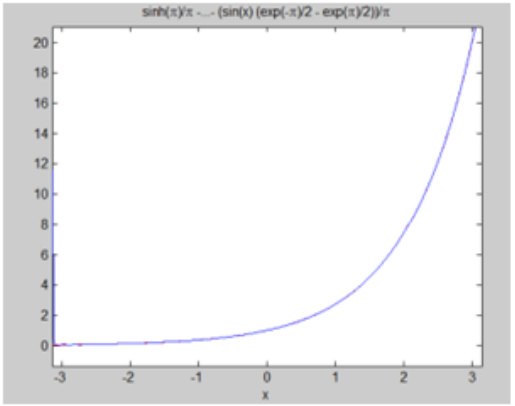
m=5



m=10



m=100



m=1000

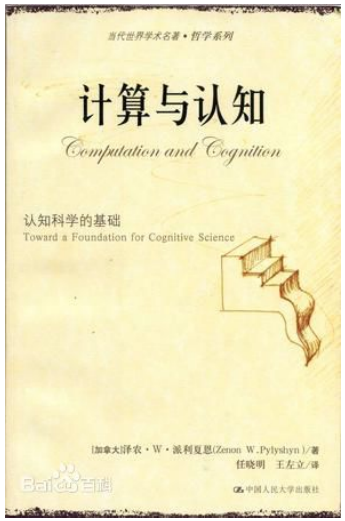
- 人脑：逻辑证明，难于计算
- 机器：难于证明，擅于计算

问题提出—知识表征

- 数学分析是现代数学基础
  - 线性代数、概率、微分方程、统计与随机过程
- 数学分析是现代工程基础
  - 电路、信号与系统、数字电路、模拟电路、超大规模电路
- 知识表示法：自然语言以及数学语言表达
  - 概念（命题）
  - 运算（命题）
  - 关系（命题）
  - 定理（命题）
- 人脑记忆和理解数学分析知识
  - 我们记忆多少？
  - 应用数学分析知识解决实际问题吗？
- 机器难于记忆数学分析知识，更难于实现问题解决。

认知科学与认知计算

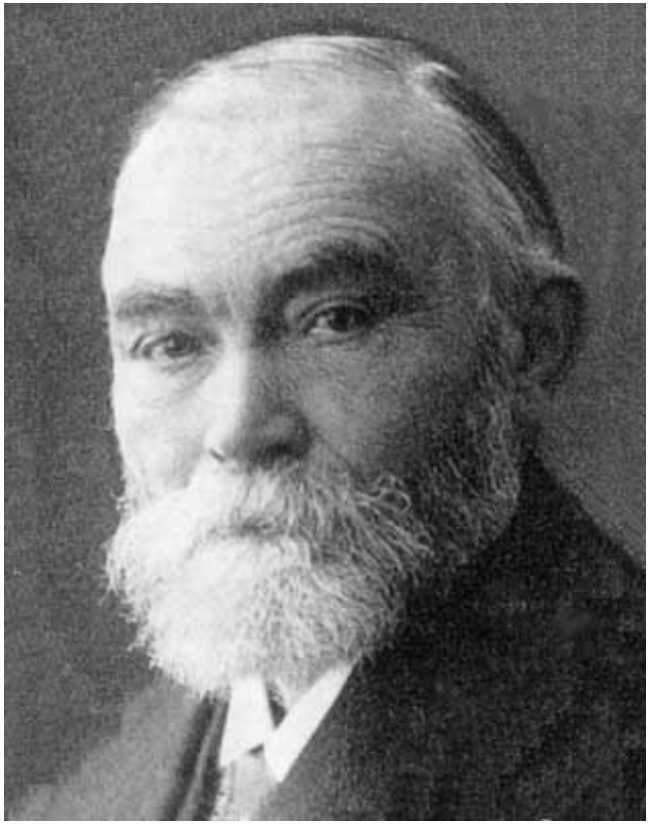
- 认知是知识的获取、存储、转换和使用的过程
- 认知哲学：行为主义、认知主义和联结主义
- 认知过程：感觉、知觉、注意、学习、记忆、思维、推理、问题解决、决策。
- 认知表征：知识表示为人理解的形式。
  - 陈述性知识（是什么）
  - 程序性知识（做什么）
- 认知科学
  - 计算机科学
  - 神经科学
  - 认知心理学



- 泽农·W·派利夏恩提出核心命题：
  - 计算是心理行为的实际模型而不仅仅是模拟。
  - 提出了认知模型的计算概念。

逻辑、计算与语言

- 弗雷格思想理论
  - 思想是陈述句的含义
  - 思想有真假
  - 思想有结构
  - 思想通过语言来表达和传递
  - 存在判定思想同一性的标准
  - 思想影响人的意志
- 自然语言符合逻辑语言
  - 在保持思想的情况下，自然语言变换为逻辑语言



Gottlob Frege 1848-1925

知识表征（数学及专业基础）

- 数学分析
- 线性代数
- 概率、统计与随机过程
- 微分方程
- 电路
- 信号与系统
- 数字电路
- 数字信号处理
- 现代控制系统

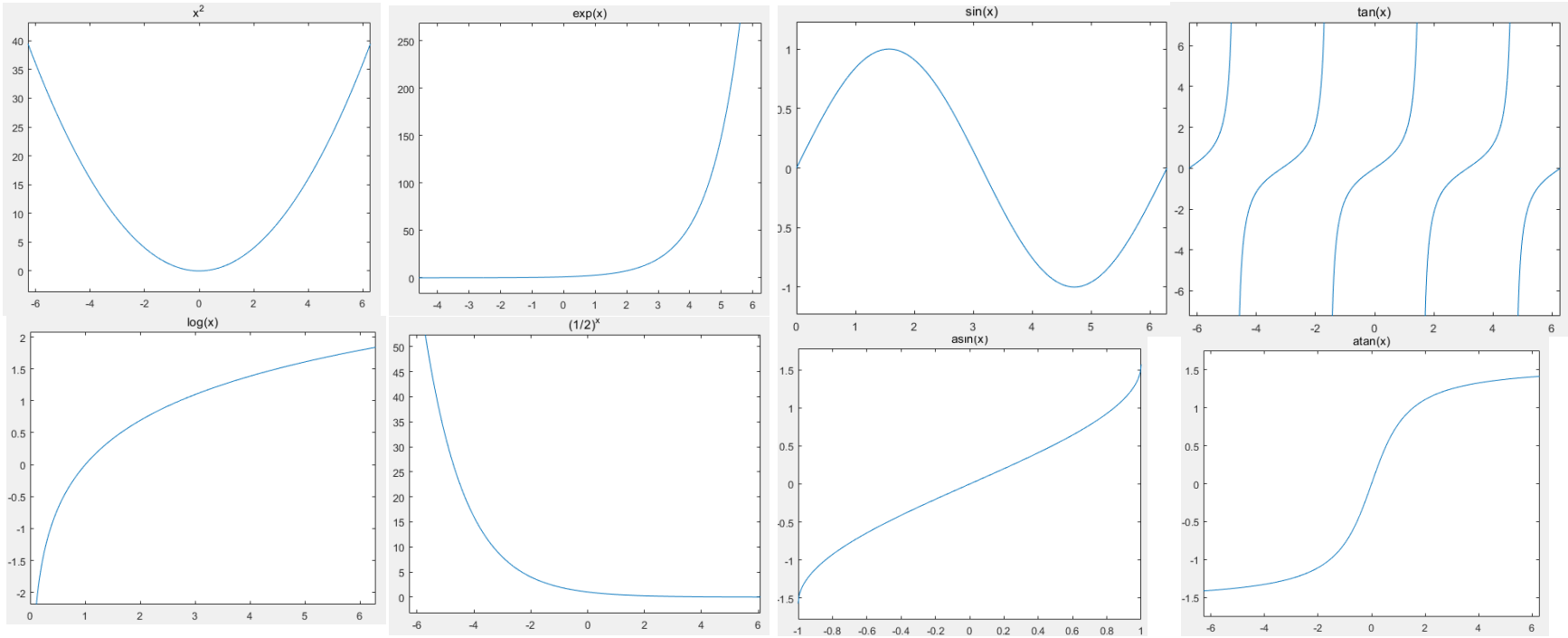
- 离散数学
- 数据结构
- 数字逻辑
- 计算机组成（CPU）
- 操作系统
- 编译系统
- 软件工程

知识表征（续）

- 在线教学实验基础问题
  - 知识的机器记忆、检索与使用
- 数学分析与离散数学是专业基础
  - 知识传授，面向人的知识理解
- 如何解决机器的知识表征问题
- 认知计算
  - 逻辑与计算表征知识

对象：函数及初等函数

- 定义：设X、Y和f是集合，若对每个 $x \in X$，存在唯一 $y \in Y$，使得 $<x, y> \in f$，则称 f 为从 X 到 Y 的函数，记为 $f: X \to Y$。



- 算术运算：
- $\mathbf{dom}\,(f \pm g) = \mathbf{dom}\,(f) \cap \mathbf{dom}\,(g)$
- $\mathbf{dom}\,(f * g) = \mathbf{dom}\,(f) \cap \mathbf{dom}\,(g)$
- $\mathbf{dom}\,(f / g) = \mathbf{dom}\,(f) \cap \mathbf{dom}\,(g) \wedge g(x) \neq 0$

- 复合与逆运算 $f \circ g(x)$
- f^{-1}

极限概念表征

- 定义（维尔斯特拉斯）：设函数$f(x)$在点x_0邻域有定义，对于任意$\varepsilon>0$，存在$\delta>0$，对于任意x，当$|x-x_0|<\delta$时，都有$|f(x)-A|<\varepsilon$，则称x趋于x_0时，函数$f(x)$的极限为A，记为$\lim_{x\to x0} f(x)=A$。
- 逻辑表征
  - $\forall\varepsilon(\varepsilon>0\to\exists\delta(\delta>0\wedge\forall x(|x-x_0|<\delta\to|f(x)-A|<\varepsilon)))$
- 计算表征
  - $\lim_{x\to x0} f(x)\equiv \text{limit}(f(x),x,x0)$

$\text{is\_limit}(f(x), x_0)$ $\leftrightarrow\forall\varepsilon(\varepsilon>0\to\exists\delta(\delta>0\wedge\forall x(\|x-x_0\|<\delta\to\|f(x)-A\|<\varepsilon)))$
$\text{limit}(f(x), x_0)=A$ $\leftrightarrow\text{is\_limit}(f(x), x_0)$

极限概念表征（续）

- 在极限语境下，逻辑表征极限概念人能够完全理解其意义，并且没有二义性，机器能够记忆、检索与使用。
  - 逻辑表征奠定定理逻辑证明的基础。
- 在极限语境下，计算表征函数极限，机器容易求值。
- 极限的陈述性知识和程序性知识分别用逻辑表征、计算表征方法实现。
- 计算定义谓词（**is_limit(f(x), x_0)**）

极限运算

- 定理：设$\lim_{x\to x0} f(x)=A$，$\lim_{x\to x0} g(x)=B$，则
- （1）. $\lim_{x\to x0} f(x)\pm g(x)=A\pm B$
- （2）. $\lim_{x\to x0} f(x)*g(x))=A*B$
- （3）. $\lim_{x\to x0} f(x)/g(x))=A/B$

```
function [ tv ] = limit_addition_operation(sv,ef,eg )
eval(sv);
    f(x)=eval(ef);
    g(x)=eval(eg);
    tv=limit(f(x)+g(x))==limit(f(x))+limit(g(x));
end
```

```
function [tv] = main(    )
syms x x0
    sv='syms x';
    ef='sin(x)';
    eg='cos(x)';
    tv=limit_addition_operation(sv,ef,eg);
end
```

- 给出10，100，1000个函数实例，验证极限运算性质为真的事实。
- 逻辑证明极限运算性质

极限性质

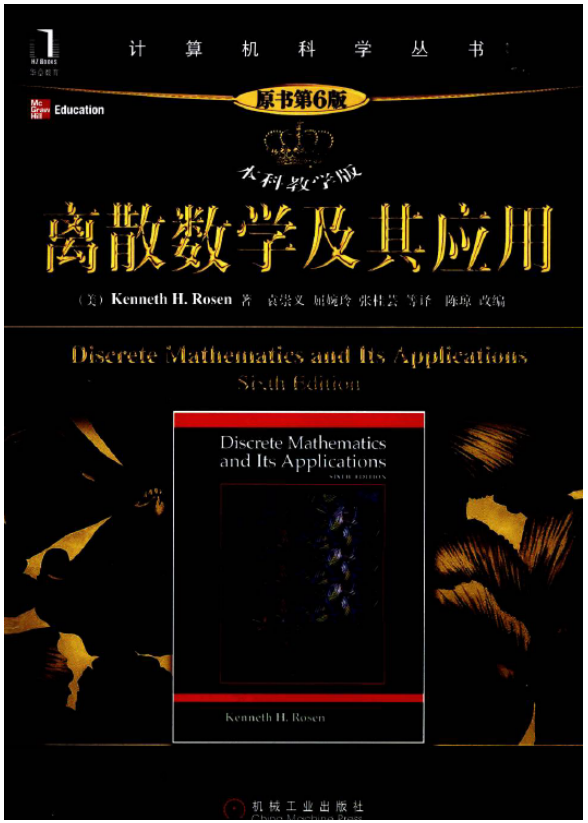
- 定理(唯一性)：设$f(x)$是邻域$U(x_0)$上函数，若极限$\lim_{x\to x0} f(x)$存在，则极限值唯一。
- 定理(有界性)：设$f(x)$是邻域$U(x_0)$上函数，若极限$\lim_{x\to x0} f(x)$存在，则函数$f(x)$有界。
- 定理（保序性）：设$f(x)$和$g(x)$是邻域$U(x_0)$上函数，$f(x)\le g(x)$并且$\lim_{x\to x0} f(x)=a$，$\lim_{x\to x0} g(x)=b$，则$a\le b$。
- 定理(夹逼性)：设$f(x)$, $g(x)$和$h(x)$是邻域$U(x0)$上函数，$f(x)\le g(x)\le h(x)$ 并且$\lim_{x\to x0} f(x)=a$，$\lim_{x\to x0} h(x)=a$

  则$\lim_{x\to x0} g(x)=a$。

limit (f(x),x0)=a∧limit (f(x),x0)=b ⊢a=b
is_limit (f(x),x0) ⊢∃M(M>0∧∀x(|x- x_0|<δ→|f(x)|<M))
f(x)≤g(x) ∧limit (f(x),x0)=a∧limit (f(x),x0)=b ⊢a≤b
f(x)≤g(x)≤h(x)∧limit (f(x),x0)=a∧limit (h(x),x0)=a ⊢limit (g(x),x0)=a

离散数学的知识表征

- 逻辑与计算表征
  - 定义概念是命题（表征）
  - 定义运算是命题（表征）
  - 定义关系是命题（表征）
  - 性质（概念、运算及关系）是命题（证明）
- 程序构建方法
  - 概念、运算及关系
  - 判断、验证与证明
- 思想表达为程序，行为由程序实现。



集合与关系的构建

- 集合是一些能够明确区分的对象构成的整体。
  - 如果对象 a 在集合 A 中，则称 a 是A 的元素。
- 元素与集合有“属于” 关系， 记为∈。
  - a 属于A，记为 a∈A，否则记为a∉A。

```
def createset(m,n):
    A=random.sample(range(m), n)
    A=set(A)
    return A
```

```
def  Cartesianproduct( X,Y):
    XY=set({})
    for x in X:
        for y in Y:
            XY.add((x,y))
    return  XY
```

```
def createrelation(m,n):
    global X,XY
    X=range(m)
    XY=Cartesianproduct( X,X)
    R=random.sample(XY, n)
    return R
```

并运算

- 定义：设**A,B**为二集合，称由**A**和**B**的所有元素组成的集合为**A**与**B**的并集，记作**A ∪ B**，称∪为并运算符。
- **A ∪ B ={ x | x∈A ∨ x∈B}**
- **A ∪ B ⇔ ∀x (x∈ A ∪ B ↔ x∈A ∨ x∈B)**

```
def unionset(A,B):
    C=A | B
    return C
def  main( ):
    A={1,2,3,4}
    B={2,4,5,'a','b','c'}
    C=unionset(A,B)
    return C
```

```
x in A | B==x in A | x in B
```

集合运算性质的验证

结合律	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
交换律	$A \cup B = B \cup A$ $A \cap B = B \cap A$
分配律	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
德•摩根律	$\sim(A \cup B) = \sim A \cap \sim B$ $\sim(A \cap B) = \sim A \cup \sim B$
幂等律	$A \cup A = A$ $A \cap A = A$
吸收律	$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$
对合律	$\sim\sim A = A$

计算验证

验证： (A ∪ B) ∪ C? = A ∪ (B ∪ C)	验证结果

def main( ):
global A,B,C
m=20
n=10
A=createset(m,n)
B=createset(m,n)
C=createset(m,n)
tv=(A|B)|C==A|(B|C)
return tv
	>>> A
	set([0, 3, 4, 7, 8, 9, 10, 13, 14, 16])
	>>> B
	set([1, 2, 3, 4, 8, 9, 12, 13, 15, 16])
	>>> C
	set([1, 2, 4, 6, 7, 8, 10, 11, 12, 15])
	>>> (A|B)|C
	set([0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16])
	>>> A|(B|C)
	set([0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16])

复合运算构建

- 定义 关系R和关系S的复合
- $R \circ S = \{< x, z > \mid \exists y(<x, y> \in R \wedge <y, z> \in S\}$

```
def composite(R,S):
    RS=set({})
    for (x,y) in R:
        for (u,v) in S:
            if (y==u):
                RS.add((x,v))
    return  RS
```

```
def  main( ):
    global R,S
    R=createrelation(6,15)
    S=random.sample(XY, 10)
    RS=composite(R,S)
    return sorted(RS)
```

R=[(0, 3), (0, 4), (0, 5), (1, 0), (1, 3), (2, 0), (2, 5), (3, 2), (3, 3), (3, 4), (4, 0), (4, 1), (4, 2), (5, 0), (5, 3)]

S=[(0, 1), (2, 1), (2, 2), (2, 4), (3, 2), (4, 0), (4, 1), (4, 4), (5, 1), (5, 5)]

RS=[(0, 0), (0, 1), (0, 2), (0, 4), (0, 5), (1, 1), (1, 2), (2, 1), (2, 5), (3, 0), (3, 1), (3, 2), (3, 4), (4, 1), (4, 2), (4, 4), (5, 1), (5, 2)]

关系复合性质验证

- (1) $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
- (2) $(R_1 \cup R_2) \circ R_3 = R_1 \circ R_3 \cup R_2 \circ R_3$
- (3) $R_1 \circ (R_2 \cup R_3) = R_1 \circ R_2 \cup R_1 \circ R_3$
- (4) $R \circ I_A = I_A \circ R = R$

计算验证

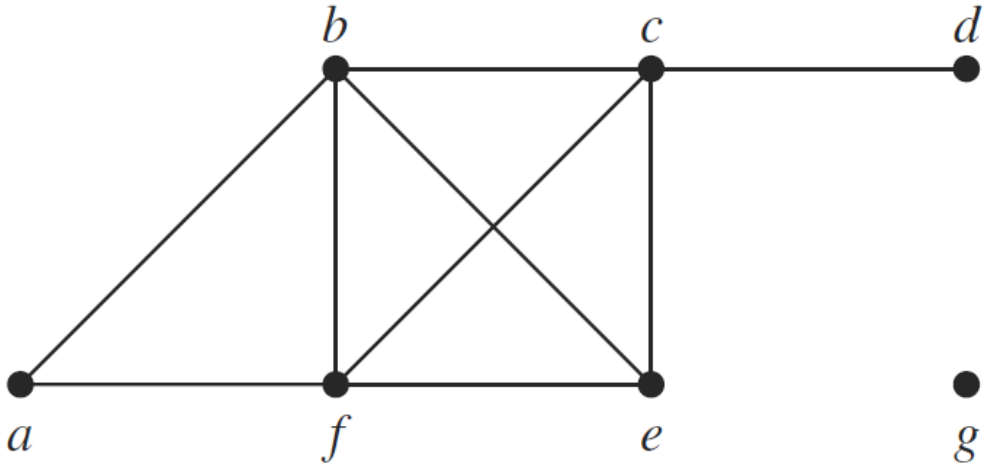
验证：$(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$	
def main( ): global X,Y,XY,R1,R2,R3,E1,E2,E m=6 n=10 X=range(m) Y=range(m) XY=set(Cartesianproduct(X,Y)) R1=set(random.sample(XY, n)) R2=set(random.sample(XY, n)) R3=set(random.sample(XY, n)) E1=composite(composite(R1,R2),R3) E2=composite(R1,composite(R2,R3)) E=E1==E2 return E	R1=set([(1, 2), (4, 5), (1, 4), (2, 2), (1, 0), (0, 3), (4, 0), (3, 4), (2, 4), (3, 5)]) R2=set([(1, 2), (5, 5), (1, 5), (2, 1), (2, 0), (0, 5), (5, 0), (2, 2), (5, 2), (4, 0)]) R3=set([(3, 2), (5, 5), (4, 4), (1, 4), (0, 2), (0, 4), (5, 1), (4, 2), (0, 3), (1, 1)]) E1=set([(1, 2), (3, 2), (1, 3), (3, 3), (4, 5), (3, 4), (3, 1), (4, 4), (2, 1), (2, 4), (1, 5), (2, 3), (1, 4), (4, 3), (2, 2), (4, 2), (4, 1), (1, 1), (3, 5)]) E2=set([(1, 2), (3, 2), (1, 3), (3, 1), (3, 3), (4, 5), (3, 4), (4, 4), (1, 4), (2, 4), (1, 5), (2, 3), (2, 1), (4, 3), (2, 2), (4, 2), (4, 1), (1, 1), (3, 5)]) E=True

无向图（实例）

- 定义：设V是一个非空顶点集合，E是无向边集合，则称有序偶<V, E>为无向图，记为G= <V, E>。

$$E=\{(x, y)|x\in V\wedge y\in V\}$$

V= {'a', 'b', 'c', 'd', 'e', 'f', 'g'}
E= {('a', 'b'), ('a', 'f'), ('b', 'c'), ('b', 'e'), ('b', 'f'), ('c', 'd'), ('c', 'e'), ('c', 'f'), ('f', 'e')}



图的构建

- 构建（m,n）图

```
def  Cartesianproduct( X,Y):
    XY=set({})
    for x in X:
        for y in Y:
            XY.add((x,y))
    return  XY
```

```
def creategraph(m,n):
    global V,XY
    V=range(m)
    XY=Cartesianproduct( V,V)
    E=random.sample(XY, n)
    return [V,E]
```

图判断

- 图的判断问题：

  给定顶点集合V和边集合E，对于任意e ∈ E，e=(u,v)，u ∈V并且v∈V。

```
def isgraph(V,E):
    tv=True
    for (u,v) in E:
        tv=tv and (u in V) and (v in V)
    return tv
```

邻接矩阵算法

```
def adjacencymatrix(G):
    [V,E]=G
    n=len(V)
    A=np.zeros((n,n))
    A=np.matrix(A)
    for i in range(n):
        for j in range (n):
            if (i,j) in E:
                A[i,j]=1
    return A
```

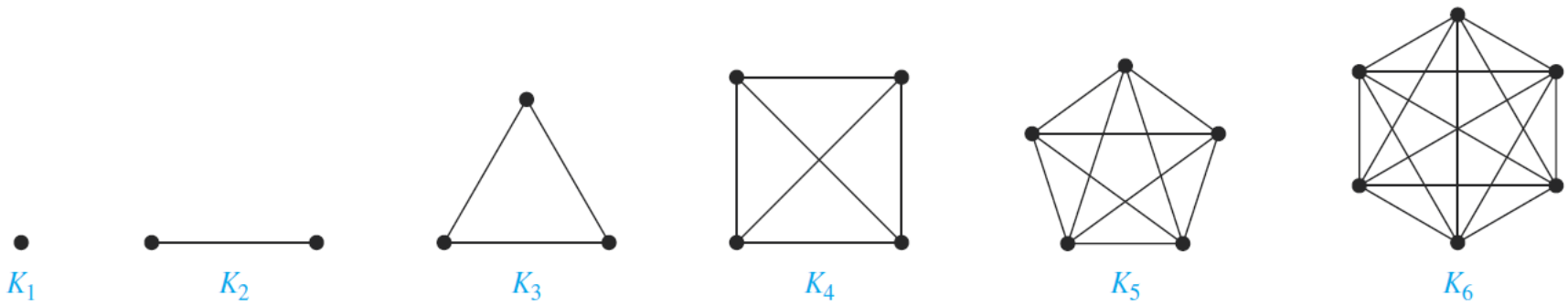
```
matrix([[ 0.,  0.,  0.,  1.,  1.,  0.,  1.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.,  1.,  0.],
        [ 1.,  0.,  0.,  1.,  0.,  1.,  0.,  0.,  0.,  0.],
        [ 0.,  1.,  0.,  0.,  1.,  0.,  1.,  0.,  1.,  0.],
        [ 1.,  0.,  1.,  0.,  0.,  1.,  0.,  1.,  0.,  1.],
        [ 0.,  1.,  1.,  0.,  1.,  0.,  0.,  1.,  0.,  0.],
        [ 0.,  1.,  0.,  0.,  0.,  1.,  0.,  1.,  0.,  0.],
        [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.]])
```

```
sorted(E)
[(0, 3), (0, 4), (0, 6), (1, 4), (1, 8), (2, 5), (3, 3), (3, 7), (3, 8), (4, 0), (4, 3),
(4, 5), (5, 1), (5, 4), (5, 6), (5, 8), (6, 0), (6, 2), (6, 5), (6, 7), (6, 9), (7, 1),
(7, 2), (7, 4), (7, 7), (8, 1), (8, 5), (8, 7), (9, 1), (9, 8)]
```

完全图（实例）

- 定义：设G= <V, E>是 (n,m)图，n≥2，若任何两个不同顶点间都恰有一条边，则称图G为完全图，记为K_n(G)。

$$\forall u \forall v(u \in V \wedge v \in V \rightarrow (u, v) \in E)$$



完全图集合构建算法

- 完全图
- $\forall u \forall v(u \in V \wedge v \in V \rightarrow (u, v) \in E)$

```
def completeset(n):
    V=set({})
    E=set({})
    for i in range(n):
        V=V | {i}
        for j in range(n):
            if(i < j):
                E=E | {(i,j)}
    return (V,E)
```

```
def completeset(n):
    V=set({})
    E=set({})
    for i in range(n):
        V.add(i)
        for j in range(n):
            if(i != j):
                E.add((i,j))
    return (V,E)
```

完全图集合构建算法统计

- n完全图（set,list）
- 给定n，统计生成完全图的边集合及时间
- 边集合存入文件
  - completegraph.txt
- E=E | {(i,j)}
- E.add((i,j))

顶点	边数	时间
10	45	0
100	4950	0, 0, 260015
200	19900	0, 5, 202298
300	44850	0, 34, 863995
400	79800	0, 136, 522808
500	124750	0, 348, 308922
600	179700	0, 799, 725741
700	244650	0, 1564, 495484
800	319600	0, 2589, 139090

顶点数	边数	时间
1000	999000	0, 0, 462000
2000	3998000	0, 1, 698000
4000	15996000	0, 8, 251000
4700	22085300	0, 12, 154000
5000		

完全图列表构建算法

```
def completelist(n):
    V=set({})
    E=[]
    for i in range(n):
        V=V | {i}
        E0=[]
        for j in range(n):
            if(i != j):
                E0=E0 + [j]
        E=E + [[i,E0]]
    return (V,E)
```

```
def completelist(n):
    V=set({})
    E=[]
    for i in range(n):
        V.add(i)
        E0=[]
        for j in range(n):
            if(i != j):
                E0.append(j)
        E.append([i,E0])
    return (V,E)
```

完全图列表构建算法统计

顶点数	边数	时间
500	249500	0, 0, 431000
600	359400	0, 0, 628000
700	489300	0, 1, 78000
800	639200	0, 1, 459000
1000	999000	0, 2, 352000
1200	1438800	0, 3, 999000
1500	2248500	0, 7, 345000
2000	3998000	0, 16, 113000
3000	8997000	0, 52, 842000
4000	15996000	0, 116, 979000
5000	24995000	0, 227, 615000

顶点数	边数	时间
2000	3998000	0, 0, 612000
3000	8997000	0, 1, 291000
4000	15996000	0, 2, 326000
5000	24995000	0, 3, 829000
11000	120989000	0, 19, 399000
12000		

小结

- 凝练计算机专业的复杂工程问题，培养学生解决复杂工程问题的意识和能力。
- 在线教学实验是新的教学实验手段，并且支持MOOC教学实验。
- 在线教学实验基础是逻辑与计算表征知识，实现机器表征和使用知识。
- 程序构建概念、运算以及关系，实现性质判断及定理验证。
- 机器表征知识
  - 一次记忆，永不遗忘


谢谢