欢迎大家选修本课程！

数据结构与算法

宋国杰

北京大学信息科学技术学院

欢迎加盟我的团队！

- **研究**：数据挖掘、机器学习与人工智能（基础）

  社会网络、智能交通系统等（应用）

- **项目**：主持国家863、科技支撑计划、国家自然基金、北京自然基金等（纵向）；NEC研究院、中兴通信、北京高速、安徽高速、山西高速、国家电网（横向）

- **成果**：国际顶级期刊和会议论文，IEEE Transaction on (TKDE、TPDS、TITS)等，KDD、AAAI等

  中国公路学会科学技术奖（一等奖）2项等

关于本课程

《数据结构与算法》

- **逻辑结构**
  - 图 ⊇ 树 ⊇ 线性
- **存储结构**
  - 顺序方法、链接方法
  - 索引方法、散列方法
- **运算**
  - 增、删、改等简单运算
  - 遍历、排序等复杂运算

时间复杂性　　空间复杂性

课程内容

第1章 概论

第2章 线性表

第3章 栈和队列

第4章 字符串

第5章 二叉树

第6章 树

第7章图

第8章 内排序

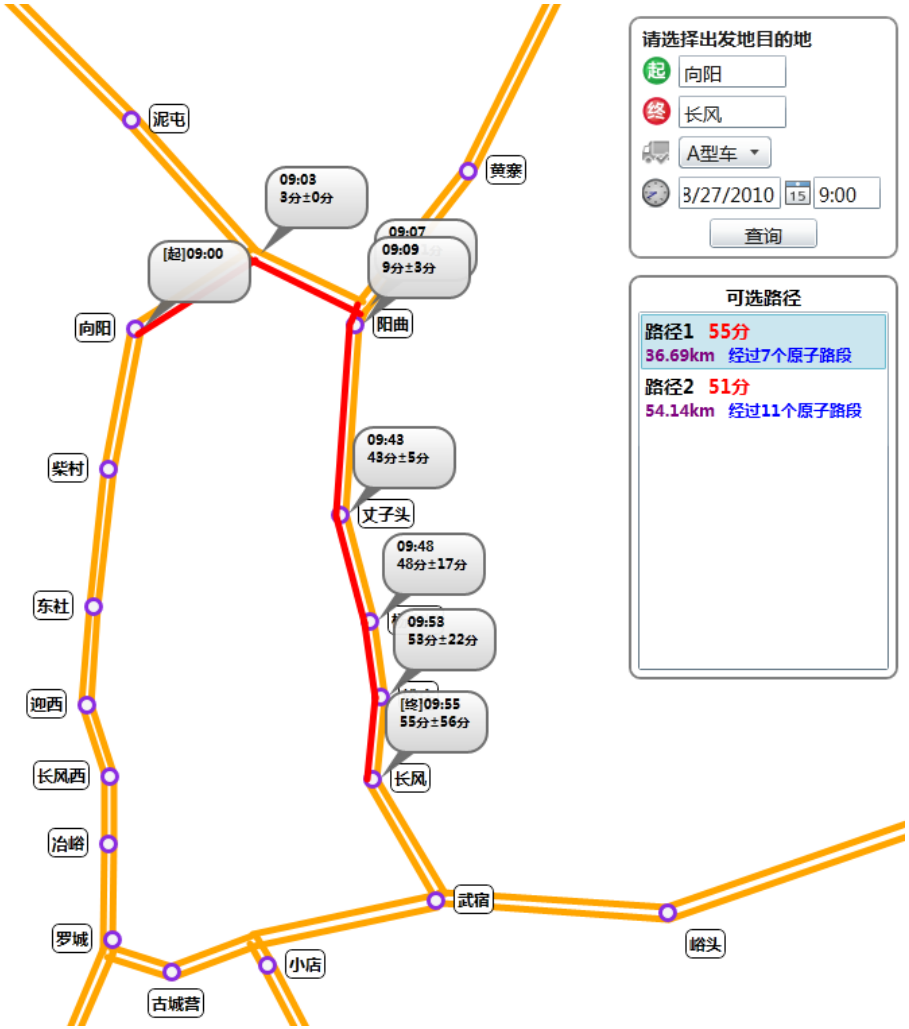
第9章 外排序

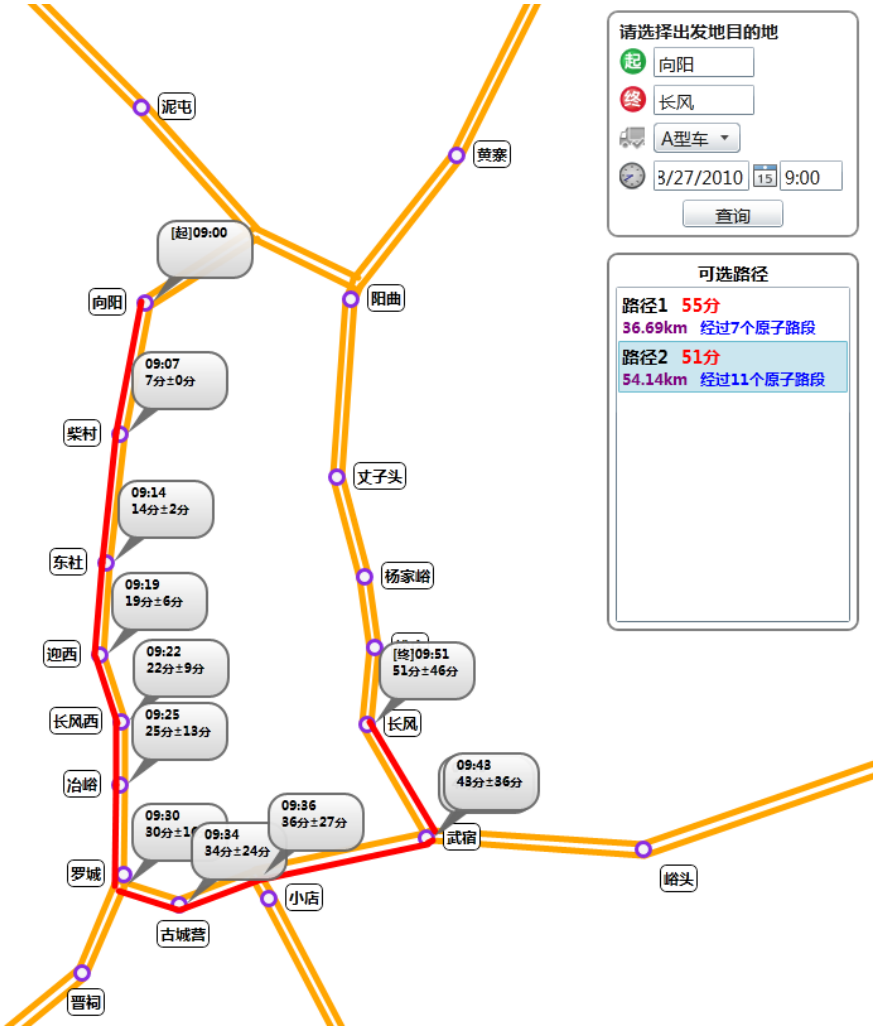
第10章 检索

第11章 索引

第12章 高级数据结构

出行导航问题


泥屯
黄寨
09:03
3分±0分
09:07
09:09
9分±3分
[起]09:00
向阳
阳曲
09:43
43分±5分
柴村
丈子头
09:48
48分±17分
东杜
09:53
53分±22分
迎西
[终]09:55
55分±56分
长风西
长风
冶峪
武宿
罗城
小店
峪头
古城营
请选择出发地目的地
起
向阳
终
长风
A型车
3/27/2010
15
9:00
查询
可选路径
路径1 55分
36.69km 经过7个原子路段
路径2 51分
54.14km 经过11个原子路段


泥屯
黄寨
[起]09:00
向阳
阳曲
09:07
7分±0分
柴村
丈子头
09:14
14分±2分
东杜
杨家峪
09:19
19分±6分
迎西
09:22
22分±9分
[终]09:51
51分±46分
长风西
09:25
25分±13分
长风
冶峪
09:43
43分±36分
09:30
09:34
34分±24分
09:36
36分±27分
武宿
罗城
小店
峪头
古城营
晋祠
请选择出发地目的地
起
向阳
终
长风
A型车
3/27/2010
15
9:00
查询
可选路径
路径1 55分
36.69km 经过7个原子路段
路径2 51分
54.14km 经过11个原子路段

课程地位


Web信息处理
队列、图、字符、矩阵散列、排序、索引、检索
人工智能
广义表、集合、搜索树及各种有向图
图形图像
队列、栈、图、矩阵、空间索引树、检索
数据库概论
线性表、多链表、排序及B+索引树
操作系统
队列、存储管理表、排序及目录树
编译原理
字符串、栈、散列表及语法树
数据结构与算法实习
算法分析与设计
数据结构与算法
概率统计
集合论与图论
程序设计实习
计算概论

课程特点

- 基础性
- 挑战性
- 理论性
- 实践性

教学目标

- **数据组织和算法分析能力**
  - 合理的组织数据、表示数据和处理数据
- **抽象能力**
  - 问题➔数据➔算法
- **算法的设计与分析**
- **编程能力**
  - 提高程序设计的质量

教学要求

- **课程讲授**
  - 讲核心、重点和难点内容，因此需要同学们预习、自学
- **诚信作业**
  - 提倡讨论，但严禁抄袭
- **加强训练**
- **有效反馈**

书面作业提交要求

- 写学号、名字
- 每次作业，都在纯文本（或PDF）中写上“我保证没有抄袭他人作业”的诚实保证。否则，计零分或根据抄袭情况倒扣分。

课程评估

- **课程作业**
  - 书面作业，分析证明类题目
  - MOOC作业
    - 概念测试、POJ算法填空、POJ算法设计与实现
- **考核方式（作业50% +考试50%）**
  - 平时作业+考勤 20%
  - MOOC作业 30%
  - POJ考试 20%
  - 期末考试 30%

教材

- 张铭，王腾蛟，赵海燕，《数据结构与算法》，高等教育出版社，2008年 6月。——普通高等教育“十一五”国家级规划教材

- 张铭、赵海燕、王腾蛟，《数据结构与算法－－学习指导与习题解析》，高等教育出版社，2005年 10月。——“十五”国家级规划教材配套参考书

课程网站

- http://course.pku.edu.cn/webapps/login/

助教信息

姓名	手机	邮箱
杜思臻	15201645119	dusizhen@126.com
刘丹萌	18707191855	304680632@qq.com
李　晨	18911840601	57577743@qq.com
焦振宇	15302182199	j.shower@163.com
李　肯	15650709373	1401214344@pku.edu.cn

第一章 概论

1. 数据结构

- 数据结构(Data Structure)
  - 涉及数据之间的逻辑关系、数据在计算机中的存储表示和在这种结构上的一组能执行的操作（运算）三个方面。
- 三要素
  - 逻辑结构
  - 存储（物理）结构
  - 运算

逻辑结构

- 逻辑结构(Logical Structure)
  - 具体问题的数学抽象，反映事物组成和逻辑关系
- 逻辑结构的表示
  - 用一组数据（表示为结点集合K），及数据间的二元关系（关系集合R）表示：（K，R）
    - K由有限个结点组成，代表一个或一组有明确结构的数据
    - R是定义在集合K上的一组关系，每个关系 r∈R都是K × K上的二元关系，描述结点间的逻辑关系

举例

- 家族成员数据结构
  - 把每个成员个体作为数据结点，而全部人员组成结点集K
  - 家族中各类亲属关系就是一组关系R
    - 其中如母子关系r、兄弟关系r*、和妯娌关系r'等

一个家族的结构

节点的类型

- 基本数据类型
  - 整数类型(integer)、实数类型(real) 、字符类型(char)和指针类型(pointer)
- 复合数据类型
  - 由基本数据类型组合而成的数据结构类型。
    - 如数组、结构类型等
  - 复合数据类型本身，又可参与定义更为复杂的结点类型
- 结点的类型可以根据应用的需要来灵活定义

结构(关系)的分类

- 逻辑结构（K,R）的分类，讨论重点在关系集R上
- 用R的性质来刻划数据结构的特点，并进行分类
  - 集合结构（set structure）
  - 线性结构（linear structure）
  - 树型结构（tree structure）
  - 图结构（graph structure）

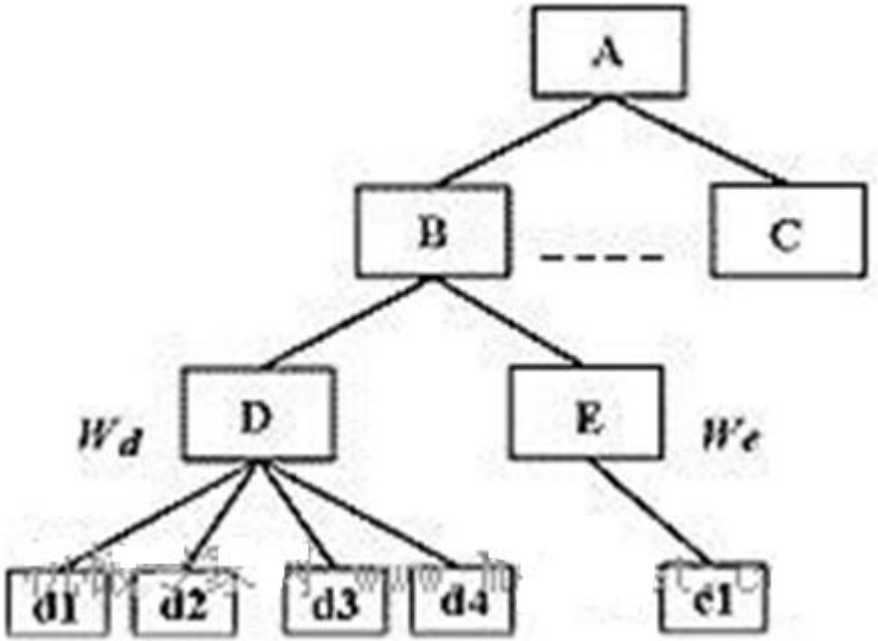
线性结构

- 亦称‘前驱关系’。关系r是有向的，且满足全序性和单索性等约束条件
  - 全序性：全部结点两两皆可比较前后
  - 单索性：每个结点都存在唯一一个前驱和后继结点

0	1	2	3	4	5	…	N-2	N-1	N

树型结构

- 亦称层次结构，每个结点可有多于1个‘直接下级’，但只有唯一的‘直接上级’
- 最高层结点称为根（root）结点，无父结点

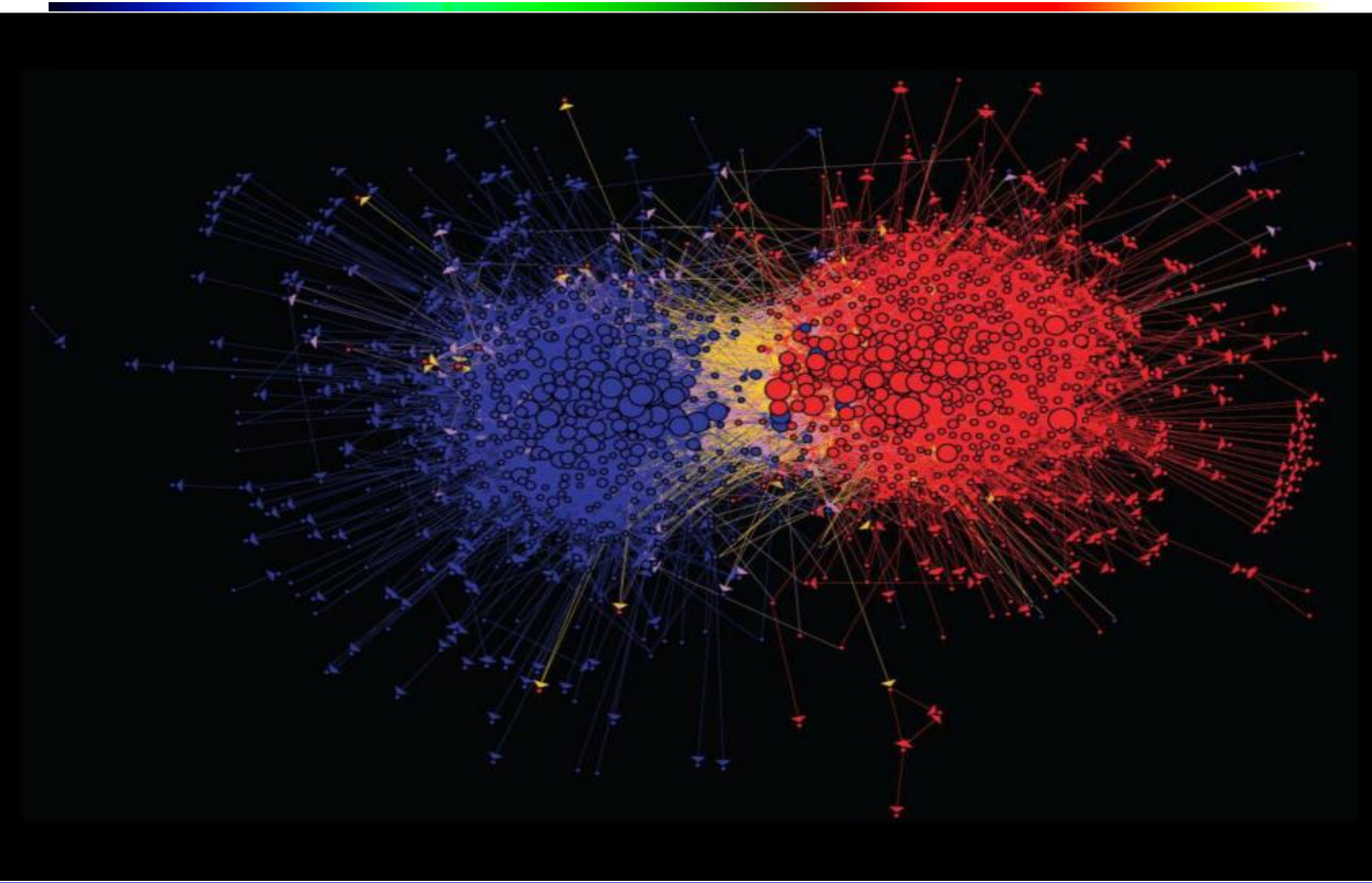


图型结构

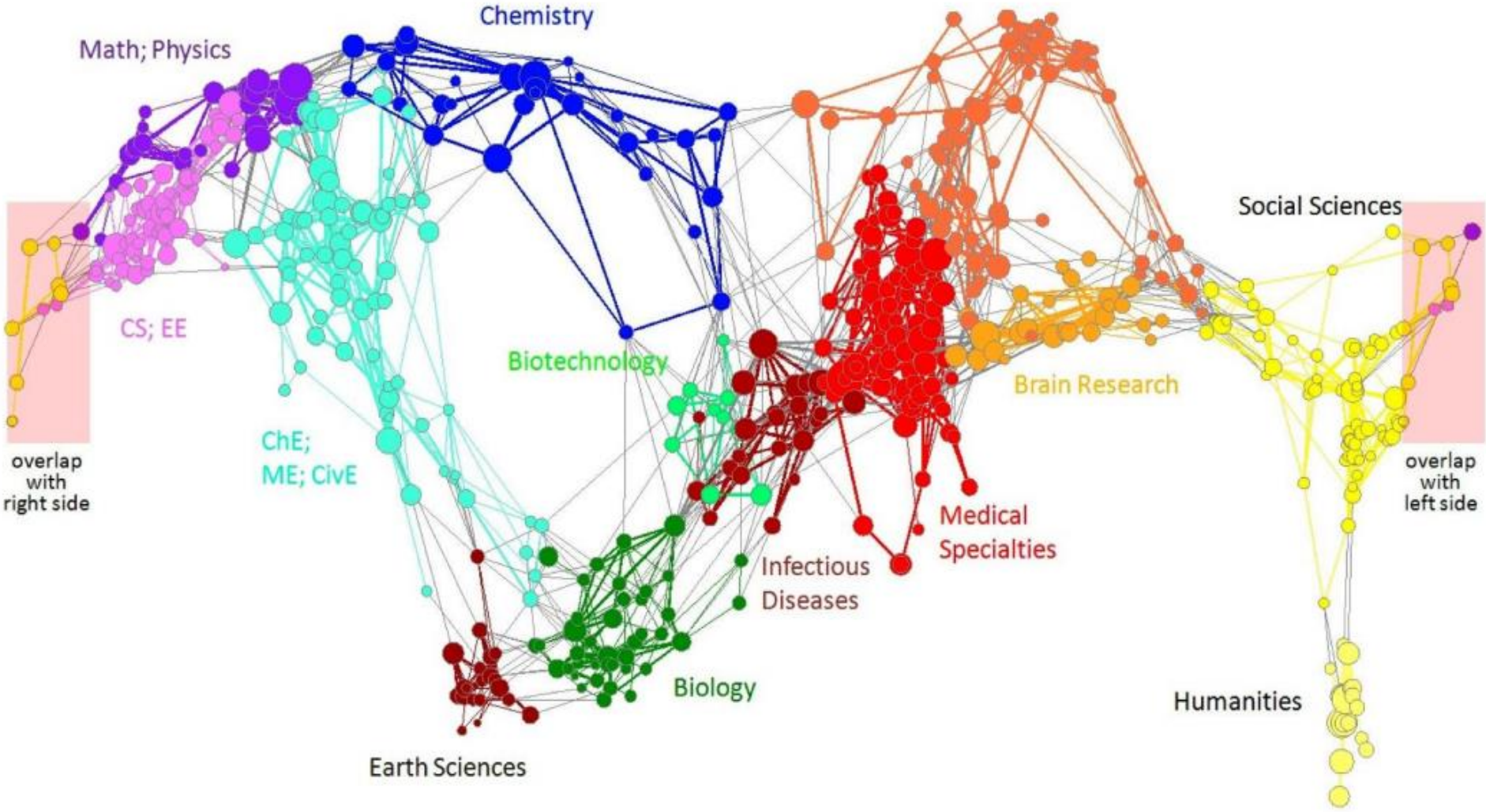
- 图结构有时称为结点互联的网络结构
  - 交通网、因特网、社会网络等
- 对于图结构的关系r没有加任何约束
- 树型结构和图型结构的基本区别是“每个结点是否仅仅属于一个直接上级”。
- 线性结构和树型结构的基本区别是“每个结点是否仅仅有一个直接的后继”

Network: Facebook Social Graph

Network: Connections between political blogs

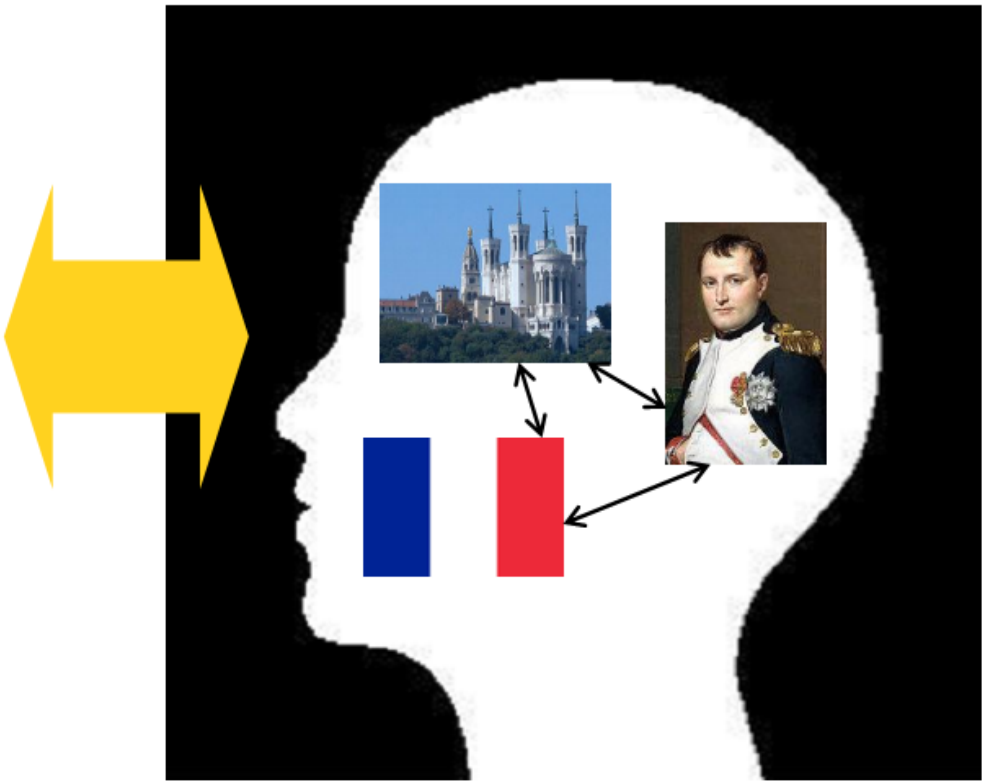


Network: Information network



Citation networks and Maps of science

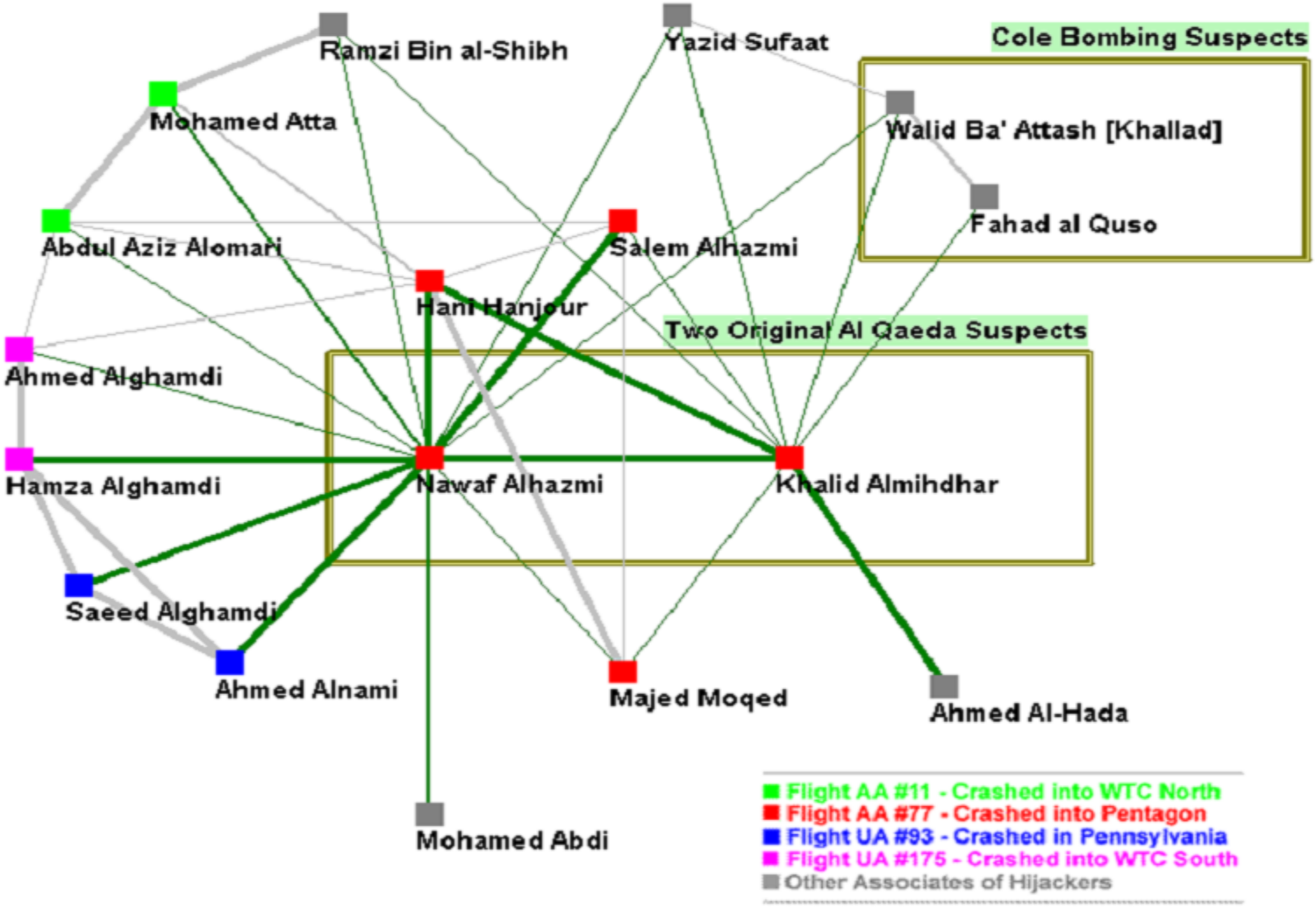
Network: Knowledge network



Understand how humans navigate Wikipedia

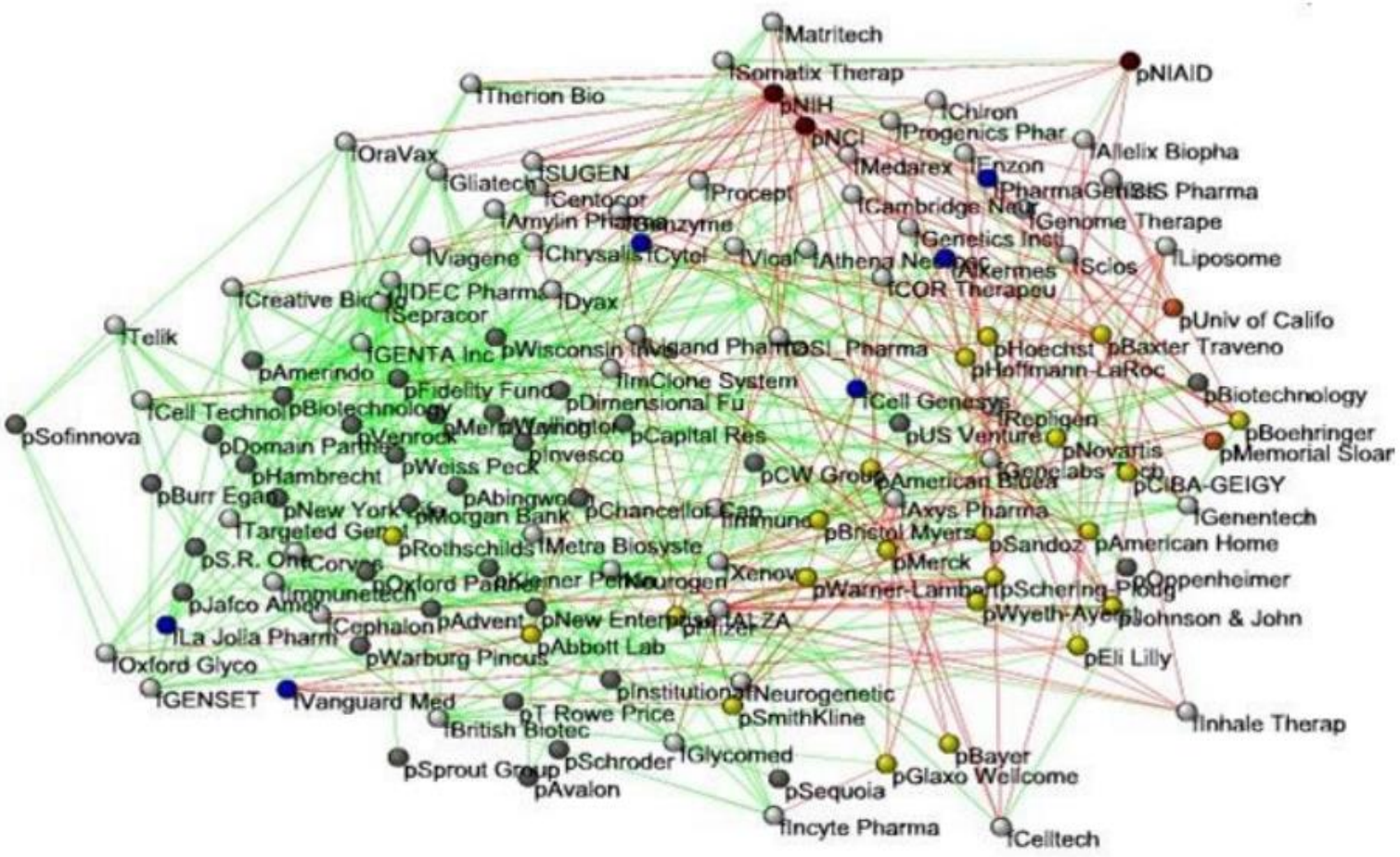
Get an idea of how people connect concepts

Network: Organization



9/11 terrorist network

Network: Economy



Nodes:

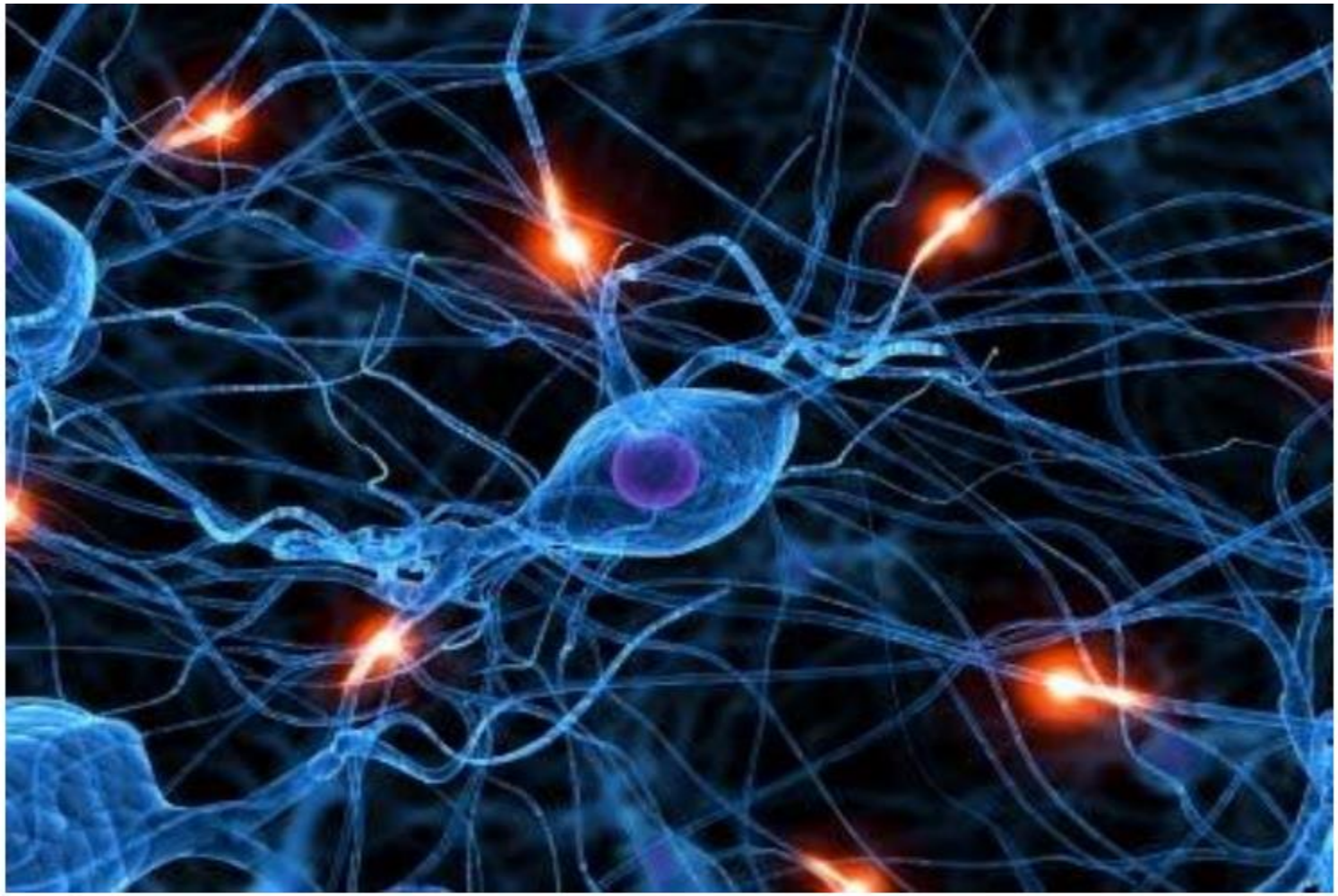
- Companies
- Investment
- Pharma
- Research Labs
- Public
- Biotechnology

Links:

- Collaborations
- Financial
- R&D

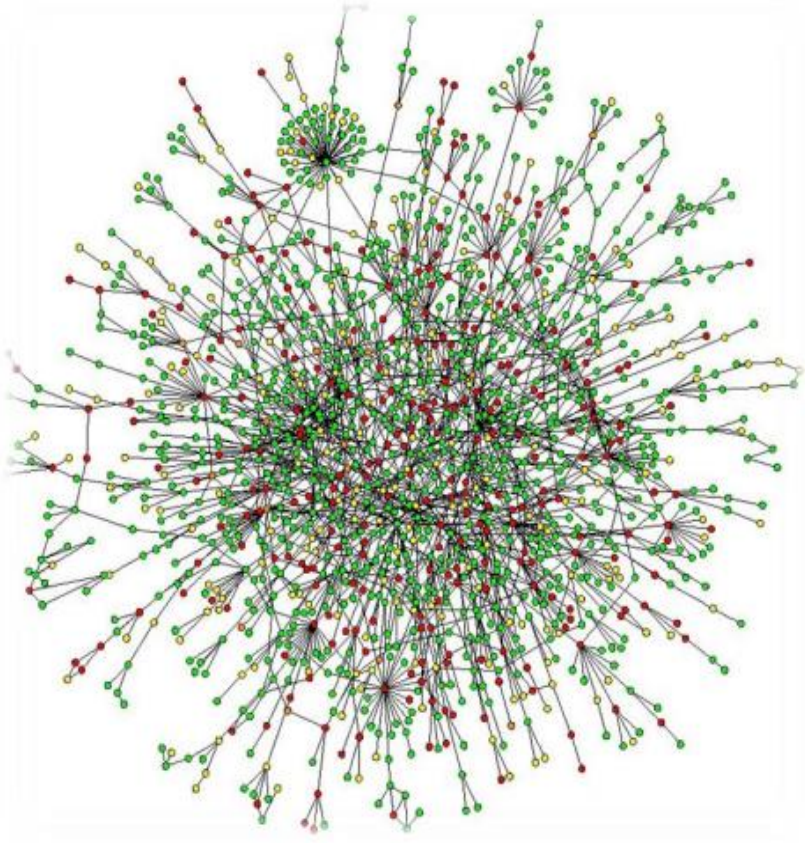
Bio-tech companies

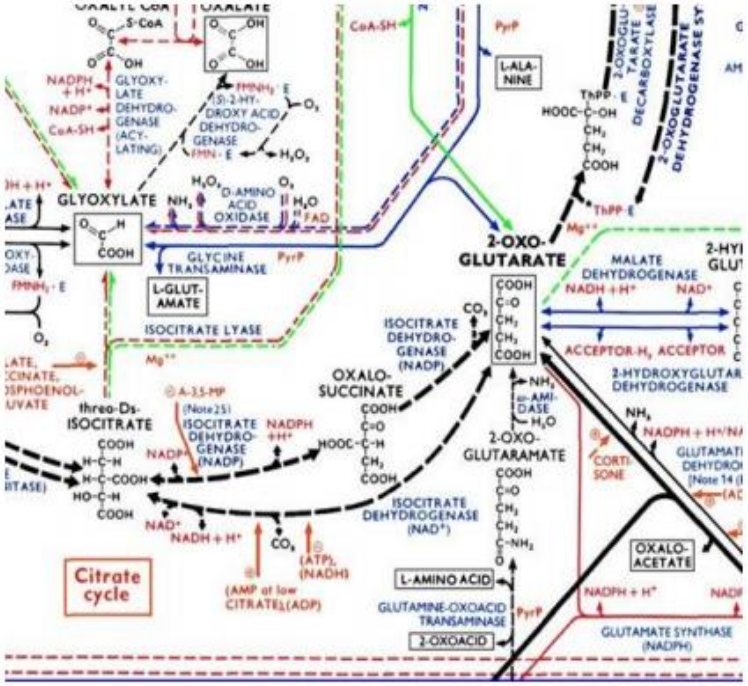
Network: Brain



Human brain has between
10-100 billion neurons

Network: Biology





Protein-Protein Interaction Networks:

Nodes: Proteins

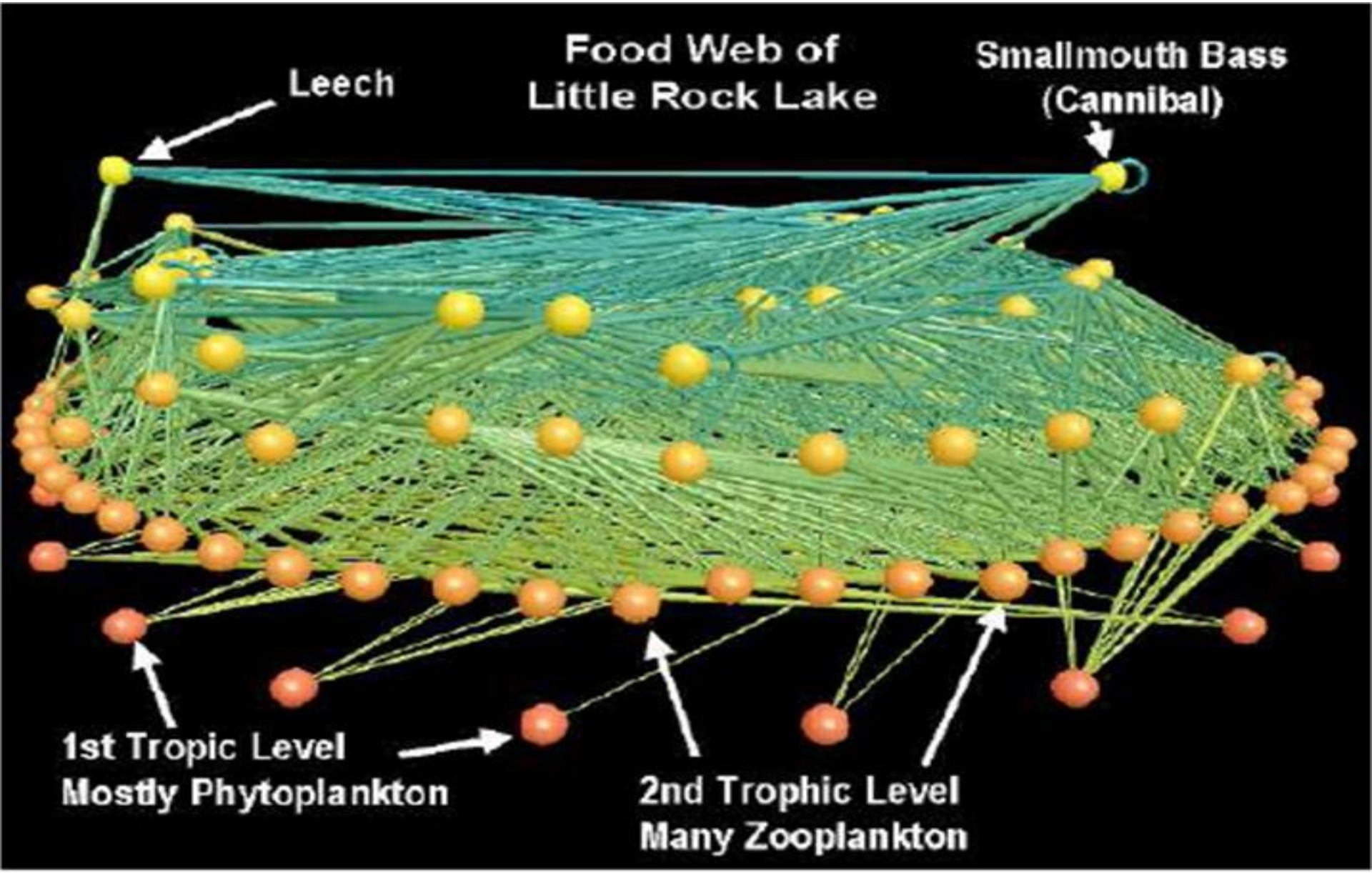
Edges: 'physical' interactions

Metabolic networks:

Nodes: Metabolites and enzymes

Edges: Chemical reactions

Network: Food Webs



结点和结构(关系)

- 自顶向下的逻辑结构分析设计方法
  - 数据结构的设计可以是一层一层地进行的
  - 先明确数据结点，及其主要关系r
  - 在分析关系r时，也要分析其数据结点的数据类型
  - 如果数据结点的逻辑结构比较复杂，那么把它作为下一个层次，再分析下一层次的逻辑结构

存储结构

- 数据的存储结构（Storage Structure）
  - 也称物理结构（Physical Structure）
  - 是逻辑结构在计算机中的物理存储表示
- 计算机主存储器
  - 空间相邻：其存储空间提供了一种具有非负整数地址编码的、存储空间相邻的单元集合，其基本的存储单元是字节
  - 随机访问：计算机的指令具有按地址随机访问存储空间内任意单元的能力，访问不同地址所需的访问时间基本相同

存储结构(Con.)

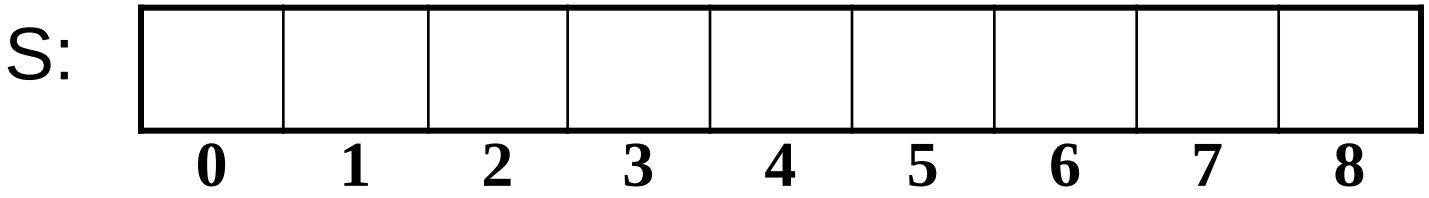
- 存储结构建立一种逻辑结构到物理结构的映射
  - 逻辑结点K到物理空间M的映射（K→M）：对于每一个结点j∈K都对应一个唯一的存储区域c∈M。
  - 逻辑关系到物理关系的映射：每一关系元组（j_1,j_2）∈r（其中j_1, j_2∈K是结点），亦即j_1, j_2的逻辑关系映射为存储单元的地址顺序关系（或指针地址指向关系）
- 四种基本存储映射方法
  - 顺序、链接、索引、散列

顺序方法

- 顺序存储
  - 结点按地址相邻关系顺序存储，结点间逻辑关系用存储单元的自然顺序关系来表达
  - 数组是顺序存储法的具体实例
- 顺序存储是紧凑存储结构
  - 紧凑指存储空间除存储有用数据外，不存储其他附加信息
  - 存储密度：存储结构所存储‘有用数据’和该结构（包括附加信息）整个存储空间大小之比

举例

- 顺序存储结构



- 支持整数编码访问

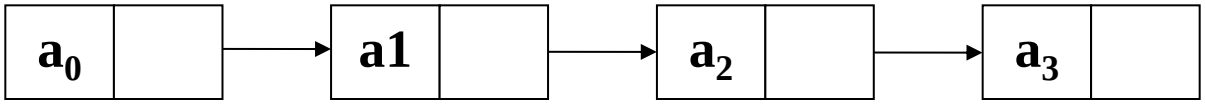
M_{00}	M_{01}	M_{02}
M_{10}	M_{11}	M_{12}
M_{20}	M_{21}	M_{22}

$$M[i][j]=M[0][0]+(k*i+j)*(\text{元素尺寸})$$

链接方法

- 链接法
  - 利用存储结构中附加指针字段来表示两个结点间逻辑关系
- 任意的逻辑关系r，都可以使用这种指针地址来表达。

  一般的做法是将数据结点分为两部分：
  - 数据字段：存放结点本身的数据
  - 指针字段：存放指针，链接到某个后继结点，指向它的存储单元的开始地址。
  - 多个相关结点的依次链接就会形成链索

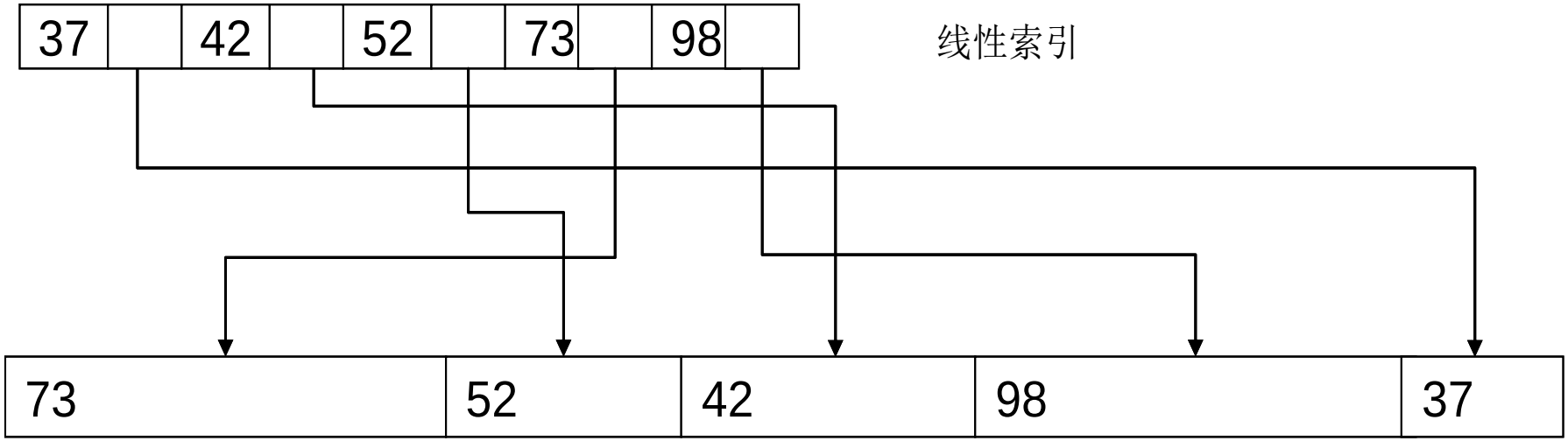


链接方法(Con.)

- 优点：增删容易
  - 顺序方法对于经常增、删结点情形，往往遇到困难
  - 链接方法结合动态存储可以解决这些复杂的问题
- 缺点：定位困难
  - 访问结点必须知道该结点的指针
  - 否则需要沿着链接指针逐个搜索，花费时间较大

索引方法

- 是顺序存储法的一种推广
- 索引方法是要建造一个由整数域 Z 映射到存储地址域D 的函数Y：Z→D，把结点的整数索引值 z∈Z映射到结点的存储地址 d∈D 。Y称为索引函数



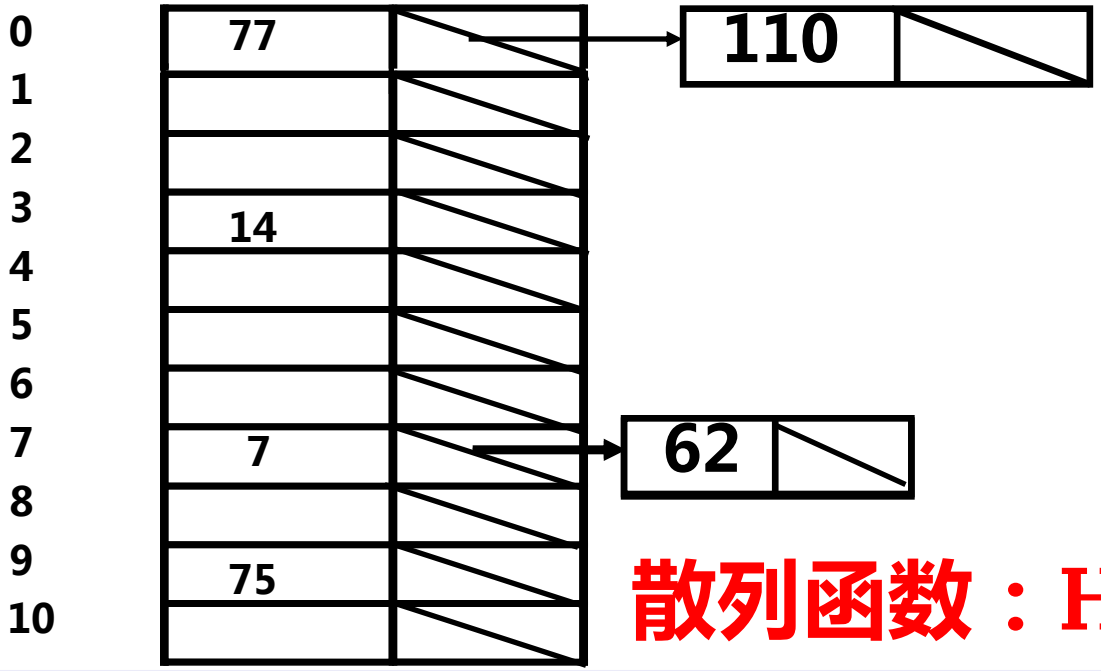
目　录

散列方法

- 散列是索引方法的延伸和扩展
- **散列函数**：将关键码 s **映射**到非负整数 z

$$h: S \rightarrow Z$$

对任意的 $s \in S$，散列函数 $h(s)=z$，$z \in Z$



散列函数：H(k) = k % 11

小结

- 数据结构
  - 逻辑结构
    - 节点
      - 基本数据类型

        整数、实数、布尔、字符，指针
      - 复合数据类型
    - 结构（关系）
      - 线性结构
      - 树型结构
      - 图型结构
  - 存储结构
    - 顺序的方法
    - 链接的方法
    - 索引的方法
    - 散列的方法
  - 数据运算

抽象数据类型(ADT)

- 抽象问题得到解决，同类具体问题都可得到解
- ADT是对多种可能的结构和实现的抽象
- 模块化思想的发展，提供了抽象数据类型的实现手段，简称ADT (Abstract Data Type)
  - 可以看作是定义了一组操作的一个抽象模型
- 一个抽象数据类型要包括哪些操作，这一点由设计者根据需要确定

抽象数据类型(续)

- 用数学方法定义对象集合和运算集合，仅通过运算的性质刻画数据对象，而独立于计算机中可能的表示方法
- 目的在于隐藏运算实现的细节和内部数据结构

抽象数据类型(续)

- 抽象数据类型由

  <数据对象，数据关系，数据操作>

  三个不可分割的部分组成的三元组:

  ADT抽象数据类型名{

  数据对象D: <数据对象的定义>

  数据关系S: <数据关系的定义>

  数据操作P: <基本操作的定义>

  }ADT抽象数据类型名

ADT示例

```
template<class Type>
class className{
    private:                    //数据结构的取值类型与取值空间
        Type  dataList;         //私有变量，存储向量元素
        …
    public:                     //运算集
          methodName();         //定义对数据的操作
        …
};
```

2. 算法

问题 —— 算法 —— 程序

目标：问题求解

- **问题（problem）一个函数**
  - 从输入到输出的一种映射
- **算法（algorithm）一种方法**
  - 对特定问题求解过程的描述，是指令的有限序列
- **程序（program）**
  - 是算法在计算机程序设计语言中的实现

数据结构 + 算法 = ？

- Pascal之父、结构化程序设计的先驱Niklaus Wirth最著名的一本书，叫作《算法 + 数据结构 = 程序》
- 程序设计的实质
  - 为计算机处理问题编制一组"指令“
- 需要解决两个 问题：算法和数据结构
  - 数据结构=问题的数学模型
  - 算法=处理问题的策略

“算法＋数据结构 ＝ 程序”

- 表达了算法与数据结构的联系及其在程序中的地位
  - 程序：采用特定逻辑结构和存储表示的数据基础上，算法的实现描述
  - 算法与数据结构是程序设计中相辅相成、不可分割的两个方面

算法的性质

- **通用性**
  - 对参数化输入进行问题求解，保证计算结果的正确性
- **有效性**
  - 算法是有限条指令组成的指令序列，即由一系列具体步骤组成
- **确定性**
  - 算法描述中的下一步应执行的步骤必须明确
- **有穷性**
  - 算法的执行必须在有限步内结束
  - 换句话说，算法不能含有死循环

算法分类

算法设计与算法分析是计算机科学的核心问题

- 穷举法(百钱买百鸡)——万能，低效
  - 避免穷举测试
- 回溯(迷宫、八皇后)、搜索(DFS, BFS)
  - 跳过无解分支、最优化问题的通法
- 递归分治(二分检索、快速排序、分治排序)
  - 子结构不重复
  - 分、治、合

算法分类(续)

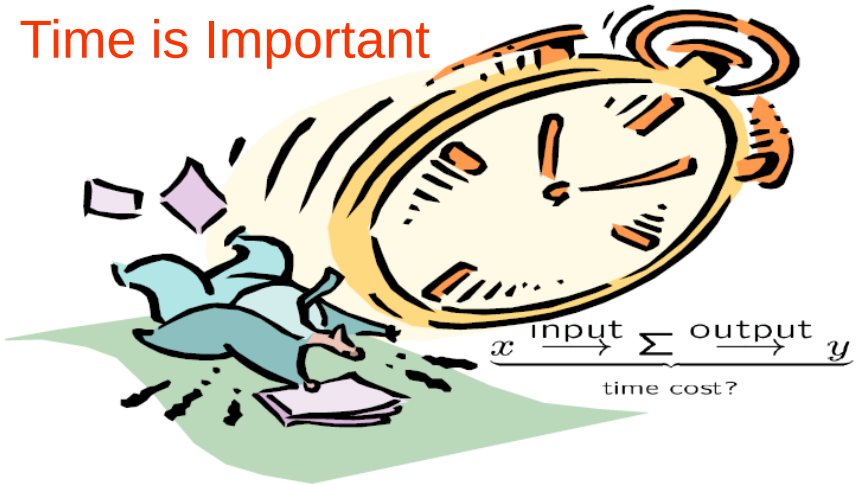
- 贪心法（Greedy）
  - 最优子结构——最优解
  - 否则，只是快速得到次优解
- 动态规划（Floyd算法）

——自底向上，利用中间结果，迅速构造

  - 最优子结构、重复子结构、无后效性
  - 搜索中分支定界的特例
  - 空间换时间

3、算法分析

- 算法分析是对一个算法需要多少计算时间和存储空间作定量分析。
- 时间资源和空间资源之间采取折衷
- 可以通过算法分析，判断所提出的算法是否符合现实情况



算法复杂性度量

- 不能用诸如微秒、纳秒这样的真实时间单位
  - 环境不同而不同
    - 一个运行在巨型机上的算法若放在PC机会慢很多，反之亦然
  - 语言不同而不同
    - C vs LISP
  - 可扩展性的不同
    - 两个不同的算法也许在输入规模为100时表现不相上下，而在输入规模扩大10倍后却表现迥异

算法复杂性度量(续)

- 重要的不是具体的时间，而是算法复杂性与输入数据规模（N）之间的关系
  - 例如，对于排序算法来说，输入规模一般就是待排序元素的个数。
- 即对应输入规模n的所需“基本操作(B)”的数来描述时间效率，与被操作的具体数值无关
  - 通常是算法最内层循环中最费时的操作

示例

- 下面是一段对数组中的各个元素求和的代码:

```
for (i = sum = 0; i<n; i++）
    sum += a[i];
```

- 数据规模: n
- 基本操作: 赋值运算
  - 在循环开始之前有两次赋值，分别对 i 和对sum进行;
  - 循环进行了n 次，每次循环中执行两次赋值，分别对sum和对 i 进行更新操作;
  - 总共有f(n)=2 + 2n次赋值操作

分析框架：增长趋势

- 增长次数
  - 小规模输入在运行时间上的差别不足以将高效的算法和低效的算法区分开来。

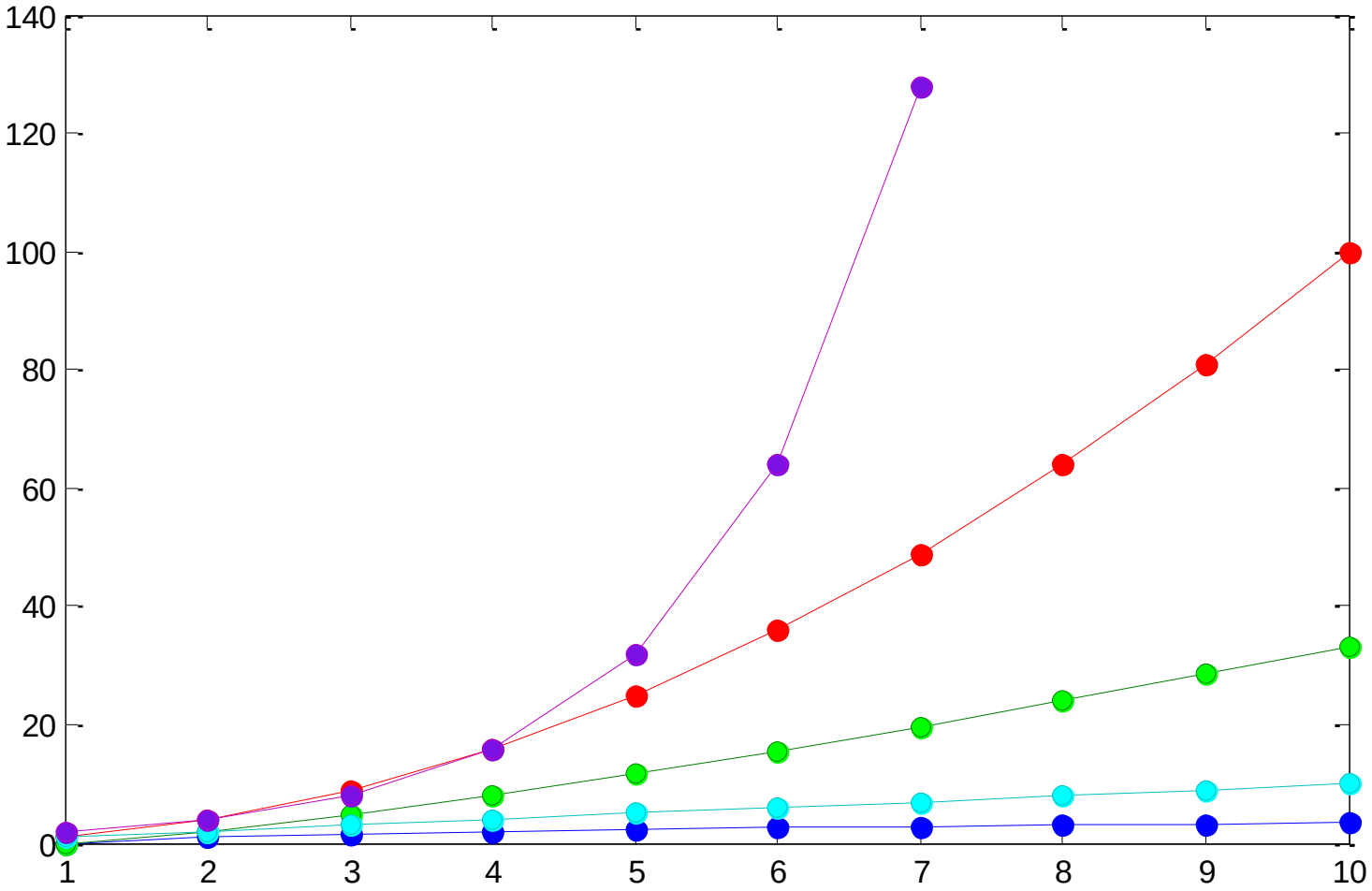
对于算法分析具有重要意义的函数值（有些是近似值）

n	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3\cdot10^1$	10^2	10^3	10^3	$3.6\cdot10^6$
10^2	6.6	10^2	$6.6\cdot10^2$	10^4	10^6	$1.3\cdot10^{30}$	$9.3\cdot10^{157}$
10^3	10	10^3	$1.0\cdot10^4$	10^6	10^9		
10^4	13	10^4	$1.3\cdot10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7\cdot10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0\cdot10^7$	10^{12}	10^{18}		

一个需要指数级操作次数的算法只能用来解决规模非常小的问题

常用函数的增长趋势

增长率函数曲线



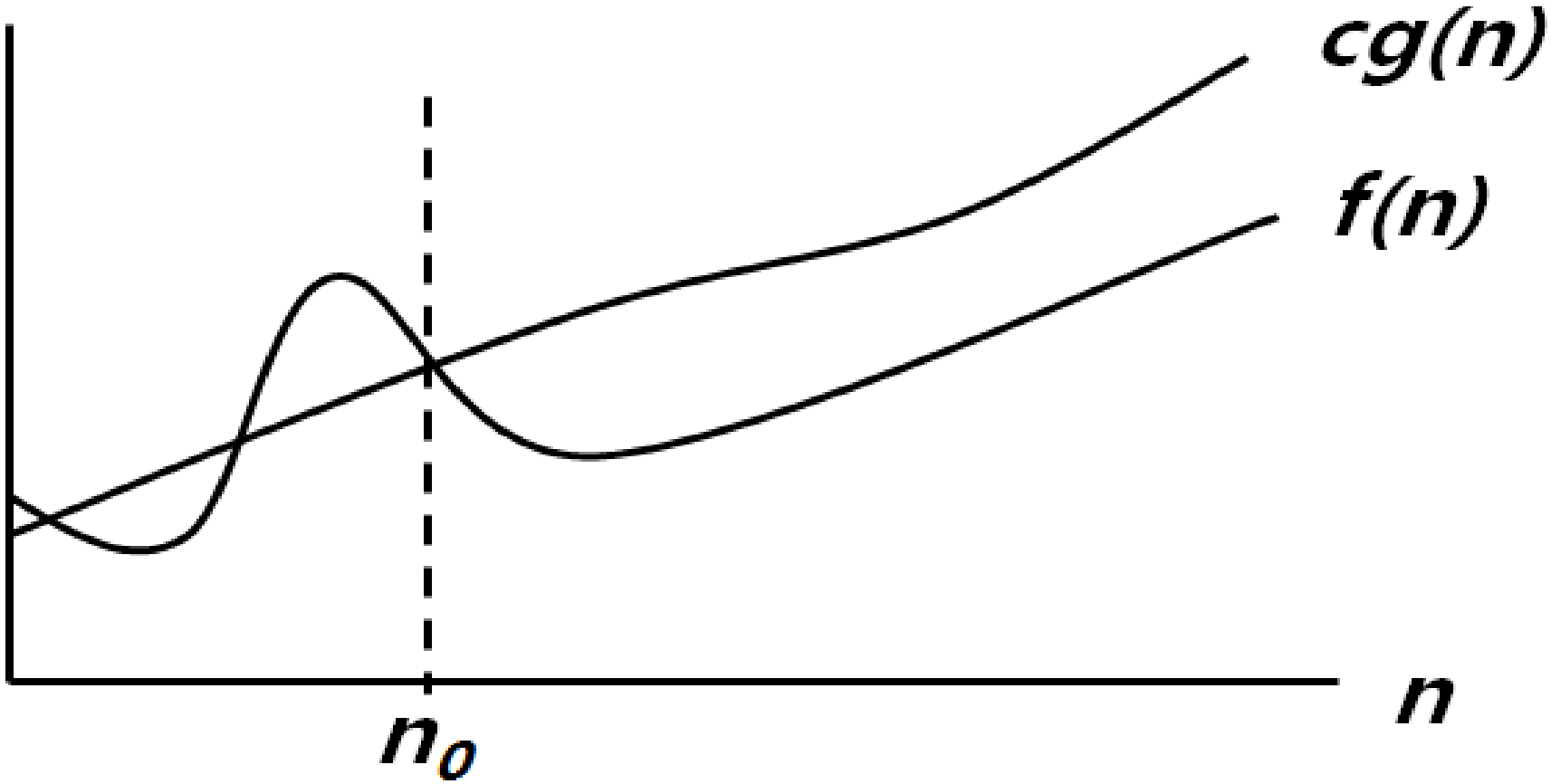
算法的渐进分析

$$f(n) = n^2 + 100n + \log_{10} n + 1000$$

- 算法分析就是要估计，当数据规模 n逐步增大时，资源开销f(n)的增长趋势
- 从数量级大小的比较来考虑，当n增大到一定值以后，资源开销的计算公式中影响最大的就是n的幂次最高的项，其他的常数项和低幂次项均可忽略。

大O表示法

- 定义1：如果存在正数c和n0，使得对任意的n≥n0，都有



大O表示法的性质

- **若符号a是不依赖于n的任意常数**
  - 如果函数 f(n) 是O(g(n))的，g(n)是O(h(n))，那么f(n)是O(h(n))的;
  - 如果函数f(n)是O(h(n))的，g(n)是O(h(n))，那么f(n) + g(n)是O(h(n))的;
  - 函数an^k是$O(n^k)$的，a不依赖于n;
  - 若f(n) = cg(n)，则f(n) 是O(g(n))的
  - 对于任何正数a和b，且b≠1，函数$\log_a n$ 是$O(\log_b n)$的。即，**任何对数函数无论底数为何，都具有相同的增长率**
  - 对任何正数 a≠1，都有$\log_a n$ 是O(lg n)的，其中$\lg n = \log_2 n$

大O表示法的运算规则

- **加法规则**：$f_1(n)+f_2(n)=O(\max(f_1(n), f_2(n)))$
  - 顺序结构，if 结构，switch结构

- **乘法规则**：$f_1(n)\ f_2(n)=O(f_1(n)\ f_2(n))$
  - for, while, do-while结构

Ω(Omega)表示法

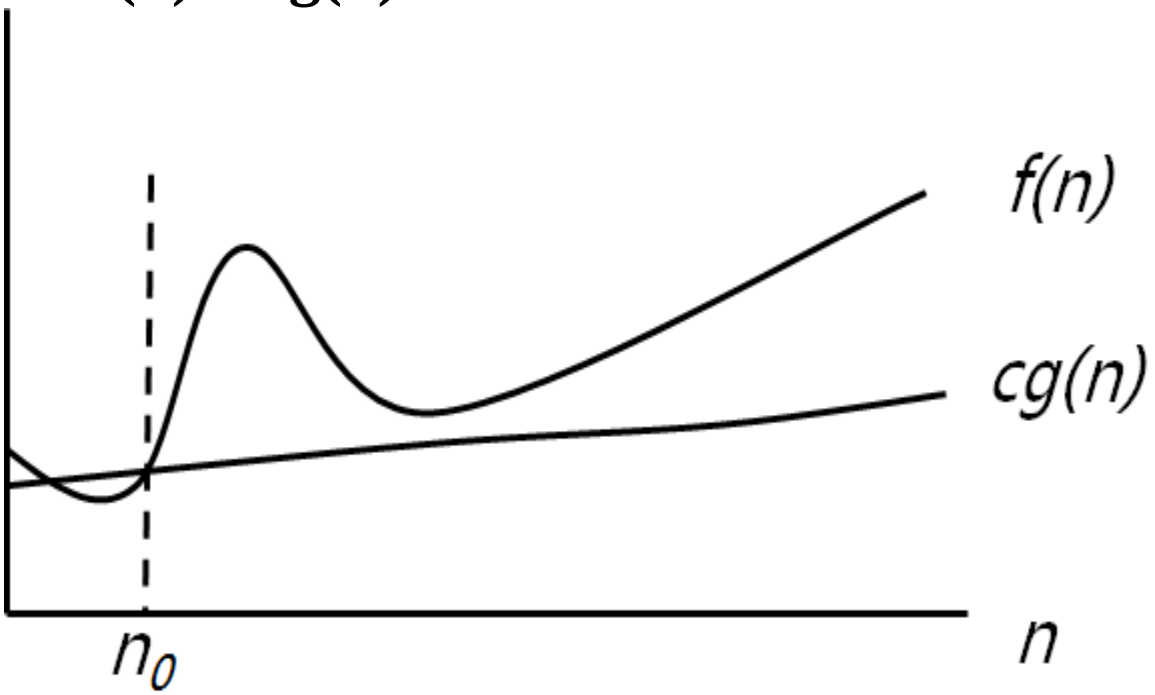
- 定义2：如果存在正数c和n0，使得对所有的n≥n0，都有

$f(n) \geq cg(n)$

$n^3 \in \Omega(n^2)$

$n(n+1) \in \Omega(n^2)$

$4n^2+5 \in \Omega(n^2)$



Θ(Theta)表示法

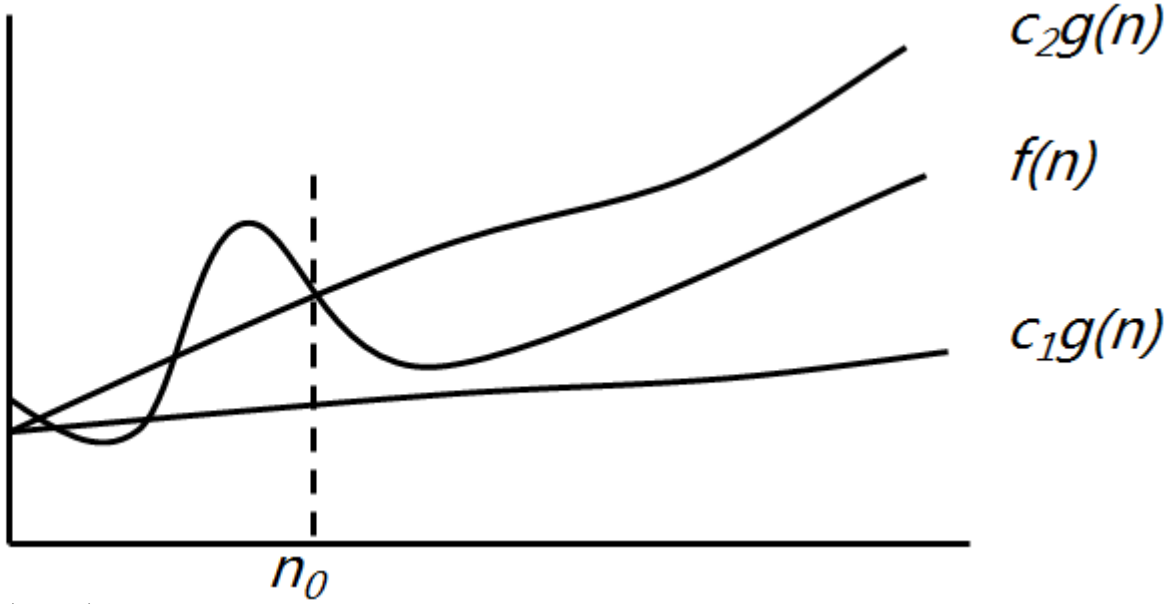
- 当上、下限相同时则可用 Θ 表示法。
- 定义3：如果一个函数既在集合O (g(n))中又在集合Ω (g(n))中，则称其为Θ (g(n))。 即存在正常数c_1, c_2，以及正整数n_0，使得对于任意的正整数n > n0，有下列两不等式同时成立：

$$c_1\, g(n) \le f(n) \le c_2\, g(n)$$

$n^2+3n+2 \in \Theta(n^2)$

$n(n-1)/2 \in \Theta(n^2)$

$4n^2+5 \in \Theta(n^2)$



算法分析示例（1）

- 依次求出给定数组的所有子数组中各元素之和:

```
for  (i = 0; i < n; i++)
        for  (j = 1, sum = a[0];  j <= i;  j++)
                sum += a[j];
```

- 循环开始前，有1次对i的赋值操作。
- 外层循环共进行n次，每个循环中包含一个内层循环，以及对i, j, sum分别进行赋值操作;
  - 每个内层循环执行2个赋值操作，分别更新sum 和 j；共执行i次（i=1,2, …,n-1）。
- 整个程序总共执行的赋值操作为:

$$1+3n+\sum_{i=1}^{n-1}2i=1+3n+2(1+2+....n-1)=1+3n+n(n-1)=O(n)+O(n^2)=O(n^2)$$

算法分析示例（2）

- 若只对每个子数组的前5个元素求和，则相应的代码可采用下面的方式:

```
for ( i=4;  i<n;  i++)
    for (j = i-3, sum = a[i-4]; j <= i; j++)
        sum += a[j];
```

- 外层循环进行n-4次
- 对每个 i 而言，内层循环只执行4次，每次的操作次数和 i 的大小无关：11次赋值操作
- 整个代码总共进行 1 + 11*(n-4) = O(n)次
- 看似双重循环，其实线性时间

最坏、最好、和平均情况

- 算法分析受条件分支的影响
  - 算法的增长率估计往往由于算法中的条件分支而遇到困难
  - 分支走向又受输入数据取值的影响，因此很多算法都无法得出独立于输入数据的渐进估计
- 估计方法
  - 最坏情况估计
  - 最好情况估计
  - 平均情况估计

平均情况下的复杂性分析

- 平均情况下的算法复杂度分析，要考虑算法的所有的输入情况，确定每种情况下的输入数目

- 简单情况下，需要考虑每种输入情况的概率

$$C_{avg} = \sum_i p(input_i)steps(input_i)$$

$p(input_i)$为第**i**种输入的出现概率，

$steps(input_i)$为算法处理第**i**种输入时所需的基本操作数目。

复杂性分析(Con.)

- 对于时间开销，一般不注意算法的‘最好估计’。特别是处理应急事件，计算机系统必须在规定的响应时间内做完紧急事件处理。这时，最坏估计是唯一的选择

- 对于多数算法而言，最坏情况和平均情况估计，它们的时间开销的公式虽然不同，但是往往只是常数因子大小的区别，或者常数项的大小区别。因此不会影响渐进分析的增长率函数估计

时间和空间的折衷

- 空间开销也可以实行类似的渐进分析方法
- 静态存储结构
  - 空间开销估计往往容易
- 动态存储结构
  - 算法运行过程中会有数量级地增大或缩小
  - 这种情况的空间开销的分析和估计是十分必要的

时空折衷(Trade-off between time and space)

- 空间换时间：为改善算法的时间开销，可通过增大空间开销而设计出时间开销小的算法
- 时间换空间：为缩小算法的空间开销，通过增大时间开销来换取存储空间的节省

总结

- 学完了本章，你达到下述要求
  - 数据结构的定义
  - 抽象数据类型
  - 算法的特性及分类
  - 算法的效率度量

再见…

联系信息:

电子邮件: gjsong@pku.edu.cn

电　　话: 62754785

办公地点: 理科2号楼2307室