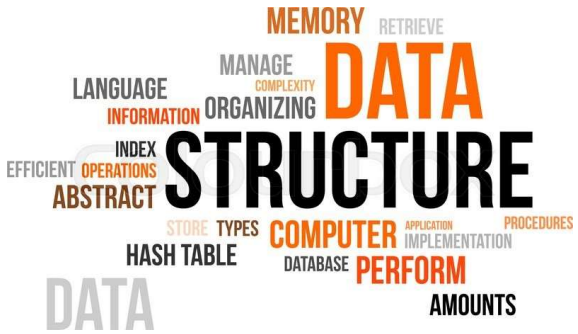
数据结构概述



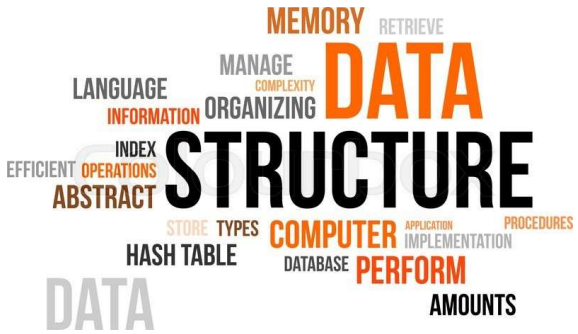
hwdong

B站 和微博： hw-dong

数据结构概述

- 什么是数据结构
- 数据元素、数据项
- 逻辑结构和物理结构
- 算法性能分析

C++版数据结构



hwdong

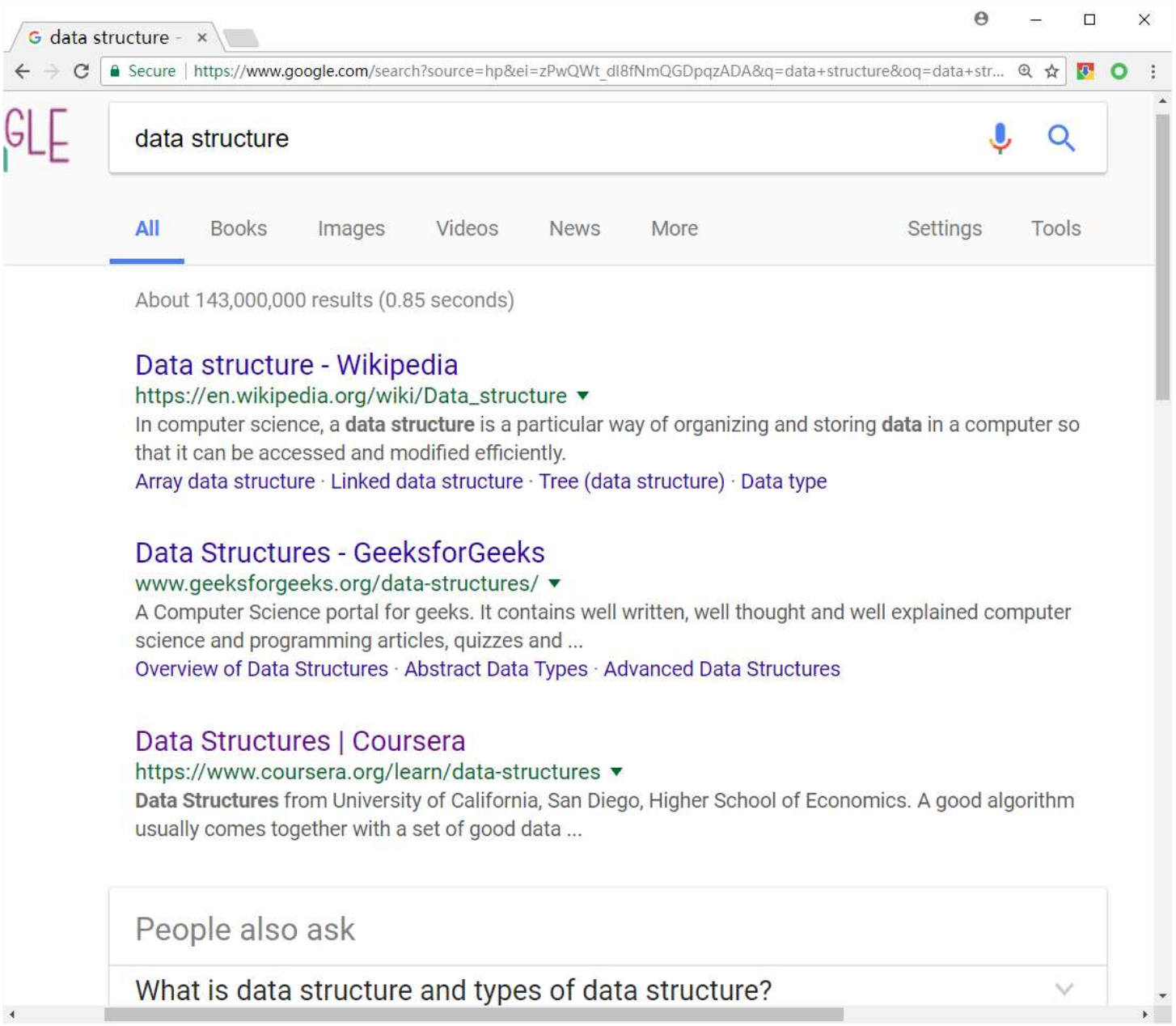
B站 和微博： hw-dong

什么是数据结构

- 数据结构就是研究以何种方式存储、组织数据，以便高效率地处理这些数据。处理数据的步骤（过程）-算法。
- 熟悉精通数据结构才能设计和开发高效率的软件系统，是软件工程师的基本内功。所有软件系统都离不开数据结构
- 硅谷、国内BAT等企业面试软件编程人员的面试题几乎都是数据结构及算法。

什么是数据结构

- 一伙人去买东西，不能乱哄哄的，可以排成一队。
- 衣橱里的衣服、书柜里的书，不能杂乱无章，应该有条理的摆放，才能方便你寻找。



搜索引擎如何时刻面对每秒几亿次点击，从大量数据中快速找到您要搜索的信息并排序？

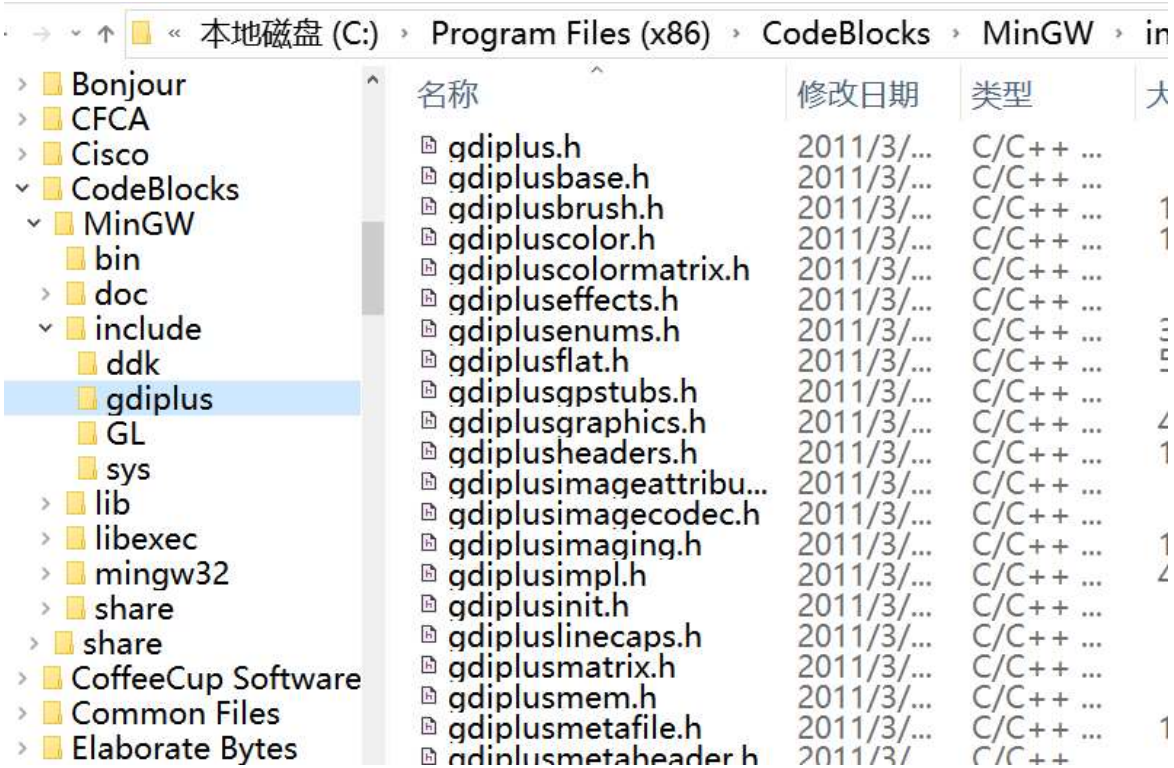
什么是数据结构

- 工资单、货物清单、学生名单等放在一张表，并按照姓名或编号排序。那么查找效率就会很高。

学号	姓名	性别	籍贯	年龄
98131	张三	男	北京	20
98164	李斯	女	上海	21
98165	王武	男	广州	19
98182	赵柳	女	香港	22
98224	...			

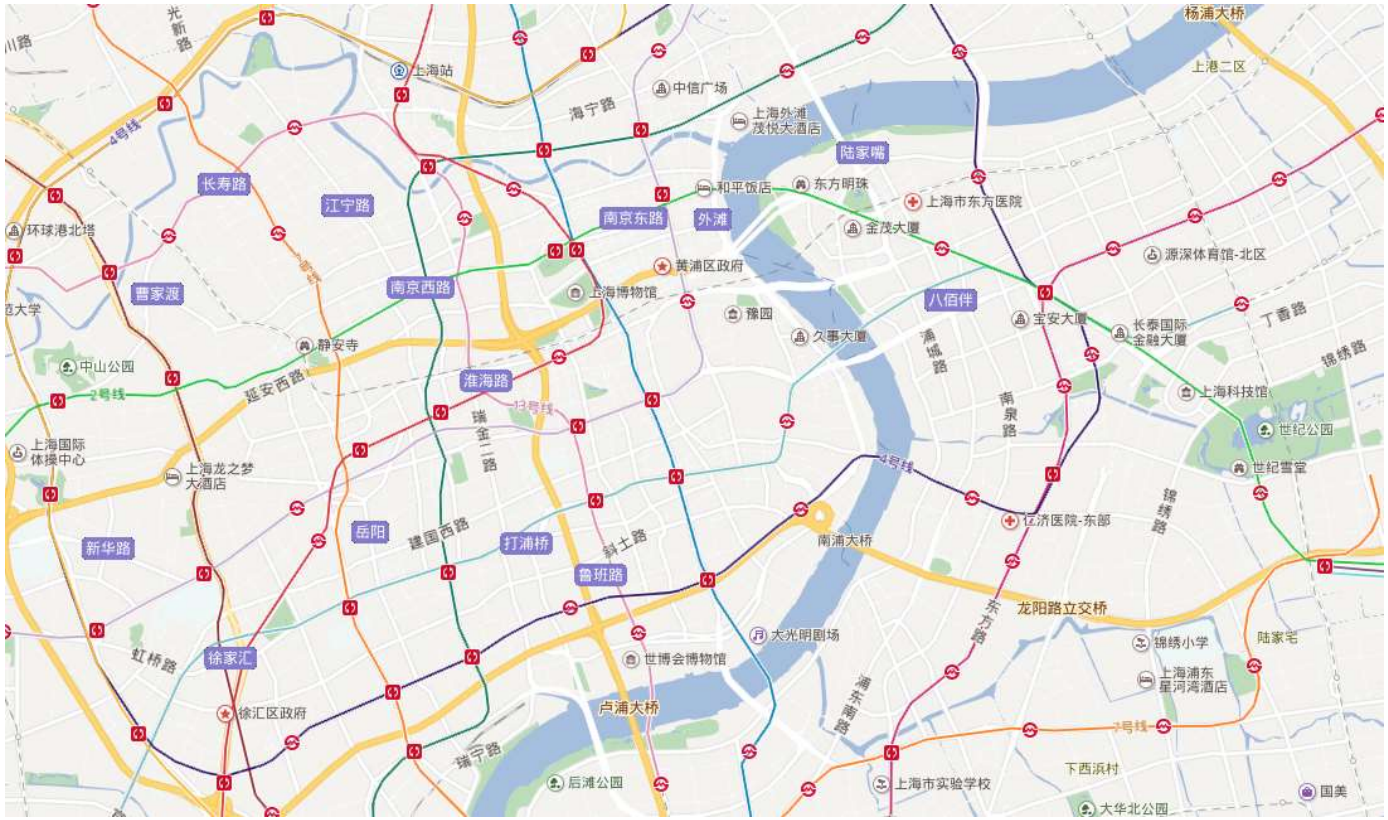
什么是数据结构

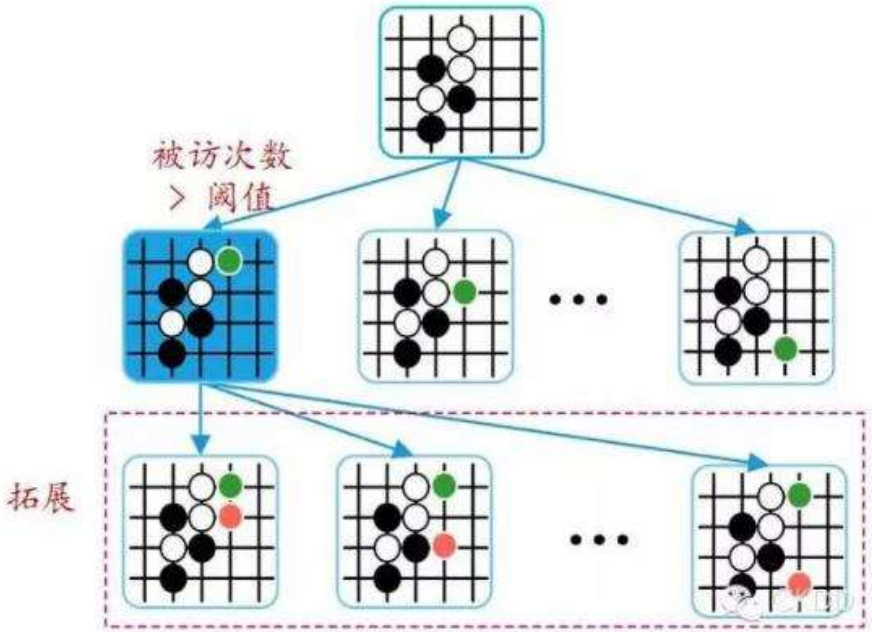
- 电脑里的文件可以放在不同文件夹，文件夹里还包含子文件夹，查找时可以按照名字或时间排序来查找。



什么是数据结构

- 地图，地点之间通过道路相连，构成一个几何图，可以用图的算法求解交通查询如最短路线等。





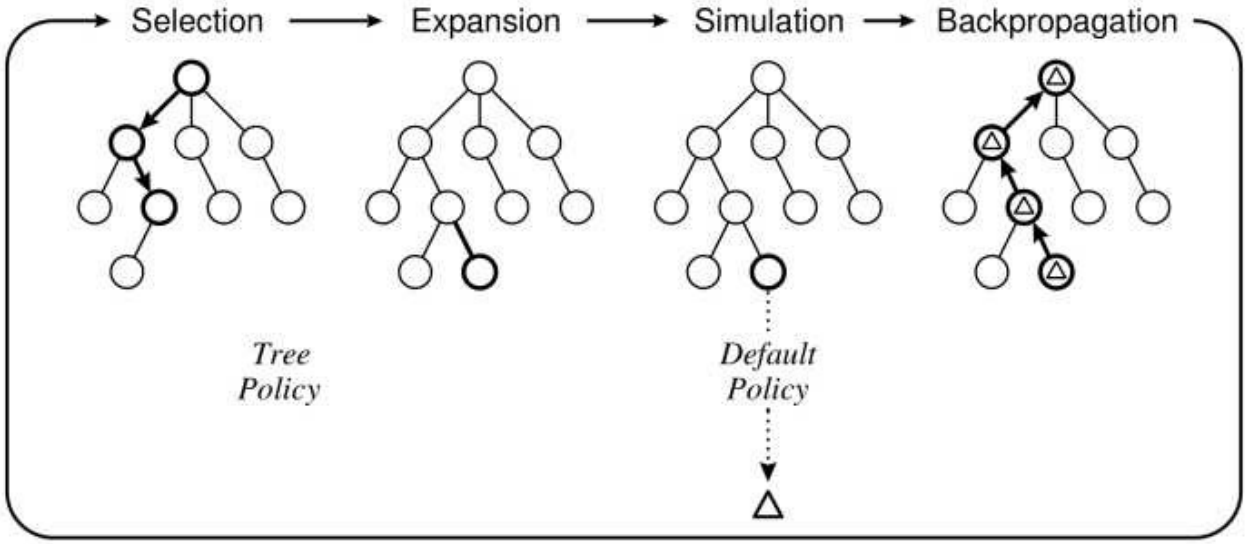
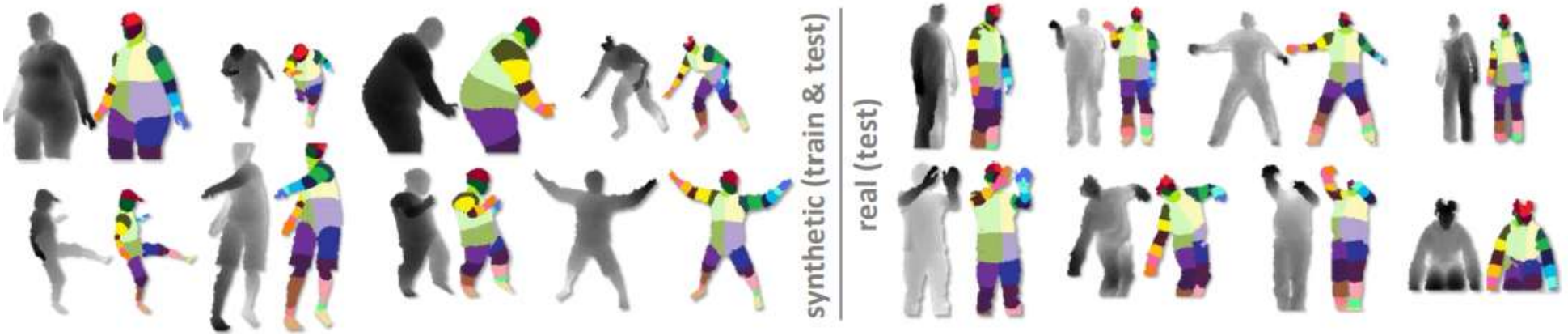
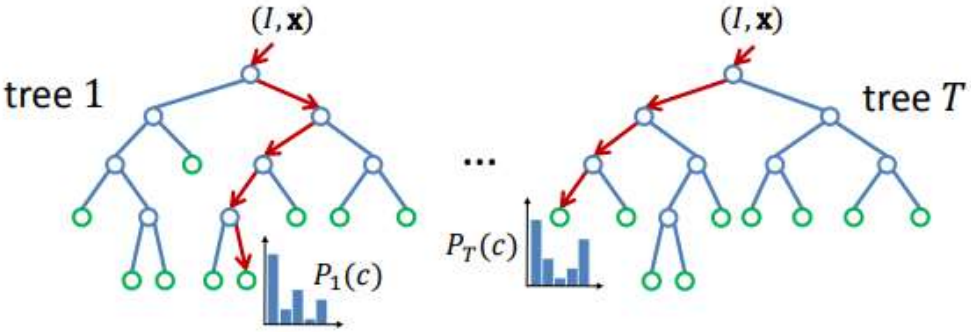


Fig. 2. One iteration of the general MCTS approach.

AlphaGo如何对棋局的大量状态空间进行快速搜索？





Kinect实时从深度图像中检查人体骨骼

什么是数据结构

- 计算机处理的数据类型各种各样，有文本、图像、视频等。
- 大数据时代数据越来越多，如果没有一个合理的结构组织这些数据，软件系统将不能有效处理这些数据。
- 数据结构研究对象：由很多同类型数据元素构成的数据集（大数据）
- 研究数据元素之间的关系及其存储表示，如何高效地对这些数据进行处理（操作）。

课程内容

- 线性表
- 栈和队列
- 字符串、多维数组
- 树和二叉树
- 图
- 查找
- 优先队列和堆
- 排序

面向面试或考研的刷题

C++类模板描述

从动画例子到原理，最后实战编程

数据元素和数据项

- 本课程主要研究同类型**数据元素**构成的数据集。**数据元素**描述一个具体对象的所有属性（属性也称为“**数据项**”），也称为“**记录**”。
- **数据元素**是数据的基本单位，在程序中常作为一个整体考虑和处理。

一个数据元素
(一条记录)

登录号	书名	借阅者编号
001	理论力学	9002
002	高等数学	9001
. . .	. . .	. . .

数据元素和数据项

- **数据项**： 数据的不可分割的最小单位
- 一个数据元素可由一或多个数据项构成

逻辑结构和物理结构

数据结构主要从2个方面研究如何有效地存储组织和处理数据。

- 逻辑结构：从抽象/高层角度来描述数据的特征和操作的抽象含义。

- 物理结构：从具体实现的角度讨论数据及其逻辑操作在计算机的表示和实现。

逻辑结构

- 数据的数学/逻辑模型：从抽象角度刻画数据的特点和操作
- 比如“整数”

1, 2, …

不涉及是1个苹果，还是2个人，2斤面粉，….

运算: 加、减、乘、除等

逻辑结构

- 数据的数学/逻辑模型：从抽象角度刻画数据的特点和操作
- 比如抽象的看“手机”是一个什么东东?

一个电子设备，有外壳、触摸屏、开关按键、触摸屏、摄像头
操作：
开关机
拨打电话
收发短信
拍照
使用APP

手机是一个抽象概念，不涉及厂家品牌的具体手机。

逻辑结构

- 数据的数学/逻辑模型：从抽象角度刻画数据的特点和操作
- 数据元素（比如各种表格中的记录）之间在抽象/逻辑上看具有什么样的关系？

  线性（一对一：一个挨着一个）　棋局（一对多）
- 对数据有哪些操作：　这些操作的抽象/逻辑含义

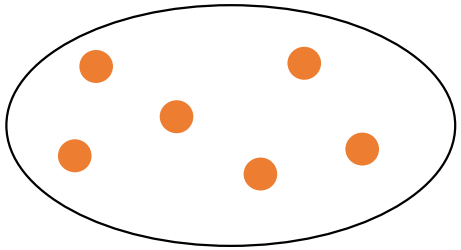
  插入　　删除　　读取　　修改　　查询

  而 不关心比如删除一个人还是一条记录

逻辑结构

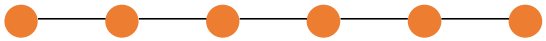
- 从抽象角度刻画数据元素之间的关系和对数据进行的操作.

- 数据元素之间的关系。有4种：集合、线性、树型、图型或网状。
- 集合关系：数据元素属于一个集合，它们之间无特定关系

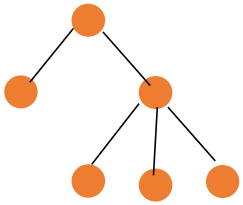


逻辑结构

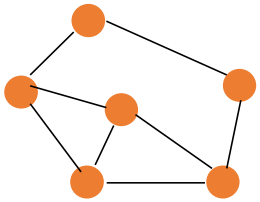
- 数据元素之间的关系。有4种：集合、线性、树型、图型或网状。
- **线性**关系： 一个接一个。比如学生花名册中的记录之间



- **树型**： 一对多。比如资源管理器中的文件夹、组织结构、家族族谱、决策树、状态树



- **图型**： 多对多。比如交通图、计算机网络、社交关系



逻辑结构

- 从抽象角度刻画数据元素之间的关系和对数据进行的操作
- 从抽象角度描述的数据结构，称为“抽象数据类型”(Abstract data type)，简称为ADT。
- 比如一个ADT叫做”线性表或列表” （List）。

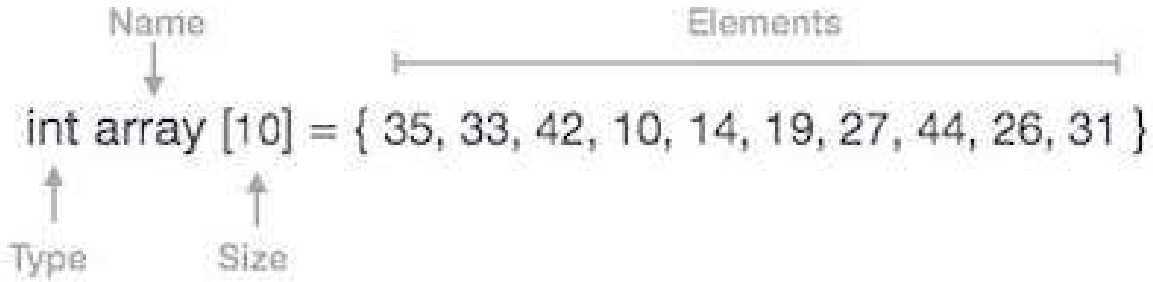
线性关系：（a_1，a_2，a_3，…）

- **存储(store)** 一系列数据元素
- **读(read)** 某个位置的数据元素
- **修改(modify)** 某个位置的数据元素

物理结构

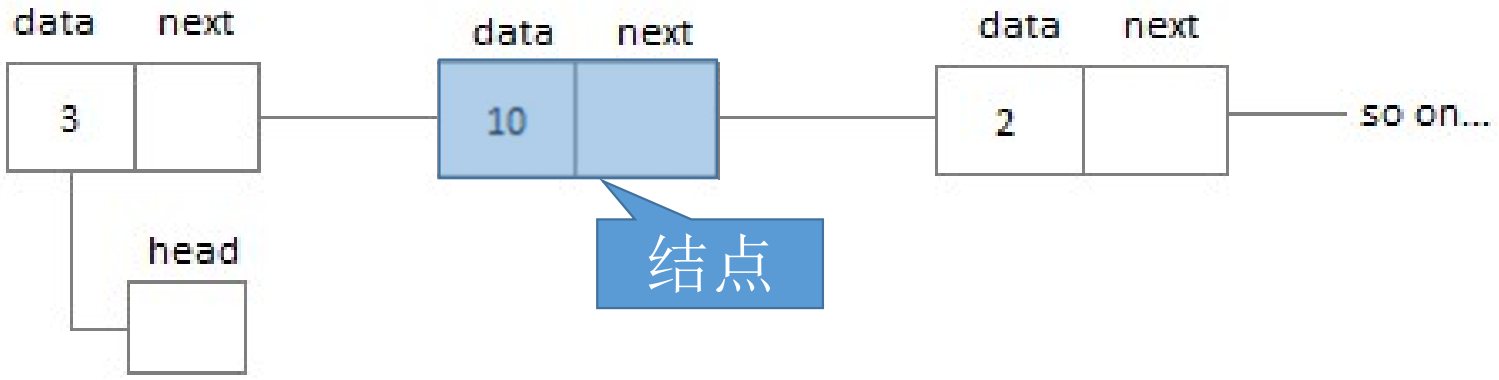
- 数据及其操作在计算机内部的表示和实现
- 如C语言的数组(array)就是ADT类型的“线性表”的具体实现

```
int  array[10];
array[2] = 30;          /*通过下标位置修改*/
printf(“%d”,array[1]);
```



物理结构

- 数据及其操作在计算机内部的表示和实现
- ADT类型的“线性表”也可以用链表实现



同一个ADT可以有不同的物理实现： 数组、链表

物理结构

- 存储结构

  数据元素及其关系在计算机中存储

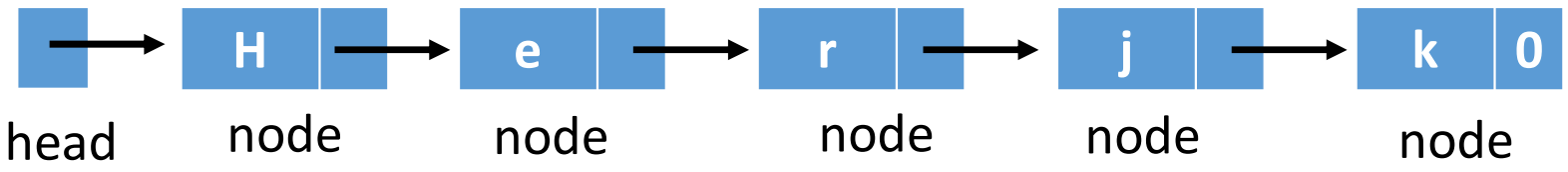
- 顺序映象（顺序存储结构），如数组

H	e	r	j	k

  以存储地址的相邻性表示数据元素间的逻辑关系

- 非顺序映象（链式存储结构），如链表

  通过指示信息表示数据元素间的逻辑关系



算法性能分析

- 算法是对数据处理过程的步骤/指令的描述, 是指令的有限序列
- 算法的特性:

有穷性：有穷步，每步有穷时间内完成

确定性：每个步骤都有确切的含义，相同的输入具有相同的执行路径和结果

可行性：每个步骤或指令应该是可行的

输入输出：有输入数据，产生或输出执行结果

算法性能分析

- 好的算法应当满足：

  正确性：算法应能满足具体问题的需求

  可读性：算法应易于阅读和理解

  健壮性：输入数据非法时，算法也能适当作出反应或进行处理

  高效性：算法执行时间短，占用存储空间少

算法性能分析: 时间复杂度

- 程序：算法在计算机上的实现
- 程序的执行时间取决于如下因素：

  算法本身

  问题的规模

  编程语言

  编译程序

  硬件性能

同一个算法程序在不同的语言、编译程序和硬件的条件下，执行时间是不同的.

比如：算法A在硬件A上执行时间为1秒，算法B在硬件B上执行时间为2秒，并不能因此就认为算法A的效率更高

算法性能分析: 时间复杂度

- 程序：算法在计算机上的实现
- 程序的执行时间取决于如下因素：

算法本身

问题的规模

编程语言

编译程序

硬件性能

同一个算法在不同的语言、编译程序和硬件的条件下，执行时间是不同的.

所以评价算法的性能应当排除这三者的影响
只需要考虑算法本身和问题的规模

算法性能分析: 时间复杂度

算法时间效率的量度

- 事后统计法:　测量一个算法执行所需要的时间

缺点:

需要编写测试程序

测量结果依赖于具体的软、硬件

- 事前分析估算法

算法性能分析: 时间复杂度

- 选择一个“基本操作”，分析基本操作执行的次数。

```
for(i=1; i<=n; ++i)
      for(j=1; j<=n; ++j) {
            c[i][j]=0;
            for(k=1; k<=n; ++k)
                  c[i][j]+=a[i][k]*b[k][j];
      }
```

- 执行次数 $F = n^3$

算法性能分析: 时间复杂度

- 选择一个“基本操作”，分析基本操作执行的次数。
- 用该基本操作的重复次数表示算法的执行时间，一般为问题规模 n 的函数 f(n)，简称“频度”。
- 算法的时间复杂度为：和f(n)同阶的简化无穷大量

$$T(n)=O(f(n))$$

算法性能分析: 时间复杂度

- 例1

```
for(i=2; i<=n; ++i)
    for(j=2; j<=i-1; ++j) {
        ++x;
        a[i][j] = x;
    }
```

i=2: 0次

i=3: 1次

.....

i=n: n-2次

$$f(n) = \frac{(n-1)(n-2)}{2} = \frac{n^2 - 3n + 2}{2}$$

$T(n)= O(f(n)) = O(n^2)$

算法性能分析: 时间复杂度

- 例2

```
for(i = 1; i <= n; i ++) {
    m = i;
    for(j = i; j <= n; j ++)
    {
        if(data[j] < data[m])
            m = j;
        if(m != i)
        {
            temp    = data[m];
            data[m] = data[i];
            data[i] = temp;
        }
    }
}
```

执行次数=n,
T(n)=O(n)

执行次数都是
n(n+1)/2,
$T(n)=O(n^2)$

$T(n)=O(n^2+n)=O(n^2)$

矩阵运算的时间复杂度

- 两个n维向量$a=(a_1,a_2,\ldots a_n)^T$，$b=(b_1,b_2,\ldots b_n)^T$的点积

$$\begin{pmatrix} a_1 & a_2 & \ldots & a_n \end{pmatrix}\begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix} = a_1b_1 + a_2b_2 + \ldots + a_nb_n$$

2n-1(flops)=n(次乘法)+(n-1)(次加法)

矩阵运算的时间复杂度

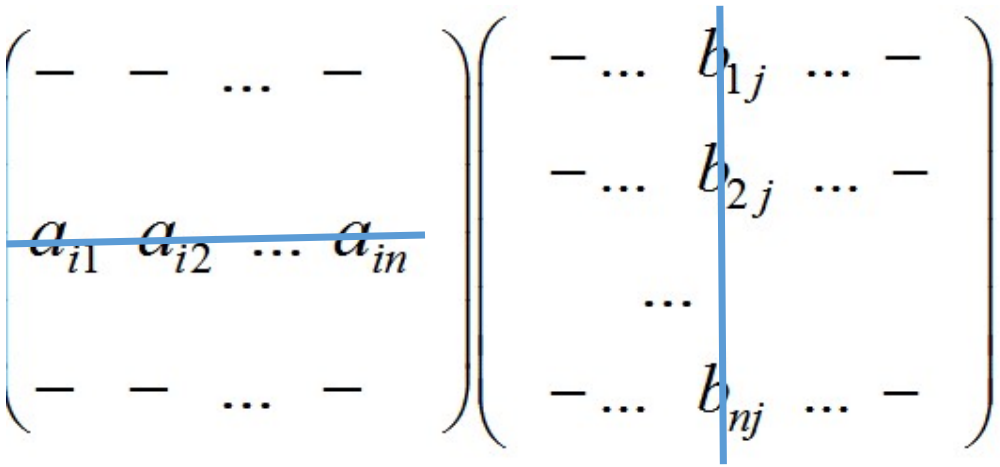
- 设A是mxn矩阵,x是n维向量，则Ax是一个m维向量，每个元素由A的一行向量和x点乘得到

$$\begin{pmatrix} - & - & \dots & - \\ a_1 & a_2 & \dots & a_n \\ - & - & \dots & - \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$$

m(2n-1) (flops)

矩阵运算的时间复杂度

- 设A是mxn矩阵, B是nxp矩阵，则AB是mxp矩阵，每个元素由A的一行和B的一列两个n维向量点乘得到



mp(2n-1) (flops)

(AB)x ? A(Bx)

- 那种方式更快?

A是mxn矩阵, B是nxp矩阵，则AB是mxp矩阵,x是px1矩阵.

AB: mp(2n-1)
(AB)x: mp(2n-1) + m(2p-1)

Bx: n(2p-1)
A(BX): n(2p-1) + m(2n-1)

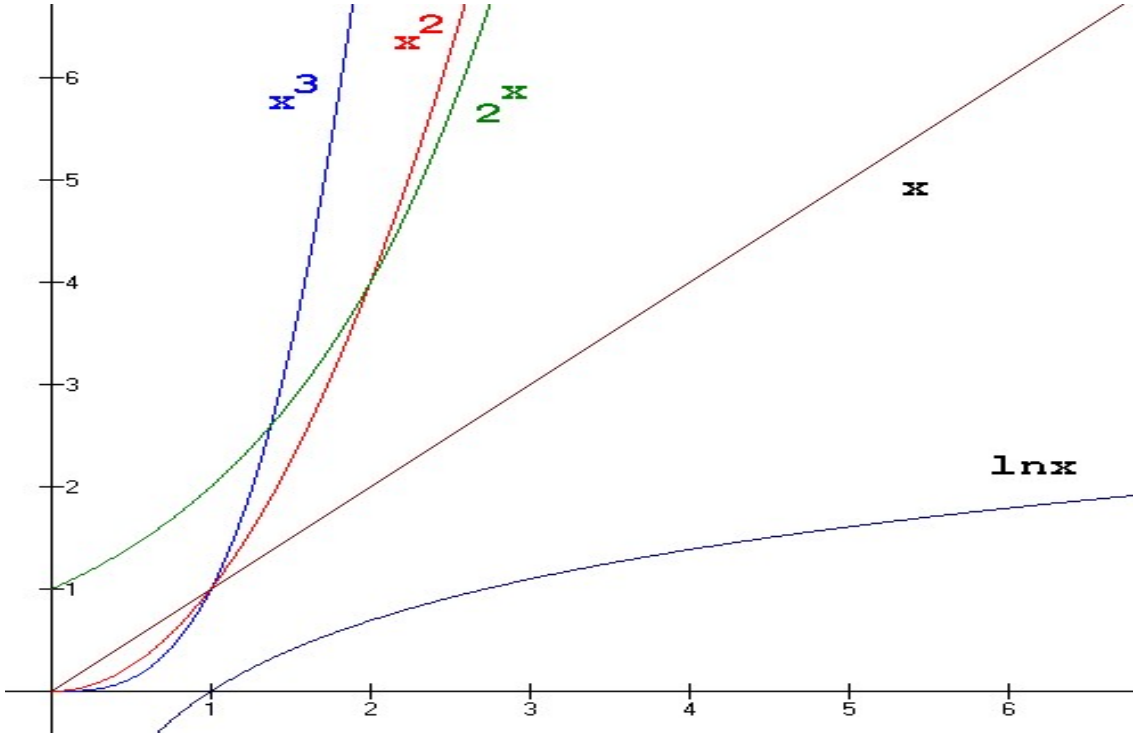
```
A = randn(n,n);  B = randn(n,n);  x = randn(n,1);
t1 = cputime;  y = (A*B)*x;  t1 = cputime - t1;
t2 = cputime;  y = A*(B*x);  t2 = cputime - t2;
```

The table shows the results on a 2.8GHz machine for eight values of n.

n	time (sec.) method 1	time (sec.) method 2
500	0.10	0.004
1000	0.63	0.02
1500	1.99	0.06
2000	4.56	0.11
2500	8.72	0.17
3000	14.7	0.25
4000	34.2	0.46
5000	65.3	0.68

算法性能分析: 时间复杂度

- 各种常见的渐进复杂度, 如：$a^n > n^b > \log_c n$



算法性能分析: 空间复杂度

- 算法执行时所需存储空间的量度，记作S(n)，其中 n 为问题的规模.
- 一般不考虑存放数据本身占用的空间，只考虑执行算法所需辅助空间，除非数据所占空间与算法本身有关.
- 排序： $(a_1, a_2, \ldots, a_n)$

	时间复杂度
冒泡排序： 1 个数据元素的辅助空间	$O(n^2)$
归并排序： n个数据元素的辅助空间	$O(n\log n)$

算法性能分析

- 有些算法的复杂度与输入数据有关
- 如冒泡排序，当输入数据基本有序时，其时间复杂度为O(n)，基本无序时，为$O(n^2)$，平均为$O(n^2)$
- 此时就应该分最好情况、最差情况、平均情况来讨论

关注

https://a.hwdong.com

B站或微博：hw-dong

网易云课堂：hwdong

腾讯课堂：hwdong.ke.qq.com

QQ群： 101132160