算法与数据结构

一、数据结构概论

setoy@qq.com 2018.8

1.1 算法是什么

DATA STRUCTURE & ALGORITHM

- 算法(Algorithm)：解决问题的步骤
  - 更快、更高效地解决问题
  - 利用经典的现成算法
  - 自己设计算法

例：Top 3

DATA STRUCTURE & ALGORITHM

- 读入10名选手的分数，按从高到低的顺序给出前3名的得分。注意，满分是100分。
- 算法1：3次搜索
  - 将各选手的得分输入到数组a[10]
  - 找到10个数中的最大值输出，然后在数组中删除这个数
  - 找到9个数中的最大值输出，然后在数组中删除这个数
  - 找到8个数中的最大值输出

例：Top 3

DATA STRUCTURE & ALGORITHM

- 算法2：排序
  - 将各选手的得分输入到数组a[10]
  - 对a数组降序排序
  - 输出a数组的前3个数

例：Top 3

DATA STRUCTURE & ALGORITHM

- 算法3：统计各个得分人数
  - 将得分为p的人的数量存入数组c[p]中
  - 然后按c[100],c[99]……的顺序进行统计，当c[i]的值大于等于1时，将i输出c[i]次，且累计达3次后停止。

更一般性的问题

DATA STRUCTURE & ALGORITHM

- Top N
- 有n个整数$a_i(i = 1,2, \dots, n)$，请按从大到小的顺序输出其中最大的m个数。限制：
- $n \leq 1\,000\,000$
- $m \leq 1000$
- $0 \leq a_i \leq 10^6$

2.1 算法评价

DATA STRUCTURE & ALGORITHM

- 影响算法（程序）执行时间的因素
  - 操作系统、硬件配置
  - 程序语言（程序语言级别越高，执行效率越低）
  - 问题的数据规模
  - 算法策略
- 衡量标准
  - 时间（每个测试点1s或3s）
  - 空间（64MB、512MB）

2.2 算法的时间复杂度

DATA STRUCTURE & ALGORITHM

```
cin >> n;
x++;
for (int i=1; i<=n; i++)
    x++
for (int i=1; i<=n; i++)
    for (int j=1; j<=n; j++)
        x++;
```

$f(n)=1$ $O(1)$ 常量阶

$f(n)=n$ $O(n)$ 线性阶

$f(n)=n^2$ $O(n^2)$ 平方阶

- 算法执行时间的增长率$T(n)$和$f(n)$的增长率相同，称为算法的渐进时间复杂度，大O表示，记作：

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

2.3 算法时间复杂度计算

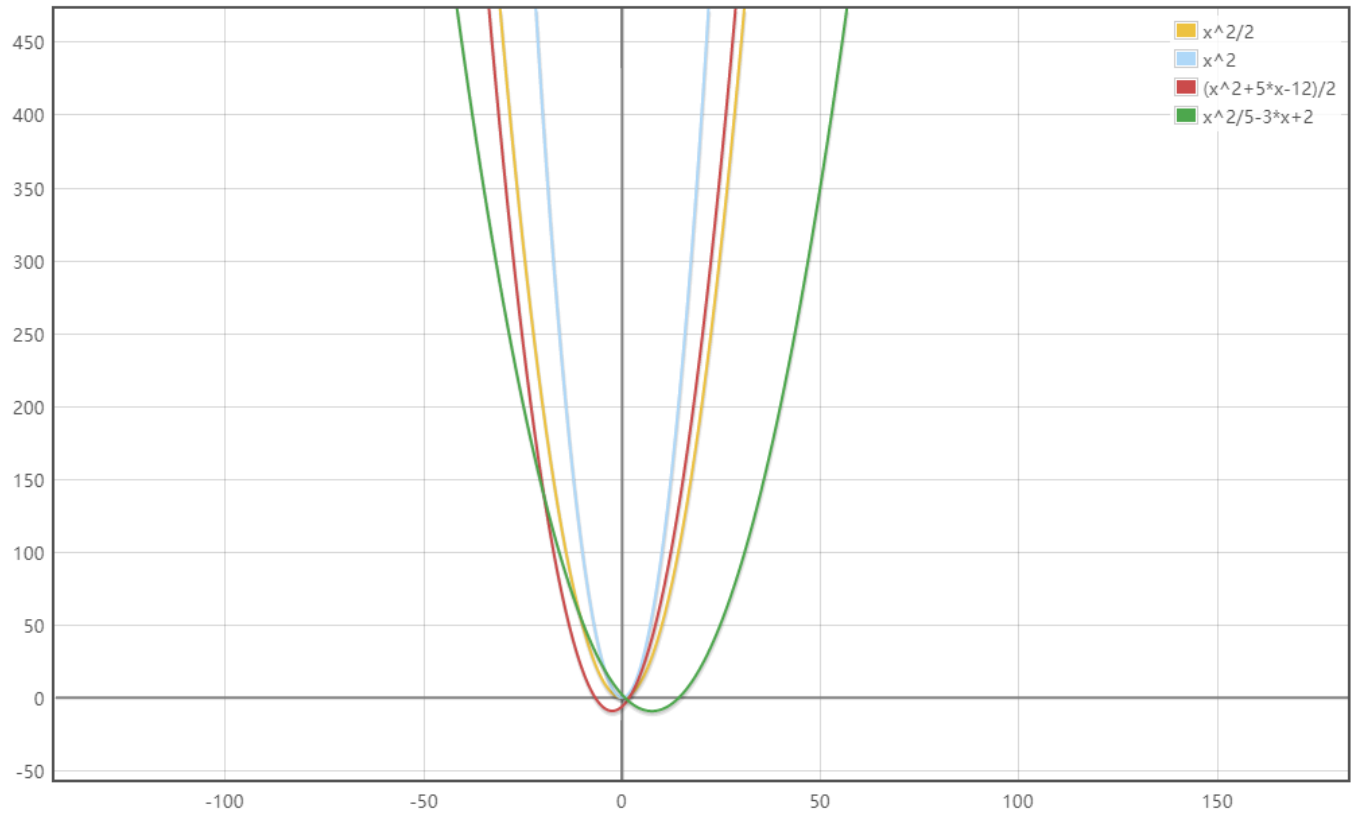
DATA STRUCTURE & ALGORITHM

- 算法的时间复杂度考虑的只是对于问题规模n的增长率，故只需求出关于n的阶即可

```
for (int i=2; i<=n; i++)
   for (int j=2; j<=i-1; j++)
       x++;
```

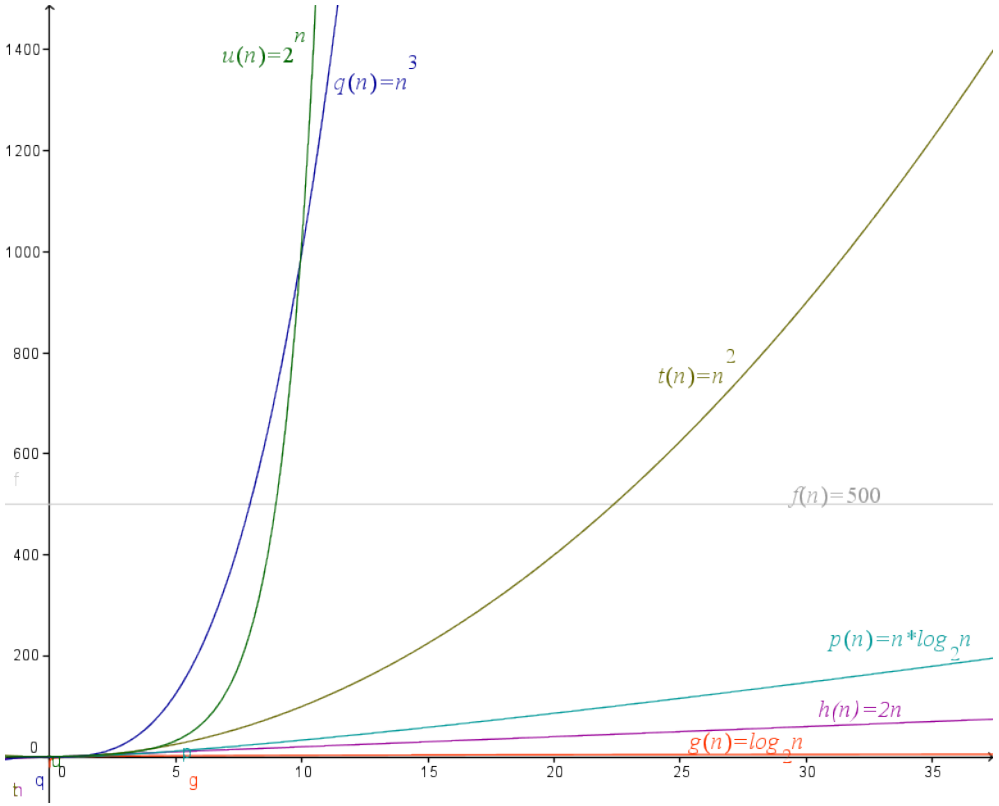
$$\frac{(n-1)(n-2)}{2}=\frac{n^2-3n+2}{2}$$

$$O(n^2)$$



2.4 不同阶的算法时间复杂度

DATA STRUCTURE & ALGORITHM



$$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3) < O(2^n)$$

2.4 不同阶的算法时间复杂度

DATA STRUCTURE & ALGORITHM

- 测评机器：Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
- 假设1秒种能进行10^9次任意循环

n	logn	sqrt(n)	nlogn	n^2	2^n	n!
5	2/-	2/-	10/-	25/-	32/-	120/-
10	3/-	3/-	30/-	100/-	1024/-	3628800
20	4/-	4/-	80/-	400/-	1048576/-	2.4×10^{18}
50	5/-	7/-	250/-	2500/-	10^{15}/27.8h	3.0×10^{64}
100	6/-	10/-	600/-	10000/-	10^{30}	9.3×10^{157}
1000	9/-	31/-	9000/-	1 000 000/-	10^{300}	4.0×10^{2567}
10 000	13/-	100/-	130 000/-	100 000 000/-	10^{3000}	10^{35660}
100 000	16/-	316/-	1 600 000/-	10^{10}	10^{30000}	10^{456574}
1 000 000	19/-	1000/-	19 000 000/-	10^{12}	10^{300000}	$10^{5565709}$

3.1 为什么要学习数据结构?

DATA STRUCTURE & ALGORITHM

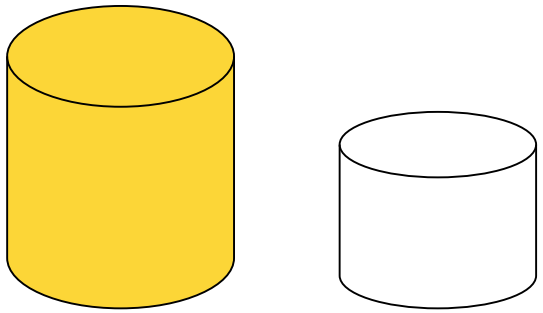
- 程序=算法+数据结构
  - 算法：处理问题的策略、步骤
  - 数据结构：问题中数据是如何存储的，数据之间的关系是怎样的，什么的关系会在使用数据的时候更方便。
- 例1：
  - Top 3问题，算法1
  - Top 3问题，算法2
  - Top 3问题，算法3

问题导引1-1

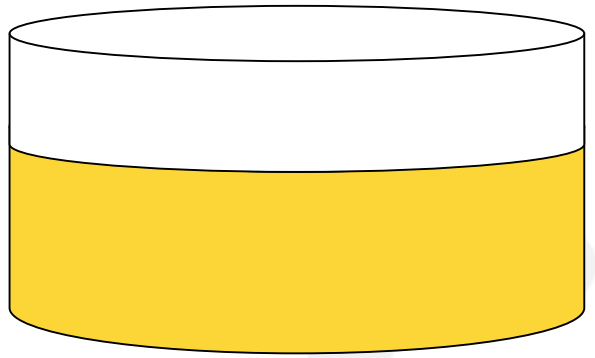
DATA STRUCTURE & ALGORITHM

- 倒水问题
- 有两个无刻度标志的水杯，分别可装x升和y升的水。设另一个水缸，可以用来向水杯灌水或从水杯向水缸里倒水，两个水杯之间也可以相互倒水。已知x升的水杯开始是盛满水的，y升的杯子是空的，问如何通过倒水和灌水操作，用最少的步数能在y升的杯子里量出z升水。

x=20, y=15, z=10



X　　Y



水缸（足够的容量足够的水）

例2:

DATA STRUCTURE & ALGORITHM

- 算法：广度优先搜索
- 数据结构：队列

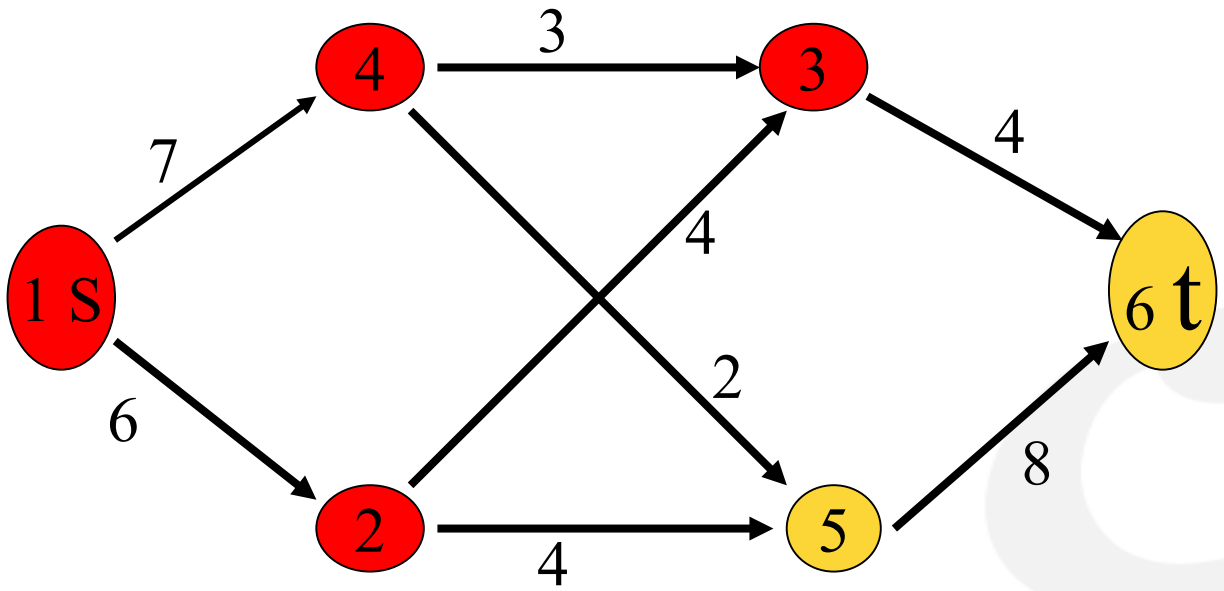
	X	Y
开始:	20	0
step 1:	5	15
step 2:	0	15
step 3:	15	0
step 4:	15	15
step 5:	20	10

例3：

DATA STRUCTURE & ALGORITHM

- 网络流问题
- 有一自来水管道输送系统，起点是S，目标是T，途中经过的管道都有一个最大的容量。
- 问从S到T的最大水流量是多少？

算法：网络流最大流算法
数据结构：图



3.2 什么是数据结构

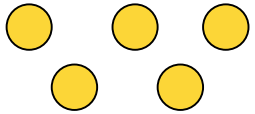
DATA STRUCTURE & ALGORITHM

- 数据结构是存储和组织数据的一种方式，以便对数据进行有效的访问和修改。
  - 数据与数据间的关系
  - 针对数据的操作：增、删、改、找
- 数据结构可以定义为一个二元组：
  - Data_Structure = (D, R)
    - $D = \{a_1, a_2, a_3, \ldots, a_n\}$
    - $R = \{<a_1, a_3>, <a_2, a_7>, <a_1, a_5>\}$

3.3 数据的逻辑结构元素间的逻辑关系

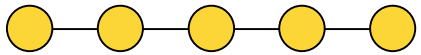
DATA STRUCTURE & ALGORITHM

- 离散结构：数据元素呈离散状，数据元素间无特殊关系。
  - 集合



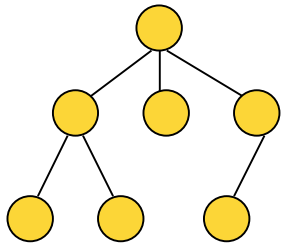
处理不重复数据

- 线性结构：每个数据元素有唯一的前驱或后继
  - 数组、字符串
  - 链表
  - 栈、队列

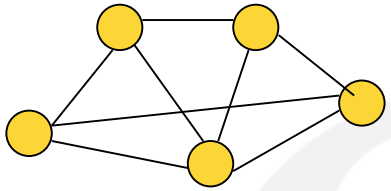


操作简单，编程复杂度低

- 非线性结构：
  - 树形结构：数据元素间存在一对多的关系
  - 图状结构：数据元素间存在多对多的关系



操作效率高，
广泛用于查找



结构复杂，
应用广泛，
有许多经典算法

3.4 数据的物理结构元素的存储方式

DATA STRUCTURE & ALGORITHM

- 顺序存储：
  - 借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系。逻辑上关联的数据元素，物理存储结构中相邻
- 链式存储：
  - 借助元素存储地址的指针（pointer）表示数据元素之间的逻辑关系 。逻辑上关联的数据元素，物理存储结构中不一定相邻。