渐变分析

asymptotic analysis

当n→∞时，函数f(n)的极限行为

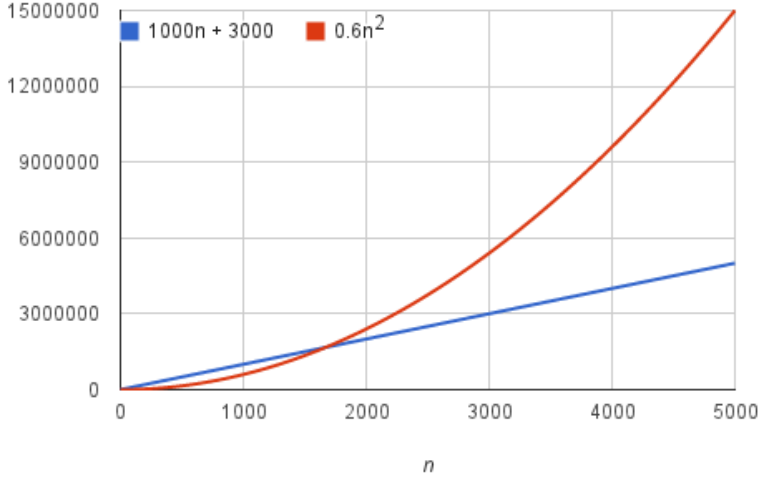
Youtube频道：hwdong

渐变分析

- 设$f(n) = n^2 + 3n$, 当n变得非常大时，与n^2 相比，3n变得微不足道，因此，当$n \to \infty$时，$f(n)$ 和n^2渐进相等。即 $f(n) \sim n^2$。
- 当$n \to \infty$时，

  1000n+3000和$0.6n^2$

  比较可以忽略不计。



渐变分析

- 通过去掉不那么重要的项和常数系数，我们可以专注于算法运行时间的重要部分，即增长率，而不会陷入使我们的理解更加复杂的细节中。
- 如$0.01n^2+700n$和n^2是同一个数量级，它们具有同样的增长率（同介无穷大量）。
- 当去掉常数系数和不太重要的项时，使用**渐变符号**表示增长的数量级。

渐变符号

- 上界O 、下界Ω 、紧界Θ 、非紧上界o、非紧下界ω。

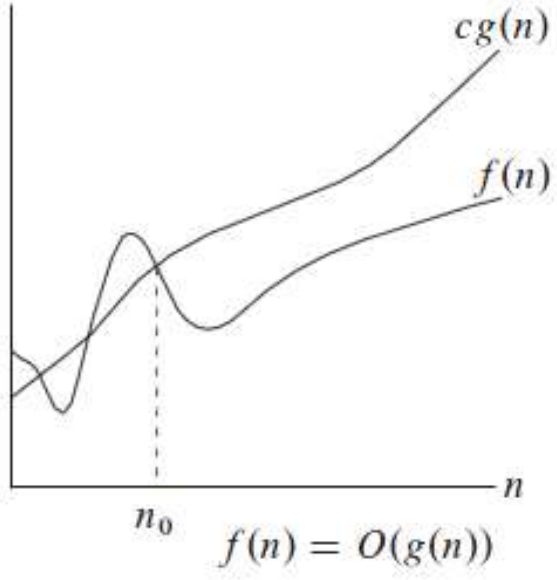
上界O

- 若存在正数c和n_0，使得对所有$n \geq n_0$有：

$$0 \leq f(n) \leq cg(n)\ ,$$

则称f(n)的渐进上界是g(n)，记作

$$f(n)=O(g(n))$$



上界O

- 如$f(n) = n^2+2n$,则:

$f(n)=O(n^2)$,可取$c = 2, n_0 = 2$

$f(n)=O(n^3)$,可取$c = 1, n_0 = 2$

$f(n)=O(0.1n^3)$,可取$c = 10, n_0 = 2$

- 如果$f(n)=O(g(n))$，说明$g(n)$的阶等于或高于$f(n)$.

上界的阶越低，则评估就越精确，就越有价值。

求证：所有度为k的多项式的上界是$O(n^k)$。

证明：设$T(n) = a_k n^k + \ldots + a_1 n + a_0 \quad \text{where } a_k \neq 0.$

Let $n_0 = 1$ and let $a^* = \max_i |a_i|$

$$\begin{aligned} T(n) &= a_k n^k + \ldots + a_1 n + a_0 \\ &\leq a^* n^k + \ldots + a^* n + a^* \\ &\leq a^* n^k + \ldots + a^* n^k + a^* n^k \\ &= (k+1) a^* \cdot n^k \end{aligned}$$

Let $c = (k+1)a^*$ which is a constant

练习

- 请证明：对函数$f(x) = 2x^2 - 5x + 1$, $f(x) = O(x^2)$
- 请证明：$For\ any\ k \geq 1,\ n^k\ is\ not\ O(n^{k-1})$.

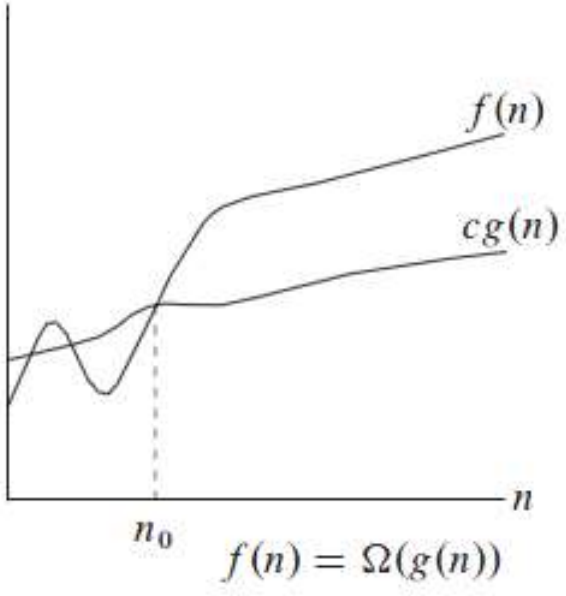
下界Ω

- 若存在正数c和n_0，使得对所有$n \geq n_0$有：

$$0 \leq cg(n) \leq f(n)\ ,$$

则称f(n)的渐进下界是g(n)，记作

$$f(n) = \Omega(g(n))$$



下界Ω

- 如$f(n) = n^2+2n$,则:

$f(n)= \Omega(n^2)$,可取$c = 1, n_0 = 1$

$f(n)= \Omega(100n)$,可取$c = 0.01, n_0 = 1$

$f(n)= \Omega(10n^2+5n)$,可取$c = 0.1, n_0 = 1$

- 如果$f(n)= \Omega(g(n))$，说明g(n)的介等于或低于f(n) .

下界的阶越高，则评估就越精确，就越有价值。

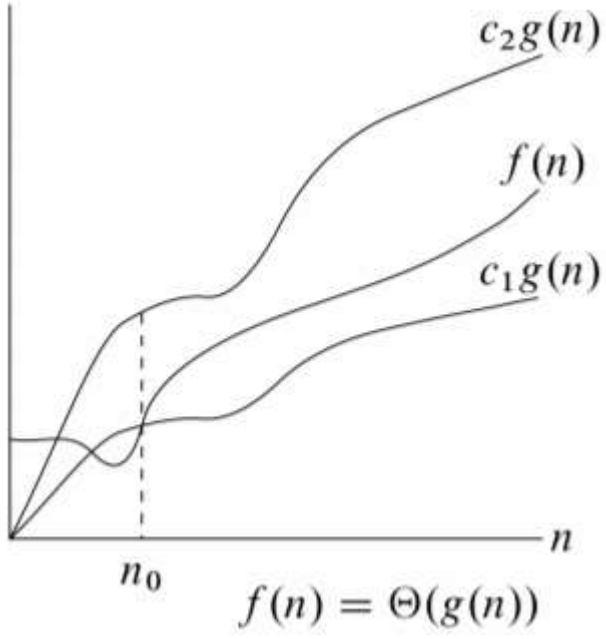
紧界Θ

- 若正数c_1,c_2和n_0使得对所有$n \geq n_0$有：

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \},$$

则称f(n)的紧界是g(n)，记作

$$f(n)= \Theta(g(n))$$



紧界Θ

- 如$f(n) = n^2+2n$, $g(n) = n^2$,则$f(n)= \Theta(n^2)$
- 如果$f(n)= \Theta(g(n))$，说明f(n)和g(n)是同阶无穷大量。
- 定理：$\lim_{n\to\infty} f(n)/g(n) =c$ $(c>0)$, 则$f(n)= \Theta(g(n))$

如：

$\lim_{n\to\infty} (n^2/3-4n)/(n^2) = 1/3$，因此$(n^2/3-4n) = \Theta(n^2)$

练习

- $f(n) = 0.1n^2+3n-7$和n,n^2,n^3的关系

非紧上界o

- 若对**任何**正数c>0，存在正数n_0 >0使得对所有n≥ n_0有：

$0 \le f(n)<cg(n)$ }

则称f(n)的非紧上界是g(n)，记作

$f(n)= o(g(n))$

- 等价于 $f(n) / g(n) \to 0$ ， as $n \to \infty$。

如： $\lim_{n\to\infty} (5n^2+9n)/(n^3) = 0$，因此$(5n^2+9n) = o(n^3)$

非紧上界o

- $\log n = o(n^c)$
- $n^c = o(a^n)$

非紧下界ω

- 若对**任何**正数c>0，存在正数n_0>0使得对所有$n \geq n_0$有：

$$0 \leq cg(n) < f(n) \}$$

则称f(n)的非紧下界是g(n)，记作

$$f(n) = \omega(g(n))$$

$f(n) = n^2$
$g(n) = 10n$

- 等价于 $f(n) / g(n) \to \infty$， as $n \to \infty$。

性质：

(1) 传递性

- 如果f=O(g),g= O(h)，则f= O(h)　　$f \le g, g \le h$, 则$f \le h$
- 如果f= Ω(g),g= Ω(h)，则f= Ω(h)　　$f \ge g, g \ge h$, 则$f \ge h$
- 如果f= Θ(g),g= Θ(h)，则f= Θ(h)
- 如果f= o(g),g= o(h)，则f= o(h)
- 如果f= ω(g),g= ω(h)，则f= ω(h)

（2）反身性：

- $f(n)= \Theta(f(n))$；
- $f(n)= O(f(n))$；
- $f(n)= \Omega(f(n))$.

（3）对称性：

- $f(n)= \Theta(g(n)) \Leftrightarrow g(n)= \Theta (f(n))$

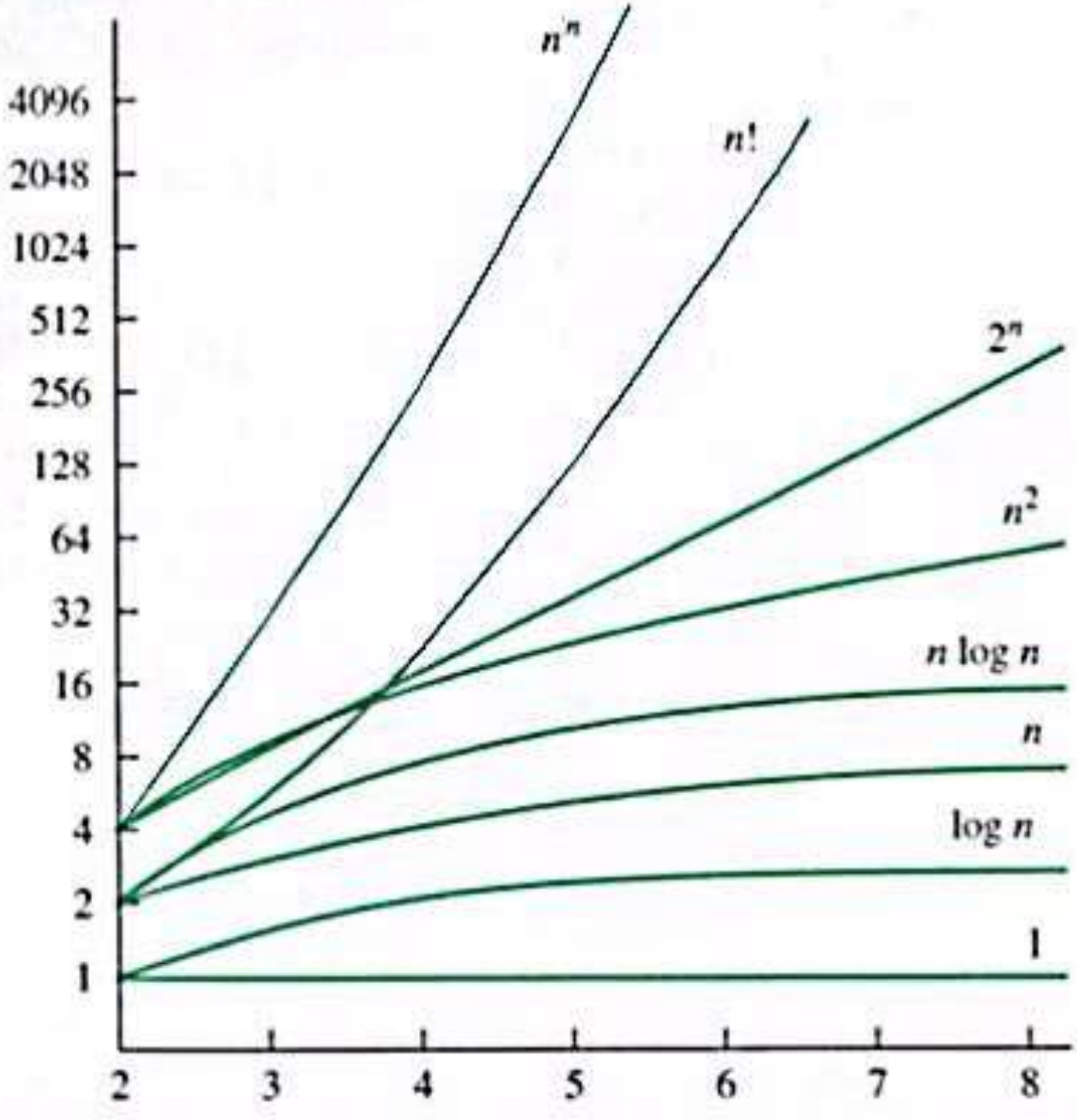
（4）互对称性：

- $f(n)= O(g(n)) \Leftrightarrow g(n)= \Omega (f(n))$ ；
- $f(n)= o(g(n)) \Leftrightarrow g(n)= \omega (f(n))$ ；

算法分析中常见的复杂性函数

FUNCTION	NAME	
c	Constant	常数
$\log N$	Logarithmic	对数
$\log^2 N$	Log-squared	
N	Linear	线性
$N \log N$	N log N	
N^2	Quadratic	二次
N^3	Cubic	三次
2^N	Exponential	指数

多项式


n^n
$n!$
2^n
n^2
$n \log n$
n
$\log n$
1
4096
2048
1024
512
256
128
64
32
16
8
4
2
1
2
3
4
5
6
7
8

- 哪种增长最能体现这些函数的特点？

	Constant	Linear	Polynomial	Exponential
$(3/2)^n$	○	○	○	○
1	○	○	○	○
$(3/2)n$	○	○	○	○
$2n^3$	○	○	○	○
2^n	○	○	○	○
$3n^2$	○	○	○	○
1000	○	○	○	○
$3n$	○	○	○	○

常见的时间复杂度

例子

Youtube频道：hwdong

线性时间linear time : O(n)

- 运行时间最多是输入规模的常数倍，如cn。
- 如：求一组数的(a_1,a_2,…a_n)最大值。

```
max = a_1
for i=2 to n:
    if a_i >max:
        max = a_i
return max
```

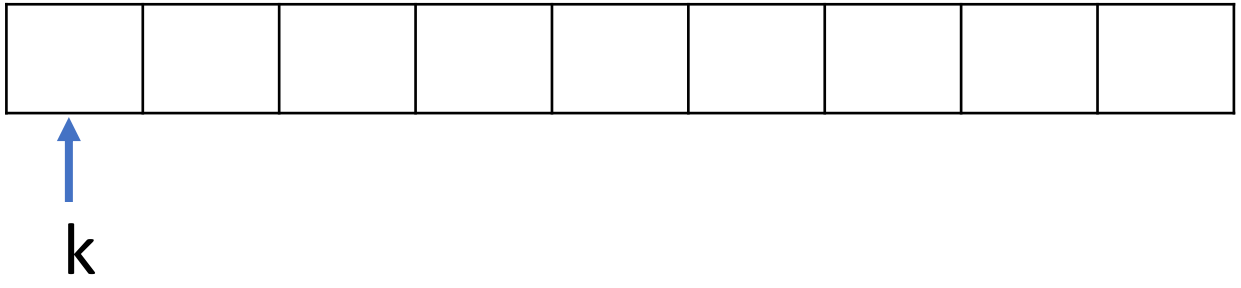
线性时间linear time : O(n)

- 2个有序数组的合并(merge)：

  $a = (a_1, a_2, \ldots a_m)$, $b = (b_1, b_2, \ldots b_n)$

如： a = (2,5,8,9) , b = (3,4,6,10,13)

c =



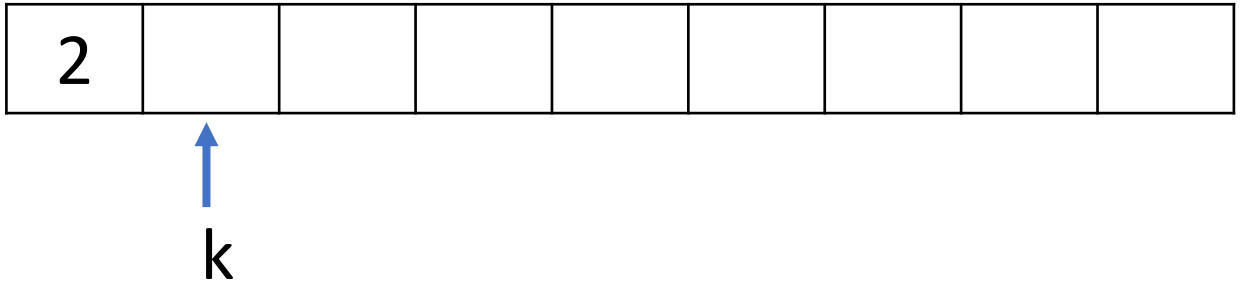
线性时间linear time : O(n)

- 2个有序数组的合并(merge)：

  $a = (a_1, a_2, \ldots a_m)$, $b = (b_1, b_2, \ldots b_n)$

如： a = (2,5,8,9) , b = (3,4,6,10,13)

c =

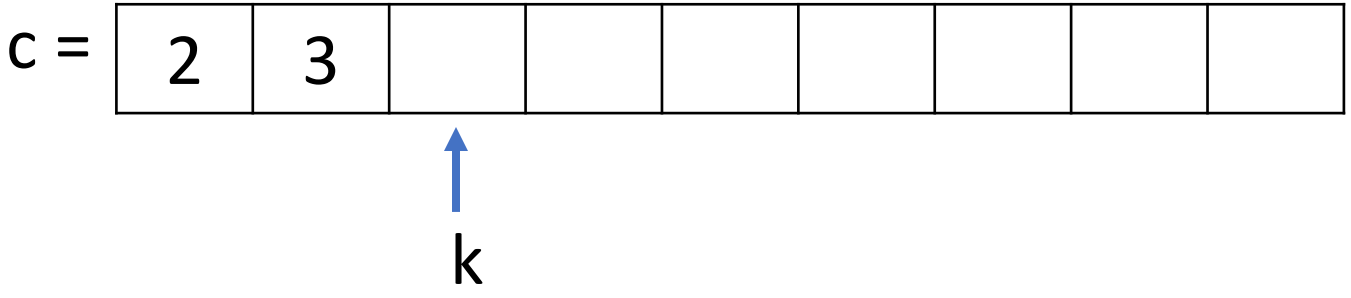


线性时间linear time : O(n)

- 2个有序数组的合并(merge)：

  $a = (a_1, a_2, \ldots a_m)$, $b = (b_1, b_2, \ldots b_n)$

如： a = (2,5,8,9) , b = (3,4,6,10,13)



线性时间linear time : O(n)

- 2个有序数组的合并(merge)：

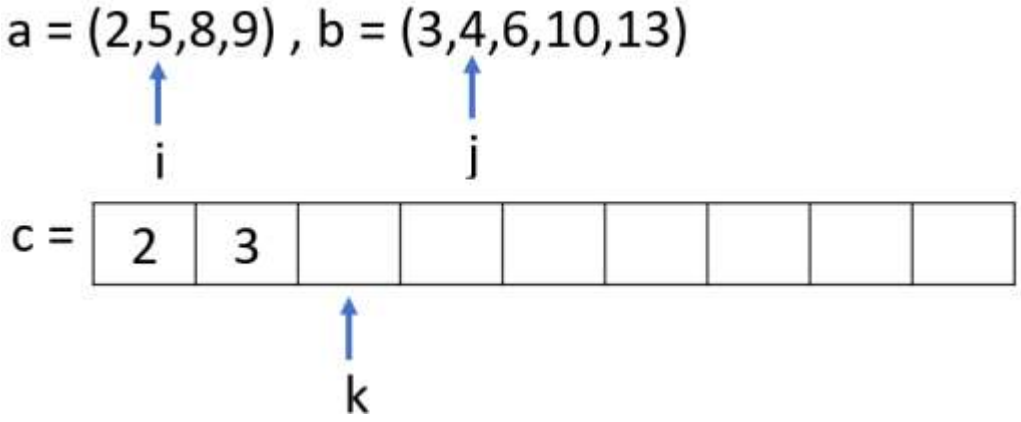
  $a = (a_1, a_2, \ldots a_m)$, $b = (b_1, b_2, \ldots b_n)$

如： a = (2,5,8,9) , b = (3,4,6,10,13)

c =

2	3	4						

merge(a,b)



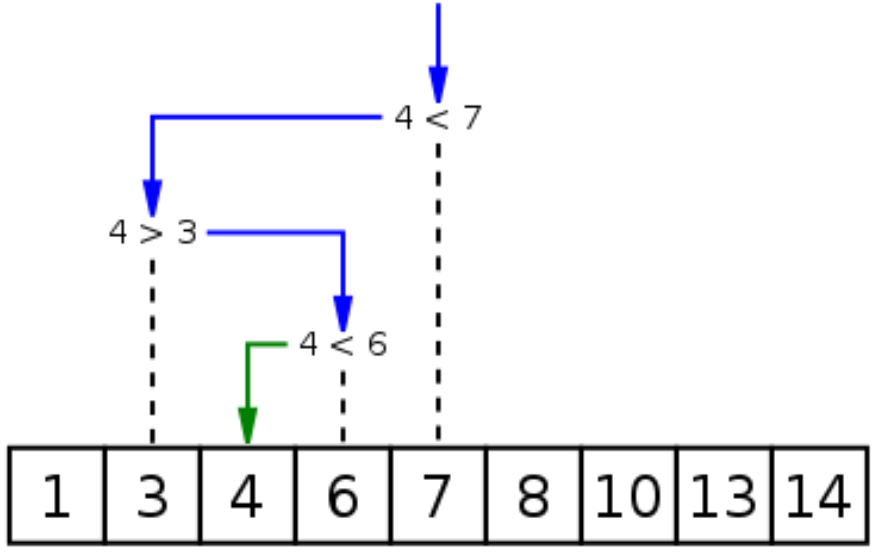

```
i=1,j=1
while i ≤m and j ≤n:
    if a_i ≤ b_j :
        c_k = a_i ,i++,k++
    else:
        c_k = b_j ,j++,k++
while i ≤m:
    c_k = a_i ,i++,k++
while j ≤n:
    c_k = b_j ,j++,k++
```

基本操作

O(m+n)

O(logn)

- 有序数组的二分查找

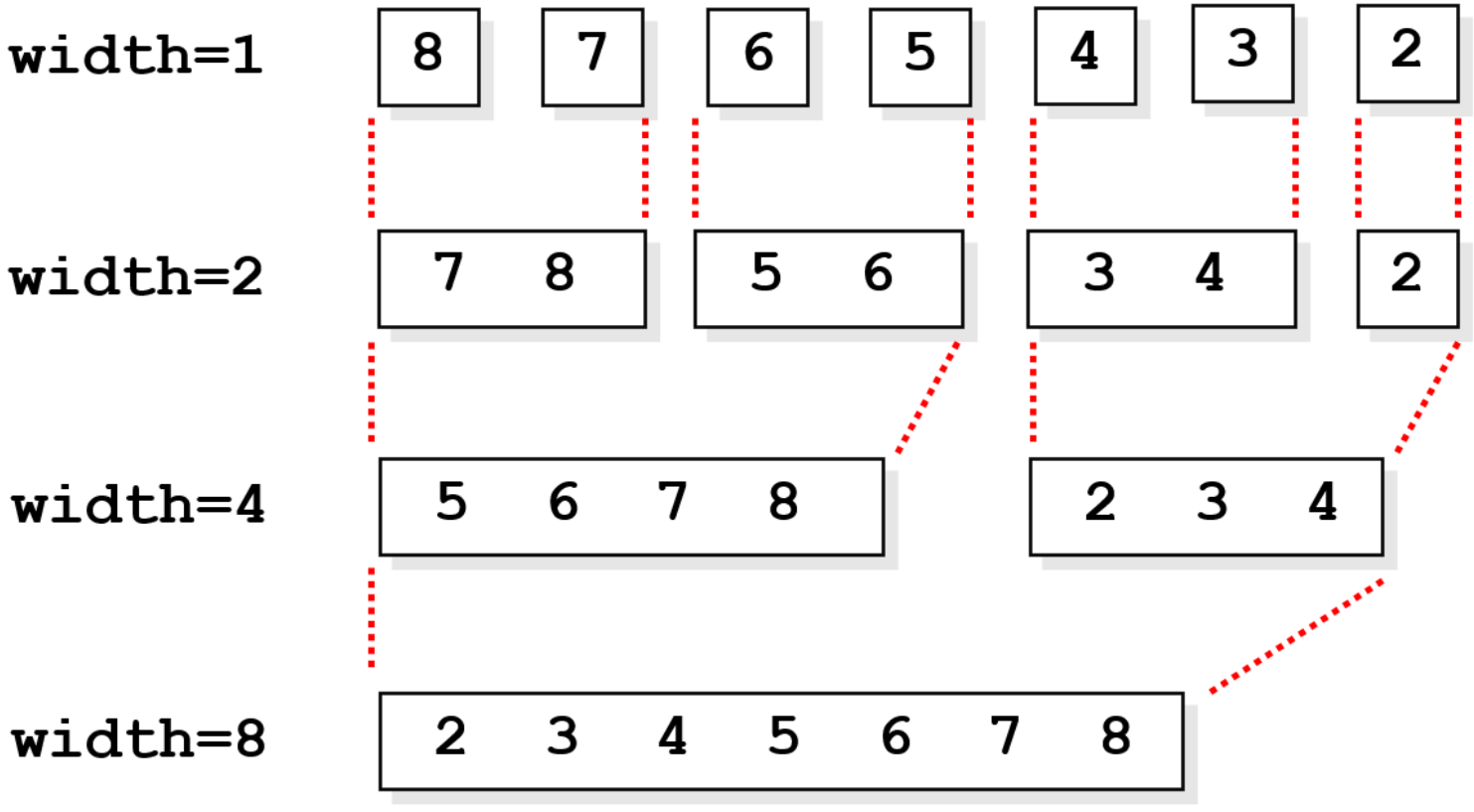


复杂度	
平均时间复杂度	$O(\log n)$
最坏时间复杂度	$O(\log n)$
最优时间复杂度	$O(1)$
空间复杂度	迭代：$O(1)$ 递归：$O(\log n)$

O(nlogn)

$2^k \geq n \Rightarrow k \geq \log n$

- 归并排序、快速排序



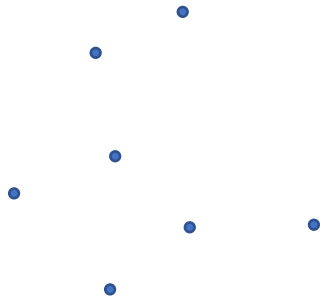
Quadratic: $O(n^2)$

- 插入排序

$$\sum_{i=2}^{n} i = (n+2)(n-1)/2$$

Quadratic: $O(n^2)$

- 平面点集的最短点对



```
min = dist(p1,p2)
for each point p:
        for each point q:
            d= dist(p,q)
             if d < dist:
                  dist  = d
return min
```

Quadratic: $O(n^3)$

- 矩阵的乘积：如2个n阶方阵相乘

```
for i=1 to n:
    for j=1 to n:
        cij = 0
        for k=1 to n:
            cij = cij+ aik*bkj
```

exponential: $O(a^n)$

- Hanoi汉诺塔问题： $O(2^n)$

```
Hanoi(n,A,B,C)
    if n==1:        move(n, A,C)
    Hanoi(n-1,A,C,B)
    move(n, A,C)
    Hanoi(n-1,B,C,A)
```

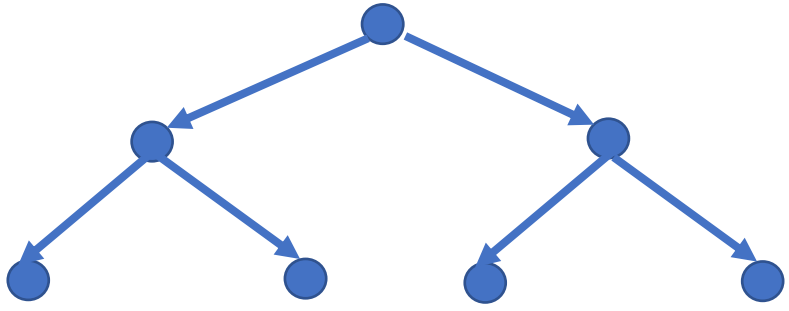
$T(n) = 2T(n-1)+1$

$T(1) = 1$

$T(n) = 2^n-1$

exponential: $O(a^n)$

- 背包问题(Knapsack problem)：给定一组物品，每种物品都有自己的重量和价格，在限定的总重量内，我们如何选择，才能使得物品的总价格最高。
- 穷举法： 每个物体选或不选，一共有2^n。

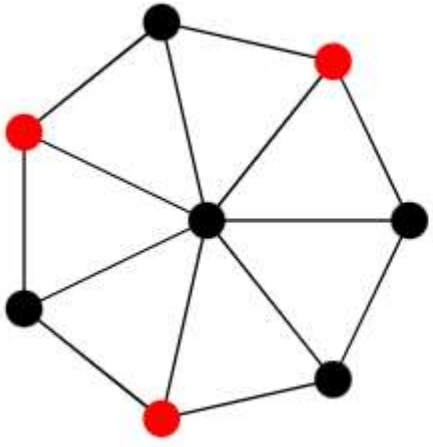
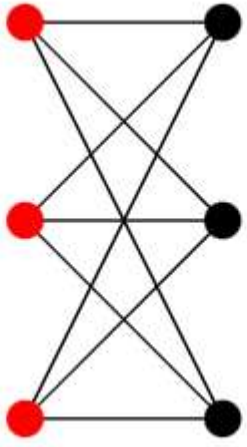
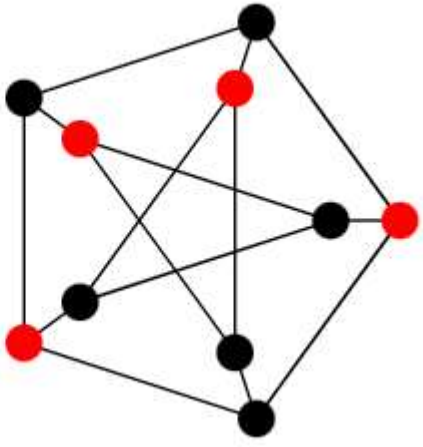
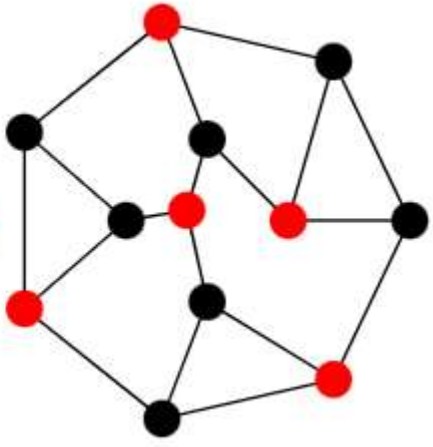


n个物品的背包问题:

```
int knapSack(int W, int w[], int v[],int n):
    if n==0:
       return 0
    if w_n>W:
       return  knapsack(W,w,v,n-1)
    return
        max( v_n +knapsack(W-w_n,w,v,n-1),
             knapsack(W,w,v,n-1),
          )
```

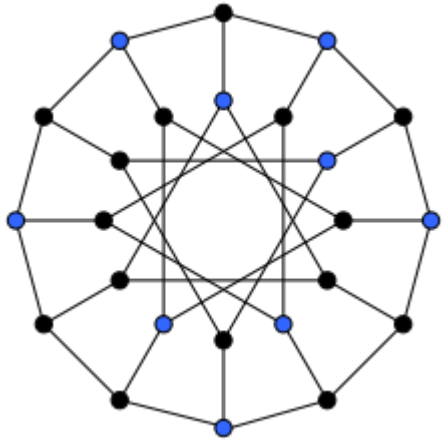
exponential: $O(a^n)$

- 独立集是指图G 中两两互不相邻的顶点构成的集合 。
- 图 G 的独立集是顶点的子集，该子集中没有两个顶点表示 G 的边。

exponential: $O(a^n)$

- 独立集是指图G 中两两互不相邻的顶点构成的集合。
- 问题：求图的最大独立集



图的子集有2^n个，

检查一个子集是否有边相连n^2。

$O(n^2\,2^n)$

常见函数

1）取整函数

⌊ *x* ⌋：不大于*x*的最大整数;

⌈ *x* ⌉：不小于*x*的最小整数。

2）指数函数

对于正整数m,n和实数a>0:

$a^0=1$;

$a^1=a$;

$a^{-1}=1/a$;

$(a^m)^n = a^{mn}$;

$(a^m)^n = (a^n)^m$;

$a^m a^n = a^{m+n}$;

$a>1 \Rightarrow$ $\Rightarrow n^b = o(a^n)$, 即a^n是n^b的非紧上界

3）对数函数

$\log n = \log_2 n$;

$\lg n = \log_{10} n$;

$\ln n = \log_e n$;

$\log^k n = (\log n)^k$;

$\log \log n = \log(\log n)$;

for a>0,b>0,c>0

$$a = b^{\log_b a}$$

$$\log_c(ab) = \log_c a + \log_c b$$

$$\log_b a^n = n \log_b a$$

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$\log_b (1/a) = -\log_b a$$

$$\log_b a = \frac{1}{\log_a b}$$

$$a^{\log_b c} = c^{\log_b a} \qquad a^{\log_b n} = n^{\log_b a}$$

P vs NP

- 能在多项式时间(polynomial time)内求解的一类问题称为 “类**P** ”或“P问题“,简称 "P"。
- 虽然不知道多项式时间的解，但一旦给了一个解，可以在多项式时间内验证这个解的一类问题称为**NP**，NP是 “非确定多项式时间 ”(“nondeterministic polynomial time“)的缩写。如：寻找一个大数的质因数。

旅行家推销问题(TSP):给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路

- 显然，$P \subseteq NP$，
- 但P=NP？这个问题的意思是："如果一个问题的解可以在多项式时间内验证，那么能否在多项式时间内找到它？"
- 克雷(Clay)数学研究所悬赏百万美元的7个千禧年大奖难题之一 （2000年5月24日）

庞加莱猜想

在拓扑学，二维球面是紧致且单连通。通俗地说，意味球面不会无限延伸，并且其上任何闭合的圈都可收紧至一点。

庞加莱猜想

庞加莱猜想考虑的是**更高维**的情况：
若闭合三维空间中每条闭曲线都可连续收缩到一点，那么拓扑地看，这空间是否就是球？

它的数学陈述为：**一个单连通三维闭流形同胚于三维球面**

这猜想是三维流形的分类问题的核心

庞加莱猜想

1962年，**斯蒂芬·斯梅尔**证明了庞加莱猜想在五维以上的等价结论，四维的情况则在二十年后由**迈克尔·弗里德曼**证明，但数学界始终对三维流形束手无策，而人类所处的宇宙是三维流形，更显出问题重要

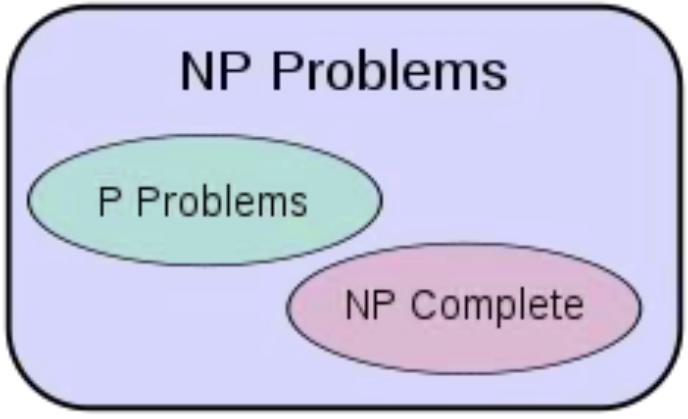
- 2003年，俄罗斯数学家**格里戈里·佩雷尔曼**在arXiv贴出了完整证明

- 2006年，多组研究者先后发表论文阐释了佩雷尔曼的成果，并认定其无误。由于这一贡献，国际数学家大会决定授予佩雷尔曼菲尔兹奖，但他本人却拒绝领奖

- 2010年3月18日，千禧年大奖正式颁发给佩雷尔曼，但他又一次拒绝领奖，也包括克雷数学研究所的百万奖金

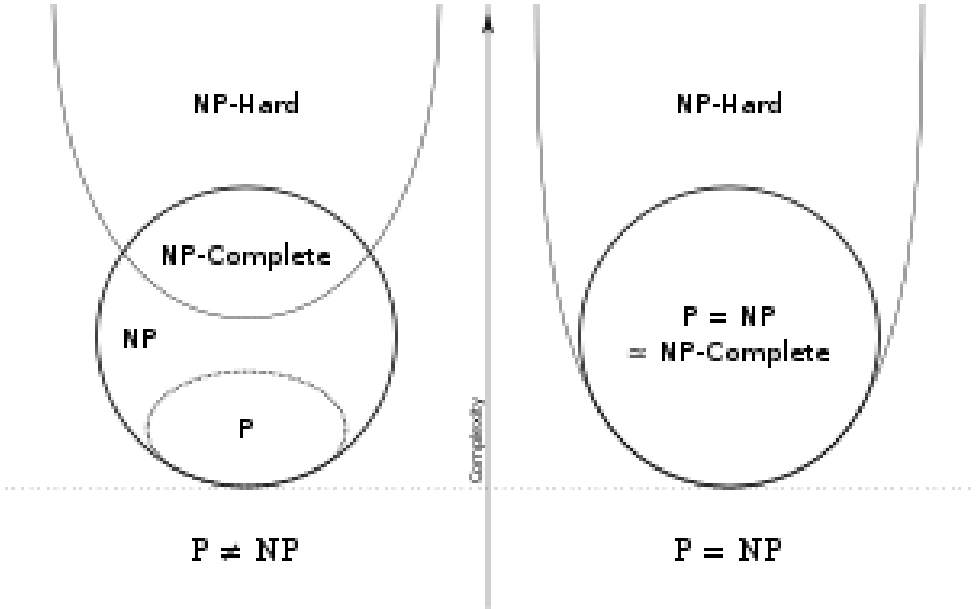
NP-Complete（NP 完备性）

- 为了攻克P=NP问题，NP-完备性的概念非常有用。NP-完备性问题是一组问题，其他的NP问题都可以在多项式时间内还原为该组问题的一个问题，而且其解仍然可以在多项式时间内得到验证。



NP-Hard（NP难问题）

- NP-hard问题是指那些至少和NP问题一样难的问题，即任何NP问题都可以（在多项式时间内）转化成它们。NP-hard问题不一定是NP问题，即它们的解不一定可以在多项式时间内验证。



关注我

博客： hwdong-net.github.io

Youtube频道