Reviews

from chap1 to chap4

songyou@buaa.edu.cn

Reviews from chap1 to chap4

- Framework for describing algorithms
- Correctness of algorithms
- Efficiency of algorithms

- Divide and Conquer
- Asymptotic analysis of function（渐近分析）
- Recurrences

Reviews from chap1 to chap4

Examples:

- Merge sort
- Multiplication of two integers
- Multiplication of two matrices
- Finding Minimum and Maximum
- Majority problem （多数问题）
- Branch growth（树枝生长）

Exam1 Merge Sort

.....

Exam2 Multiplication of two integers（整数相乘）

Multiplication of two integers （整数相乘问题）

$$\begin{array}{r} 31415962 \\ \times 27182818 \\ \hline ? \end{array}$$

X 和 Y 是两个 n 位的十进制整数，分别表示为

$X = x_{n-1}x_{n-2}\cdots x_0$，$Y = y_{n-1}y_{n-2}\cdots y_0$，其中 $0 \le x_i, y_j \le 9$ $(i, j = 0, 1, \ldots, n\text{-}1)$，设计一个算法求 $X \times Y$，并分析其计算复杂度。说明：算法中“基本操作”约定为两个个位整数相乘 $x_i \times y_j$、两个整数相加、除以10、等等；这里的输入规模 n 表示输入数据的大小（位长），而不是输入数据的个数。

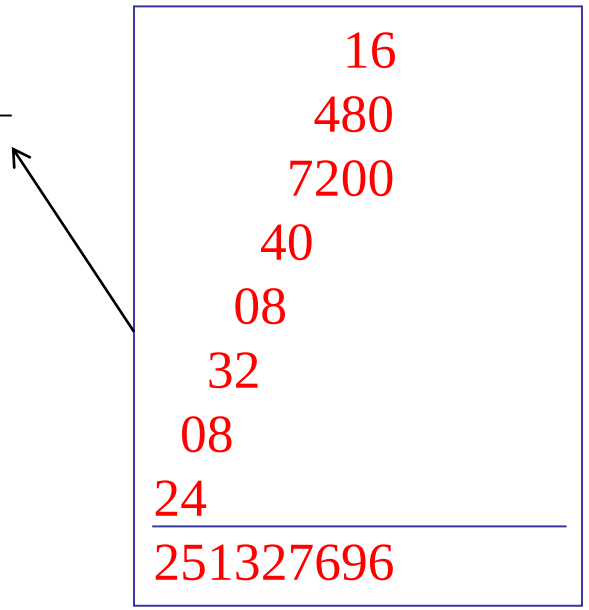
Exam2 Multiplication of two integers（整数相乘）

two n-digit numbers X and Y, Complexity($X \times Y$) = ?

(1) Naive (原始的) pencil-and-paper algorithm

```
   12
 ×23
----
    6
   3
----
   4
  2
----
  276
```

```
   31415962
 ×27182818
----------
  251327696
   31415962
  251327696
   62831924
 251327696
   31415962
 219911734
   62831924
----------
 853974377340916
```



Complexity analysis: n^2 multiplications and at most n^2-1 additions (加法).

Exam2 Multiplication of two integers（整数相乘）

two n-digit numbers X and Y, Complexity($X \times Y$) = ?

(2) Divide and Conquer algorithm

Let $X = ab$

$Y = cd$

where a, b, c and d are $n/2$ digit numbers, e.g. $1364 = 13\times10^2 + 64$.

Let $m = n/2$. Then

$$XY = (10^m a+b)(10^m c+d)$$
$$= 10^{2m}ac+10^m(bc+ad)+bd$$

Exam2 Multiplication of two integers（整数相乘）

two n-digit numbers X and Y, Complexity($X \times Y$) = ?

(2) Divide and Conquer algorithm

Let $X = ab$, $Y = cd$

then $XY = (10^m a+b)(10^m c+d) = 10^{2m}ac+10^m(bc+ad)+bd$

Multiply(X; Y; *n*)
  if $n = 1$
    return X×Y
  else
    $m = \lceil n/2 \rceil$
    $a = \lfloor X/10^m \rfloor$, $b = X \bmod 10^m$
    $c = \lfloor Y/10^m \rfloor$, $d = Y \bmod 10^m$
    e = **Multiply(*a*; *c*; *m*)**
    f = **Multiply(*b*; *d*; *m*)**
    g = **Multiply(*b*; *c*; *m*)**
    h = **Multiply(*a*; *d*; *m*)**
    return $10^{2m}e + 10^m(g + h) + f$

Complexity analysis:

$T(1) = 1$,
$T(n) = 4T(\lceil n/2 \rceil) + O(n)$.
Applying Master Theorem,

$T(n)$ = ?

Exam2 Multiplication of two integers（整数相乘）

two n-digit numbers X and Y, Complexity($X \times Y$) = ?

(3) Divide and Conquer (Karatsuba's algorithm) 卡拉茨巴算法

Let $X = ab$, $Y = cd$

then $XY = (10^m a+b)(10^m c+d) = 10^{2m}ac+10^m(bc+ad)+bd$

Note that $bc + ad = ac + bd - (a - b)(c - d)$. So, we have

FastMultiply(X; Y; *n*)
  if $n = 1$
    return X×Y
  else
    $m = \lceil n/2 \rceil$
    $a = \lfloor X/10^m \rfloor$, $b = X \bmod 10^m$
    $c = \lfloor Y/10^m \rfloor$, $d = Y \bmod 10^m$
    e = **FastMultiply(*a*; *c*; *m*)**
    f = **FastMultiply(*b*; *d*; *m*)**
    g = **FastMultiply(*a*–*b*; *c*–*d*; *m*)**
    return $10^{2m}e + 10^m(e + f - g) + f$

Complexity analysis:

$T(1) = 1$,
$T(n) = 3T(\lceil n/2 \rceil) + O(n)$.
Applying Master Theorem,

$$T(n) = ?$$

Exam2 Multiplication of two integers（整数相乘）

two n-digit numbers X and Y, Complexity($X \times Y$) = ?

Divide and Conquer (Karatsuba's algorithm)

```
FastMultiply(X; Y; n)
    if n = 1
        return X×Y
    else
        m = ⌈n/2⌉
        a = ⌊X/10^m⌋, b = X mod 10^m
        c = ⌊Y/10^m⌋, d = Y mod 10^m
        e = FastMultiply(a; c; m)
        f = FastMultiply(b; d; m)
        g = FastMultiply(a–b; c–d; m)
        return 10^(2m)e + 10^m(e + f – g ) + f
```

$$T(n) = ?$$

1960 年，一位名叫 Anatoly Karatsuba（安德烈·阿列克谢耶维奇·卡拉茨巴）的 23 岁的俄罗斯数学家发明，于1962年发表。

A. Karatsuba, Yu. Ofman, "Multiplication of many-digital numbers by automatic computers", Doklady Akademii Nauk SSSR, 1962, Volume 145, Number 2, Pages 293–294.

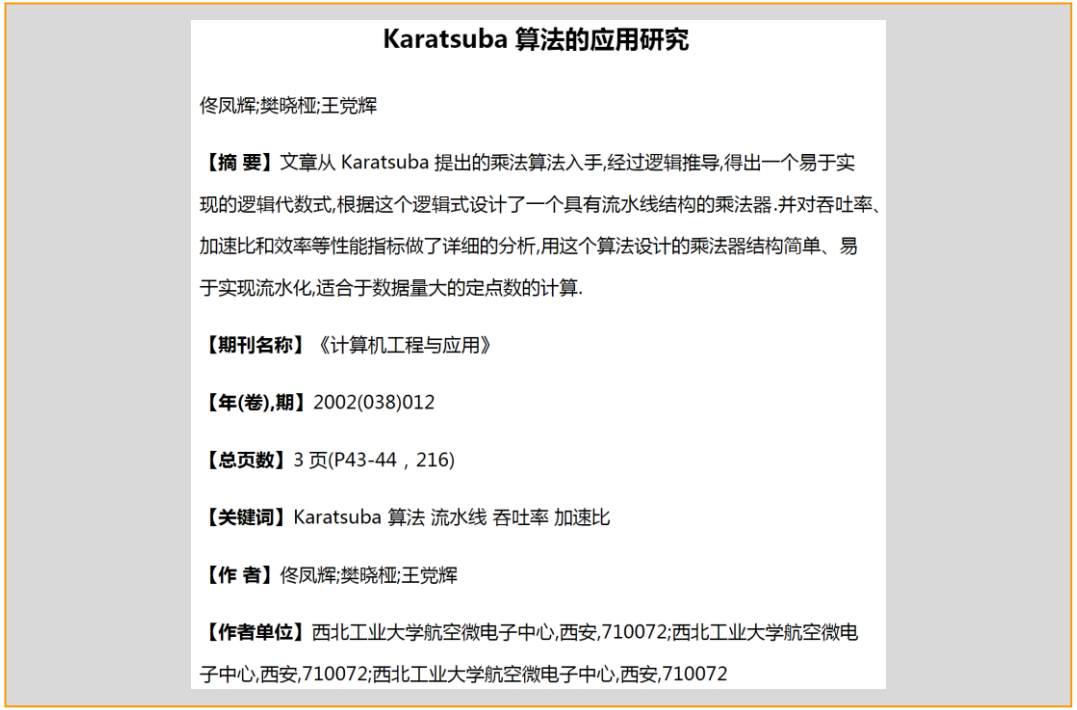
n 很大时（如10000，或更大），可用 FFT 或中国余数定理

Exam2 Multiplication of two integers（整数相乘）

(3) Divide and Conquer (Karatsuba's algorithm)

学算法，不满足于刷题（虽然有的同学AC能力很强，有的同学WA的很痛苦），读原著，算法研究与应用，也很有趣。希望同学们可以思考算法问题，尝试写点小论文，如 →

A. Karatsuba, Yu. Ofman, "Multiplication of many-digital numbers by automatic computers", Doklady Akademii Nauk SSSR, 1962, Volume 145, Number 2, Pages 293–294.



Karatsuba 算法的应用研究

佟凤辉;樊晓桠;王党辉

【摘 要】文章从 Karatsuba 提出的乘法算法入手,经过逻辑推导,得出一个易于实现的逻辑代数式,根据这个逻辑式设计了一个具有流水线结构的乘法器.并对吞吐率、加速比和效率等性能指标做了详细的分析,用这个算法设计的乘法器结构简单、易于实现流水化,适合于数据量大的定点数的计算.

【期刊名称】《计算机工程与应用》

【年(卷),期】2002(038)012

【总页数】3 页(P43-44，216)

【关键词】Karatsuba 算法 流水线 吞吐率 加速比

【作 者】佟凤辉;樊晓桠;王党辉

【作者单位】西北工业大学航空微电子中心,西安,710072;西北工业大学航空微电子中心,西安,710072;西北工业大学航空微电子中心,西安,710072

Exam3 Multiplication of two matrices（矩阵相乘）

Multiplication of two matrices （矩阵相乘问题）

A 和 **B** 是两个 n 阶实方阵，表示为 $\mathbf{A}=\begin{pmatrix} a_{11}\ldots a_{1n} \\ \ldots\ldots\ldots \\ a_{n1}\ldots a_{nn} \end{pmatrix}$ ， $\mathbf{B}=\begin{pmatrix} b_{11}\ldots b_{1n} \\ \ldots\ldots\ldots \\ b_{n1}\ldots b_{nn} \end{pmatrix}$

设计一个算法求 **A**×**B**，并分析计算复杂度。

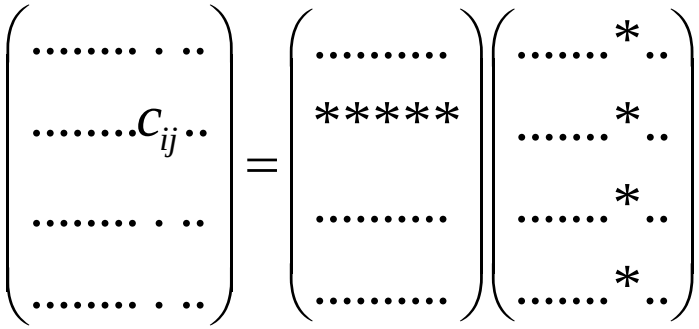
说明：算法中“基本操作”约定为两个实数相乘，或两个实数相加。

Exam3 Multiplication of two matrices（矩阵相乘）

two $n \times n$ matrices **A** and **B**, Complexity(**C** = **A**×**B**) = ?

(1) Standard method

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$



MATRIX-MULTIPLY(A, B)

```
for i ← 1 to n
    for j ← 1 to n
        C[i, j] ← 0
        for k ← 1 to n
            C[i, j] ← C[i, j]+A[i, k]·B[k, j]
return C
```

Complexity:

$O(n^3)$ multiplications and additions.

$T(n) = O(n^3)$.

Exam3 Multiplication of two matrices（矩阵相乘）

two $n \times n$ matrices **A** and **B**, Complexity(**C** = **A**×**B**) = ?

(2) Divide and conquer

An $n \times n$ matrix can be divided into four $n/2 \times n/2$ matrices,

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$\mathbf{C}_{11} = \mathbf{A}_{11}\mathbf{B}_{11}+\mathbf{A}_{12}\mathbf{B}_{21}$, $\mathbf{C}_{12} = \mathbf{A}_{11}\mathbf{B}_{12}+\mathbf{A}_{12}\mathbf{B}_{22}$

$\mathbf{C}_{21} = \mathbf{A}_{21}\mathbf{B}_{11}+\mathbf{A}_{22}\mathbf{B}_{21}$, $\mathbf{C}_{22} = \mathbf{A}_{21}\mathbf{B}_{12}+\mathbf{A}_{22}\mathbf{B}_{22}$

Complexity analysis:

Totally, 8 multiplications (subproblems), and 4 additions ($n/2 \times n/2 \times 4$).

$T(1) = 1$, $T(n) = 8T(\lceil n/2 \rceil)+n^2$.

Applying Master Theorem, we have $T(n) = O(n^3)$.

Exam3 Multiplication of two matrices（矩阵相乘）

two $n \times n$ matrices **A** and **B**, Complexity(**C** = **A**×**B**) = ?

(3) Divide and conquer (Strassen Algorithm, 斯特拉森)

Volker Strassen. Gaussian elimination is not optimal. Numerische Mathematik, 14(3):354–356, 1969.

[引用] Gaussian elimination is not optimal

V Strassen - Numerische mathematik, 1969 - Springer

… This fact should be compared with the result of KLYUYEV and KOKOVKINSHCHERBAK [1] that Gaussian elimination for solving a system of linearequations is optimal if one restricts …

☆ 保存 99 引用 被引用次数：3774 相关文章 所有 8 个版本

Exam3 Multiplication of two matrices（矩阵相乘）

two $n \times n$ matrices **A** and **B**, Complexity(**C** = **A**×**B**) = ?

(3) Divide and conquer (Strassen Algorithm, 斯特拉森)

An $n \times n$ matrix can be divided into four $n/2 \times n/2$ matrices,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix}$$

Define
$\mathbf{P}_1 = (\mathbf{A}_{11}+\mathbf{A}_{22})(\mathbf{B}_{11}+\mathbf{B}_{22})$
$\mathbf{P}_2 = (\mathbf{A}_{11}+\mathbf{A}_{22})\mathbf{B}_{11}$
$\mathbf{P}_3 = \mathbf{A}_{11}(\mathbf{B}_{11}-\mathbf{B}_{22})$
$\mathbf{P}_4 = \mathbf{A}_{22}(-\mathbf{B}_{11}+\mathbf{B}_{22})$
$\mathbf{P}_5 = (\mathbf{A}_{11}+\mathbf{A}_{12})\mathbf{B}_{22}$
$\mathbf{P}_6 = (-\mathbf{A}_{11}+\mathbf{A}_{21})(\mathbf{B}_{11}+\mathbf{B}_{12})$
$\mathbf{P}_7 = (\mathbf{A}_{12}-\mathbf{A}_{22})(\mathbf{B}_{21}+\mathbf{B}_{22})$

Then
$\mathbf{C}_{11} = \mathbf{P}_1+\mathbf{P}_4-\mathbf{P}_5+\mathbf{P}_7$, $\mathbf{C}_{12} = \mathbf{P}_3+\mathbf{P}_5$
$\mathbf{C}_{21} = \mathbf{P}_2+\mathbf{P}_4$, $\mathbf{C}_{22} = \mathbf{P}_1+\mathbf{P}_3-\mathbf{P}_2+\mathbf{P}_6$

Complexity analysis:
Totally, 7 multiplications,
And 18 additions.
$T(1) = 1$,
$T(n) = 7T(\lceil n/2 \rceil) + cn^2$.
Applying Master Theorem,

$$T(n) = ?$$

Exam3 Multiplication of two matrices（矩阵相乘）

two $n \times n$ matrices **A** and **B**, Complexity(**C** = **A**×**B**) = ?

Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions.

$T(n) = O(n^{2.376})$

Matrix multiplication via arithmetic progressions

D Coppersmith, S Winograd

Proceedings of the nineteenth annual ACM symposium on Theory of computing, 1987 • dl.acm.org

> We present a new method for accelerating matrix multiplication asymptotically. This work builds on recent ideas of Volker Strassen, by using a basic trilinear form which is not a matrix product. We make novel use of the Salem-Spencer Theorem, which gives a fairly dense set of integers with no three-term arithmetic progression. Our resulting matrix exponent is 2.376.
>
> ACM Digital Library

☆ 保存 引用 被引用次数：3995 相关文章 所有 11 个版本

作业：读原著。实现算法。出一道上机题？用作某一次考试 ($n = 1000$)？

$$O(n^3) \Rightarrow O(n^{2.807})$$

$$\Rightarrow O(n^{2.376}) \Rightarrow ?$$

启示：求解一类问题的某个算法被提出，如果未被证明是最优的，通常还有更优的算法存在（等着你去发现）。

Exam4 Finding Minimum and Maximum（MaxMin）

Find the lightest and heaviest of *n* elements using a balance that allows you to compare the weight of 2 elements.（对于一个具有 *n* 个元素的数组，用一个天平，通过比较 2个元素的重量，求出最轻和最重的一个）

Goal

Minimize the number of comparisons.（最少的比较次数？）

Exam4 Finding Minimum and Maximum（MaxMin）

Max element: Find element with max weight (重量) from $w[0, n-1]$

```
maxElement = 0;
for (int i = 1; i < n; i++)
    if (w[maxElement] < w[i])
        maxElement = i;
```

Number of comparisons (比较次数) is $n-1$.

(1) Obvious method（直接法）

- Find the max of n elements making $n-1$ comparisons.
- Find the min of the remaining $n-1$ elements making $n-2$ comparisons.
- Total number of comparisons is $2n-3$.

Exam4 Finding Minimum and Maximum（MaxMin）

(2) Divide and conquer

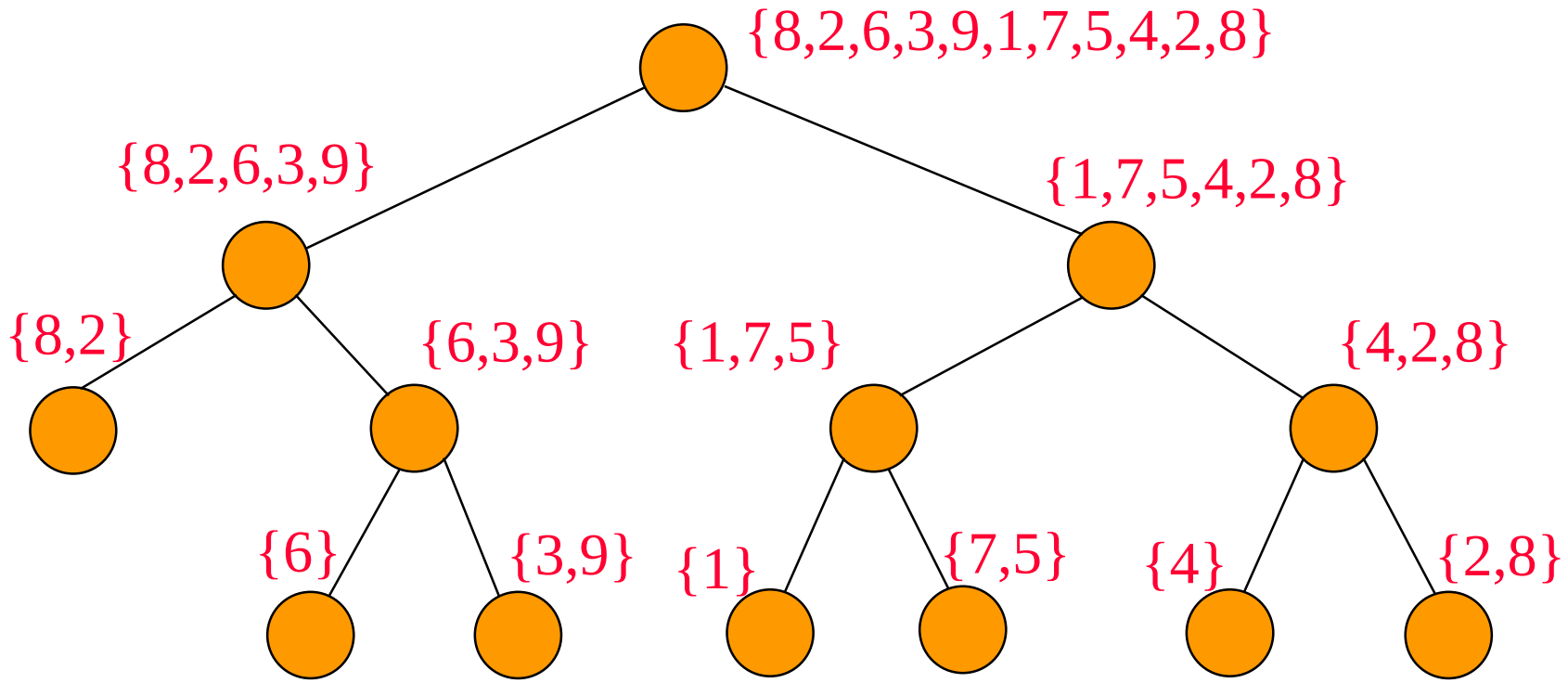
Example

- Find the min and max of {3, 5, 6, 2, 4, 9, 3, 1}.
  - **A** = {3, 5, 6, 2} and **B** = {4, 9, 3, 1}.
  - min(**A**) = 2, min(**B**) = 1.
  - max(**A**) = 6, max(**B**) = 9.
  - min{min(**A**), min(**B**)} = 1.
  - max{max(**A**), max(**B**)} = 9.
- 选苹果；挑运动员；.....

Exam4 Finding Minimum and Maximum（MaxMin）

(2) Divide and conquer

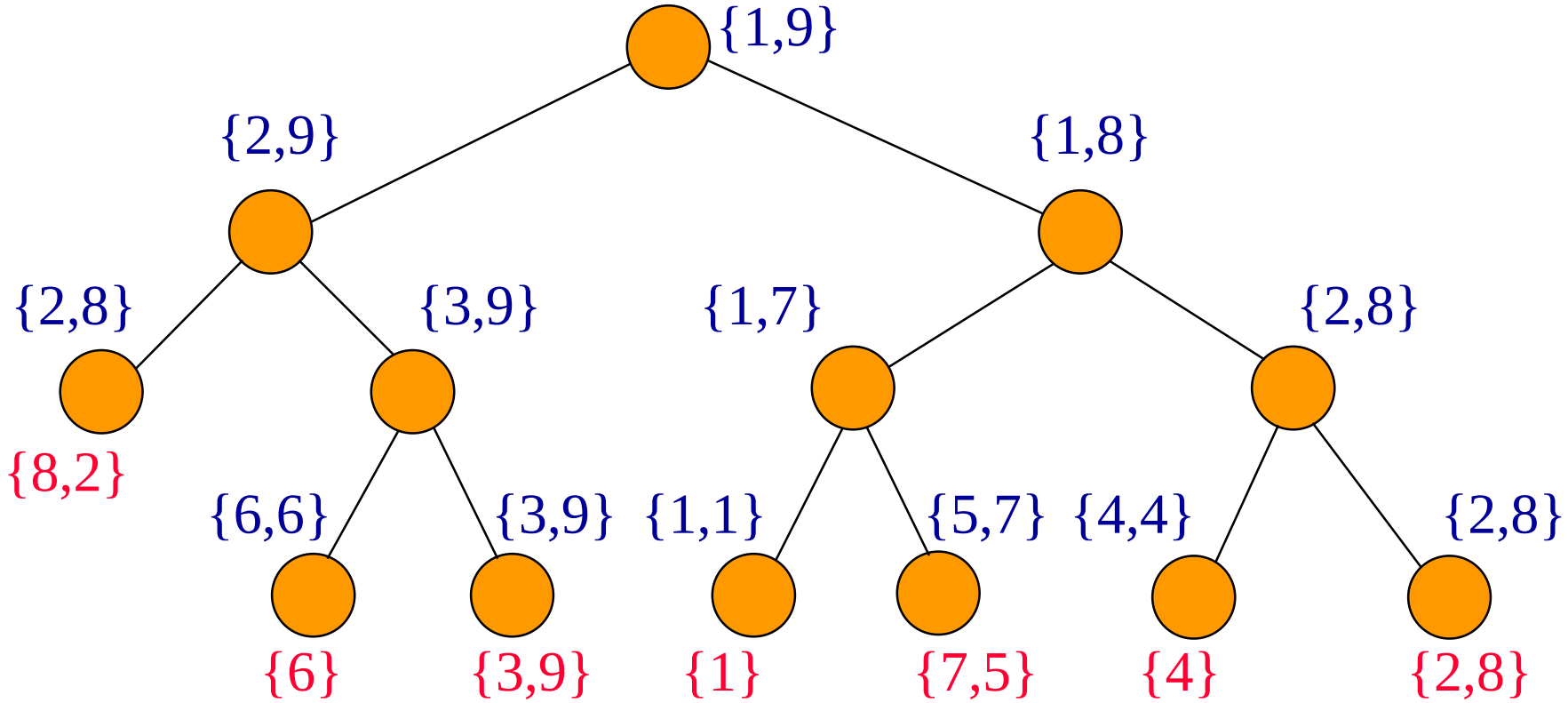
Dividing Into Smaller Problems



Exam4 Finding Minimum and Maximum（MaxMin）

(2) Divide and conquer

Dividing Into Smaller Problems



Exam4 Finding Minimum and Maximum（MaxMin）

(2) Divide and conquer

MaxMin(L)

if length(**L**) = 1 or 2, we use at most one comparison.

else

{ split (分裂) **L** into lists **L**1 and **L**2, each of $n/2$ elements

($min1$, $max1$) = **MaxMin(L1)**

($min2$, $max2$) = **MaxMin(L2)**

return (Min($min1$, $min2$), Max($max1$, $max2$))

}

Complexity analysis (Number of Comparisons):

$T(1) = 0$, $T(2) = 1$,

$T(n) = 2T(n/2)+2$

$= 4T(n/4)+2^2+2 = 2^3T(n/2^3) + 2^3+2^2+2 = \ldots$

$= 2^{k-1}T(n/2^{k-1})+2^{k-1}+\ldots+2 = 2^{k-1}+2^{k-1}+\ldots+2$

$= 2^{k-1}+2^k-2 = 3n/2 - 2$ (There, assume $n = 2^k$, $S_n = a(1-q^n)/(1-q)$)

Exam4 Finding Minimum and Maximum（MaxMin）

Comparison between Obvious method (2*n*-3) and Divide-and-Conquer method (3*n*/2-2)

Assume that one comparison takes one second.

Time	2*n*–3	3*n*/2–2
1 minute	$n = 31$	$n = 41$
1 hour	$n = 1801$	$n = 2401$
1 day	$n = 43201$	$n = 57601$

Exam5 Majority Problem（多数问题）

- Problem: Given an array **A** of *n* elements, only use "==" test to find the *majority element* (which appears more than *n*/2 times) in **A**.
- For example, given (2, 3, 2, 1, 3, 2, 2), then 2 is the majority element because 4 > 7/2.

(1) Trivial solution: $O(n^2)$

```
Majority(A[1, n])
    for(i = 1 to n)
        M = 1
        for(j = 1 to n)
            if (i != j and A[i] == A[j])
                M++
        if (M > n/2) return "A[i] is the majortiy"
    return "No majortity"
```

Exam5 Majority Problem（多数问题）

(2) Divide and conquer

Majority(A[1, *n*])
    if *n* = 1, then
        return **A**[1]
    else
        *m*1 = Majority(**A**[1, *n*/2])
        *m*2 = Majority(**A**[*n*/2+1, *n*])
    test if *m*1 or *m*2 is the majority for **A**[1, *n*]
    return majority or no majority.

A = (2, 1, 3, 2, 1, 5, 4, 2, 5, 2)

Complexity analysis (Counting):

$T(n) = 2T(n/2) + O(n) = O(n\log n)$

Exam5 Majority Problem（多数问题）

(3) However, there is a linear time algorithm for the problem. （假设最大元素不超过n）

```
for(i = 1 to n)
      ++counting[ A[i] ]
M = Max(counting[ .. ] ) // A[j] 时取得MAX
if (M > n/2)
      return "A[j] is the majority"
```

A = (2, 1, 3, 2, 1, 5, 4, 2, 5, 2)

counting[1] = 2
counting[2] = 4
counting[3] = 1
counting[4] = 1
counting[5] = 2

What's the space complexity?

Exam5 Majority Problem（多数问题）

(1) Trivial solution: $O(n^2)$

(2) Divide and conquer: $O(n\log n)$

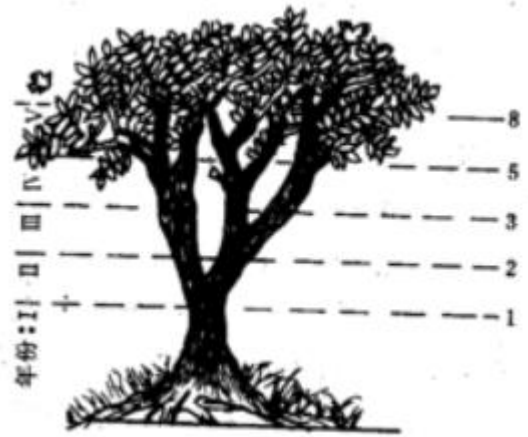
(3) Counting (计数): Linear time algorithm $O(n)$

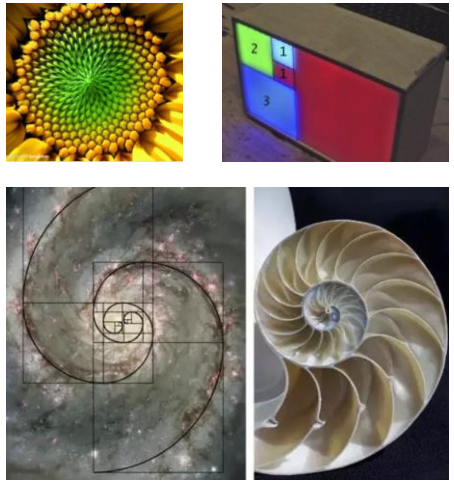
Moral (寓意) of the story?

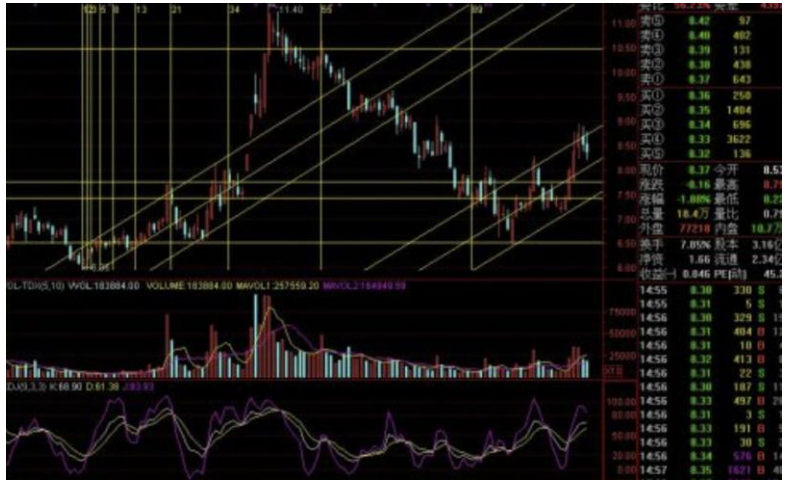
Divide and conquer may not always give you the best solution!

Exam6 Branch growth（树枝生长）

1 棵树枝 2 年后分叉（找出新树枝），第 n 年，顶端有多少棵树枝?

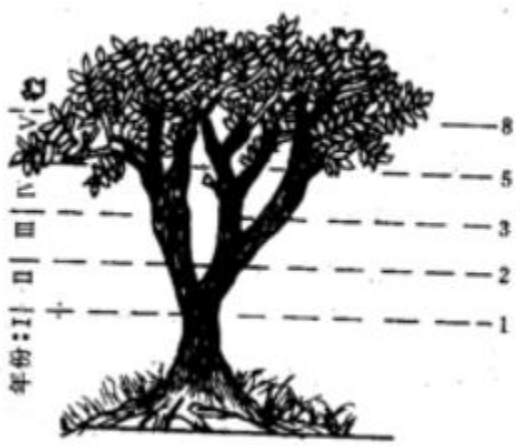


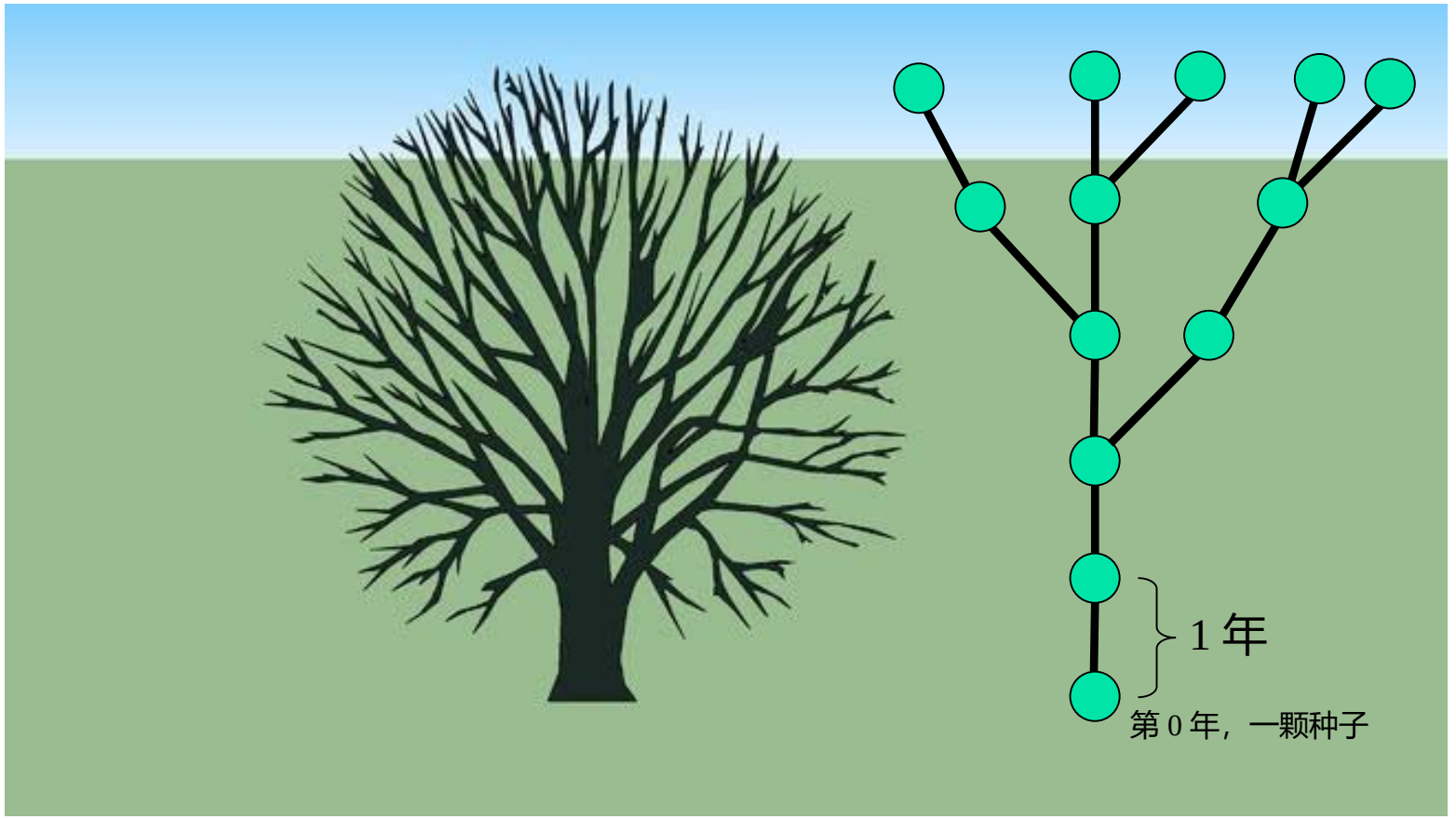




Exam6 Branch growth（树枝生长）

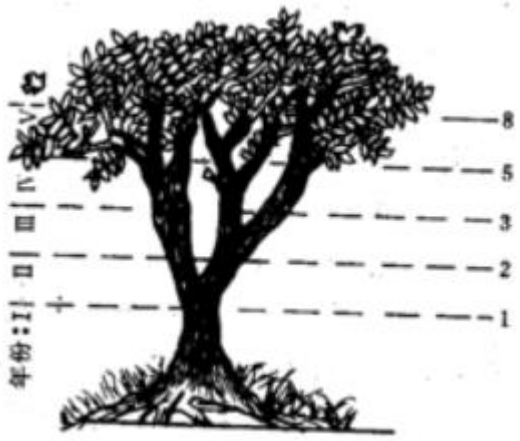
1 棵树枝 2 年后分叉（找出新树枝），第 n 年，顶端有多少棵树枝?





Exam6 Branch growth（树枝生长）

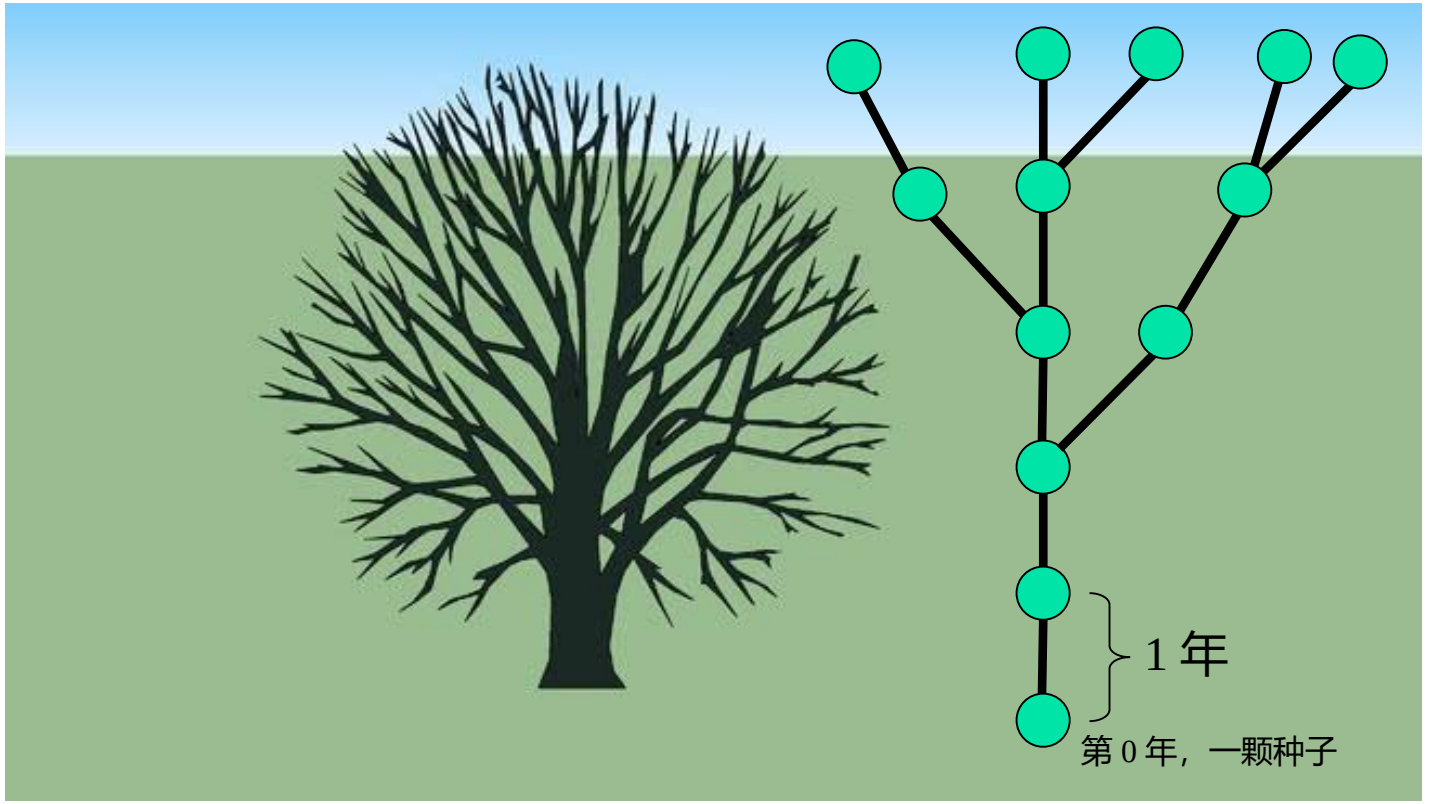
1 棵树枝 2 年后分叉（找出新树枝），第 n 年，顶端有多少棵树枝?



$F(1)$	$F(2)$	$F(3)$	$F(4)$	$F(5)$	$F(6)$	…	$F(n)$
1	1	2	3	5	8	…	?

递归解：$F(n) = F(n\text{-}1) + F(n\text{-}2)$

计算时间：$T(n) = T(n\text{-}1) + T(n\text{-}2)$



Fibonacci Number

$F(1)$	$F(2)$	$F(3)$	$F(4)$	$F(5)$	$F(6)$	…	$F(n)$
1	1	2	3	5	8	…	?

$$T(n) = T(n\text{-}1) + T(n\text{-}2)$$

1. 直接递归计算， $T(n) = a^n$ ($a \approx 1.6$)， $S(n)$ 也不小 (S: Space)

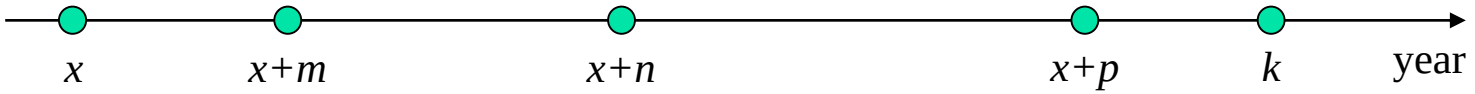
2. 用数组， $T(n) = n$, $S(n) = n$	$F[2] = F[1] = 1$; for (i: $1 \rightarrow n$) $F[i] = F[i\text{-}1]+F[i\text{-}2]$;
3. 用变量， $T(n) = n$, $S(n) = \text{const}$	for(i: $1 \rightarrow n$) $f = f1+f2$, $f2=f1$, $f1 = f$;
4. 通项公式（黄金分割）， $T(n) = ?$	$F(n) = (1/\sqrt{5})*\{[(1+\sqrt{5})/2]^n - [(1-\sqrt{5})/2]^n\}$;
5. 用矩阵法，$\lg(n)$	$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$

6. 其他方法……

Fibonacci Number -（递归与分治也可以比较复杂）

例：母牛的数量（F题 SkyLee的艾露猫）（鼹鼠）（兔子）

一头 x 年出生的母牛从 $x+m$ 年到 $x+n$ 年间每年生出一头母牛，并在 $x+p$ 年被淘汰。写一个程序，按顺序读入整数 m, n, p, k ($3<m<n<p<60, 0<k<60$)，设第 0 年有一头刚出生的母牛，计算第 k 年时共存有多少头未被淘汰的母牛。（母牛会老也会死的情况）



- 第 k 年母牛的总数量 $T(k)$
- 第 k 年新生母牛 $N(k)$

详细分析见PPT a03-1

$$T(k) = N(k\text{-}p+1) + N(k\text{-}p) + \ldots + N(k\text{-}0) \quad \text{----------} \quad (1)$$

$$N(k) = N(k\text{-}n) + N(k\text{-}n+1) + \ldots + N(k\text{-}m) \quad \text{----------} \quad (2)$$

$T(n)$的时间复杂度分析？

Exercises

Find the second heaviest of n elements using a balance that allows you to compare the weight of 2 elements. （对于一个具有 n 个元素的数组，用一个天平，通过比较 2个元素的重量，求出第二重的一个）

Try to give an algorithm, and analyze its running time. If using divide-and-conquer method, how to describe the algorithm? And its running time?

Exercises

two $n \times n$ matrices **A** and **B**, Complexity(**C** = **A**×**B**) = ?

Standard method

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$T(n) = O(n^3)$

Strassen Algorithm

$T(n) = O(n^{2.807})$

把这两种算法分别编程实现，给出若干输入，对比两种方法的基本操作的步数，并对结果给出你的认识和体会。