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)**
- **Fibonacci Number**

Exam1 Merge Sort

……

Exam2 Multiplication of two integers（整数相乘）

Multiplication of two integers （整数相乘问题）

X和Y是两个n位的十进制整数，分别表示为

$X = x_{n-1}x_{n-2}...x_0$, $Y = y_{n-1}y_{n-2}...y_0$ ，其中$0 \le x_i , y_j \le 9$ ($i , j = 0, 1, \ldots , n-1$)，设计一个算法求X×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。

two *n*-digit numbers X and Y, Complexity(X × Y) = ?

- **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
```

```
        16
       480
      7200
      40
     08
    32
   08
  24
  ---------
  251327696
```

- **Complexity analysis: n^2 multiplications and at most n^2-1 additions (加法). So, $T(n)=O(n^2)$.**

Exam2 Multiplication of two integers（整数相乘）

two n-digit numbers X and Y, Complexity(X × Y) = ?

- **Divide and Conquer algorithm**

Let $\quad X = a\ b$

$\quad\quad\quad Y = c\ d$

where a, b, c and d are $n/2$ digit numbers, e.g. $1364 = 13 \times 10^2 + 64$.

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 × Y) = ?

- **Divide and Conquer algorithm**

  Let $X = a\ b$, $Y = c\ d$

  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 mod 10^m
  $c = \lfloor Y/10^m \rfloor$; d=Y mod 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, we have
$T(n)= O(n^2)$.

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)**

  Let $X = a\ b$, $Y = c\ d$

  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 mod 10^m
$c = \lfloor Y/10^m \rfloor$; d=Y mod 10^m
e = FastMultiply(a; c; m)
f = FastMultiply(b; d; m)
g = FastMultiply($a-b$; $c-d$; m)
return $10^{2m}e + 10^m(e + f - g) + f$

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

$$T(n) = O(n^{\log_2 3}) = O(n^{1.585})$$

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) = ?

- **Standard method**

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

$$\begin{pmatrix} \cdots\cdots\ \cdot\ \cdot\cdot \\ \cdots\cdots c_{ij} \cdot\cdot \\ \cdots\cdots\ \cdot\ \cdot\cdot \\ \cdots\cdots\ \cdot\ \cdot\cdot \end{pmatrix} = \begin{pmatrix} \cdots\cdots\cdots \\ ***** \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \end{pmatrix} \begin{pmatrix} \cdots\cdots * \cdot\cdot \\ \cdots\cdots * \cdot\cdot \\ \cdots\cdots * \cdot\cdot \\ \cdots\cdots * \cdot\cdot \end{pmatrix}$$

MATRIX-MULTIPLY(A, B)

for $i \leftarrow 1$ **to** n

  for $j \leftarrow 1$ **to** n

    $C[i, j] \leftarrow 0$

    for $k \leftarrow 1$ **to** n

      $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$

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) = ?

- **Divide and conquer**

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

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

$C_{11}=A_{11}B_{11}+A_{12}B_{21}$, $C_{12}=A_{11}B_{12}+A_{12}B_{22}$

$C_{21}=A_{21}B_{11}+A_{22}B_{21}$, $C_{22}=A_{21}B_{12}+A_{22}B_{22}$

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) = ?

- **Divide and conquer (Strassen Algorithm)**

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

Gaussian elimination is not optimal

V Strassen - Numerische mathematik, 1969 - Springer

Received December 12, t 968 t. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with tess than 4.7-nlg7 arithmetical operations (all logarithms in this paper are for …

Exam3 Multiplication of two matrices（矩阵相乘）

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

- **Divide and conquer (Strassen Algorithm)**

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

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

Define
$$P_1 = (A_{11}+A_{22})(B_{11}+B_{22})$$
$$P_2 = (A_{11}+A_{22})B_{11}$$
$$P_3 = A_{11}(B_{11}-B_{22})$$
$$P_4 = A_{22}(-B_{11}+B_{22})$$
$$P_5 = (A_{11}+A_{12})B_{22}$$
$$P_6 = (-A_{11}+A_{21})(B_{11}+B_{12})$$
$$P_7 = (A_{12}-A_{22})(B_{21}+B_{22})$$

Then
$$C_{11}=P_1+P_4-P_5+P_7,\quad C_{12}=P_3+P_5$$
$$C_{21}=P_2+P_4,\quad C_{22}=P_1+P_3-P_2+P_6$$

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

$$T(n) = O(n^{\log_2 7}) = O(n^{2.807})$$

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 progression. Journal of Symbolic Computation, 9(3):251–280, 1990.

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

Matrix multiplication via arithmetic progressions

D Coppersmith, S Winograd - … of the nineteenth annual ACM symposium …, 1987 - dl.acm.org

Abstract 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.

Exam4 Finding Minimum and Maximum（MaxMin）

- **Background**

  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. **（正确找出需要的元素，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.

- **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）

- **Divide and conquer**

Example

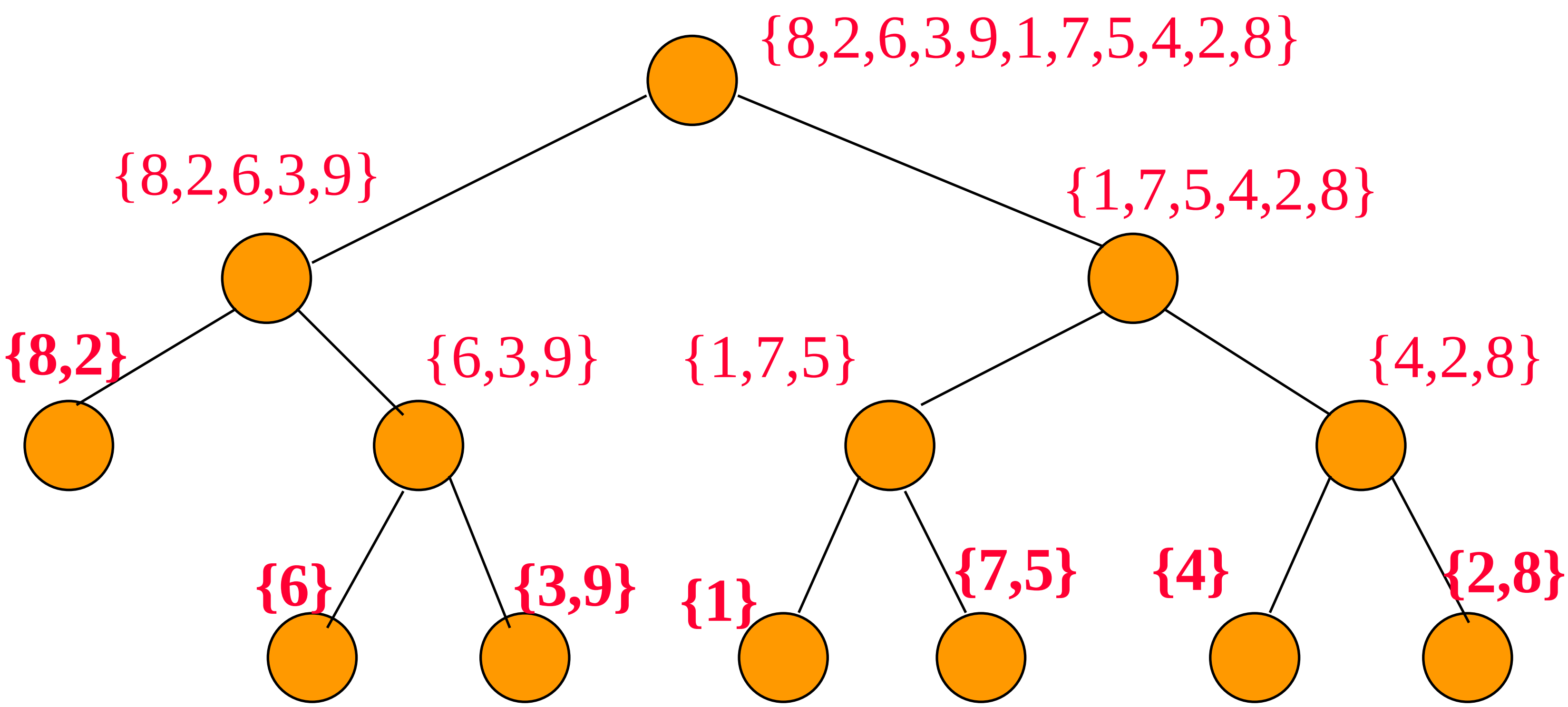
- **Find the min and max of {3,5,6,2,4,9,3,1}.**
  - **A = {3,5,6,2} and B = {4,9,3,1}.**
  - **min(A) = 2, min(B) = 1.**
  - **max(A) = 6, max(B) = 9.**
  - **min{min(A),min(B)} = 1.**
  - **max{max(A), max(B)} = 9.**
- 选苹果；挑运动员；……

Exam4 Finding Minimum and Maximum（MaxMin）

- **Divide and conquer**

  Example

  - **Dividing Into Smaller Problems**

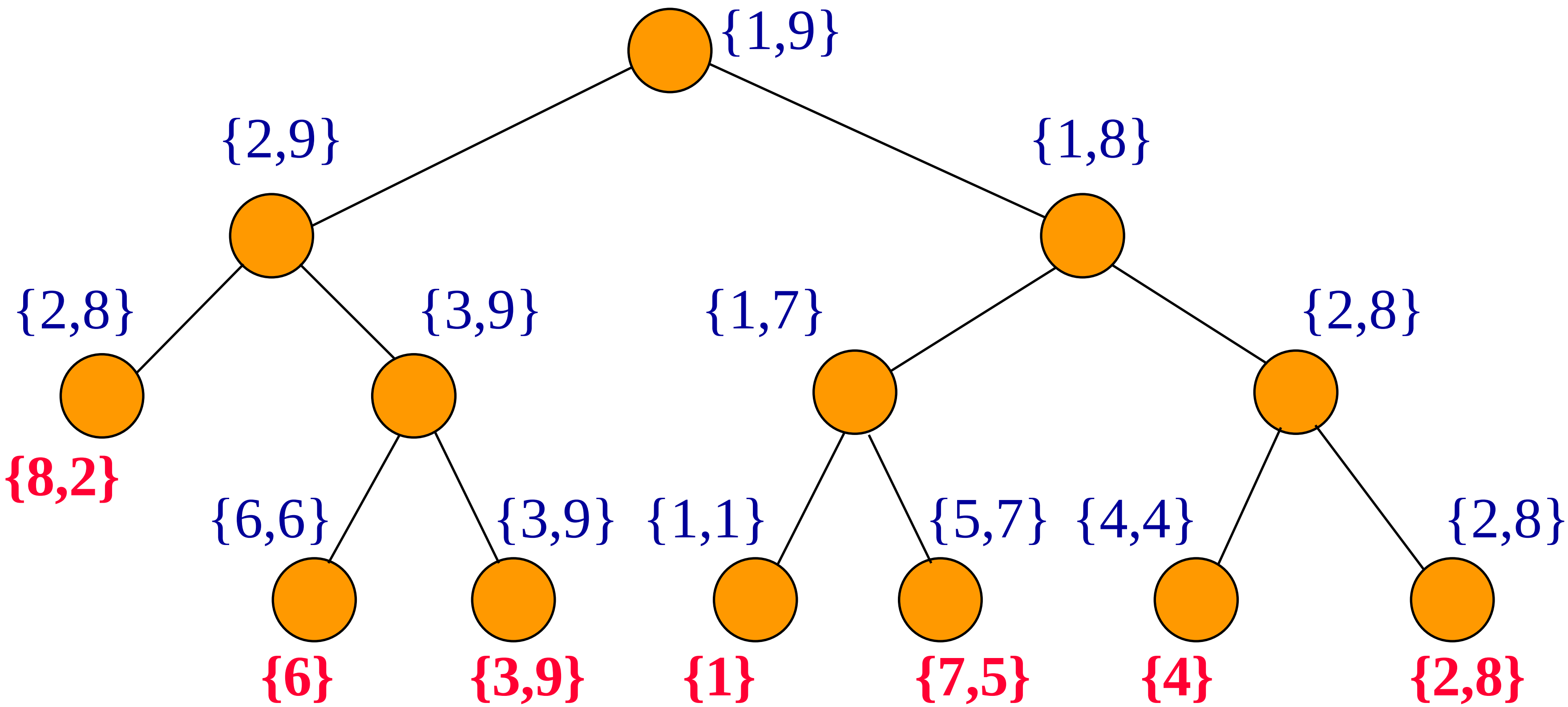


Exam4 Finding Minimum and Maximum（MaxMin）

- **Divide and conquer**

  Example

  - **Solve Small Problems and Combine**



Exam4 Finding Minimum and Maximum（MaxMin）

- **Divide and conquer**

```
MaxMin(L)
    if length(L)=1 or 2, we use at most one comparison.
    else
    {   split (分裂) L into lists L1 and L2, 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	$2n–3$	$3n/2–2$
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.**

- **Trivial solution:**

  counting (计数) is $O(n^2)$.

```
Majority(A[1, n])
  for(i = 1 to n)
    M = 0
    for(j = 1 to n)
        if (i != j and A[i]==A[j])  M++
    end
    if (M>n/2) return "A[i] is the majortiy"
  end
  return "No majortity"
```

Exam5 Majority Problem（多数问题）

- **Divide and conquer**

Complexity analysis (Counting):
$T(n) = 2T(n/2)+O(n) = O(n\log n)$

Majority(A[1, *n*])
if *n*=1, then
return A[1]
else
m1=Majority(A[1, *n*/2])
m2=Majority(A[*n*/2+1, *n*])
test if m1 or m2 is the majority for A[1, *n*]
return majority or no majority.

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

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

However, there is a linear time algorithm for the problem.

for(*i*=1 to *n*) ++frequency[A[*i*]]
M = Max(frequency(A[*j*]))
if (M > *n*/2)
return “A[*j*] is the majority”

- **Moral (寓意) of the story?**

Exam5 Majority Problem（多数问题）

- **Divide and conquer**

Majority(A[1, *n*])
if *n*=1, then
    return A[1]
else
    m1=Majority(A[1, *n*/2])
    m2=Majority(A[*n*/2+1, *n*])
test if m1 or m2 is the majority for A[1, *n*]
return majority or no majority.

Complexity analysis (Counting):
$$T(n) = 2T(n/2)+O(n) = O(n\log n)$$

However, there is a linear time algorithm for the problem.

- **Moral (寓意) of the story: Divide and conquer may not always give you the best solution!**

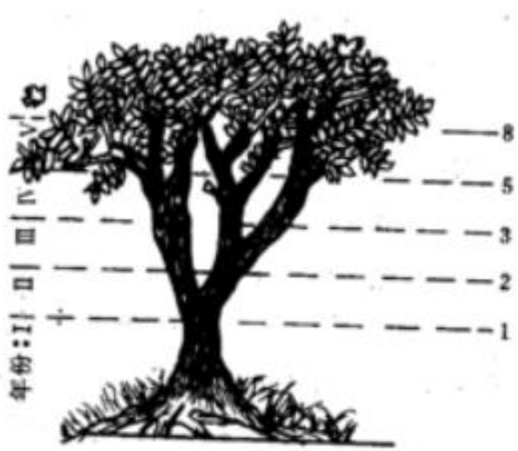
Exam6 Fibonacci Number

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

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

Why?

这里把fib数列的递归关系看作某个算法的复杂度的递归关系，于是这里就用T表示，不用F了。



Exam6 Fibonacci Number

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

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

这里把fib数列的递归关系看作某个算法的复杂度的递归关系，于是这里就用T表示，不用F了。

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
3. 用变量，T(n) = n, S(n) = const
4. 通项公式（黄金分割），T(n) = ?
5. 用矩阵法，lg(n)
6. 其他方法……

F[2] = F[1]=1; for (i: F[i] = F[i-1]+F[i-2];)

for(i: f = f1+f2; f2=f1; f1 = f;)

F(n)=(1/√5)*{[(1+√5)/2]^n - [(1-√5)/2]^n}

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

Exercise

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?