数据结构与算法（三）

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008. 6（“十一五”国家级规划教材）

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

第3章　栈与队列

- 栈
- **栈的应用**
  - **递归到非递归的转换**
- 队列

递归转非递归

- 递归函数调用原理
- **机械的递归转换**
- **优化后的非递归函数**

(2) 机械的递归转换

递归转换为非递归的方法

$$fu(n)=\begin{cases} n+1 & \text{当}n<2\text{时} \\ fu(\lfloor n/2 \rfloor)*fu(\lfloor n/4 \rfloor) & n\geq 2\text{时} \end{cases}$$

- 机械方法

1. 设置一工作栈当前工作记录
2. 设置 t+2个语句标号
3. 增加非递归入口
4. 替换第 i (i = 1, …, t)个递归规则
5. 所有递归出口处增加语句：goto label t+1;
6. 标号为t+1的语句的格式
7. 改写循环和嵌套中的递归
8. 优化处理

rd=2: n=0 f=? u1=? u2=?
rd=1: n=3 f=? u1=2 u2=?
rd=3: n=7 f=? u1=? u2=?

(2) 机械的递归转换

1. 设置一工作栈

$$fu(n)=\begin{cases} n+1 & \text{当}n<2\text{时} \\ fu(\lfloor n/2 \rfloor)*fu(\lfloor n/4 \rfloor) & n\geq 2\text{时} \end{cases}$$

- 在函数中出现的所有参数和局部变量都必须用栈中相应的数据成员代替
  - 返回语句标号域 (t+2个数值)
  - 函数参数(值参、引用型)
  - 局部变量

```
typedef struct elem { // 栈数据元素类型
    int rd;                 // 返回语句的标号
    Datatypeofp1 p1;        // 函数参数
    …
    Datatypeofpm pm;
    Datatypeofq1 q1;        // 局部变量
    …
    Datatypeofqn qn;
} ELEM;
```

rd=2: n=0 f=? u1=? u2=?
rd=1: n=3 f=? u1=2 u2=?
rd=3: n=7 f=? u1=? u2=?

(2) 机械的递归转换

2. 设置t+2个语句标号

- label *0* ：第一个可执行语句
- label t+1 ：设在函数体结束处
- label i (1<=i<=t)： 第i个递归返回处

```
void exmp(int n, int& f) {
    int u1, u2;
    if (n<2)
        f = n+1;
    else {
        exmp((int)(n/2), u1);
        exmp((int)(n/4), u2);
        f = u1*u2;
    }
}
```

(2) 机械的递归转换

3. 增加非递归入口

// 入栈

S.push(t+1，p1, ...，pm，q1，...qn);

(2) 机械的递归转换

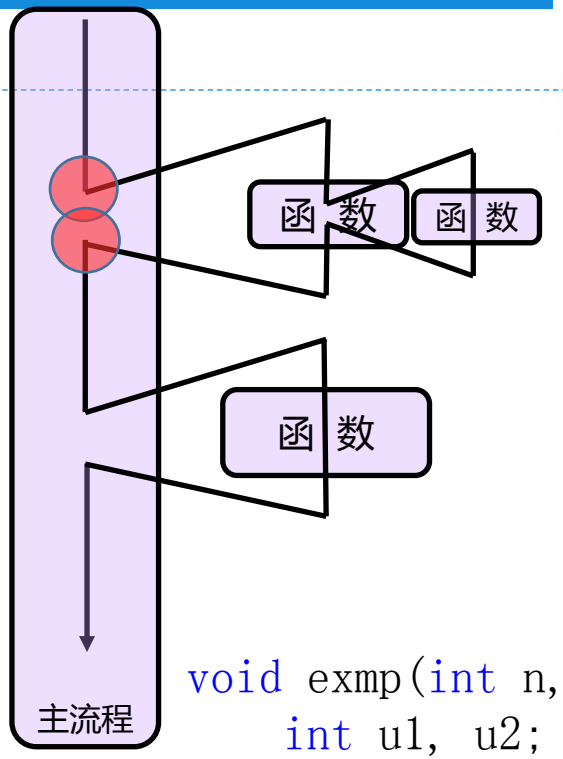
4. 替换第i (i=1,...,t)个递归规则

- 假设函数体中第i (i=1, ..., t)个

  递归调用语句为：recf(a1, a2, ...,am) ;
- 则用以下语句替换：

```
S.push(i, a1, ..., am) ;  // 实参进栈
goto label 0 ;
.....
label i :  x = S.top() ; S.pop();
```

/* 退栈，然后根据需要，将x中某些值赋给栈顶的工作记录S.top () — 相当于把引用型参数的值回传给局部变量 */




```
void exmp(int n, int& f) {
    int u1, u2;
    if (n<2)
      f = n+1;
    else {
      exmp((int)(n/2), u1);
      exmp((int)(n/4), u2);
      f = u1*u2;
    }
}
```

5. 所有递归出口处增加语句

- goto label *t+1*;

```
void exmp(int n, int& f) {
    int u1, u2;
    if (n<2)
        f = n+1;
    else {
        exmp((int)(n/2), u1);
        exmp((int)(n/4), u2);
        f = u1*u2;
    }
}
```

6. 标号为t+1的语句

```
switch ((x=S.top ()).rd) {
    case 0 :    goto label 0;
                break;
    case 1 :    goto label 1;
                break;
    ……
    case t+1 :  item = S.top(); S.pop(); // 返回
                break;
    default :   break;
}
```

7. 改写循环和嵌套中的递归

- 对于循环中的递归，改写成等价的goto型循环
- 对于嵌套递归调用

  例如，recf (... recf())

  改为：

  $exmp_1$ = recf ( );

  $exmp_2$ = recf ($exmp_1$);

  ...

  $exmp_k$ = recf ($exmp_{k-1}$)

  然后应用规则 4 解决

8. 优化处理

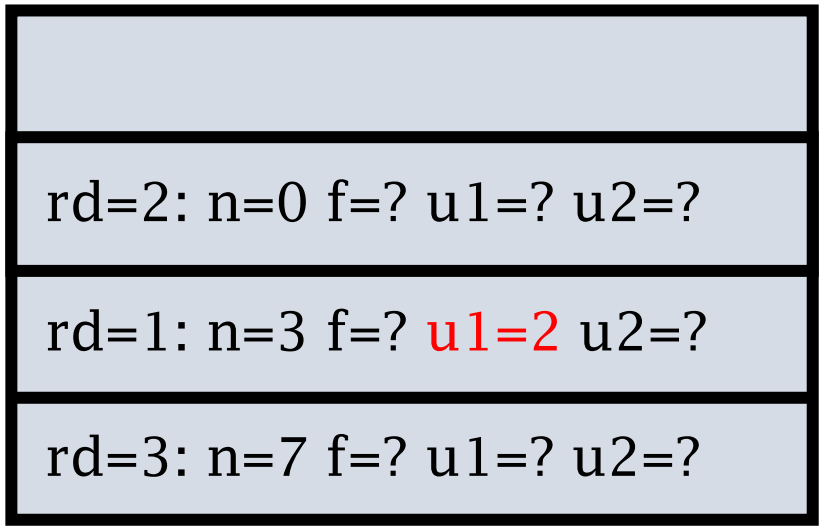
- 进一步优化
  - 去掉冗余进栈/出栈
  - 根据流程图找出相应的循环结构，从而消去goto语句

(2) 机械的递归转换

数据结构定义

$$fu(n)=\begin{cases} n+1 & \text{当}n<2\text{时} \\ fu(\lfloor n/2 \rfloor)*fu(\lfloor n/4 \rfloor) & n\geq 2\text{时} \end{cases}$$

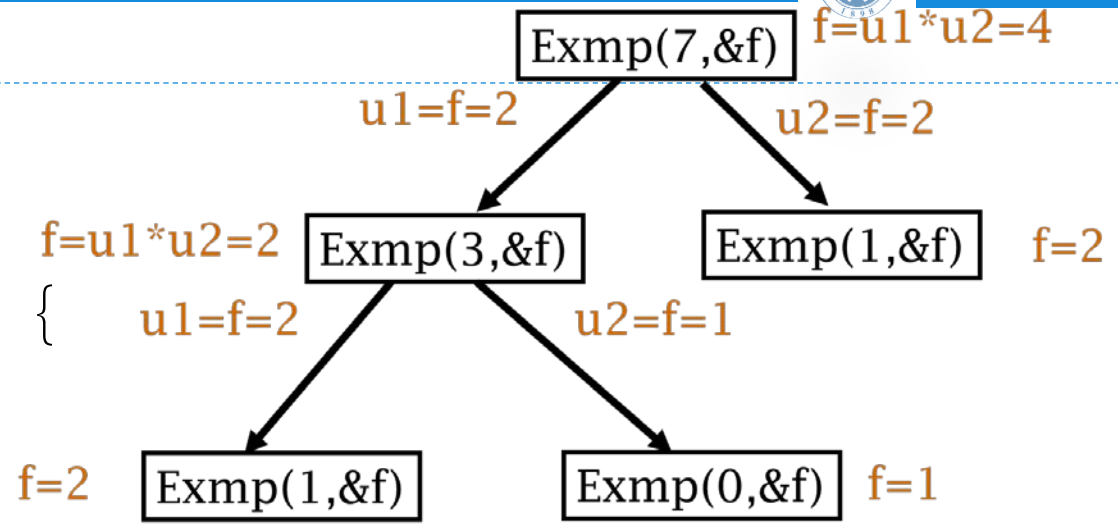
```
typedef struct elem {
    int rd, pn, pf, q1, q2;
} ELEM;
class nonrec {
    private:
        stack <ELEM> S;
    public:
        nonrec(void) { }      // constuctor
        void replace1(int n, int& f);
};
```



(2) 机械的递归转换

递归入口

```
void nonrec::replace1(int n, int& f)   {
    ELEM x, tmp
    x.rd = 3;      x.pn = n;
    S.push(x);      // 压在栈底作"监视哨"
label0: if ((x = S.top()).pn < 2)   {
            S.pop( );
            x.pf = x.pn + 1;
            S.push(x);
            goto label3;
        }
```



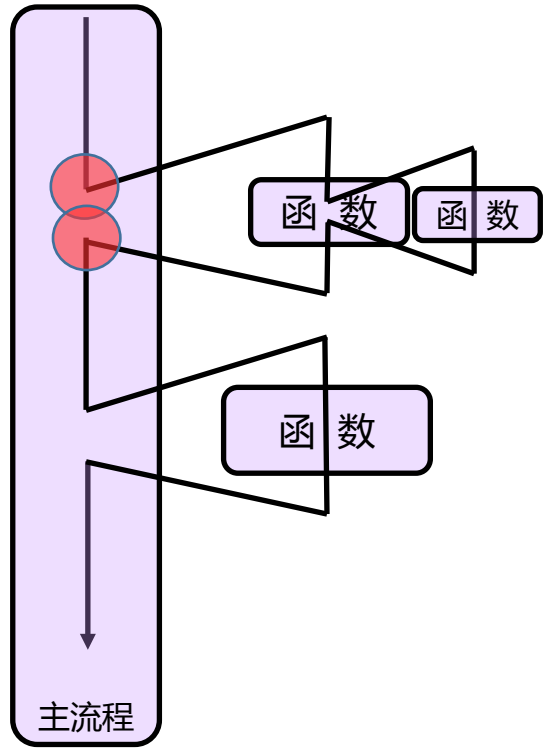

```
void exmp(int n, int& f) {
    int u1, u2;
    if (n<2)
      f = n+1;
    else {
      exmp((int)(n/2), u1);
      exmp((int)(n/4), u2);
      f = u1*u2;
    }
}
```

(2) 机械的递归转换

第一个递归语句

$$fu(n)=\begin{cases} n+1 & \text{当}n<2\text{时} \\ fu(\lfloor n/2 \rfloor)*fu(\lfloor n/4 \rfloor) & n\geq 2\text{时} \end{cases}$$

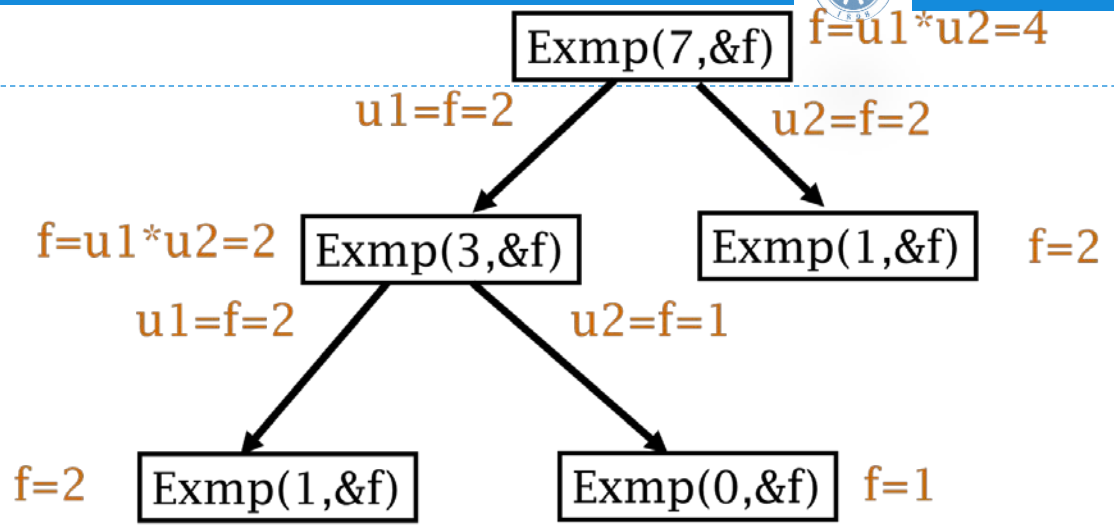
```
        x.rd = 1;          // 第一处递归
        x.pn = (int)(x.pn/2);
        S.push(x);
        goto label0;
label1: tmp = S.top(); S.pop();
        x = S.top(); S.pop();
        x.ql = tmp.pf; // 修改ul=pf
        S.push(x);
```



(2) 机械的递归转换

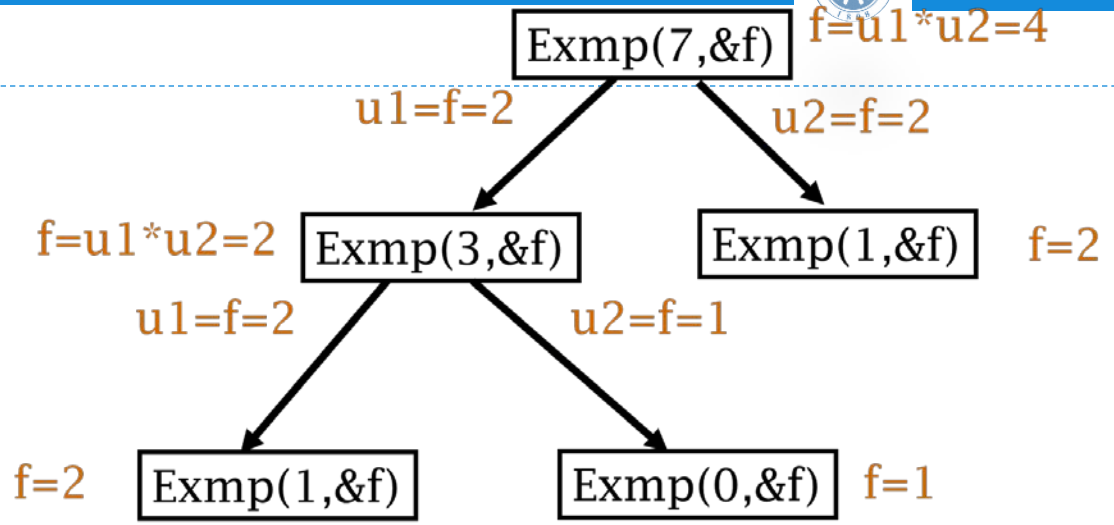
第二个递归语句

```
        x.pn = (int)(x.pn/4);
        x.rd = 2;
        S.push(x);
        goto label0;
label2: tmp = S.top(); S.pop();
        x = S.top(); S.pop();
        x.q2 = tmp.pf;
        x.pf = x.q1 * x.q2;
        S.push(x);
```




```
void exmp(int n, int& f) {
    int u1, u2;
    if (n<2)
      f = n+1;
    else {
      exmp((int)(n/2), u1);
      exmp((int)(n/4), u2);
      f = u1*u2;
    }
}
```

(2) 机械的递归转换

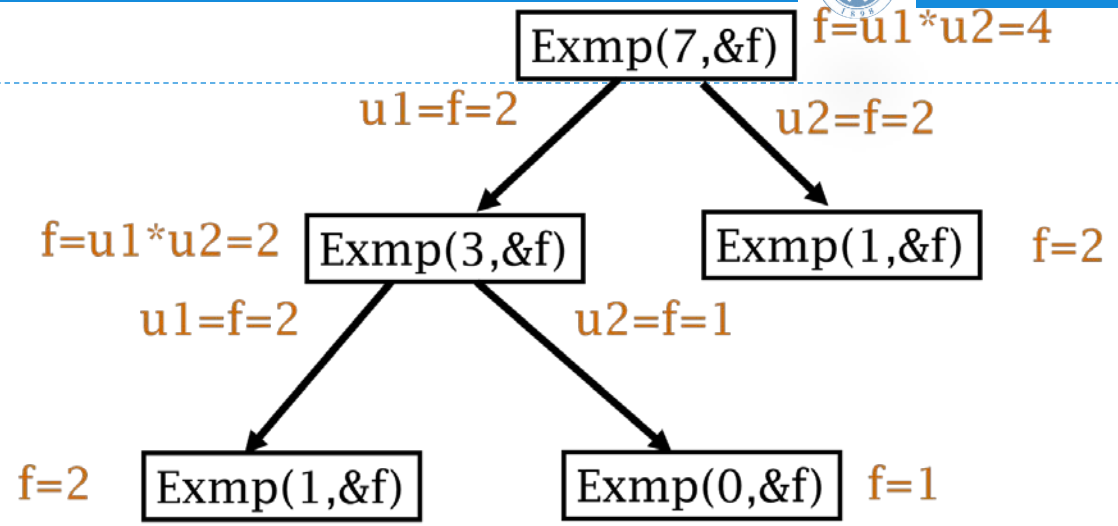



```
label3: x = S.top();
      switch(x.rd)  {
         case 1 : goto label1;
                  break;
         case 2 : goto label2;
                  break;
         case 3 : tmp = S.top(); S.pop();
                  f = tmp.pf;            //计算结束
                  break;
         default : cerr << "error label number in stack";
                   break;
       }
}
```

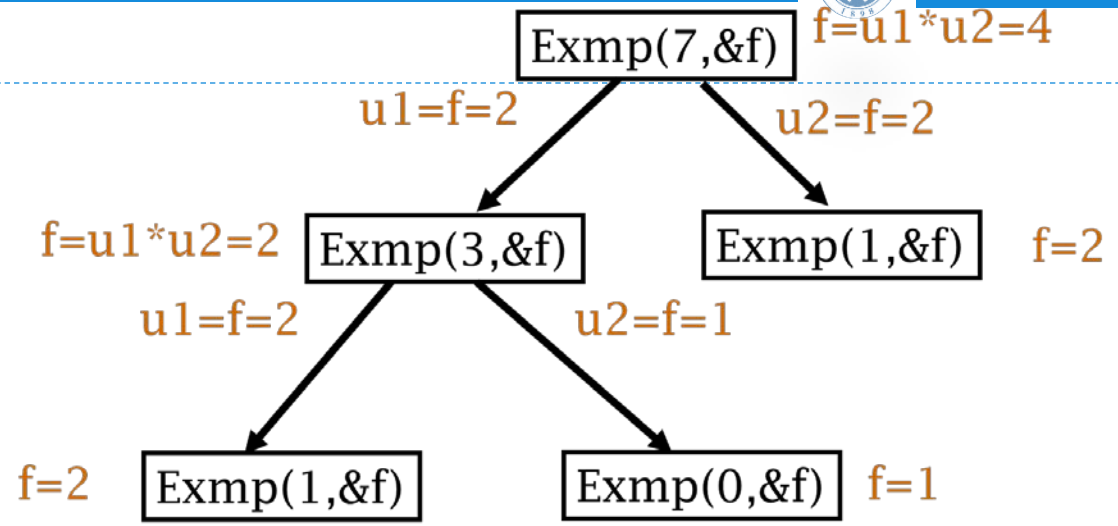
（3）优化后的非递归函数

优化后的非递归算法

```
void nonrec::replace2(int n, int& f)  {
    ELEM x, tmp;
     // 入口信息
    x.rd = 3;   x.pn = n;   S.push(x);
    do  {
      // 沿左边入栈
       while ((x=S.top()).pn >= 2){
          x.rd = 1;
          x.pn = (int)(x.pn/2);
          S.push(x);
       }
```



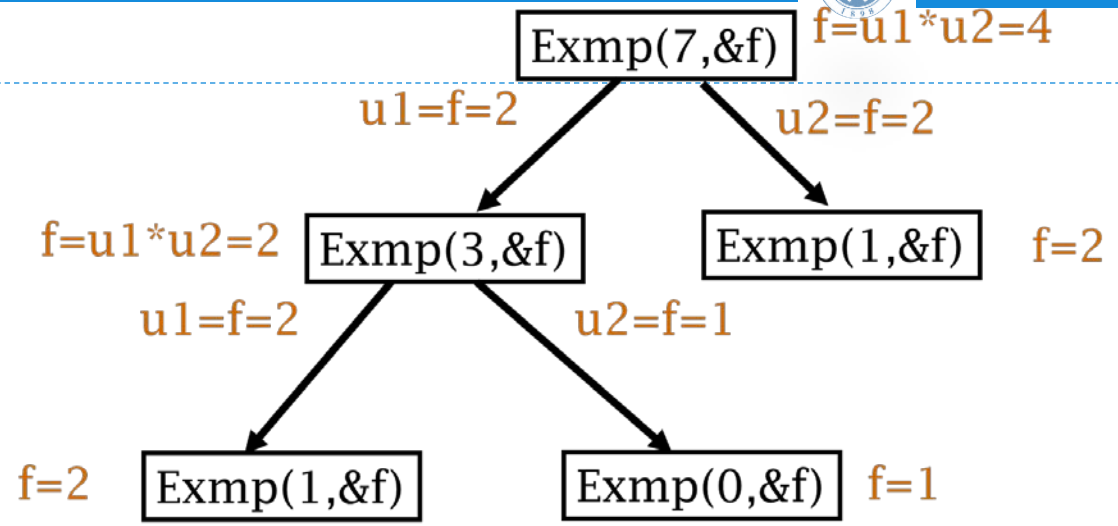
(3) 优化后的非递归函数




```
x = S.top(); S.pop();   // 原出口，n <= 2
x.pf = x.pn + 1;
S.push(x);
// 如果是从第二个递归返回，则上升
while ((x = S.top()).rd==2) {
  tmp = S.top(); S.pop();
  x = S.top(); S.pop();
  x.pf = x.q * tmp.pf;
  S.push(x);
}
```

(3) 优化后的非递归函数

```
        if ((x = S.topValue()).rd == 1) {
            tmp = S.top(); S.pop();
            x = S.top(); S.pop();
            x.q = tmp.pf;    S.push(x);
            tmp.rd = 2;  // 进入第二个递归
            tmp.pn = (int)(x.pn/4);
            S.push(tmp);
        }
    } while ((x = S.top()).rd != 3);
    x = S.top(); S.pop();
    f = x.pf;
}
```



快排的时间对照（单位ms）

方法 \ 数据量	10000	100000	1000000	10000000
递归快排	4.5	29.8	268.7	2946.7
机械方法的非递归快排	1.6	23.3	251.7	2786.1
非机械方法的非递归快排	1.6	20.2	248.5	2721.9
STL中的sort	4.8	59.5	629.8	7664.1

注：测试环境
Intel Core Duo CPU T2350
内存 512MB
操作系统 Windows XP SP2
编程环境 Visual C++ 6.0

递归与非递归处理问题的规模

- 递归求f(x)：

```
int f(int x) {
    if (x==0) return 0;
    return f(x-1)+1;
}
```

  - 在默认设置下，当x超过 11,772 时会出现堆栈溢出
- 非递归求f(x)，栈中元素记录当前x值和返回值
  - 在默认设置下，当x超过32,375,567时会出现错误

思考

- 用机械的转换方法
  - 阶乘函数
  - 2阶斐波那契函数

$$f_0=0,\ f_1=1,\ f_n = f_{n-1}+ f_{n-2}$$

  - Hanoi 塔算法

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材