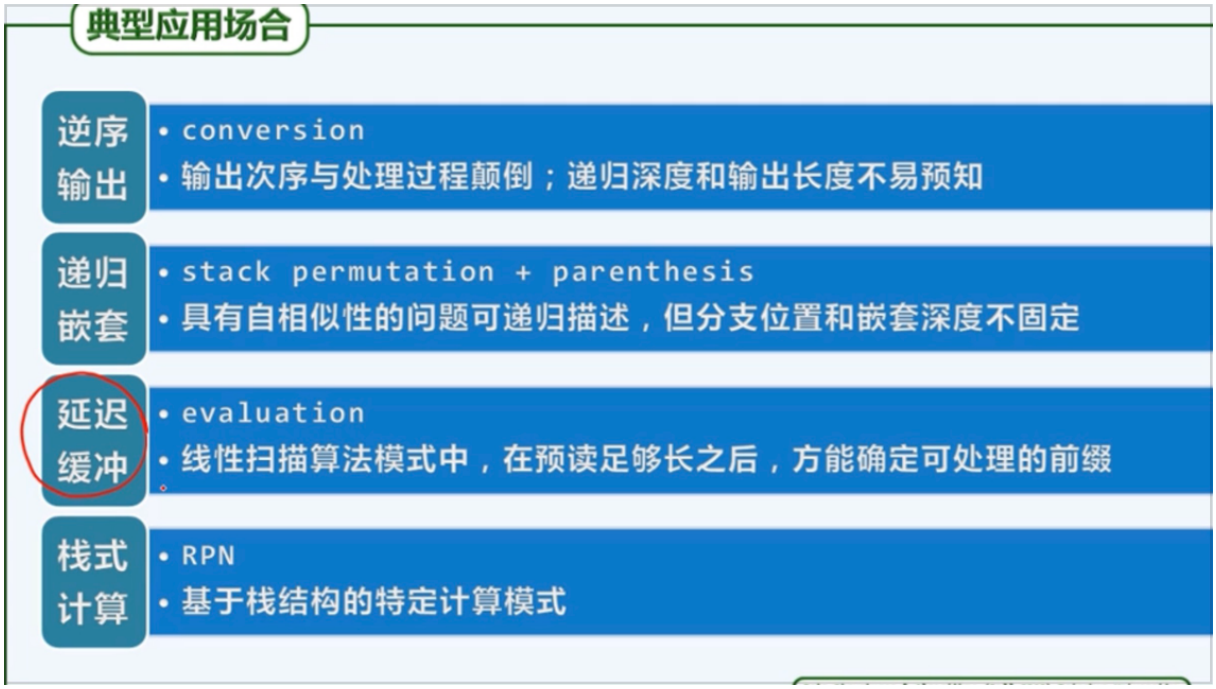
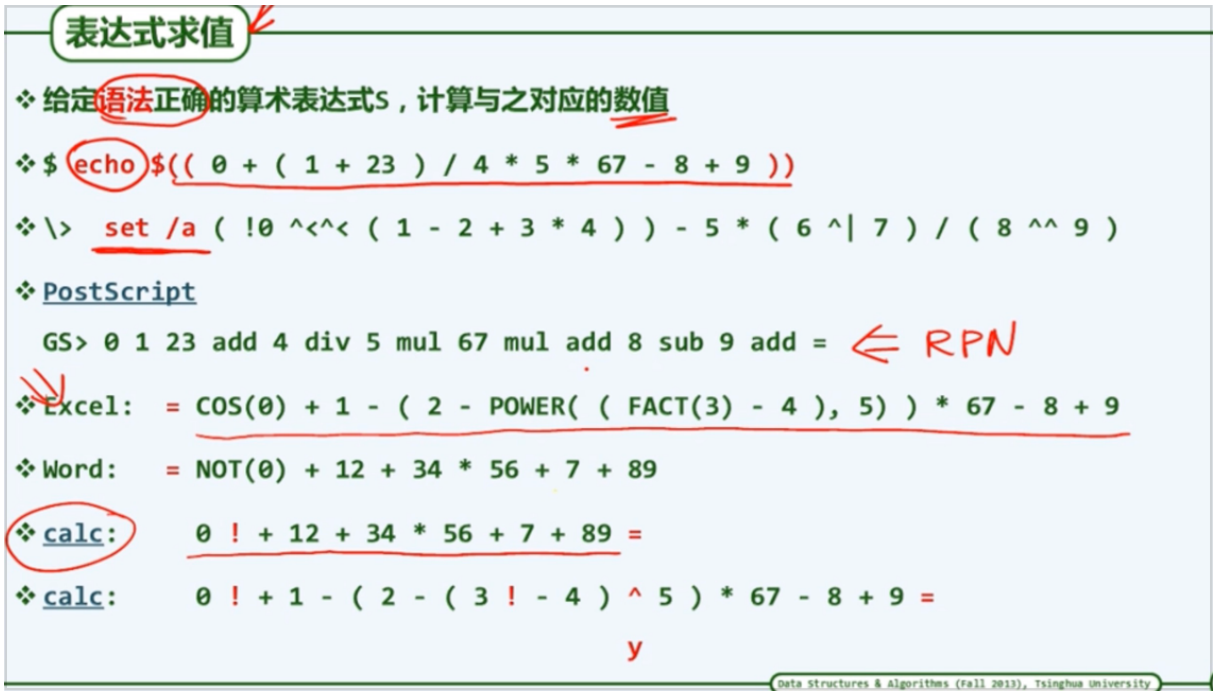
04-F 中缀表达式求值

#数据结构邓神

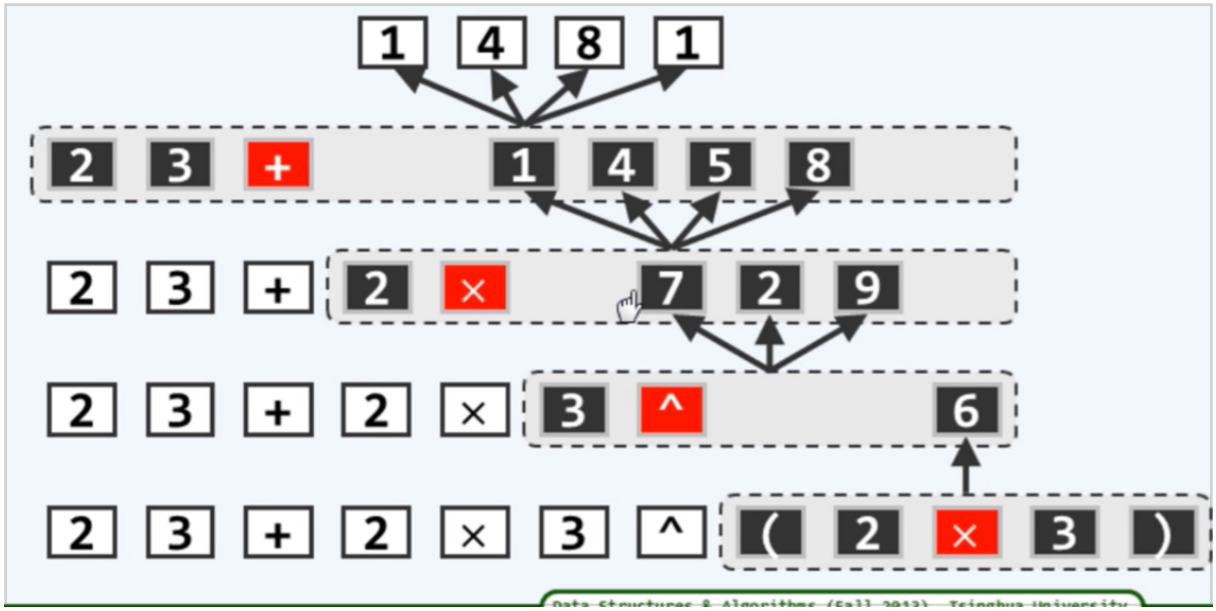
把玩



表达式求值：



构思

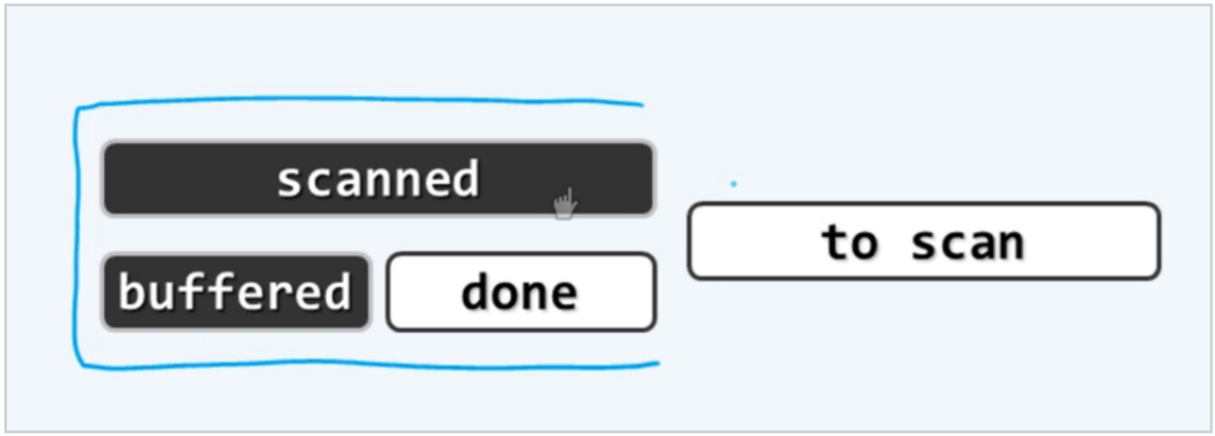


每次找到一个优先的表达式，然后计算将其表达式转换为一个值

实现减而治之的算法。

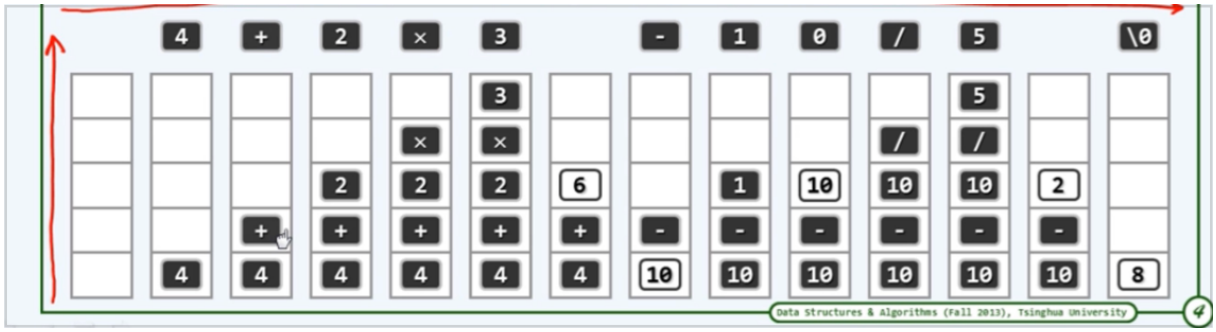
如果用线性扫描的算法，每次扫描到一个运算符都无法确定这个运算符号是否可以优先计算，换句话说我们扫描到次序无法与计算次序相对应

借助栈结构



我们吧所有扫描过的运算符存为一个栈， done表示经过判断可以处理的，而Buffer就是待处理的

实例



我们发现如果在一个高优先级的运算符的后面出现一个低（或者同级别运算符，或者结束）优先级的运算符就表示之前的运算符可以计算，

这种算法可以线性完成的表达式计算
最后在栈内唯一的元素就是我们需要的值。

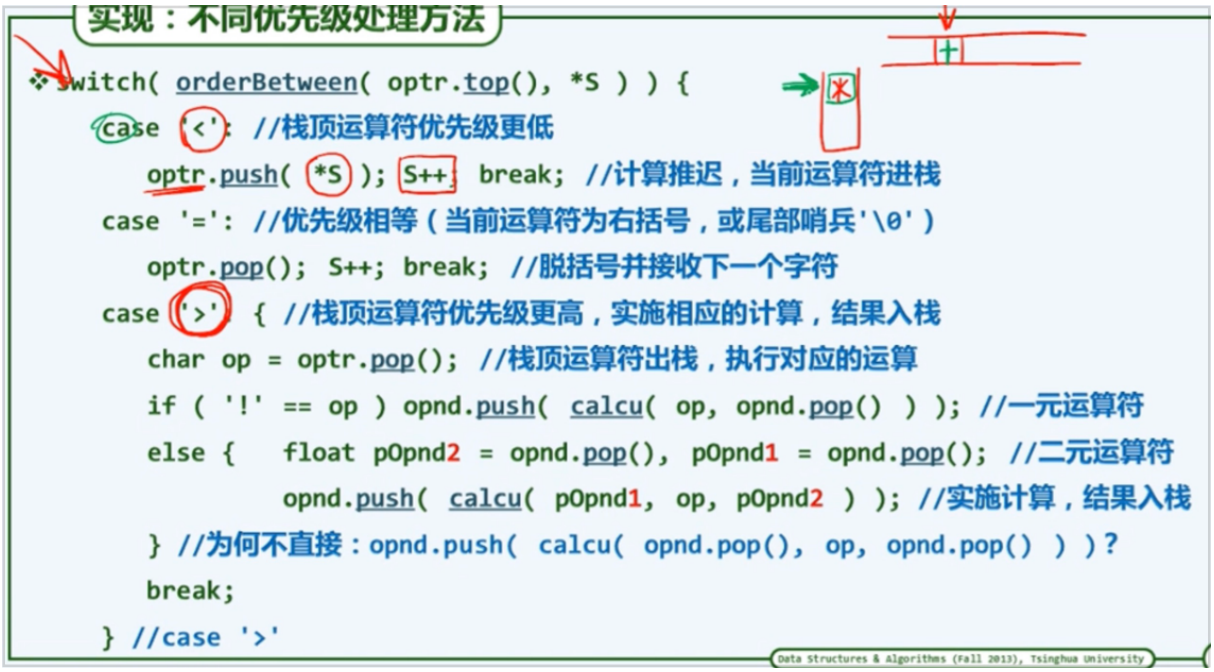
我们很难自然检测出需要的表达式，比如说 2*3 或者 10/5
那么应该如何处理呢，就是把运算符号和数值区别对待
也就是两个栈结构

算法框架


```
float evaluate(char * S,char* & RPN){
    stack<float> opnd;
    stack<char> optr;
    optr.push('\0');
    while(!optr.empty()){
        if (isdigit(*S)){
            readNumer(S,opnd);
        }
        else {
            switch(orderBetween(optr.top(),*S)){
                /* 分别处理 */
            }
        }
    }
    return opnd.top();
}
```

如何判定优先级关系：表格

一共四种可能 > < = ' '(空字符)

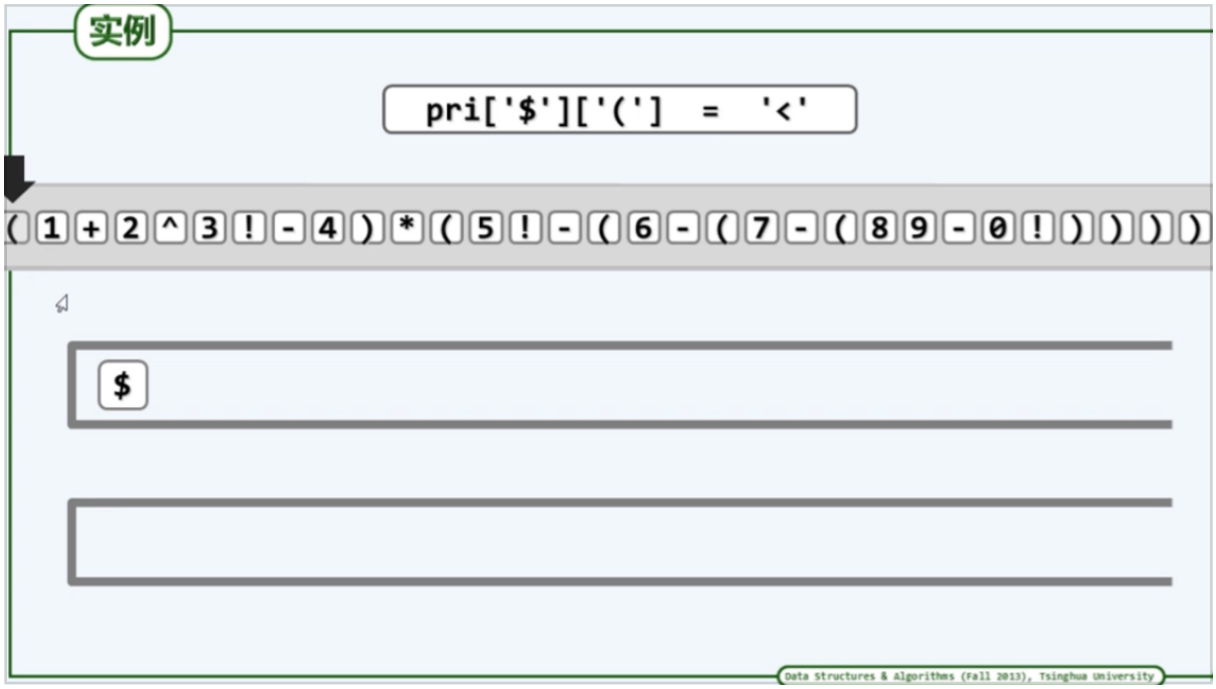


为什么不直接：会有歧义，不确定两个pop的执行次序，会导致某些非常隐蔽的问题

```
switch(orderBetween(optr.top(),*S)){
    case '<' : // 栈顶部的运算符号比扫描到的运算符号优先级低，高的先入栈，继续等待
        optr.push(*S);
        S++;
        break;
```

```
    case '=' : // 等于只有两种情况（扫描到到元素为右括号，或者尾部哨兵），我们可以将开始的\0 视为一个左括号，后面的/0，视为一个有右括号，如果两者相遇就代表完全处理完毕
        optr.pop();
        S++;
        break;
    case '>': // 如果扫描到运算符号比扫描到到运算符号优先级更高，就说明栈顶运算符可以进行运算
        char op = optr.top();
        if ('!' == op){ // 处理一元运算符
            opnd.push(calcu(op,opnd.top()));
            opnd.pop();
        }else { // 处理二元
             // 取出两个操作数
            float pOpnd2 = opnd.top();
            opnd.pop();
            float pOpnd1 = opnd.top();
            opnd.pop();
            // 将操作数结果回到栈中
            opnd.push(calcu(pOpnd1,op,pOpnd2));
        }
        break;
}
```

实例



是一种线性扫描的过程