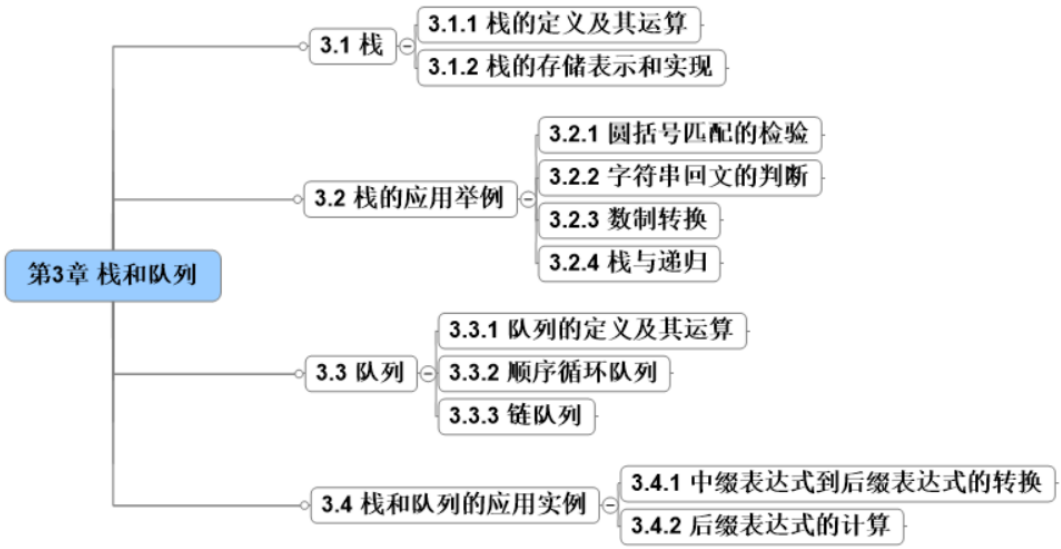
数据结构第四节课官方笔记

目录

一、课件下载及重播方法

二、教材结构图



三、本章知识点及考频总结

（一）选择题（共 4 道）

1. 栈（Stack)是限定在表的一端进行插入和删除运算的线性表，通常将插入、删除的一端称为栈顶（top），另一端称为栈底（bottom）。不含元素的空表称为空栈。

根据上述栈的定义，每次删除（退栈）的总是当前栈中最后插入（进栈）的元素，而最先进栈的元素在栈底，要到最后才能删除。假设栈 S=(a_1, a_2, … a_n)，若栈中元素按 a_1,

a_2, … a_n 的次序进栈，其中 a_1 为栈底元素，a_n 为栈顶元素，而退栈的次序却是 a_n, a_{n-1}, … a_1 。也就是说，栈的修改是按后进先出的原则进行的。因此，栈又称为**后进先出**（Last In First Out)的线性表，简称为 LIFO 表。

2. 设 S 是 SeqStack 类型的顺序栈。S.data[0]是栈底元素，那么栈顶 S.data[top]是正向增长的，即进栈时需将 S.top 加 1，退找时需将 S.top 减 1。因此，S.top<0 表示空栈，S.top==StackSize-1 表示栈满。当栈满时再做进栈运算必定产生空间溢出，简称“上溢”；当栈空时再做退栈运算也将产生溢出，简称“下溢”。

3. 顺序栈基本运算的实现：

（1）置空栈

```
void InitStack ( SeqStack * S )
{
        S->top=-1 ;
 }
```

（2）判栈空

```
int StackEmpty ( SeqStack *S )
{
        return S->top==-1 ;
}
```

（3）判栈满

```
int StackFull ( SeqStack *S )
{
        return S->top==StackSize-1 ;
```

```
 }
```

(4) 进栈（入栈）

```
 void Push ( SeqStack * S, DataType x )
 {
        if ( StackFull ( S ) )
        printf ( "stack overflow" ) ;
        else {
              S->top=S->top+1 ;          //栈顶指针加 1
              S->data[S->top]=x ;        //将 x 入栈
        }
}
```

(5) 退栈（出栈）

```
 DataType Pop ( SeqStack *S )
 {
         if ( StackEmpty ( S ) ) {
              printf ( "stack underflow" ) ;
              exit ( 0 );                  //出错退出处理
          }
         else
              return S->data [ S->top--] ;          //返回栈顶元素后栈顶指针减 1
  }
```

(6) 取栈顶元素（不改变栈顶指针）

```
DataType GetTop ( SeqStack *S )
{
    if ( StackEmpty ( S ) ) {
        printf ( "stack empty" ) ;
        exit ( 0 ) ;                    //出错退出处理
    }
    else
        return S->data[S->top] ;        //返回栈顶元素
    }
```

4. 栈的链式存储结构及基本操作

（1）判栈空

```
int StackEmpty ( LinkStack top )
{
    return top==NULL ;
}
```

（2）进找（入栈）

```
LinkStack Push ( LinkStack top , DataType x )
{    //将元素 x 插入栈顶
    StackNode * p ;
    p=(StackNode * ) malloc ( sizeof ( StackNode ) ) ; //申请新结点
    p->data=x ;
    p->next=top ;        //将新结点*p 插入栈顶
```

```
        top=p ;                  //使 top 指向新的栈顶
        return    top;                        //返回新栈顶指针
}
```

（3）退栈（出栈）

```
LinkStack Pop ( LinkStack top, DataType *x )
{     StackNode * p=top ;              //保存栈顶指针
      if ( StackEmpty ( top ) ) {
                printf ( “stack empty") ;          //栈为空
                exit ( 0 ) ; }                     //出错退出处理
      else {
                *x=p->data ;                //保存删除结点值，并带回
                top=p->next ;                    //栈顶指针指向下一个结点
                free ( p ) ;                //删除 P 指向的结点
                return top ;                //并返删除后的栈顶指针
       }
}
```

（4）取栈顶元素

```
DataType GetTop ( LinkStack top )
{
        if ( StackEmpty ( top ) ) {
                printf ( "stack empty" ) ;          //栈为空
                exit ( 0 ) ;                       //出错退出处理
```

```
    }
    else
        return top->data ;                    //返回找顶结点值
}
```

（二）主观题（共 0 道）

四、配套练习题

1、下到选项中，不宜通过栈求解的问题是（　　）。

A: 判断字符串是否是回文

B: 检验圆括号是否匹配

C: 不同数制之间进行转换

D: 图的广度优先搜索遍历

2、设栈的进栈序列为 a，b，c，d，e，经过合理的出入栈操作后，不能得到的出栈序列是（　　）。

A: d，c，e，a，b

B: d，e，c，b，a

C: a，b，c，d，e

D: e，d，c，b，a

设栈的初始状态为空，元素 1、2、3、4、5、6 依次入栈，得到的出栈序列是（2,4,3,6,5,1）,则栈的容量至少是（　　）。

A: 2

B: 3

C: 4

D: 6

[参考答案]：DAB

五、其余课程安排