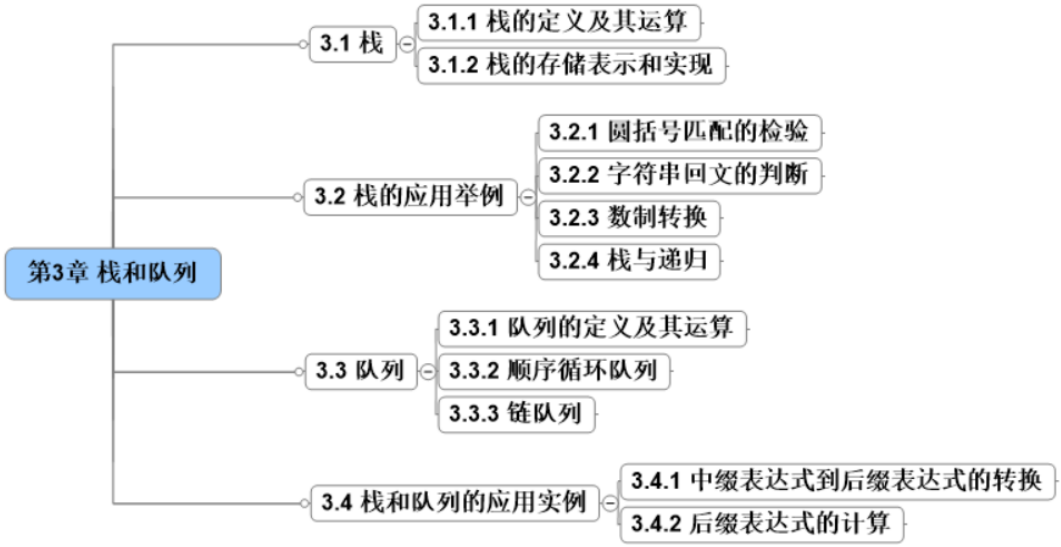
数据结构第五节课官方笔记

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. 队列（Queue）也是一种操作受限的线性表，它只允许在表的一端进行元素插入，而在另一端进行元素删除。允许插入的一端称为队尾（rear），允许删除的一端称为队头（front）。

在队列中，通常把元素的插入称为入队，而元素的删除称为出队。队列的概念与现实

生活中的排队相似，新来的成员总是加入队尾，排在队列最前面的总是最先离开队列，即先进先出，因此又称队列为先进先出（FIFO）表。

2. 队列的操作运算与栈类似，有关队列的基本运算如下：

（1）置空队列　InitQueue(Q)，构造一个空队列 Q。

（2）判队空　QueueEmpty(Q)，若 Q 为空队列，则返回 TRUE，否则返回 FALSE。

（3）入队列　EnQueue(Q，x)，若队列不满，则将数据 x 插入到 Q 的队尾。

（4）出队列　DeQueue(Q)，若队列不空，则删除队头元素，并返回该元素。

（5）取队头　GetFmm(Q)，若队列不空，则返回队头元素。

3. 顺序循环队列的基本运算：

（1）置空队列

```
void InitQueue ( CirQueue *Q )
{
        Q->front=Q->rear=0;
}
```

（2）判队空

```
int QueueEmpty ( CirQueue *Q )
{
        return Q->rear==Q->front;
}
```

（3）判队满

```
int QueueFull ( CirQueue *Q )
{
```

```
        return (Q->rear+1) % QueueSize == Q->front;
}
```

（4）入队列

```
void EnQueue ( CirQueue *Q, DataType x)
{   //插入元素 x 为队列 Q 新的队尾元素
if ( QueueFull ( Q ) )
        printf ("Queue overflow");
else {
        Q->data[Q->rear]=x;
        Q->rear=(Q->rear+1) % QueueSize;    //循环意义下的加 1
        }
}
```

（5）取队头元素

```
DataType GetFront ( CirQueue *Q )
{    //获取 Q 的队头元素值
if ( QueueEmpty ( Q ) ) {
      printf ("Queue empty");
      exit ( 0 );                    //出错退出处理
}
else
      return Q->data[Q->front]          //返回队头元素值
}
```

（6）出队列

```
DataType DeQueue ( CirQueue *Q )
{ //删除 Q 的队头元素，并返回其值
DataType x;
if ( QueueEmpty ( Q ) ) {
        printf ("Queue empty");
        exit(0);                //出错退出处理
    }
else {
        x=Q->data[Q->front];     //保存待删除元素值
        Q->front=(Q->front+1) % QueueSize; //头指针加 1
        return x;                //返回删除元素值
    }
}
```

4. 带头结点链队列的基本运算：

（1）构造空队列

```
void InitQueue (LinkQueue * Q)
{
    Q->front=( QueueNode * ) malloc ( sizeof ( QueueNode ) ) ;  //申请头结点
    Q->rear=Q->front;            //尾指针也指向头结点
    Q->rear->next=NULL;
}
```

（2）判队空

```
int QueueEmpty ( LinkQueue *Q )
{
    return Q->rear==Q->front;    //头尾指针相等队列为空
}
```

（3）入队列

```
void EnQueue ( LinkQueue * Q, DataType x )
{      //将元素 x 插入链队列尾部
       QueueNode * p=(QueueNode * ) malloc ( sizeof ( QueueNode ) )
       p->data=x;
       p->next=NULL;
       Q->rear->next=p;              //*P 链到原队尾结点之后
       Q->rear=p;                    //队尾指针指向新的队尾结点
}
```

（4）取队头元素

```
DataType GetFront ( LinkQueue * Q )
{    //取链队列的队头元素值
if ( QueueEmpty ( Q ) ) {
printf ("Queue underflow");
exit(0);                    //出错退出处理
}
else
```

return Q->front->next->data;　　//返回原队头元素值

}

（5）出队列

链队列的出队操作有两种不同情况要分别考虑：

①当队列的长度大于 1 时，则出队操作只需要修改头结点的指针域即可，尾指针不变，操作步骤如下：

s=Q->front->next;

Q->front->next=s->next;

x=s->data;

free(s);return x;

②若列队长度等于 1，则出队时不仅要修改头结点指针域，而且还需要修改尾指针。

s=Q->front->next;

Q->front->next=NULL;

Q->rear=Q->front;

x=s->data;

free(s);return x;

（二）主观题（共 0 道）

四、配套练习题

1、若用一个大小为 7 的数组作为循环队列的存储结构，且当前 rear 和 front 的值分别为 2 和 4，在此之前的操作是从队列中删除了一个元素及加入两个元素，请问这 3 个操作之前 rear 和 front 的值分别是（）

A: 0 和 1

B: 0 和 3

C: 3 和 6

D: 4 和 5

2、下列关于队列的叙述中,错误的是（）

A: 队列是一种先进先出的线性表

B: 队列是一种后进后出的线性表

C: 循环队列中进行出队操作时要判断队列是否为空

D: 在链队列中进行入队操作时要判断队列是否为满

3、已知循环队列的存储空间大小为 m，队头指针 front 指向队头元素，队尾指针 rear 指向队尾元素的下一个位置，则向队列中插入新元素时，修改指针的操作是（ ）

A: rear=(rear-1)%m;

B: front=(front+1)%m;

C: front=(front-1)%m;

D: rear=(rear+1)%m;

[参考答案]：BDD

五、其余课程安排