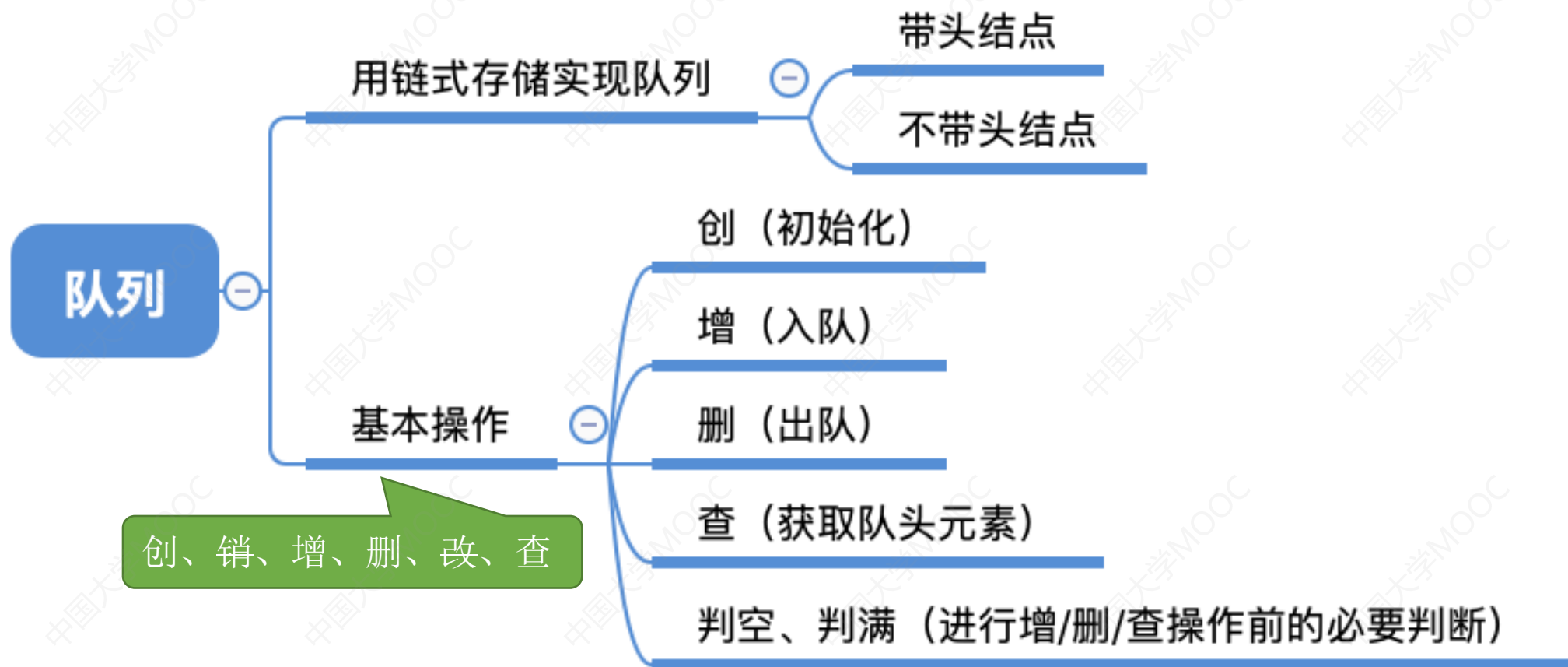


本节内容

队列

链式实现

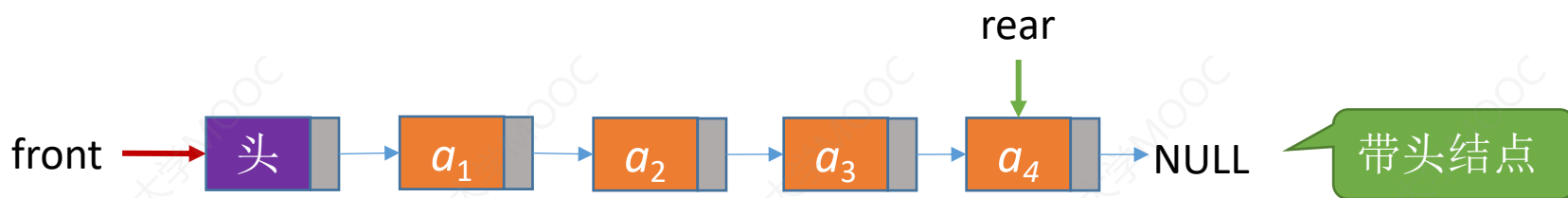
知识总览



队列的链式实现

```
typedef struct LinkNode{           //链式队列结点
    ElemType data;
    struct LinkNode *next;
}LinkNode;

typedef struct{                   //链式队列
    LinkNode *front,*rear;        //队列的队头和队尾指针
}LinkQueue;
```



初始化（带头结点）

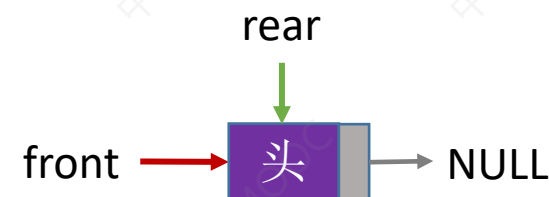
```
typedef struct LinkNode{
    ElemType data;
    struct LinkNode *next;
}LinkNode;
```

//初始化队列(带头结点)

```
void InitQueue(LinkQueue &Q){
    //初始时 front、rear 都指向头结点
    ➡ Q.front=Q.rear=(LinkNode*)malloc(sizeof(LinkNode));
    ➡ Q.front->next=NULL;
}
```

```
void testLinkQueue(){
    ➡ LinkQueue Q;    //声明一个队列
    ➡ InitQueue(Q);   //初始化队列
    //...后续操作...
}
```

```
typedef struct{
    LinkNode *front,*rear;
}LinkQueue;
```



带头结点空队列

//判断队列是否为空

```
bool IsEmpty(LinkQueue Q){
    if(Q.front==Q.rear)
        return true;
    else
        return false;
}
```

初始化（不带头结点）

//初始化队列(不带头结点)

```
void InitQueue(LinkQueue &Q){
```

//初始时 *front*、*rear* 都指向NULL

→ Q.*front*=NULL;

→ Q.*rear*=NULL;

```
}
```

//判断队列是否为空（不带头结点）

```
bool IsEmpty(LinkQueue Q){
```

if(Q.*front*==NULL)

return true;

else

return false;

```
}
```

rear → NULL

front → NULL

不带头结点的空队列

入队（带头结点）

//新元素入队（带头结点）

```
void EnQueue(LinkQueue &Q, ElemType x){
```

```
    → LinkNode *s=(LinkNode *)malloc(sizeof(LinkNode));
```

```
    → s->data=x;
```

```
    → s->next=NULL;
```

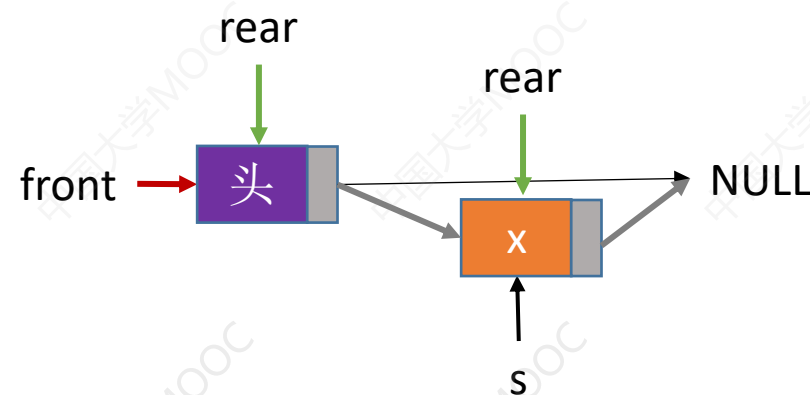
```
    → Q.rear->next=s;
```

```
    → Q.rear=s;
```

```
}
```

//新结点插入到rear之后

//修改表尾指针



入队（不带头结点）

//新元素入队（不带头结点）

```
void EnQueue(LinkQueue &Q, ElemType x){
```

```
    LinkNode *s=(LinkNode *)malloc(sizeof(LinkNode));
```

```
    s->data=x;
```

```
    s->next=NULL;
```

```
    if (Q.front == NULL){
```

```
        Q.front = s;
```

```
        Q.rear=s;
```

```
    } else {
```

```
        Q.rear->next=s;
```

```
        Q.rear=s;
```

```
    }
```

```
}
```

//在空队列中插入第一个元素

//修改队头队尾指针

不带头结点的队列，第一个元素入队时需要特别处理

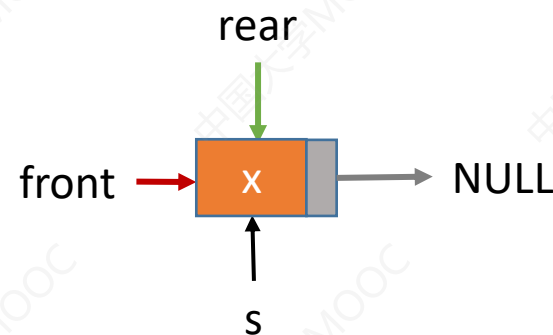
//新结点插入到 rear 结点之后

//修改 rear 指针

rear → NULL

front → NULL

不带头结点的空队列



出队（带头结点）

//队头元素出队（不带头结点）

```
bool DeQueue(LinkQueue &Q, ElemType &x){
```

```
    if(Q.front==Q.rear)
```

```
        return false;    //空队
```

```
    LinkNode *p=Q.front->next;
```

```
    x=p->data;    //用变量x返回队头元素
```

```
    Q.front->next=p->next;    //修改头结点的 next 指针
```

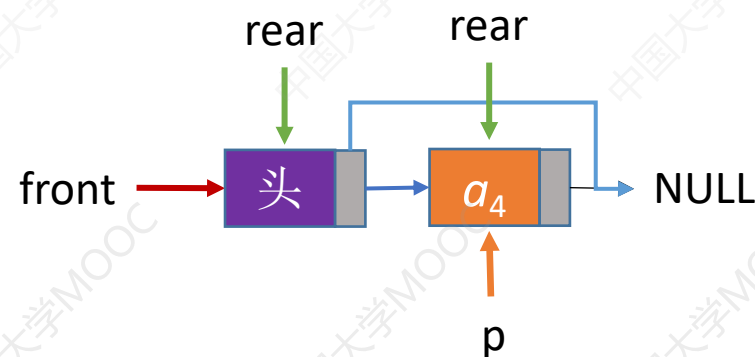
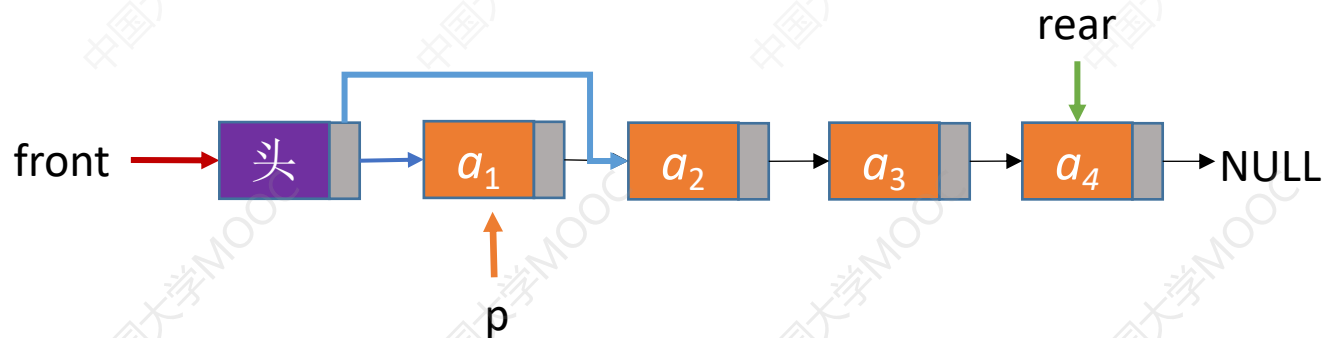
```
    if(Q.rear==p)    //此次是最后一个结点出队
```

```
        Q.rear=Q.front;    //修改 rear 指针
```

```
    free(p);    //释放结点空间
```

```
    return true;
```

```
}
```



出队（不带头结点）

//队头元素出队（不带头结点）

```
bool DeQueue(LinkQueue &Q, ElemType &x){
```

```
    if(Q.front==NULL)
```

```
        return false;
```

```
    LinkNode *p=Q.front;
```

```
    x=p->data;
```

```
    Q.front=p->next;
```

```
    if(Q.rear==p){
```

```
        Q.front = NULL;
```

```
        Q.rear = NULL;
```

```
    }
```

```
    free(p);
```

```
    return true;
```

```
}
```

//空队

//p指向此次出队的结点

//用变量x返回队头元素

//修改 front 指针

//此次是最后一个结点出队

//front 指向 NULL

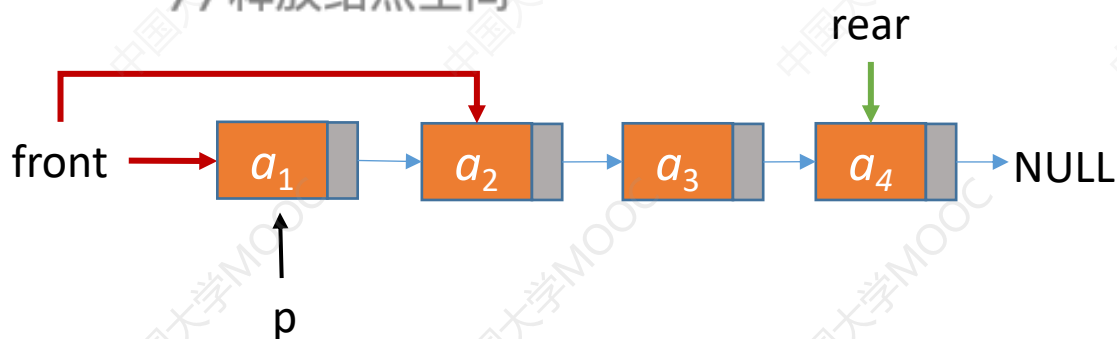
//rear 指向 NULL

//释放结点空间

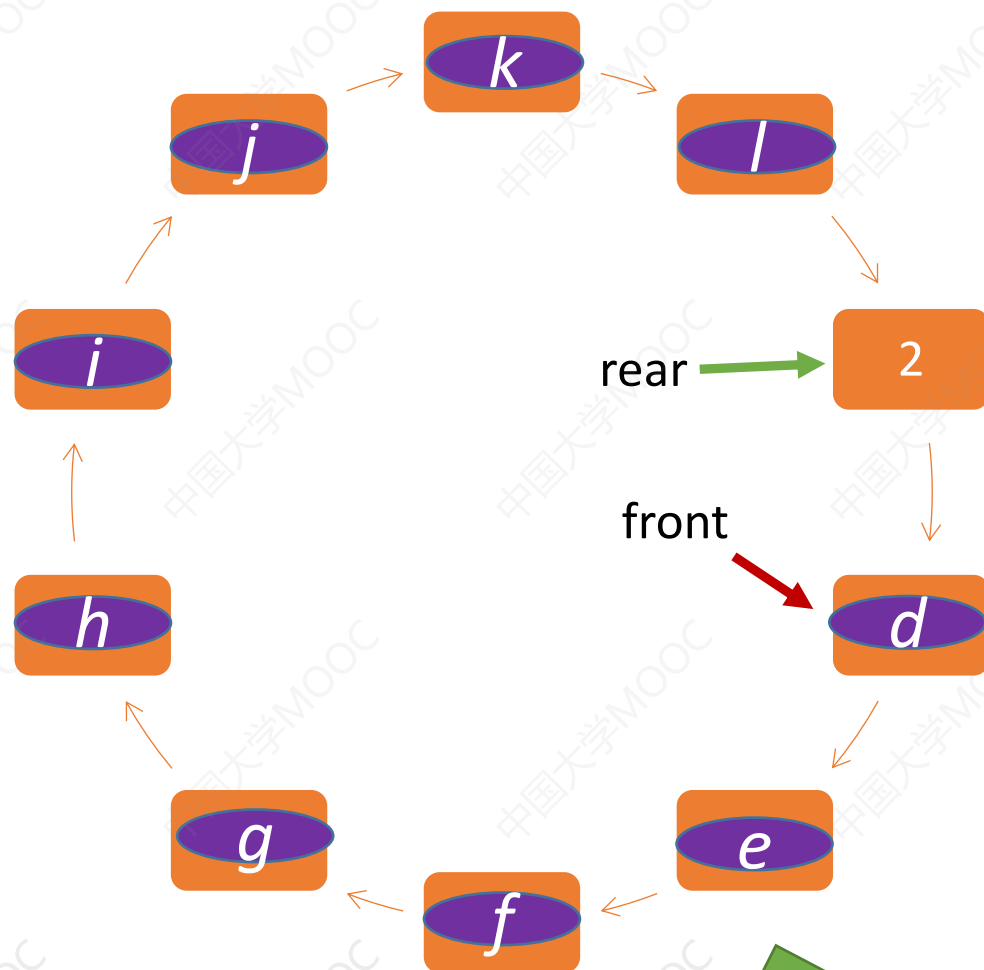
rear → NULL

front → NULL

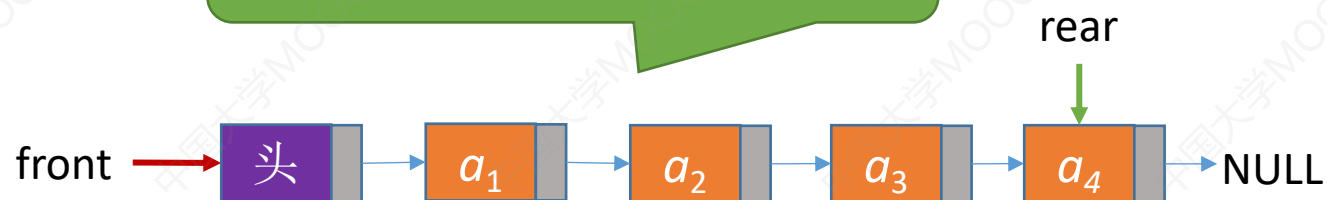
不带头结点的空队列



队列满的条件

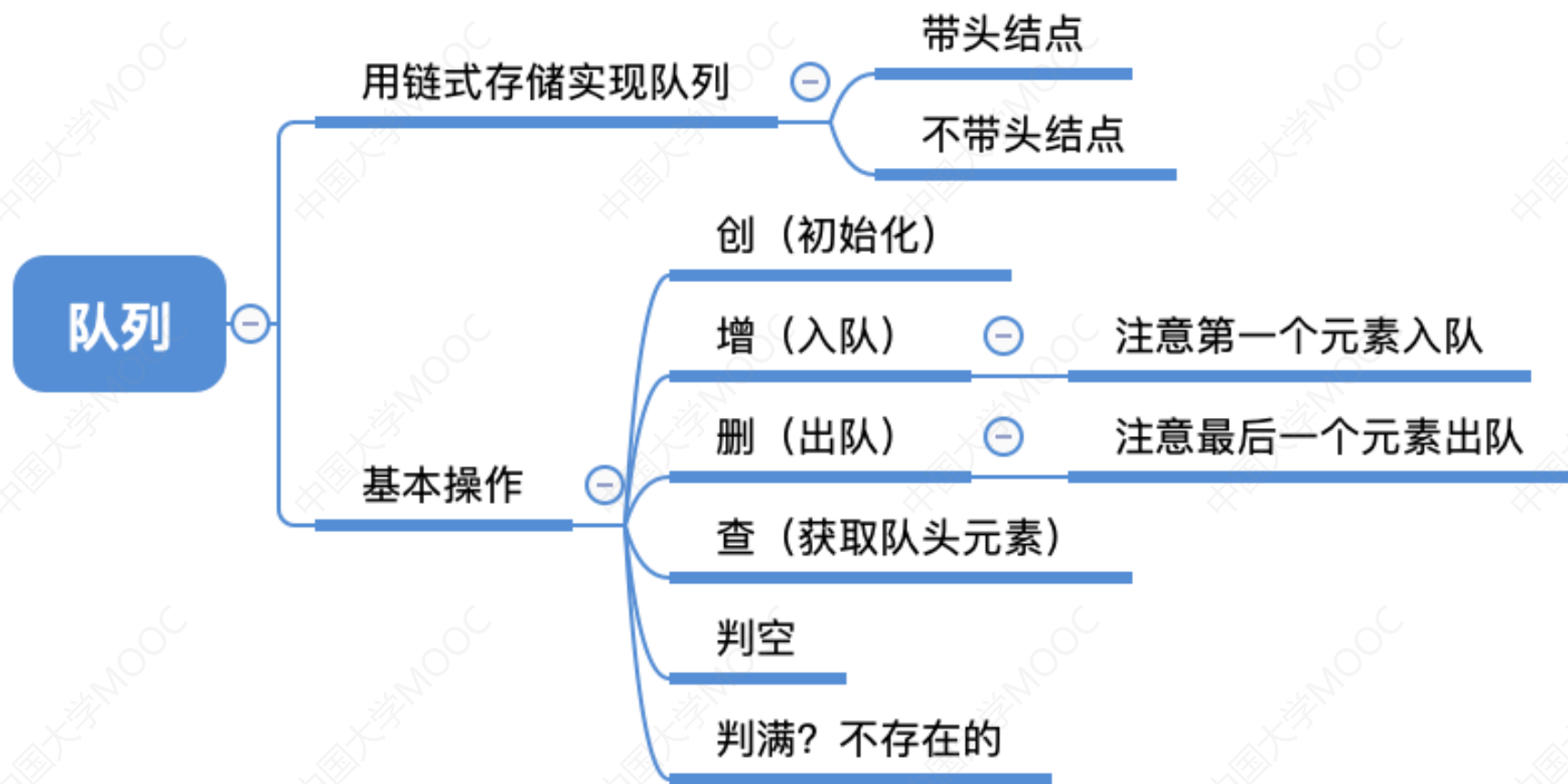


链式存储——一般不会队满，除非内存不足



顺序存储——预分配的空间耗尽时队满

知识回顾与重要考点



```
typedef struct LinkNode{  
    ElemType data;  
    struct LinkNode *next;  
}LinkNode;
```

```
typedef struct{  
    LinkNode *front,*rear;  
}LinkQueue;
```

欢迎大家对本节视频进行评价~



学员评分：3.2_3 队列...

扫一扫二维码打开或分享给好友



— 腾讯文档 —

可多人实时在线编辑，权限安全可控



公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研