

本节内容

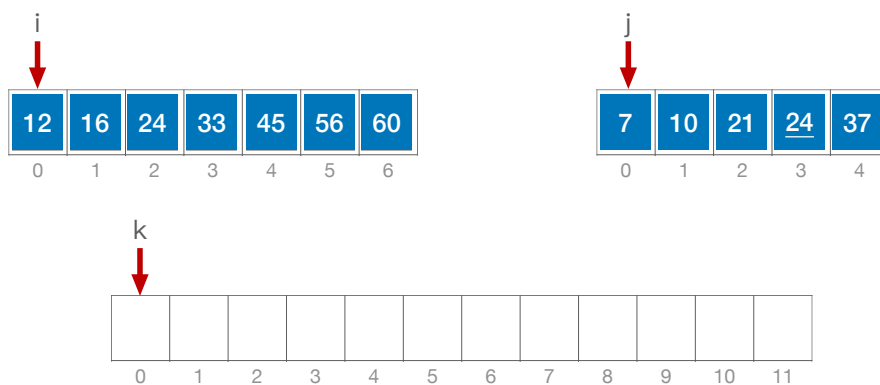
归并排序

(Merge Sort)

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?

归并：把两个或多个已经有序的序列合并成一个

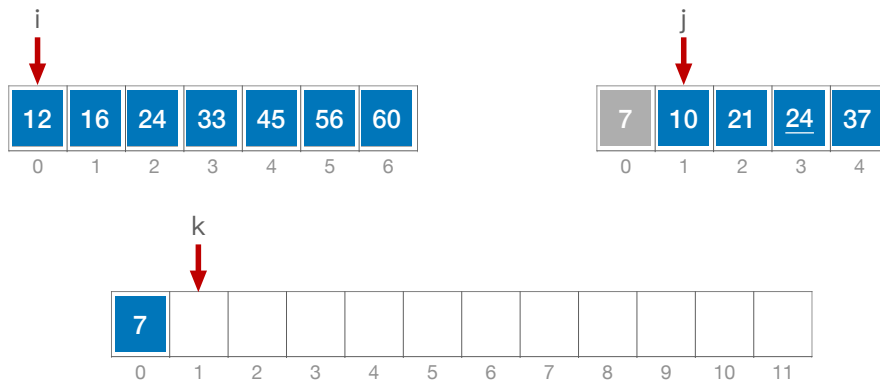


对比 i 、 j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?

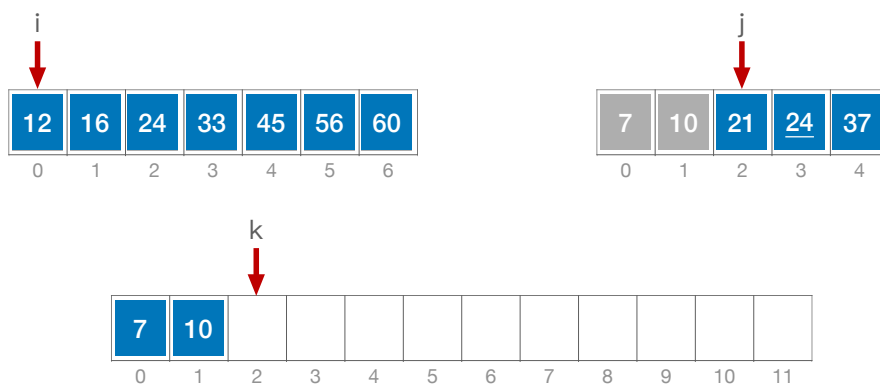
归并：把两个或多个已经有序的序列合并成一个



王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?

归并：把两个或多个已经有序的序列合并成一个

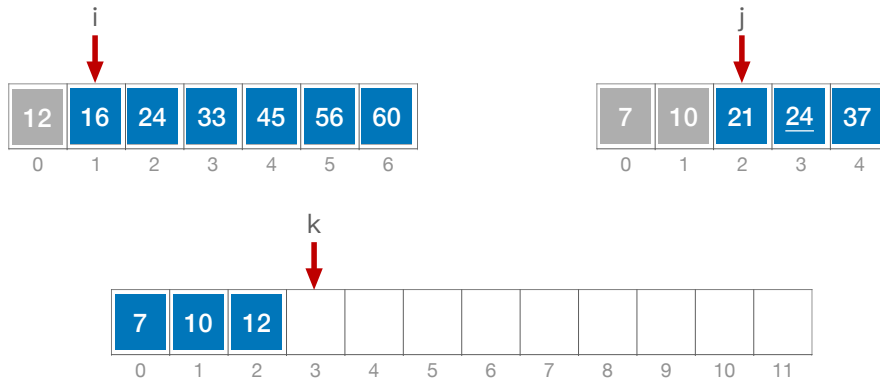


王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?



归并：把两个或多个已经有序的序列合并成一个



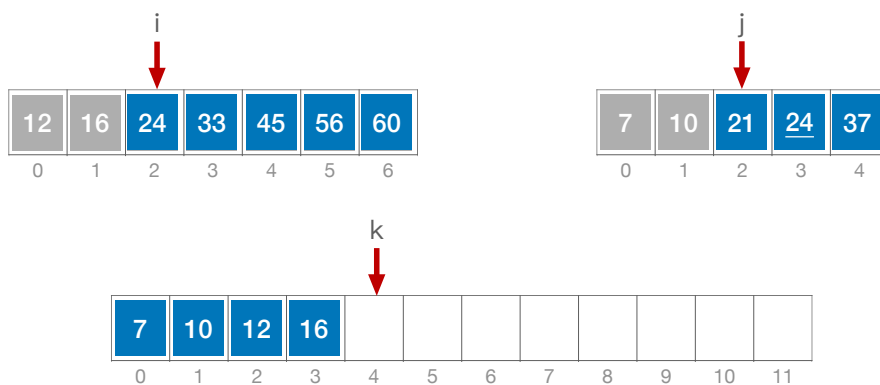
对比 i、j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?



归并：把两个或多个已经有序的序列合并成一个



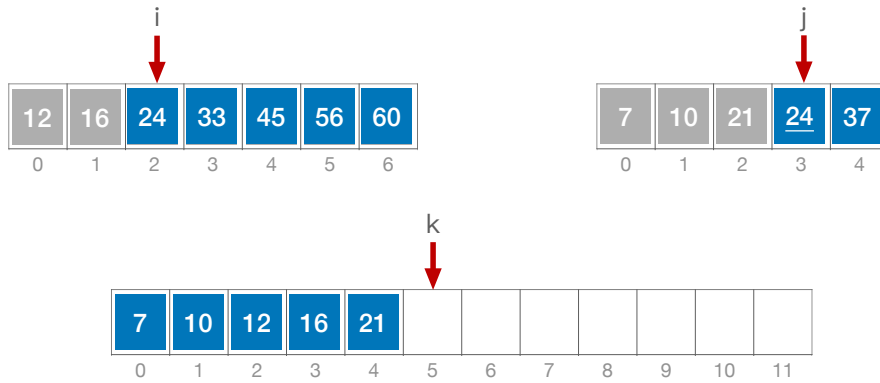
对比 i、j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?



归并：把两个或多个已经有序的序列合并成一个



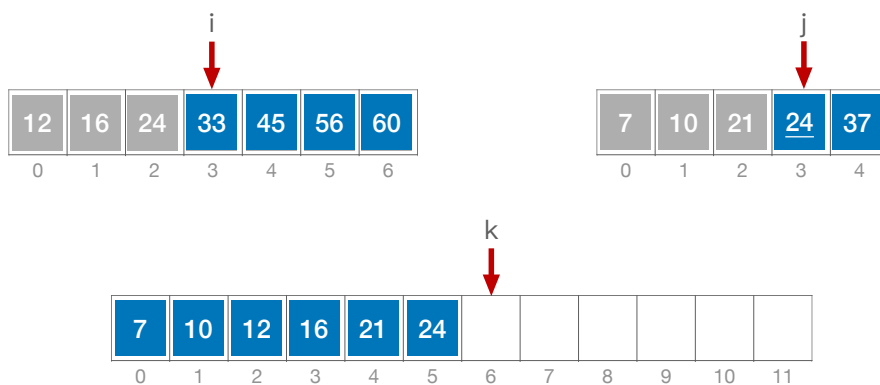
对比 i、j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?



归并：把两个或多个已经有序的序列合并成一个



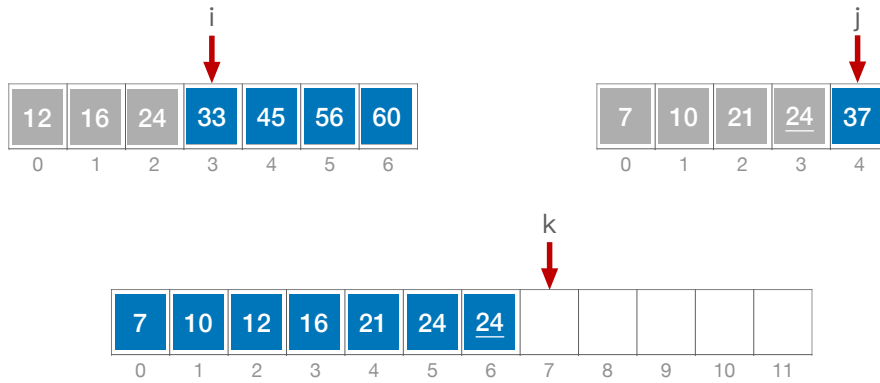
对比 i、j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?



归并：把两个或多个已经有序的序列合并成一个



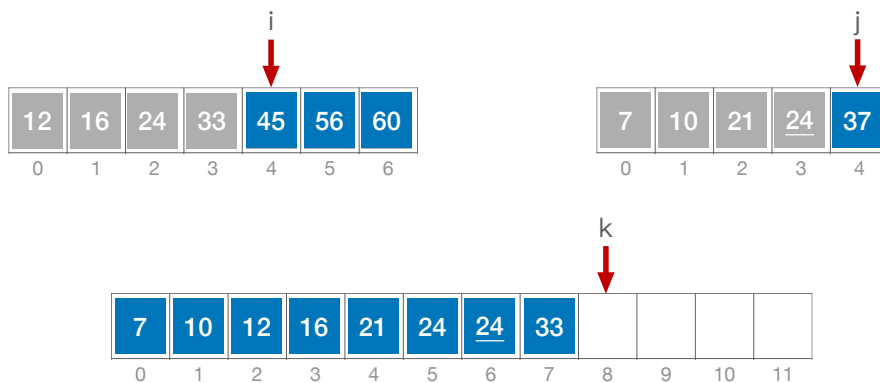
对比 i、j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?



归并：把两个或多个已经有序的序列合并成一个

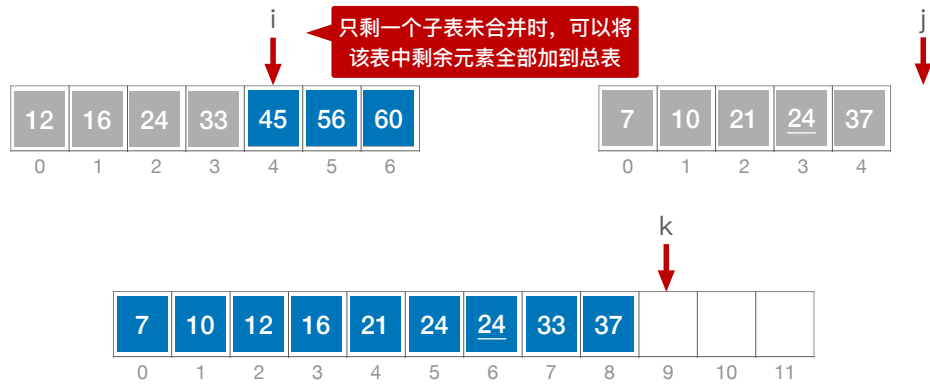


对比 i、j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?

归并：把两个或多个已经有序的序列合并成一个

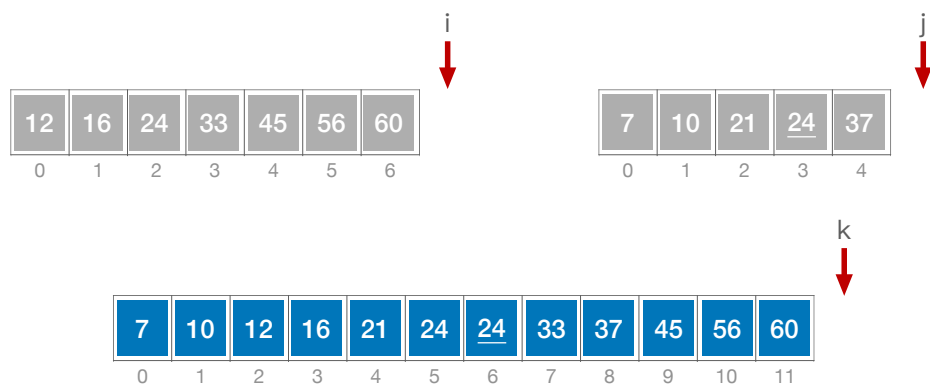


对比 i、j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

什么是 Merge (归并/合并) ?

归并：把两个或多个已经有序的序列合并成一个

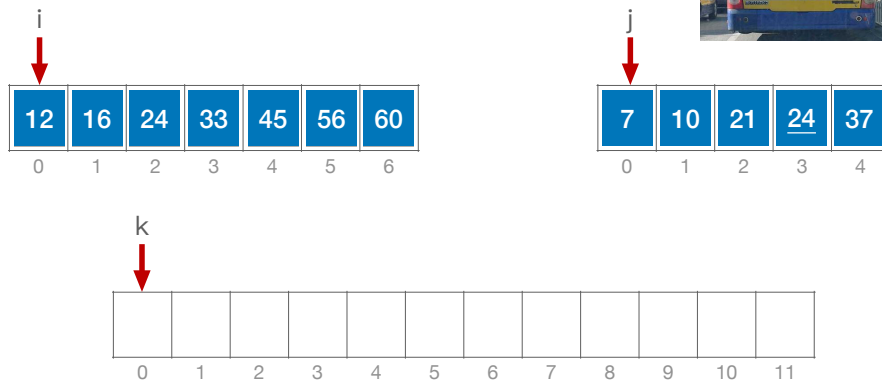


对比 i、j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

“2路”——二合一 → “2路”归并

归并：把两个或多个已经有序的序列合并成一个

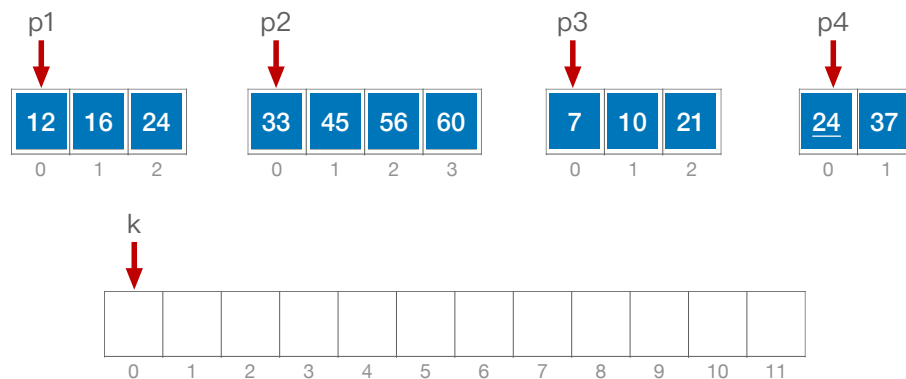


“2路”归并——每选出一个元素注需对比关键字1次
对比 i 、 j 所指元素，选择更小的一个放入 k 所指位置

王道考研/CSKAOYAN.COM

“4路”——四合一 → “4路”归并

归并：把两个或多个已经有序的序列合并成一个

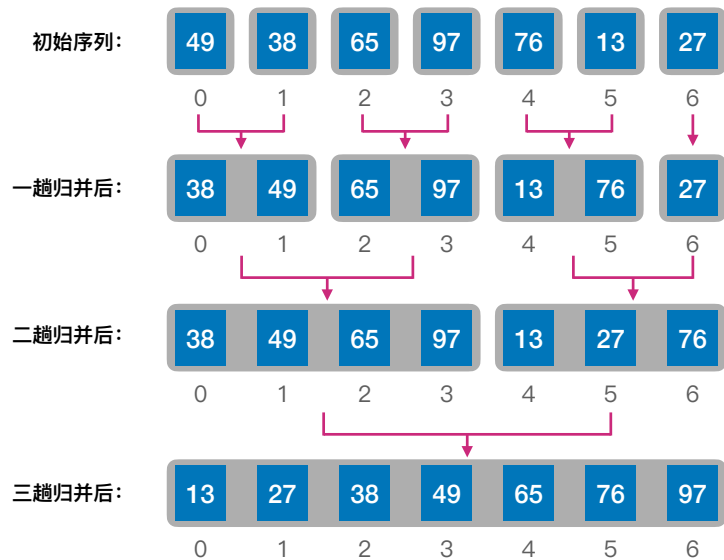


“4路”归并——每选出一个元素注需对比关键字3次
对比 $p1$ 、 $p2$ 、 $p3$ 、 $p4$ 所指元素，选择更小的一个放入 k 所指位置
结论： m 路归并，每选出一个元素需要对比关键字 $m-1$ 次

王道考研/CSKAOYAN.COM

在内部排序中一般采用2路归并

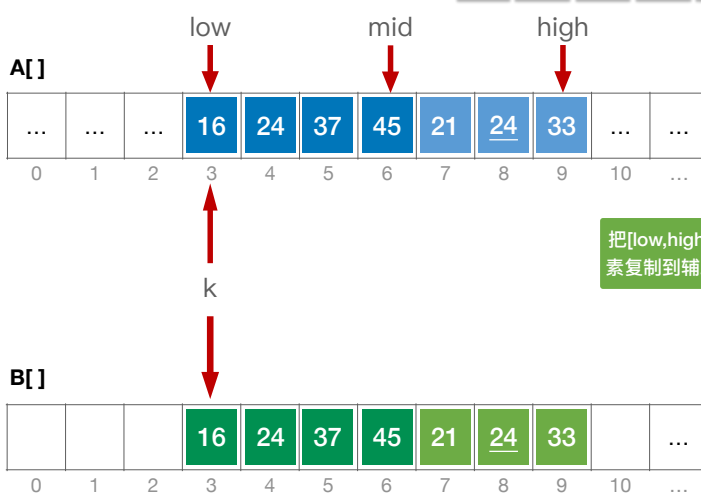
归并排序（手算模拟）



核心操作：把数组内的两个有序序列归并为一个

王道考研/CSKAOYAN.COM

代码实现

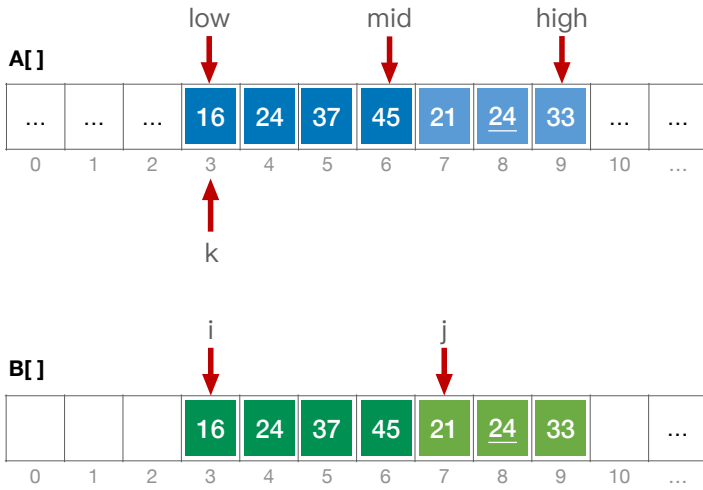


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B

//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    }
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}
```

王道考研/CSKAOYAN.COM

代码实现

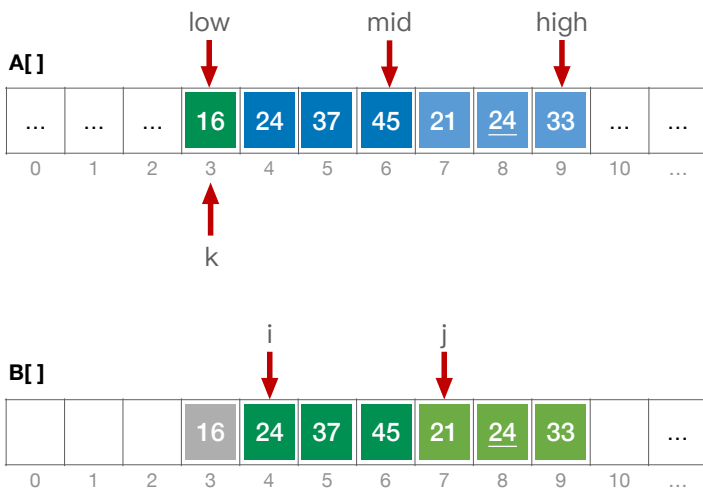


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B

//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    归并 for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}
```

王道考研/CSKAOYAN.COM

代码实现

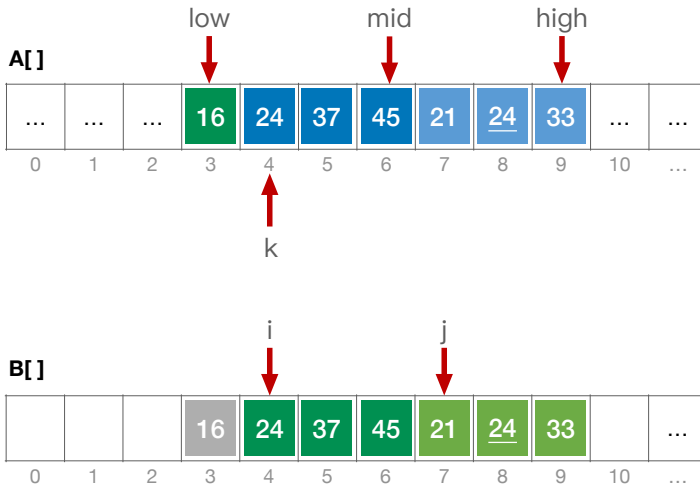


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B

//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    归并 for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}
```

王道考研/CSKAOYAN.COM

代码实现

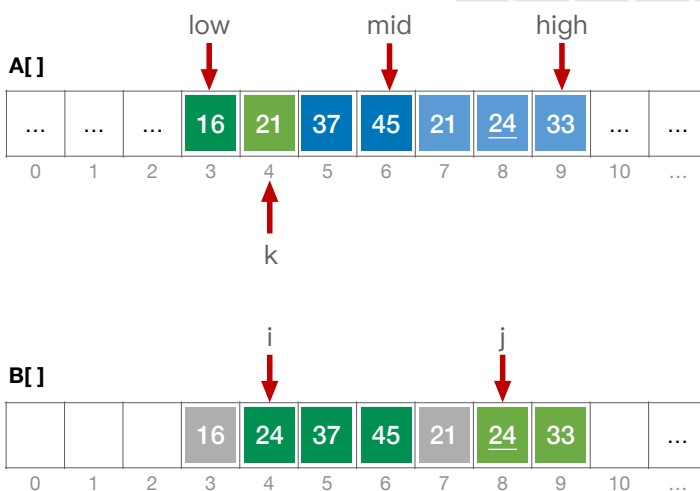


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B

//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    }
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}
```

王道考研/CSKAOYAN.COM

代码实现

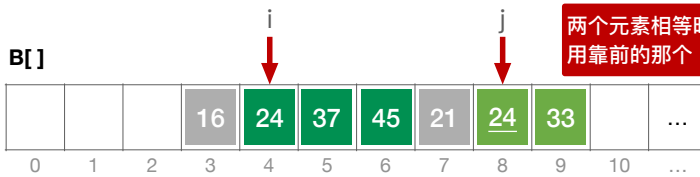
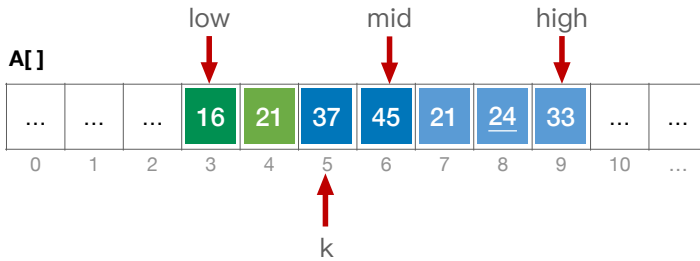


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B

//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    }
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}
```

王道考研/CSKAOYAN.COM

代码实现



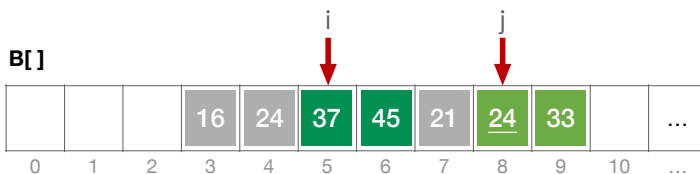
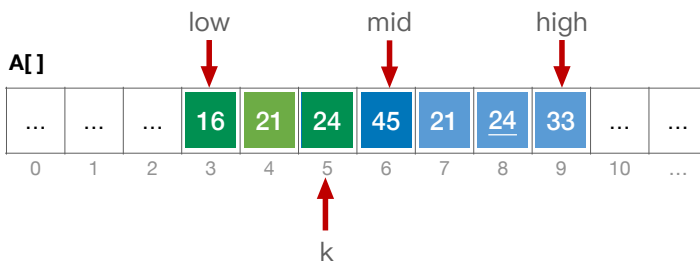
两个元素相等时，优先使用靠前的那个（稳定性）

```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B

//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    }
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}
```

王道考研/CSKAOYAN.COM

代码实现

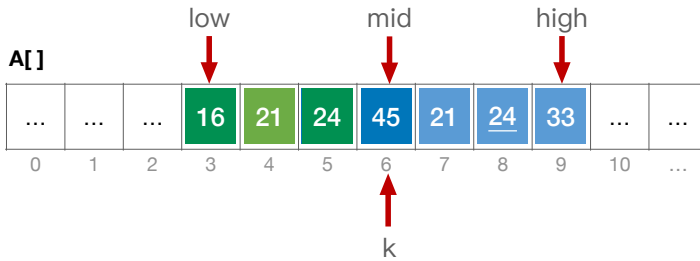


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B

//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    }
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}
```

王道考研/CSKAOYAN.COM

代码实现



```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
```

```
//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
```

```
void Merge(int A[],int low,int mid,int high){
```

```
int i,j,k;
```

```
for(k=low;k<=high;k++){
```

```
B[k]=A[k]; //将A中所有元素复制到B中
```

归并

```
for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
```

```
if(B[i]<=B[j])
```

```
A[k]=B[i++]; //将较小值复制到A中
```

```
else
```

```
A[k]=B[j++];
```

```
}//for
```

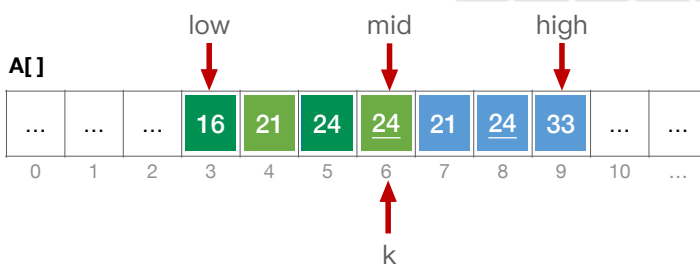
```
while(i<=mid) A[k++]=B[i++];
```

```
while(j<=high) A[k++]=B[j++];
```

```
}
```

王道考研/CSKAOYAN.COM

代码实现



```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
```

```
//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
```

```
void Merge(int A[],int low,int mid,int high){
```

```
int i,j,k;
```

```
for(k=low;k<=high;k++){
```

```
B[k]=A[k]; //将A中所有元素复制到B中
```

归并

```
for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
```

```
if(B[i]<=B[j])
```

```
A[k]=B[i++]; //将较小值复制到A中
```

```
else
```

```
A[k]=B[j++];
```

```
}//for
```

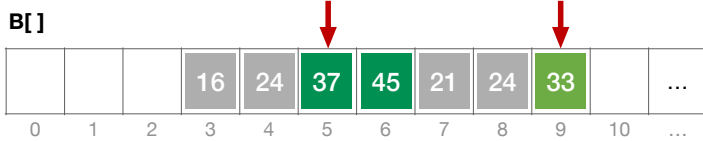
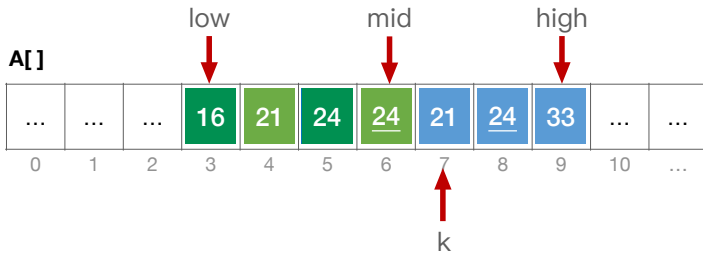
```
while(i<=mid) A[k++]=B[i++];
```

```
while(j<=high) A[k++]=B[j++];
```

```
}
```

王道考研/CSKAOYAN.COM

代码实现



```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
```

```
//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
```

```
void Merge(int A[],int low,int mid,int high){
```

```
int i,j,k;
```

```
for(k=low;k<=high;k++){
```

```
B[k]=A[k]; //将A中所有元素复制到B中
```

归并

```
for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
```

```
if(B[i]<=B[j])
```

```
A[k]=B[i++]; //将较小值复制到A中
```

```
else
```

```
A[k]=B[j++];
```

```
}//for
```

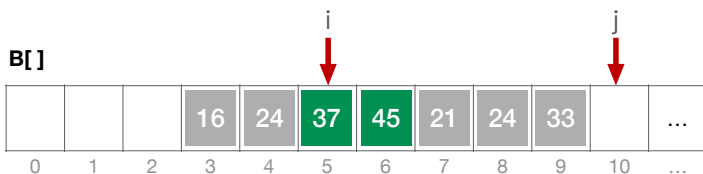
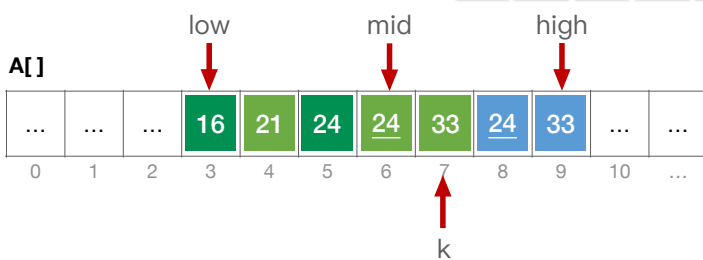
```
while(i<=mid) A[k++]=B[i++];
```

```
while(j<=high) A[k++]=B[j++];
```

```
}
```

王道考研/CSKAOYAN.COM

代码实现



```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
```

```
//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
```

```
void Merge(int A[],int low,int mid,int high){
```

```
int i,j,k;
```

```
for(k=low;k<=high;k++){
```

```
B[k]=A[k]; //将A中所有元素复制到B中
```

归并

```
for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
```

```
if(B[i]<=B[j])
```

```
A[k]=B[i++]; //将较小值复制到A中
```

```
else
```

```
A[k]=B[j++];
```

```
}//for
```

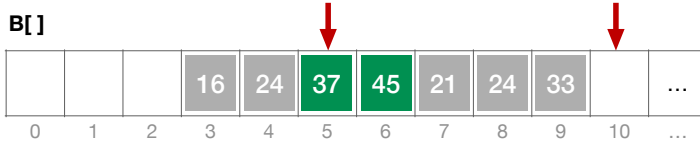
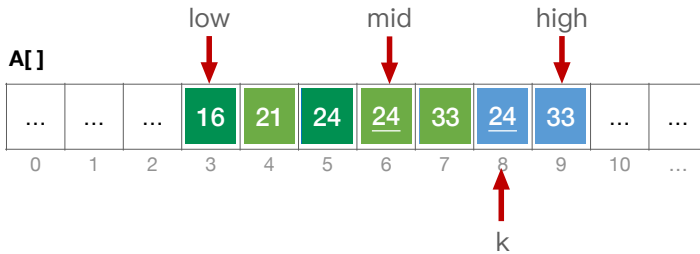
```
while(i<=mid) A[k++]=B[i++];
```

```
while(j<=high) A[k++]=B[j++];
```

```
}
```

王道考研/CSKAOYAN.COM

代码实现



```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
```

```
//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
```

```
void Merge(int A[],int low,int mid,int high){
```

```
int i,j,k;
```

```
for(k=low;k<=high;k++){
```

```
B[k]=A[k]; //将A中所有元素复制到B中
```

归并

```
for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
```

```
if(B[i]<=B[j])
```

```
A[k]=B[i++]; //将较小值复制到A中
```

```
else
```

```
A[k]=B[j++];
```

```
}//for
```

```
while(i<=mid) A[k++]=B[i++];
```

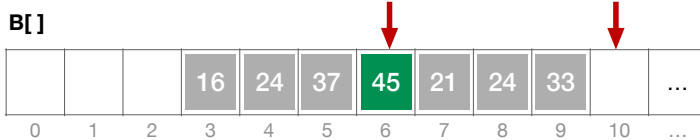
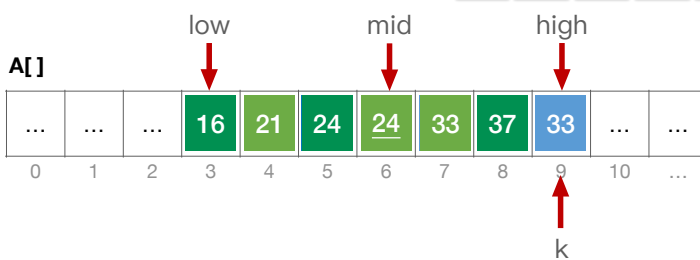
```
while(j<=high) A[k++]=B[j++];
```

```
}
```

没有归并完的部分复制到尾部

王道考研/CSKAOYAN.COM

代码实现



```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
```

```
//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
```

```
void Merge(int A[],int low,int mid,int high){
```

```
int i,j,k;
```

```
for(k=low;k<=high;k++){
```

```
B[k]=A[k]; //将A中所有元素复制到B中
```

归并

```
for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
```

```
if(B[i]<=B[j])
```

```
A[k]=B[i++]; //将较小值复制到A中
```

```
else
```

```
A[k]=B[j++];
```

```
}//for
```

```
while(i<=mid) A[k++]=B[i++];
```

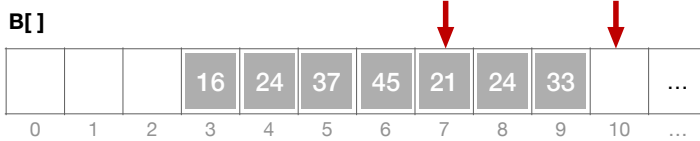
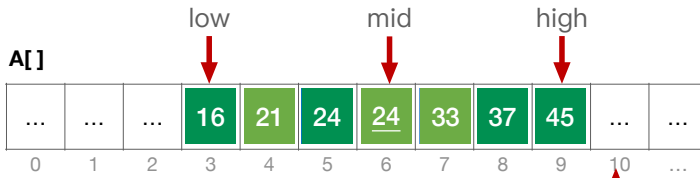
```
while(j<=high) A[k++]=B[j++];
```

```
}
```

没有归并完的部分复制到尾部

王道考研/CSKAOYAN.COM

代码实现



```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
```

```
//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
```

```
void Merge(int A[],int low,int mid,int high){
```

```
int i,j,k;
```

```
for(k=low;k<=high;k++)
```

```
B[k]=A[k]; //将A中所有元素复制到B中
```

归并

```
for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
```

```
if(B[i]<=B[j])
```

```
A[k]=B[i++]; //将较小值复制到A中
```

```
else
```

```
A[k]=B[j++];
```

```
}//for
```

```
while(i<=mid) A[k++]=B[i++];
```

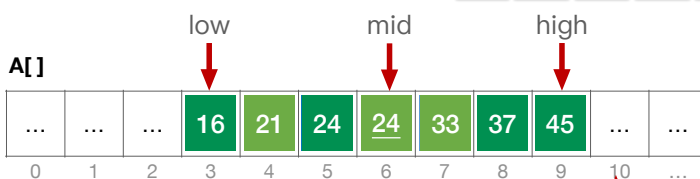
```
while(j<=high) A[k++]=B[j++];
```

```
}
```

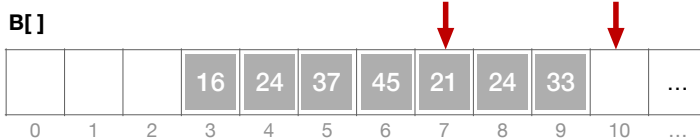
没有归并完的部分复制到尾部

王道考研/CSKAOYAN.COM

代码实现



经过 Merge 处理,
A[low~high]整体有序



```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
```

```
//A[low..mid]和A[mid+1..high]各自有序，将两个部分归并
```

```
void Merge(int A[],int low,int mid,int high){
```

```
int i,j,k;
```

```
for(k=low;k<=high;k++)
```

```
B[k]=A[k]; //将A中所有元素复制到B中
```

归并

```
for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
```

```
if(B[i]<=B[j])
```

```
A[k]=B[i++]; //将较小值复制到A中
```

```
else
```

```
A[k]=B[j++];
```

```
}//for
```

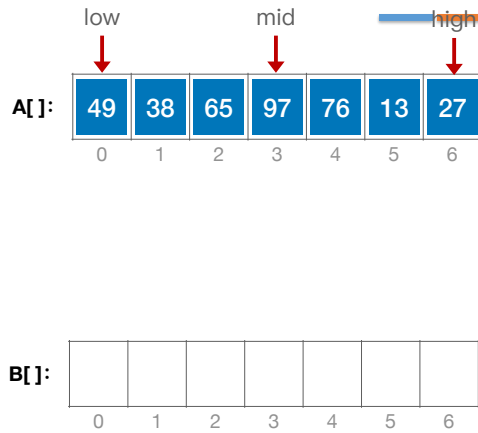
```
while(i<=mid) A[k++]=B[i++];
```

```
while(j<=high) A[k++]=B[j++];
```

```
}
```

王道考研/CSKAOYAN.COM

代码实现

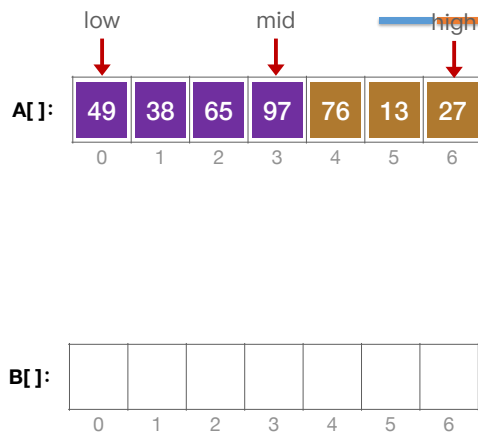


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现

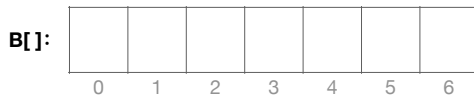
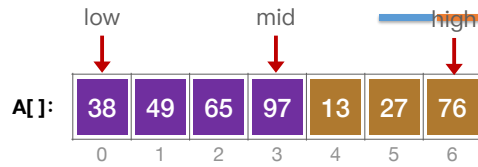


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现



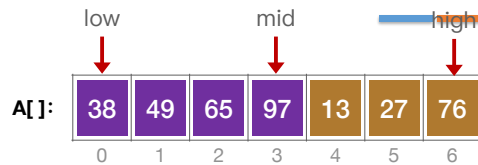
将左右两个子序列分别进行归并排序

```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现



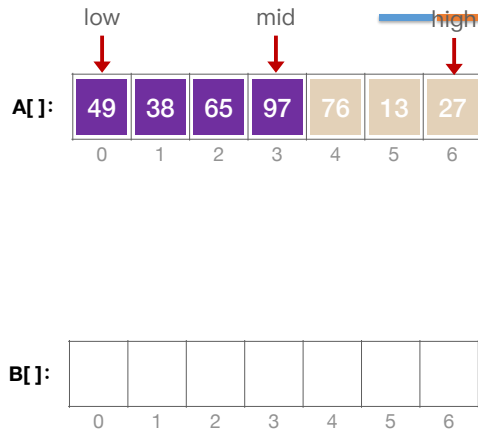
左右两个子序列分别有序之后再将二者归并

```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现

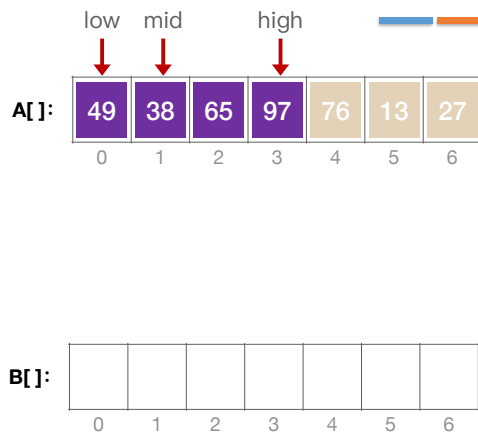


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现

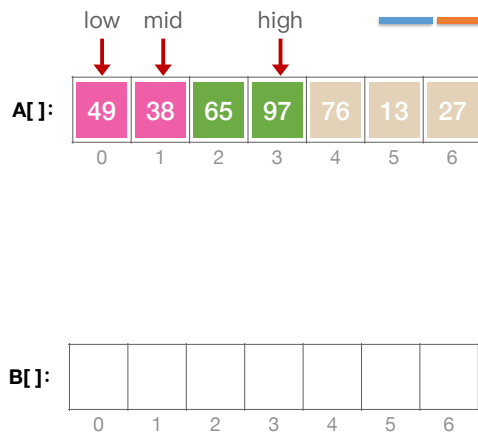


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现

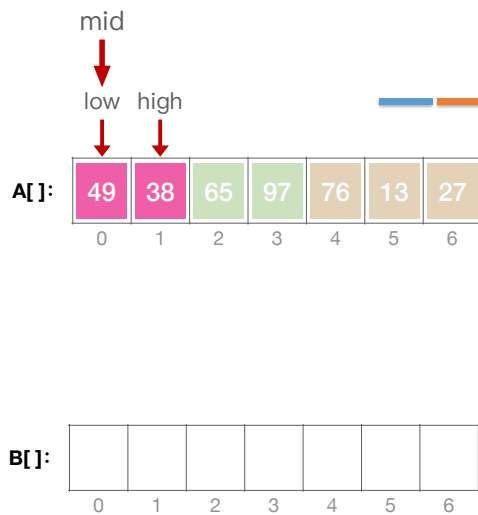


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现

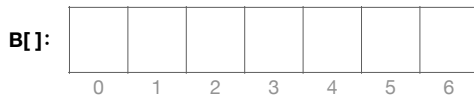
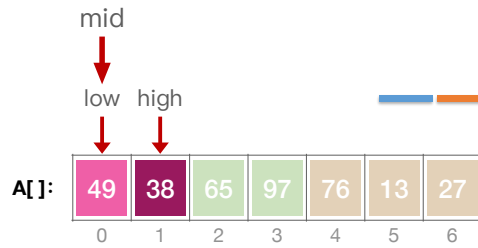


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现



禁止赛娃

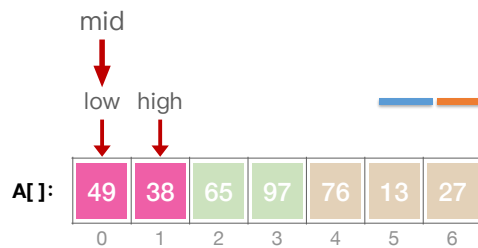
将左右两个子序列分别进行归并排序（每个子序列只含有1个元素）

```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现

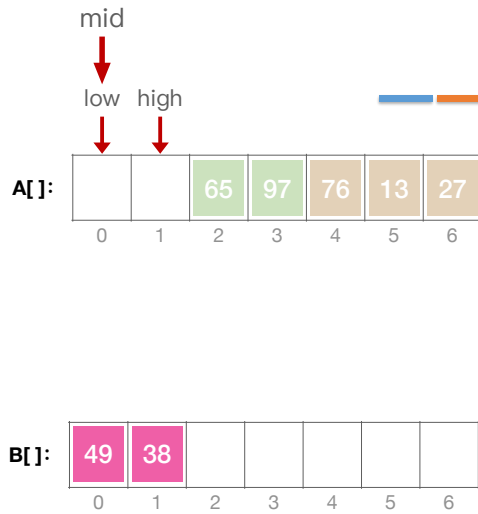


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现

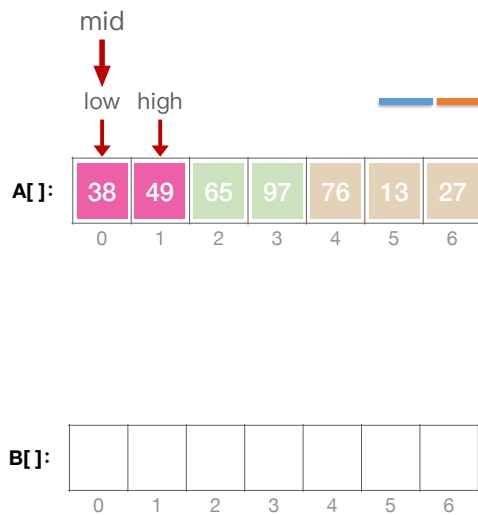


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

王道考研/CSKAQYAN.COM

代码实现

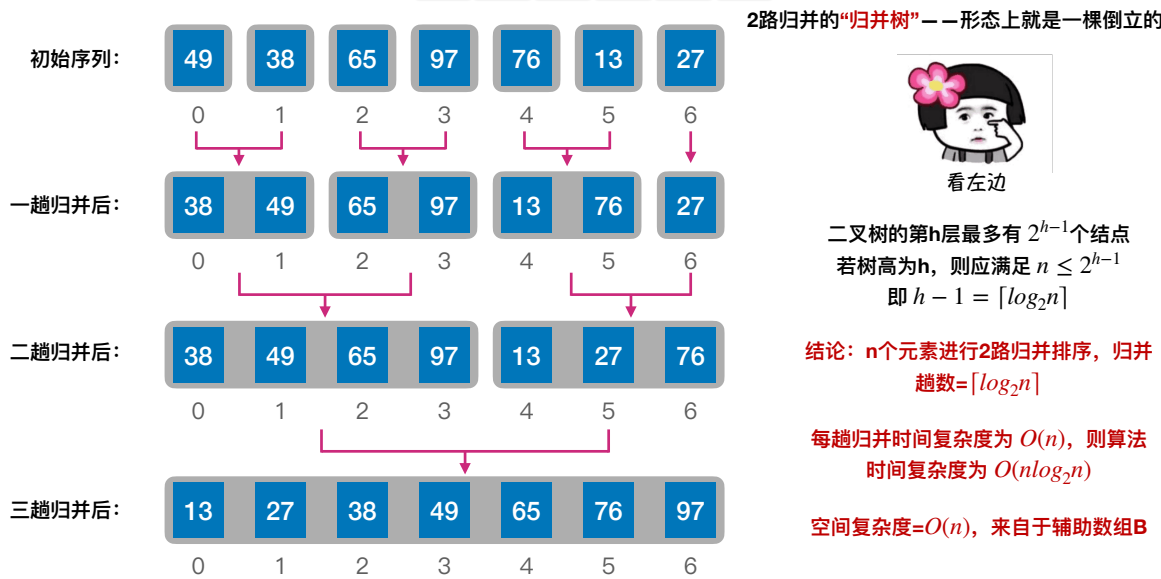


```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B
//A[low...mid]和A[mid+1...high]各自有序，将两个部分归并
void Merge(int A[],int low,int mid,int high){
    int i,j,k;
    for(k=low;k<=high;k++)
        B[k]=A[k]; //将A中所有元素复制到B中
    for(i=low,j=mid+1,k=i;i<=mid&&j<=high;k++){
        if(B[i]<=B[j])
            A[k]=B[i++]; //将较小值复制到A中
        else
            A[k]=B[j++];
    } //for
    while(i<=mid) A[k++]=B[i++];
    while(j<=high) A[k++]=B[j++];
}

void MergeSort(int A[],int low,int high){
    if(low<high){
        int mid=(low+high)/2; //从中间划分
        MergeSort(A,low,mid); //对左半部分归并排序
        MergeSort(A,mid+1,high); //对右半部分归并排序
        Merge(A,low,mid,high); //归并
    } //if
}
```

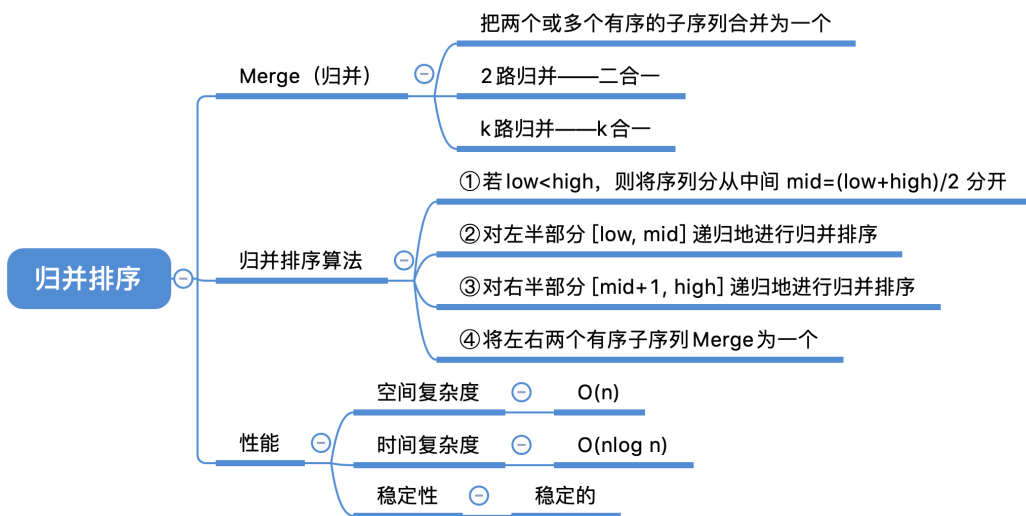
王道考研/CSKAQYAN.COM

算法效率分析



王道考研/CSKAOYAN.COM

知识回顾与重要考点



王道考研/CSKAOYAN.COM



@王道论坛



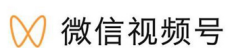
@王道计算机考研备考
@王道咸鱼老师-计算机考研
@王道楼楼老师-计算机考研



@王道计算机考研



@王道计算机考研



@王道计算机考研



@王道在线