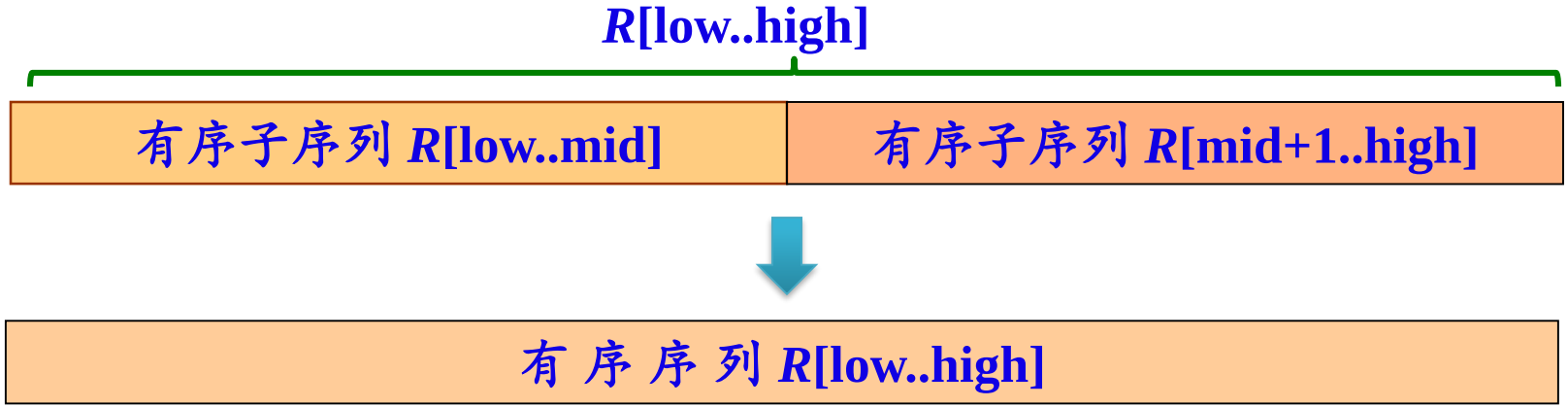
10.5 归并排序

1、归并的思路

归并排序是多次将相邻两个或两个以上的有序表合并成一个新的有序表。

最简单的归并是将相邻两个有序的子表合并成一个有序的表，即二路归并排序。

一次二路归并：将两个位置相邻的记录有序子序列归并为一个记录的有序序列。



2、二路归并算法

❶ **Merge()**：一次二路归并，将两个相邻的有序子序列归并为一个有序序列。

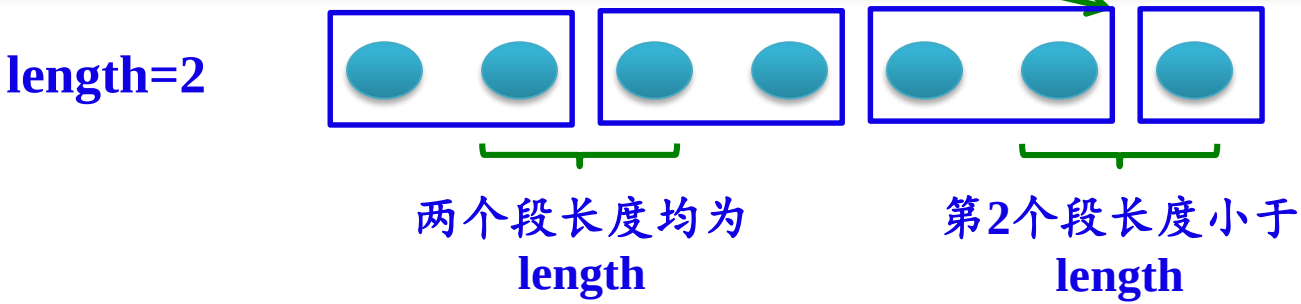
```
void Merge(RecType R[],int low,int mid,int high)
{      RecType *R1;
       int i=low,  j=mid+1,  k=0;
                        //k是R1的下标,i、j分别为第1、2段的下标
       R1=(RecType *)malloc((high-low+1)*sizeof(RecType));
       while (i<=mid && j<=high)
            if (R[i].key<=R[j].key)          //将第1段中的记录放入R1中
            {    R1[k]=R[i];  i++;k++;   }
            else                             //将第2段中的记录放入R1中
            {   R1[k]=R[j];  j++;k++;   }
```

空间复杂度为 O(high-low+1)

```
    while (i<=mid)          //将第1段余下部分复制到R1
    {    R1[k]=R[i];  i++;k++;  }
    while (j<=high)         //将第2段余下部分复制到R1
    {    R1[k]=R[j];  j++;k++;  }
    for (k=0,i=low;i<=high;k++,i++)     //将R1复制回R中
        R[i]=R1[k];
    free(R1);
}
```

❷ MergePass()：一趟二路归并（段长度为length）。

```
void MergePass(RecType R[],int length,int n)
{     int i;
      for (i=0;i+2*length-1<n;i=i+2*length)   //归并length长的两相邻子表
           Merge(R,i,i+length-1,i+2*length-1);
      if (i+length-1<n)                        //余下两个子表,后者长度小于length
           Merge(R,i,i+length-1,n-1);          //归并这两个子表
}
```

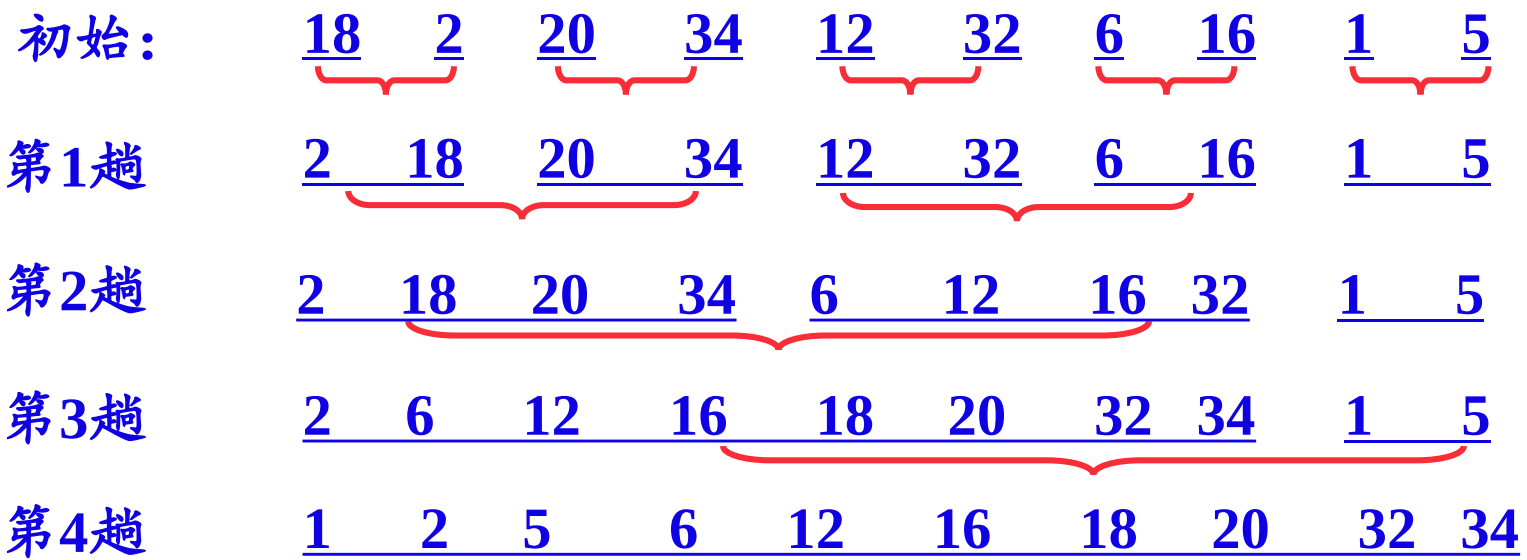


❸ MergeSort()：二路归并排序算法：

```
void MergeSort(RecType R[],int n)
{      int length;
       for (length=1;length<n;length=2*length)
            MergePass(R,length,n);
}
```

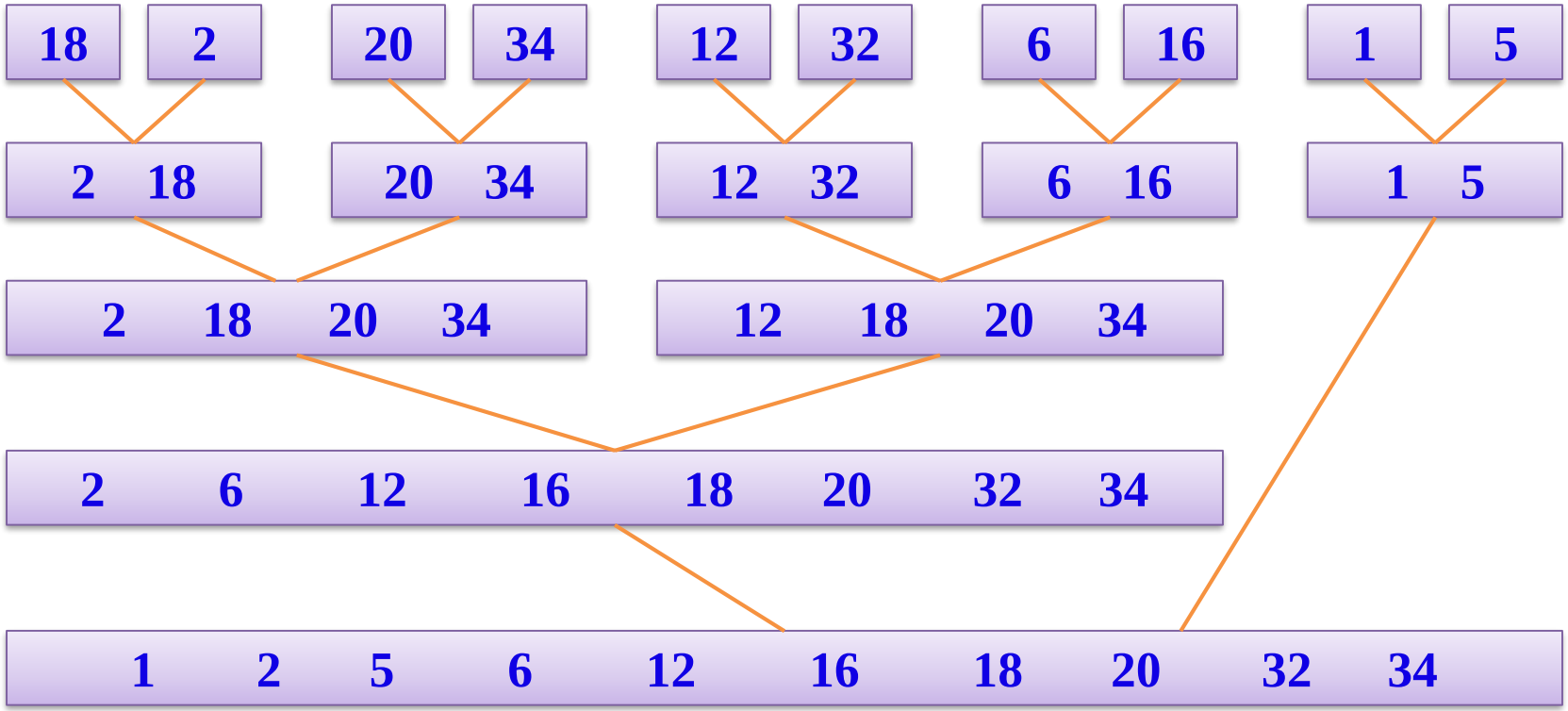
$\lceil \log_2 n \rceil$趟

【例10-7】 设待排序表有10个记录,其关键字分别为{18，2，20，34，12，32，6，16，1，5}。说明采用归并排序方法进行排序的过程。



需要$\log_2 10$取上界即4趟

更清楚的表示

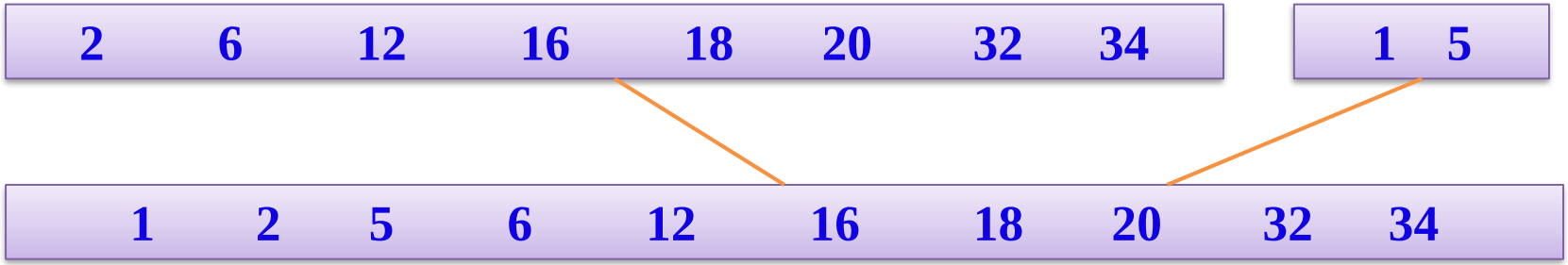


一颗归并树

3、归并算法分析

- 每一趟归并的时间复杂度为 $O(n)$
- 总共需进行$\lceil \log_2 n \rceil$趟。
- 二路归并排序的时间复杂度为$O(n\log_2 n)$。

每一次二路归并后临时空间都会释放。而最后的一次二路归需要全部记录参加归并：



占用临时空间为全部记录个数n

所以空间复杂度为$O(n)$。

【例10-8】数据序列{5,4,15,10,3,2,9,6,8}是某排序方法第一趟后的结果，该排序算法可能是_____。

A.冒泡排序　　B.二路归并排序

C. 堆排序　　D.简单选择排序

第一趟：{ 5,4, 15,10, 3,2, 9,6, 8 }

相邻的两个元素都是递减的

【例10-9】就排序算法所用的辅助空间而言，堆排序、快速排序和归并排序的关系是_____。

A.堆排序＜快速排序＜归并排序

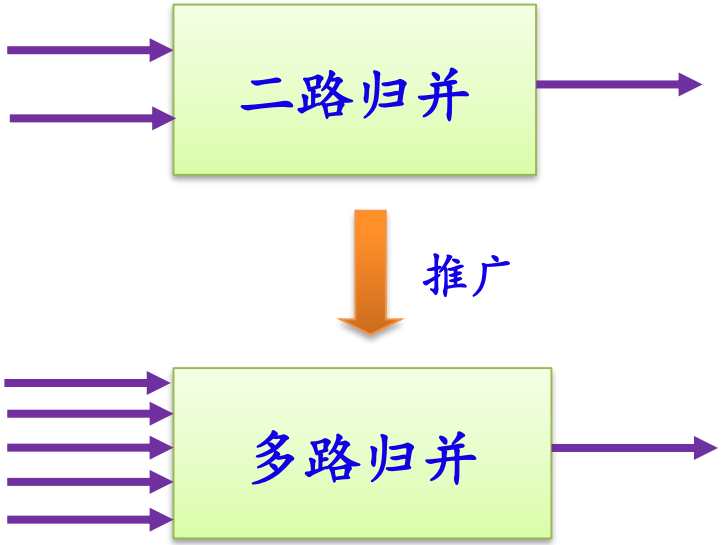
B.堆排序＜归并排序＜快速排序

C.堆排序＞归并排序＞快速排序

D.堆排序＞快速排序＞归并排序

堆排序、快速排序、归并排序

O(1)　　$O(\log_2 n)$　　$O(n)$



思考题：

以三路归并为例，多路归并算法设计有哪些难点？

三路归并和二路归并的时间比较

三路归并的时间复杂度为$O(n\log_3 n)$

$n\log_3 n = n\log_2 n / \log_2 3 = O(n\log_2 n)$

同一个级别！

——本讲完——