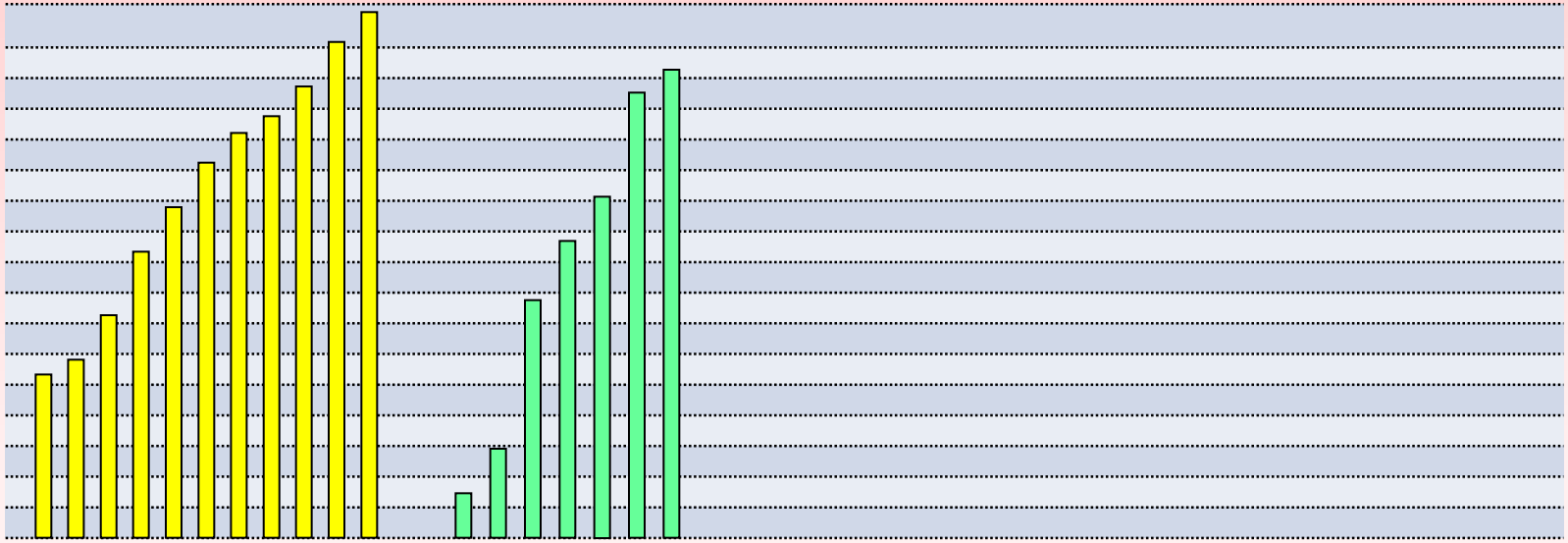
本节主题：

归并排序

归并排序思想

- 多次将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：是直接将两个有序的子表合并成一个有序的表。

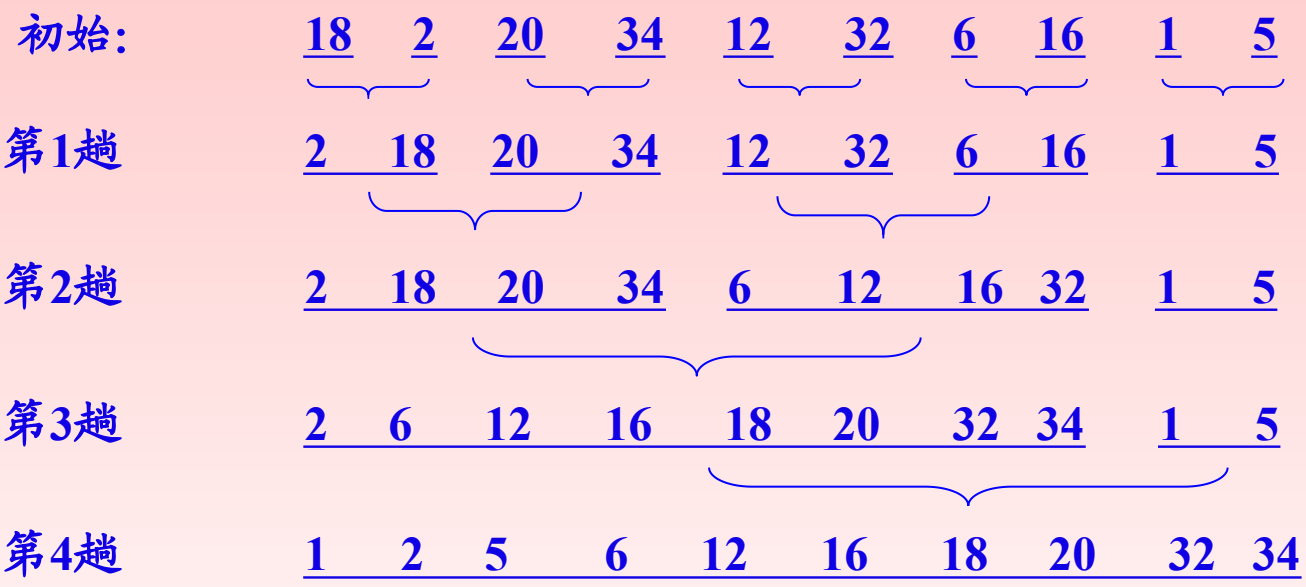


2-路归并的实现


```
void Merge(RecType R[],int low,int mid,int high)
{
   RecType *R1;
   int i=low,j=mid+1,k=0;
   //动态分配空间R1，用于保存合并结果
   R1=(RecType *)malloc((high-low+1)*sizeof(RecType));
   //两段均未扫描完时合并
   while (i<=mid && j<=high)
      if (R[i].key<=R[j].key)
      {
         R1[k]=R[i];
         i++;
         k++;
      }
      else
      {
         R1[k]=R[j];
         j++;
         k++;
      }
   //将第1段余下的部分复制到R1
   while (i<=mid)
   {
      R1[k]=R[i];
      i++;
      k++;
   }
   //将第2段余下的部分复制到R1
   while (j<=high)
   {
      R1[k]=R[j];
      j++;
      k++;
   }
   //将合并后的结果复制回R
   for (k=0,i=low; i<=high; k++,i++)
      R[i]=R1[k];
}
```

空间复杂度为 O(high-low+1)

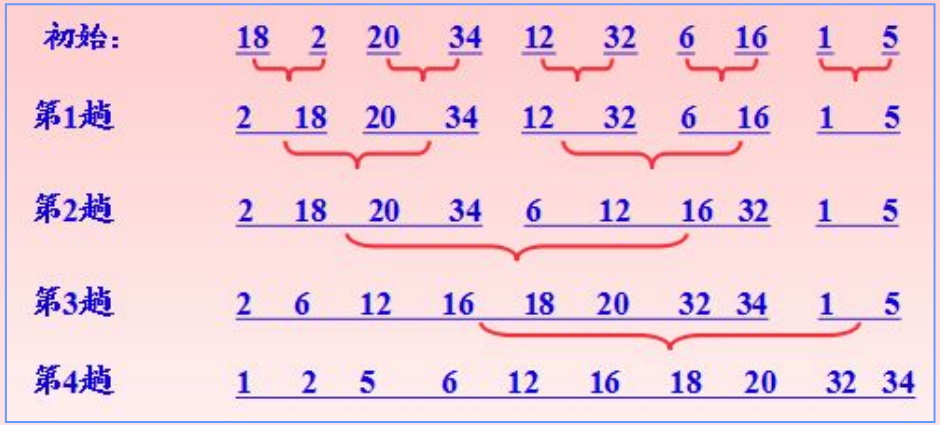
归并排序的过程



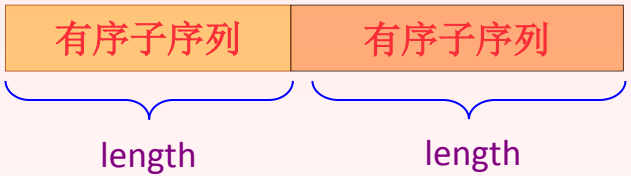
共4趟，$\log_2 10=4$

实现一趟归并MergePass()及合并排序MergeSort()的实现

```
void MergePass(RecType R[],int length,int n)
{
   int i;
   for (i=0; i+2*length-1<n; i=i+2*length)
      Merge(R,i,i+length-1,i+2*length-1);
   if (i+length-1<n)
      Merge(R,i,i+length-1,n-1);
}
void MergeSort(RecType R[],int n)
{
   int length;
   for (length=1; length<n; length=2*length)
      MergePass(R,length,n);
}
```



进行$\log_2 n$趟归并



性能及应用

- 时间复杂度
  - 每一趟归并的时间复杂度为 O(n)
  - 总共需进行⌈$\log_2 n$⌉趟。
  - ——归并排序的时间复杂度为O($n\log_2 n$)
- 空间复杂度
  - O(n)
- 实际应用
  - 当要合并的段很短时，直接用插入排序等简单算法