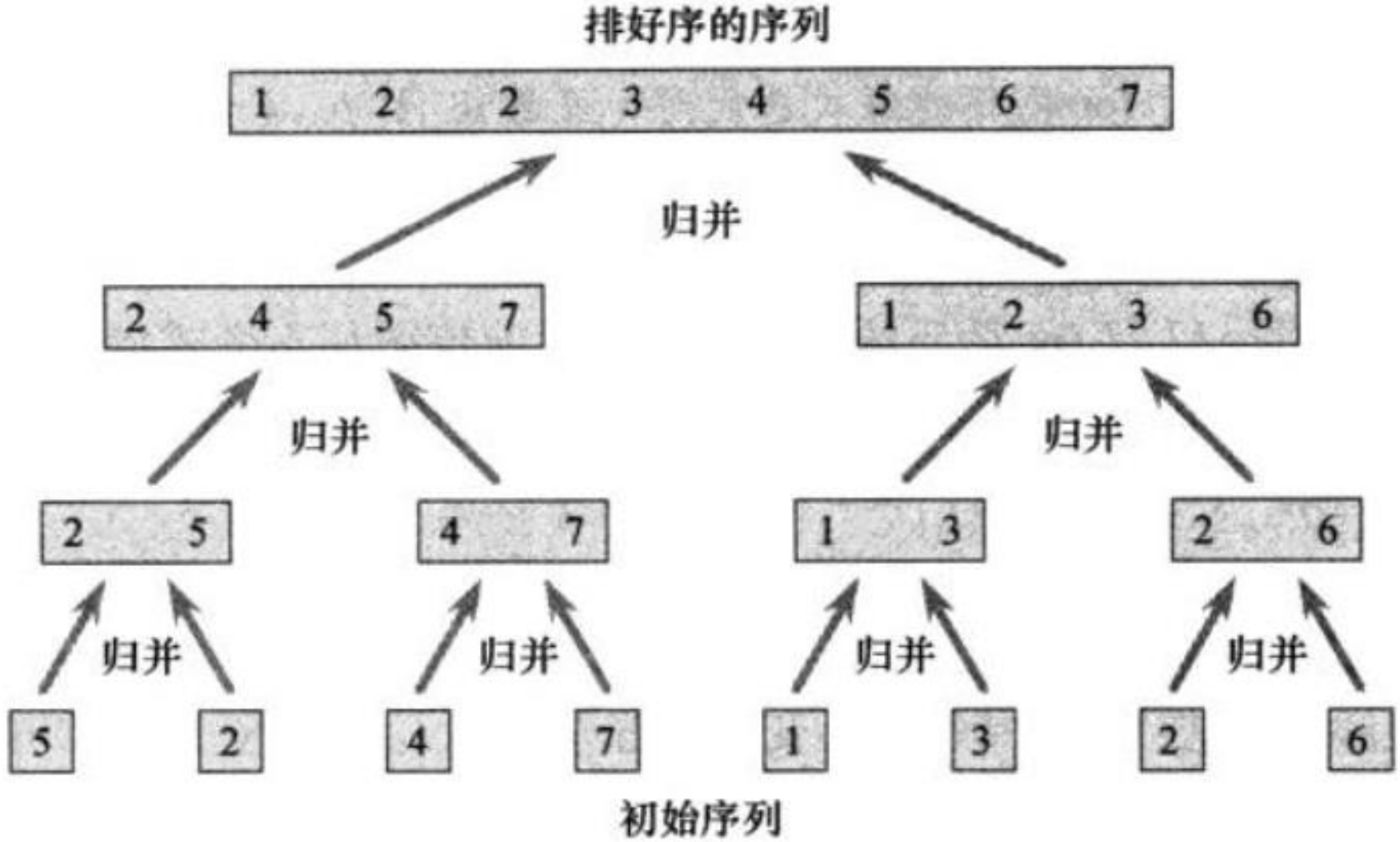
排序

归并排序

```
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
}
```

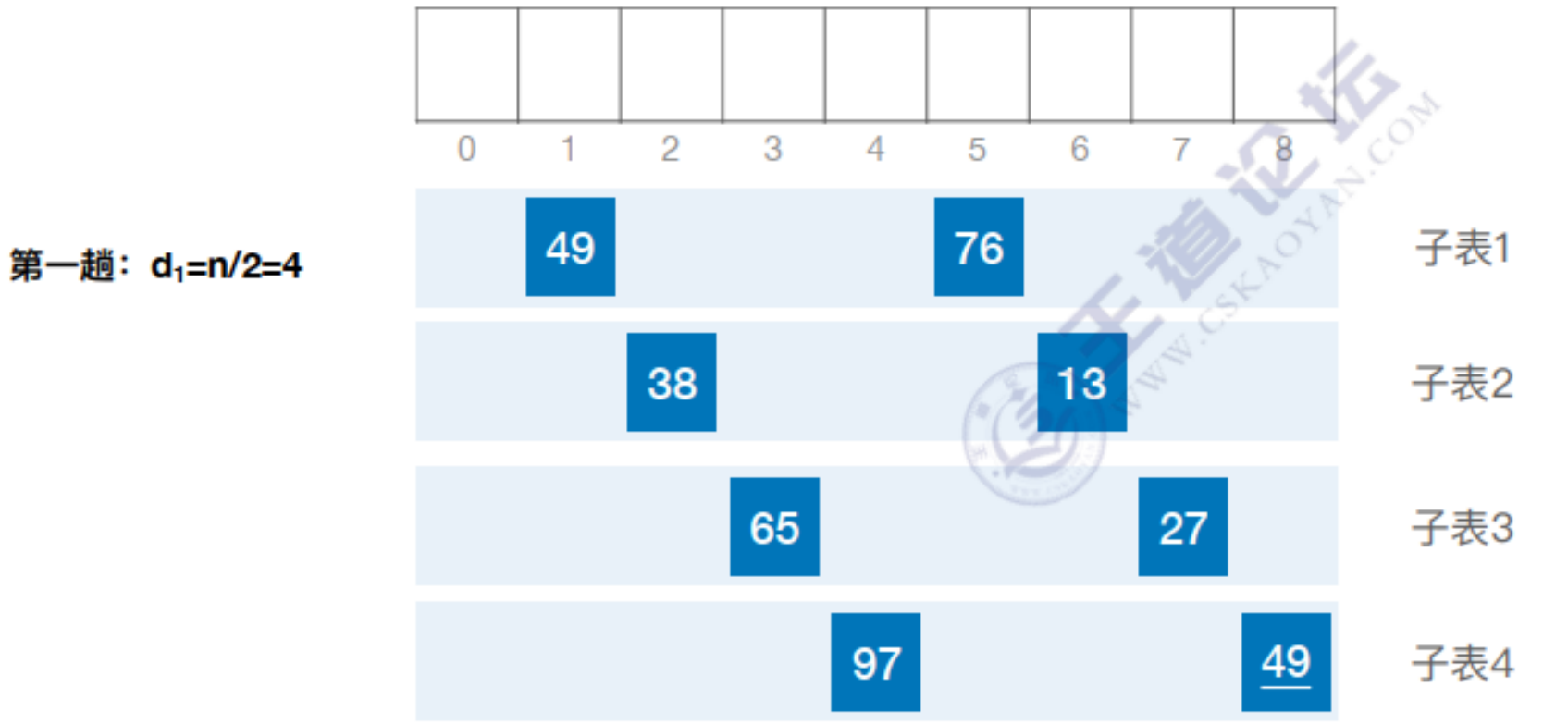
```
int *B=(int *)malloc(n*sizeof(int)); //辅助数组B

//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++];
}
```

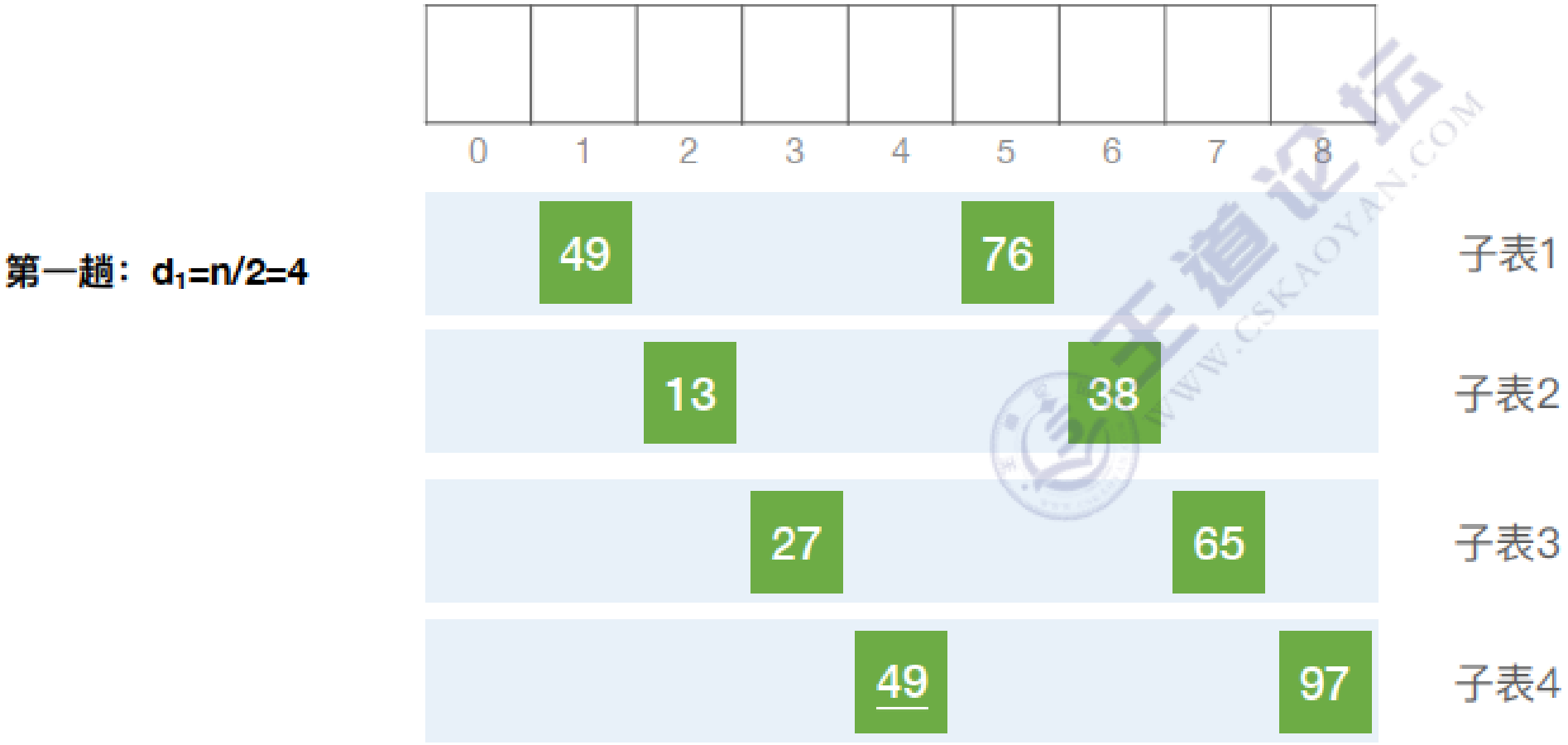
见示意图

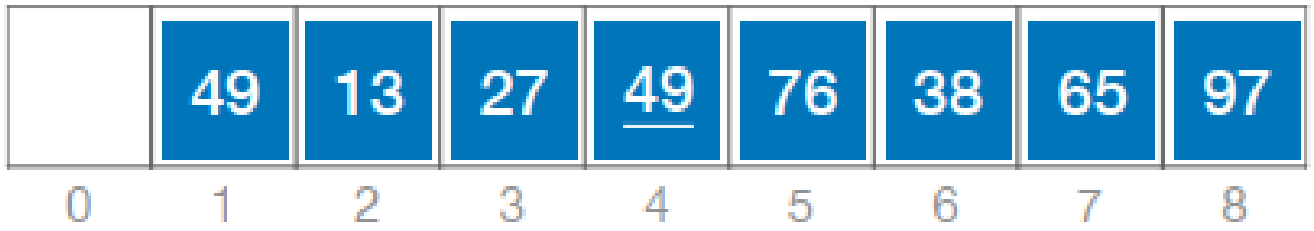

排好序的序列
1 2 2 3 4 5 6 7
归并
2 4 5 7
1 2 3 6
归并
归并
2 5
4 7
1 3
2 6
归并
归并
归并
归并
5
2
4
7
1
3
2
6
初始序列

希尔排序

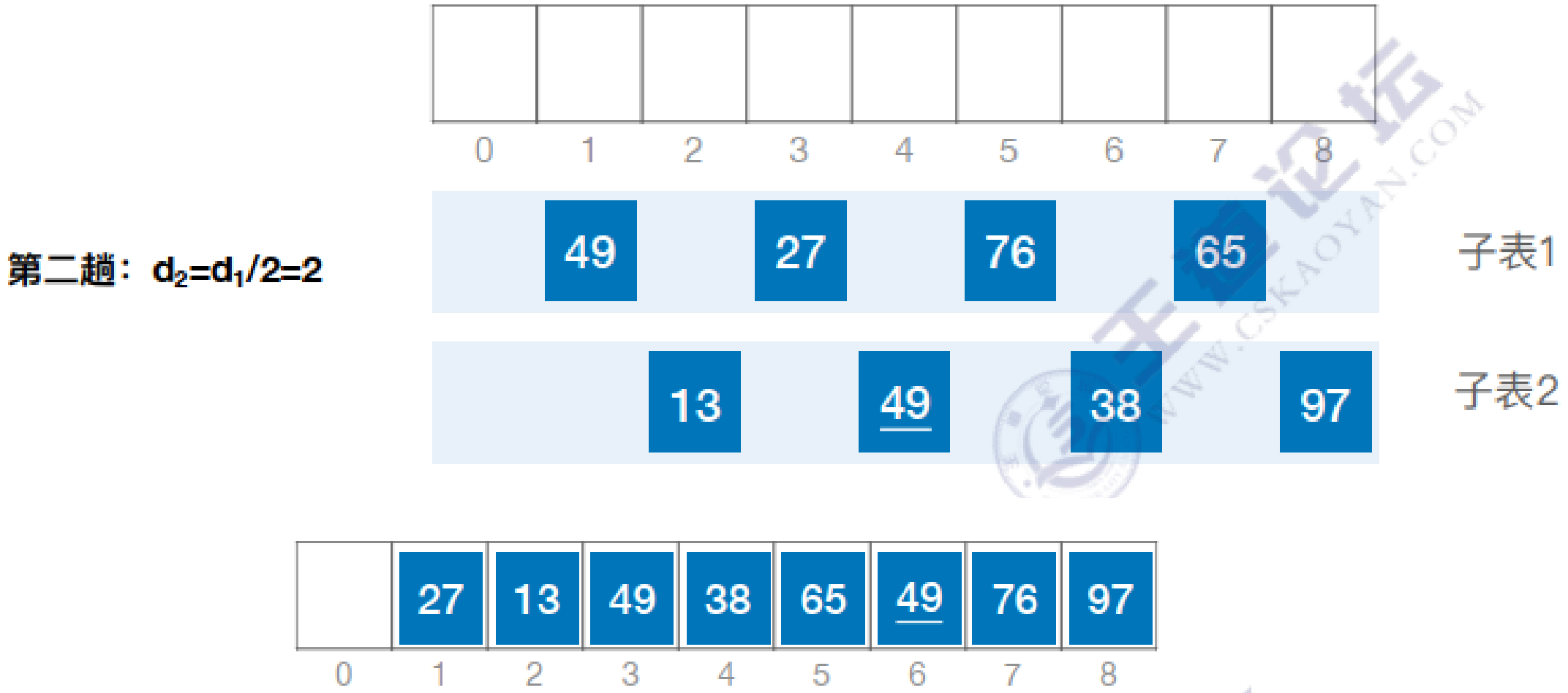
希尔排序：先将待排序表分割成若干形如 $L[i, i+d, i+2d, \ldots, i+kd]$ 的“特殊”子表，对各个子表分别进行直接插入排序。缩小增量d，重复上述过程，直到d=1为止。



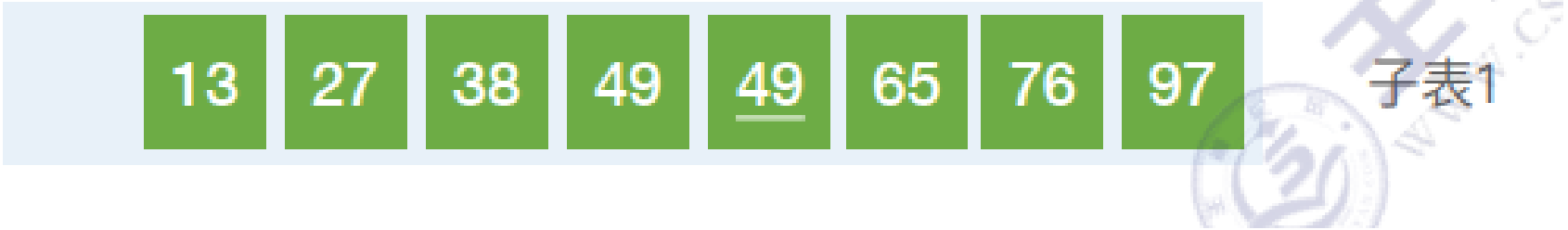

0 1 2 3 4 5 6 7 8
第一趟：$d_1=n/2=4$
49
76
子表1
13
38
子表2
27
65
子表3
49
97
子表4


49 13 27 49 76 38 65 97
0 1 2 3 4 5 6 7 8

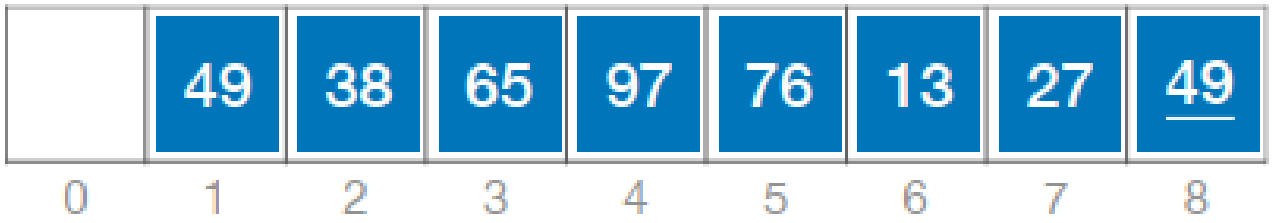
每趟都是模拟直接插入排序



第三趟：$d_3=d_2/2=1$

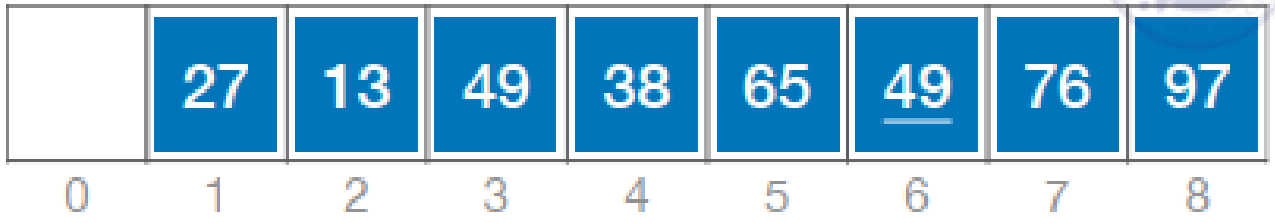


希尔本人建议：每次将增量缩小一半



第一趟：$d_1=n/2=4$

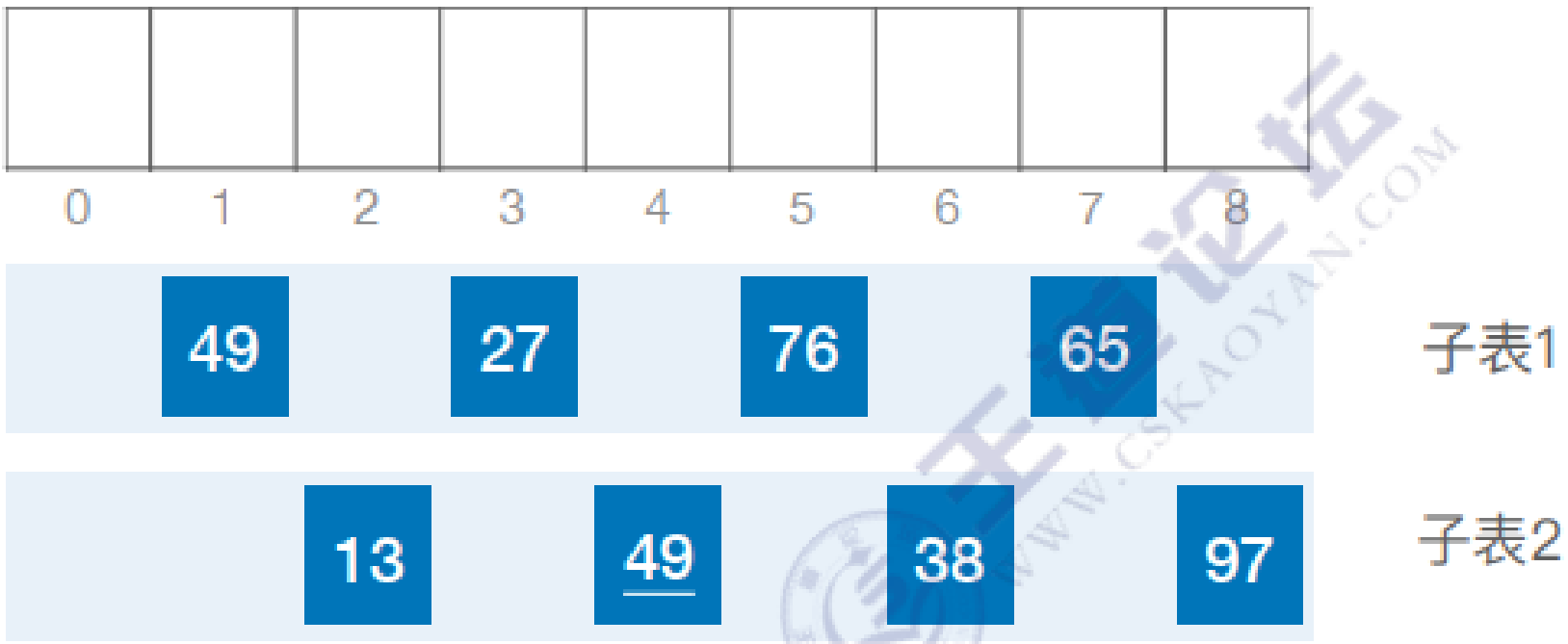
第二趟：$d_2=d_1/2=2$



第三趟：$d_3=d_2/2=1$

```
// 默认情况下 算法不人为指定增量d的值
void ShellSort(int a[], int n){
    int d, i, j;
    for (d=n/2;d>=1;d=d/2){
        for(i=d+1;i<=n;i++){
            if (a[i]<a[i-d]){
                a[0]=a[i];
                for(j=i-d;j>0 && a[0]<a[j];j=j-d)
                    a[j+d]=a[j];
                a[j+d]=a[0];
            }
        }
    }
}
```

第二趟：$d_2=d_1/2=2$



练习

- 将顺序表分成偶数和奇数两部分