Part One

第一讲 选择排序

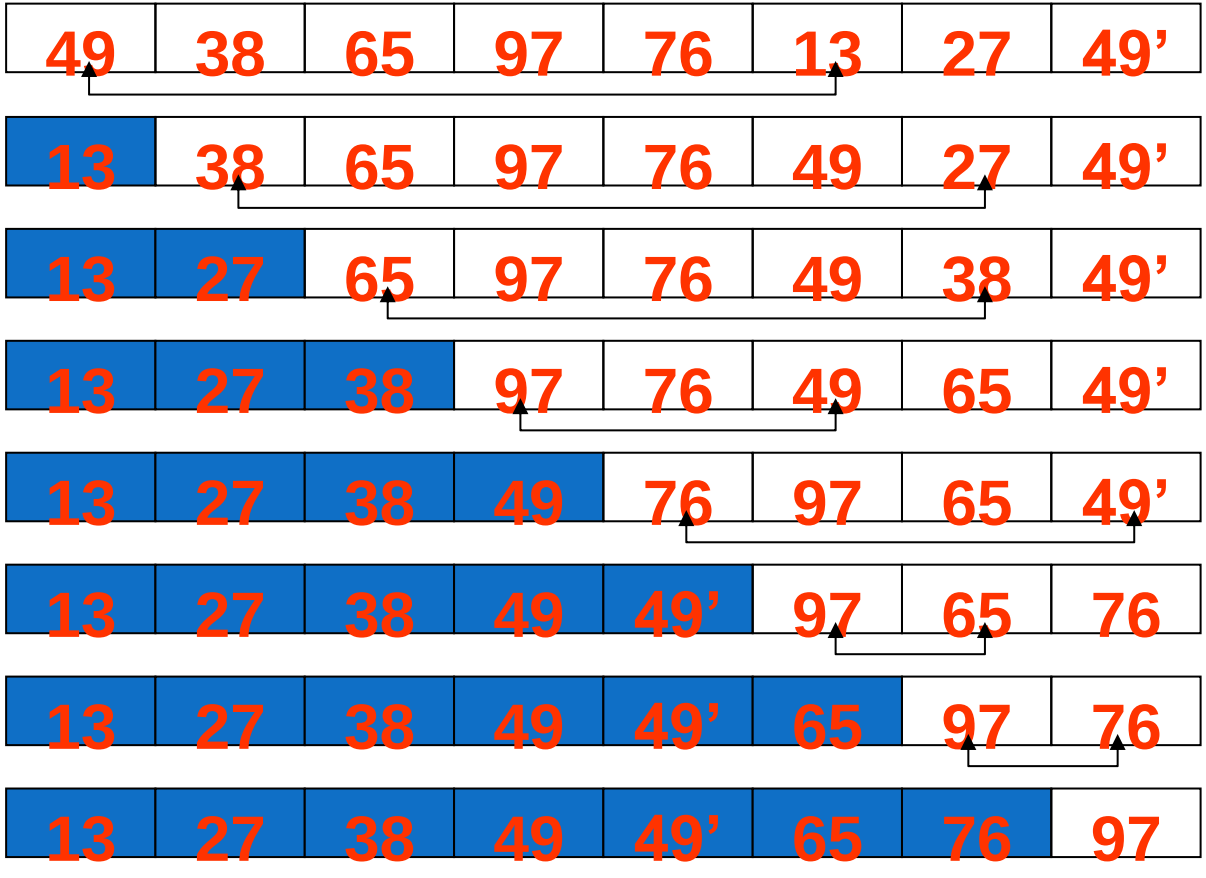
选择排序

- 排序是计算机中最为常见的操作之一。
- 排序就是将一组杂乱无章的数据按一定的规律排列起来（递增或递减）。
- 排序的对象可以是数字型，也可以是字符型，按其对应的ASCII码的顺序排列。

选择排序

- 选择排序（升序）：每次在若干无序数据中查找最小数，放在无序数据的首位。
  - 从N个元素的列表中找最小值及其下标，与第一个元素交换
  - 从第二个元素开始的N-1个元素中找最小值及其下标，与第二个元素交换
  - 以此类推，N-1轮后即为排好序的数据

选择排序


49 38 65 97 76 13 27 49’
13 38 65 97 76 49 27 49’
13 27 65 97 76 49 38 49’
13 27 38 97 76 49 65 49’
13 27 38 49 76 97 65 49’
13 27 38 49 49’ 97 65 76
13 27 38 49 49’ 65 97 76
13 27 38 49 49’ 65 76 97

选择排序算法的实现

```
a = [49,38,65,97,76,13,27,49]
for i in range(len(a)-1):
    m = i
    for j in range(i+1, len(a)):
        if a[j] < a[m]:
            m = j
    temp = a[i]
    a[i], a[m] = a[m], a[i]
    a[m] = temp
print(a)
```

- 选择排序算法的分析
  - 选择排序共需比较N-1轮，总共的比较次数为(N-1)+(N-2)+…+2+1=N(N-1)/2次
  - 选择排序执行交换的次数是N-1

Part Two

第二讲 冒泡排序

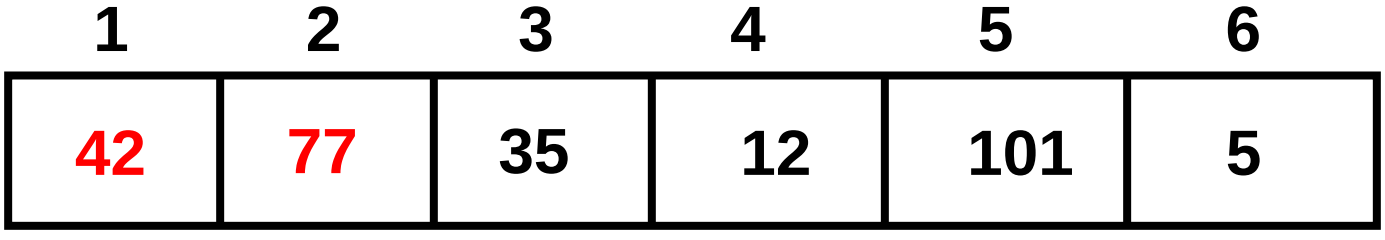
冒泡排序

- 冒泡排序（升序）：
  - 第一轮比较：从第一个元素开始，按顺序对列表中所有N个元素中连续的两个元素进行两两比较，如果两者不满足升序关系则交换。第一轮比较过后，最大数将下沉到列表最后。
  - 第二轮比较：从第一个元素开始，对列表中前N-1个元素进行两两比较，使次大数沉到最后。
  - 依此类推，N-1轮后，排序完毕

- 冒泡排序：两两比较

1	2	3	4	5	6
77	42	35	12	101	5

- 冒泡排序：两两比较



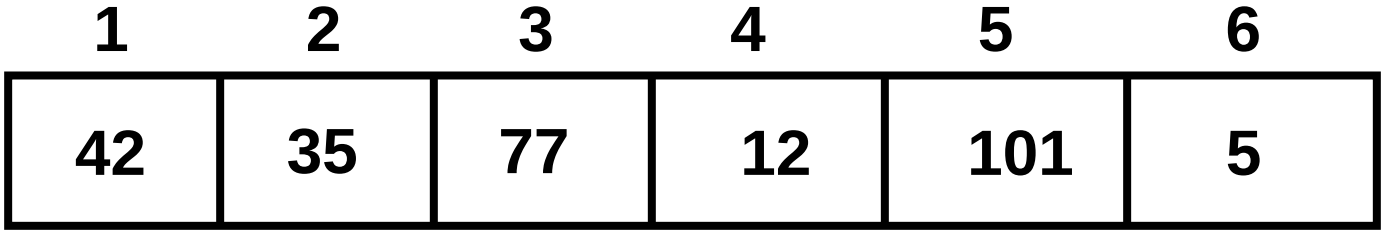
- 冒泡排序：两两比较

1	2	3	4	5	6
42	77	35	12	101	5

- 冒泡排序：两两比较

1	2	3	4	5	6
42	35	77	12	101	5

- 冒泡排序：两两比较



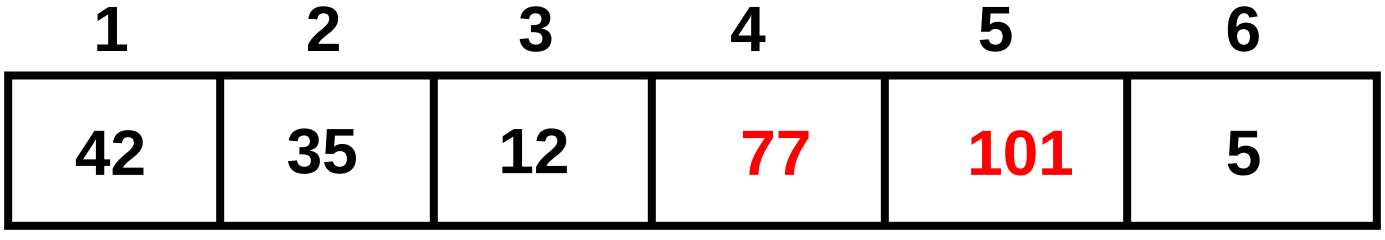
- 冒泡排序：两两比较

1	2	3	4	5	6
42	35	12	77	101	5

- 冒泡排序：两两比较

1	2	3	4	5	6
42	35	12	77	101	5

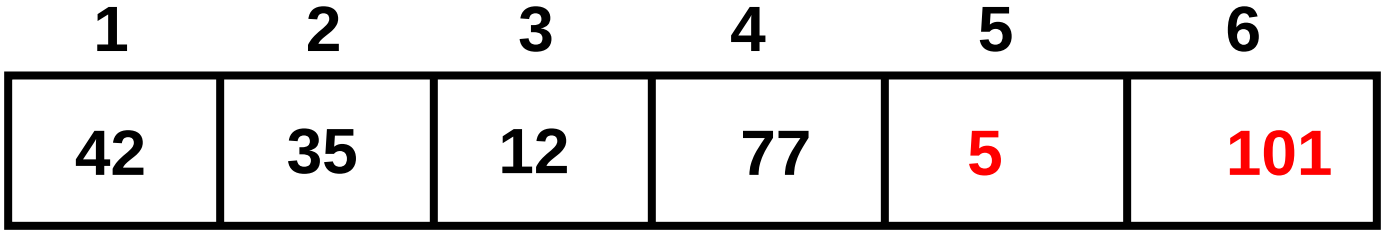
- 冒泡排序：两两比较



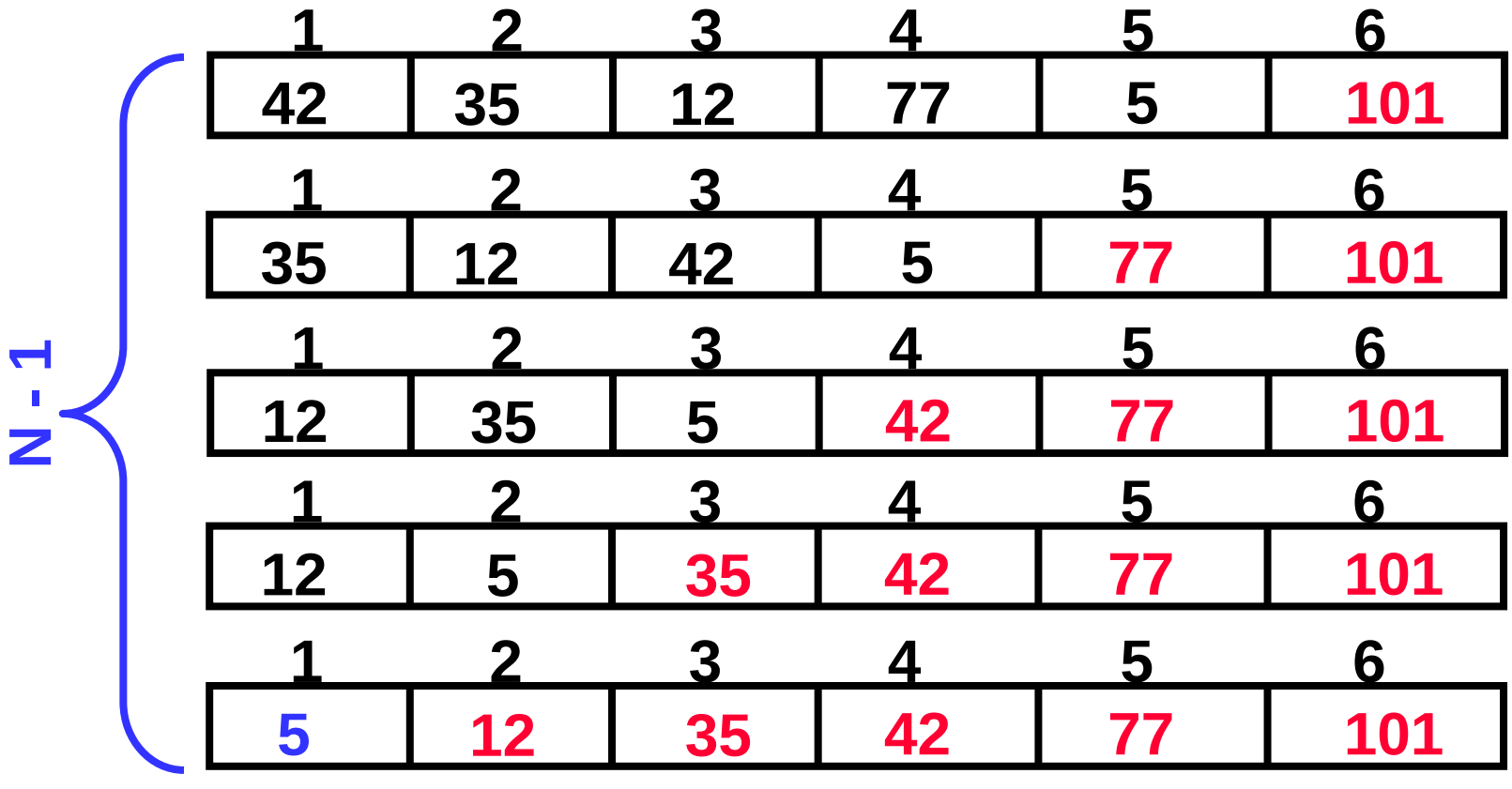
- 冒泡排序：两两比较

1	2	3	4	5	6
42	35	12	77	101	5

- 冒泡排序：两两比较



冒泡排序


N - 1
1 2 3 4 5 6
42 35 12 77 5 101
1 2 3 4 5 6
35 12 42 5 77 101
1 2 3 4 5 6
12 35 5 42 77 101
1 2 3 4 5 6
12 5 35 42 77 101
1 2 3 4 5 6
5 12 35 42 77 101

冒泡排序算法的实现

```
a=[77,42,35,12,101,5]
for i in range(len(a)-1):
    for j in range(len(a)-1-i):
        if a[j] > a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]
print(a)
```

- 冒泡排序算法的改进
  - 如果在某一轮比较中，一次交换也没有执行过，就说明已经排好序了

```
a=[77,42,35,12,101,5]
for i in range(len(a)-1):
    flag = True
    for j in range(len(a)-1-i):
        if a[j] > a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]
            flag = False
    if flag == True:
        break
print(a)
```

- 冒泡排序算法的分析
  - 算法的主要时间消耗是比较的次数
  - 冒泡排序共需比较N-1轮，总共的比较次数为(N-1)+(N-2)+…+2+1=N(N-1)/2次
  - 冒泡排序执行交换的次数不确定
  - 冒泡排序是一种效率很低的排序算法

Part Three

第三讲 函数

函数

- 函数用于在程序中分离不同的任务
- 实现结构化程序设计
- 减少程序复杂度
- 实现代码的复用
- 提高代码的质量
- 协作开发
- 实现特殊功能（递归）

- 函数的分类
  - 内置函数
    - input()、print()、int()、float()、len()、max()等
  - 标准库函数
    - math包中的sqrt()、sin()；random包中的random()等
  - 第三方库函数
    - Python是开放式语言，Python社区里提供了很多第三方的高质量的库，比如Python图像处理库等，下载安装后可像标准库函数一样使用
  - 自定义函数

自定义函数的定义

```
def 函数名([形参列表]):
    函数体
```

函数的调用

```
函数名([实参列表])
```

```
#例：创建求两个数平均值的函数
#定义
def average(a, b):
    return (a+b)/2

#调用
c = average(1, 2)
print(c)
```

- 问题：编写一个函数判定一个数是否是素数，并调用其输出200以内的素数。

```
import math
#定义函数
def IsPrime(a):
    m = int(math.sqrt(a))
    for i in range(2, m+1):
        if a%i==0:
            return False
    return True
#调用
for i in range(2, 200):
    if IsPrime(i)==True:
        print(i)
```

- 问题：将冒泡排序算法写成函数形式。

```
a=[77,42,35,12,101,5]
for i in range(len(a)-1):
    for j in range(len(a)-1-i):
        if a[j] > a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]
print(a)
```

```
def bubble_sort(a):
    for i in range(len(a)-1):
        for j in range(len(a)-1-i):
            if a[j] > a[j+1]:
                a[j],a[j+1]=a[j+1],a[j]

a=[77,42,35,12,101,5]
bubble_sort(a)
print(a)
```

- 递归函数：自调用函数，在函数体内部直接或间接地调用自己。
- 问题：编写函数求n的阶乘

```
#非递归方法
def factorial(n):
    s = 1
    for i in range(1,n+1):
        s = s * i
    return s
```

```
#递归方法
def factorial(n):
    if n == 1:
        return 1
    else:
        s = n * factorial(n-1)
        return s
```

```
def factorial(3):
    if n == 1:
        return 1
    else:
        s = n * factorial(n-1)
        return s
```

```
def factorial(3):
    if n == 1:
        return 1
    else:
        s = 3 * factorial(2)
        return s
```

```
def factorial(3):
    if n == 1:
        return 1
    else:
        s = 3 * factorial(2)
        return s
```

```
def factorial(2):
    if n == 1:
        return 1
    else:
        s = 2 * factorial(1)
        return s
```

```
def factorial(3):
    if n == 1:
        return 1
    else:
        s = 3 * factorial(2)
        return s
```

```
def factorial(2):
    if n == 1:
        return 1
    else:
        s = 2 * factorial(1)
        return s
```

```
def factorial(1):
    if n == 1:
        return 1
    else:
        s = n * factorial(n-1)
        return s
```

```
def factorial(3):
    if n == 1:
        return 1
    else:
        s = 3 * factorial(2)
        return s
```

```
def factorial(2):
    if n == 1:
        return 1
    else:
        s = 2 * 1
        return s
```

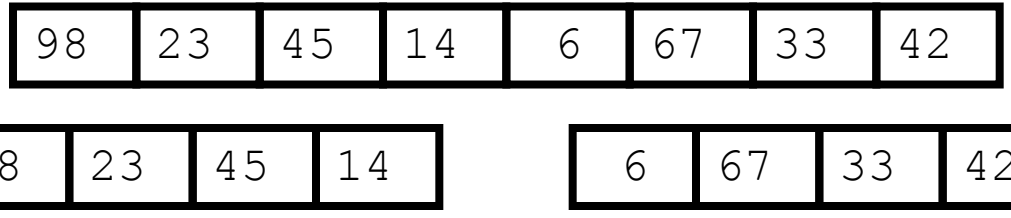
```
def factorial(3):
    if n == 1:
        return 1
    else:
        s = 3 * 2
        return s
```

Part Four

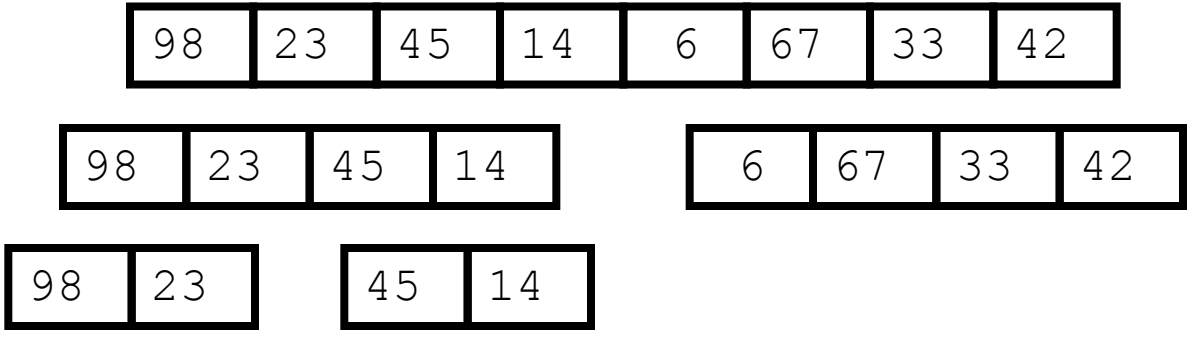
第四讲 归并排序

- 归并排序：分而治之
  - 将包含N个元素的列表拆分成两个含N/2个元素的子列表
  - 对两个子列表递归调用归并排序（最后可以将整个列表分解为N个子列表）
  - 合并两个已排好序的子列表

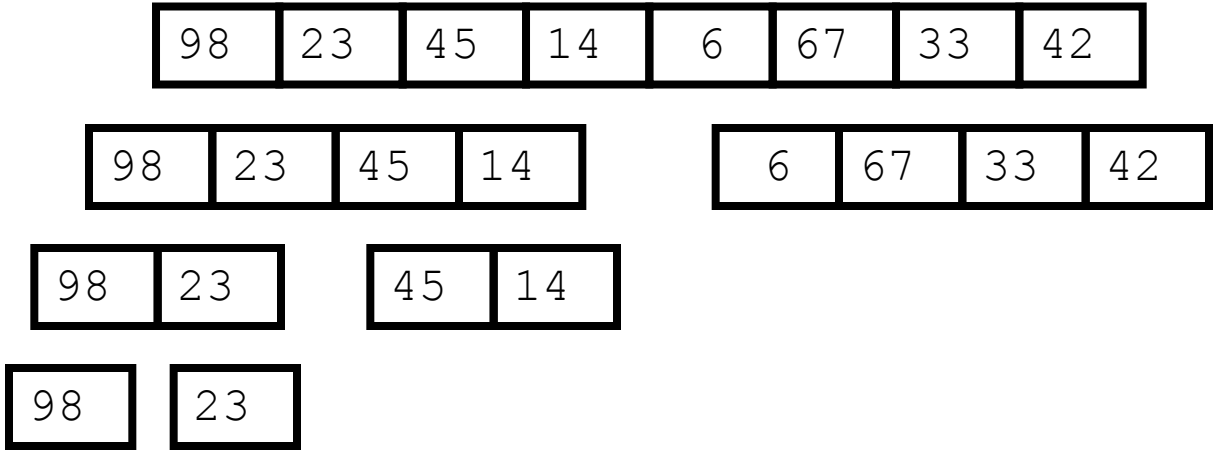
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42

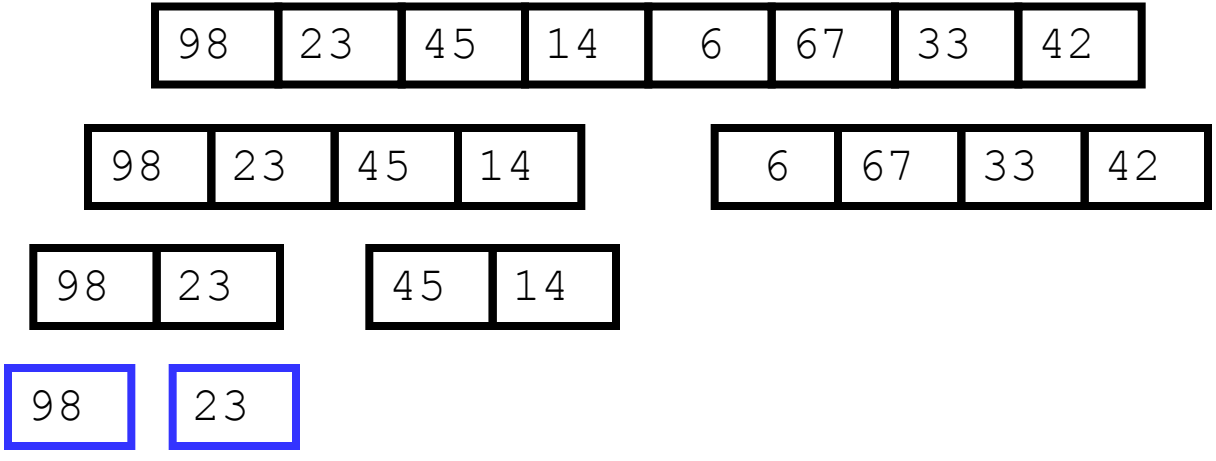
归并排序



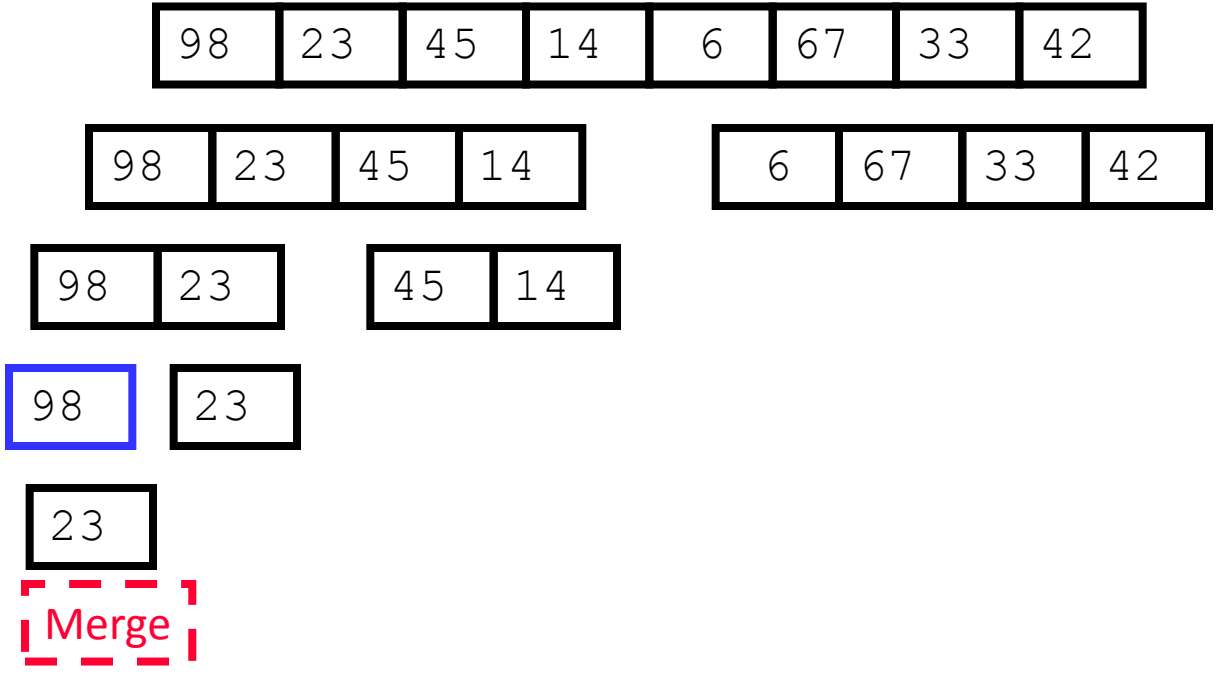
归并排序



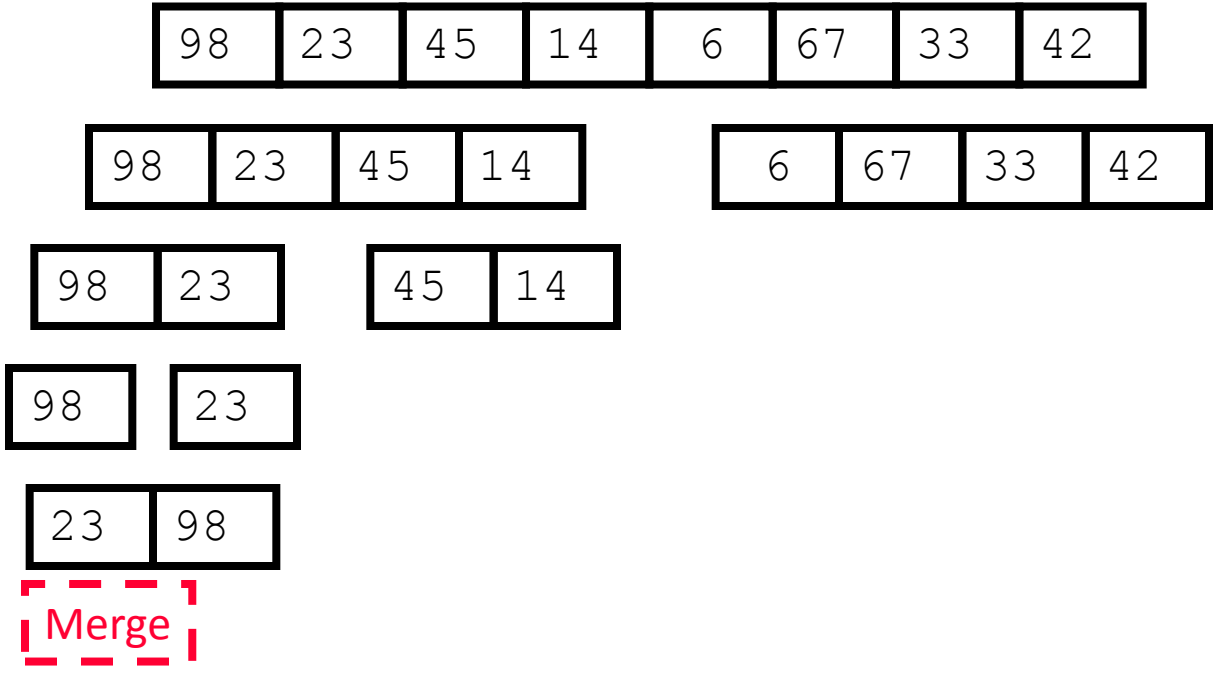
归并排序



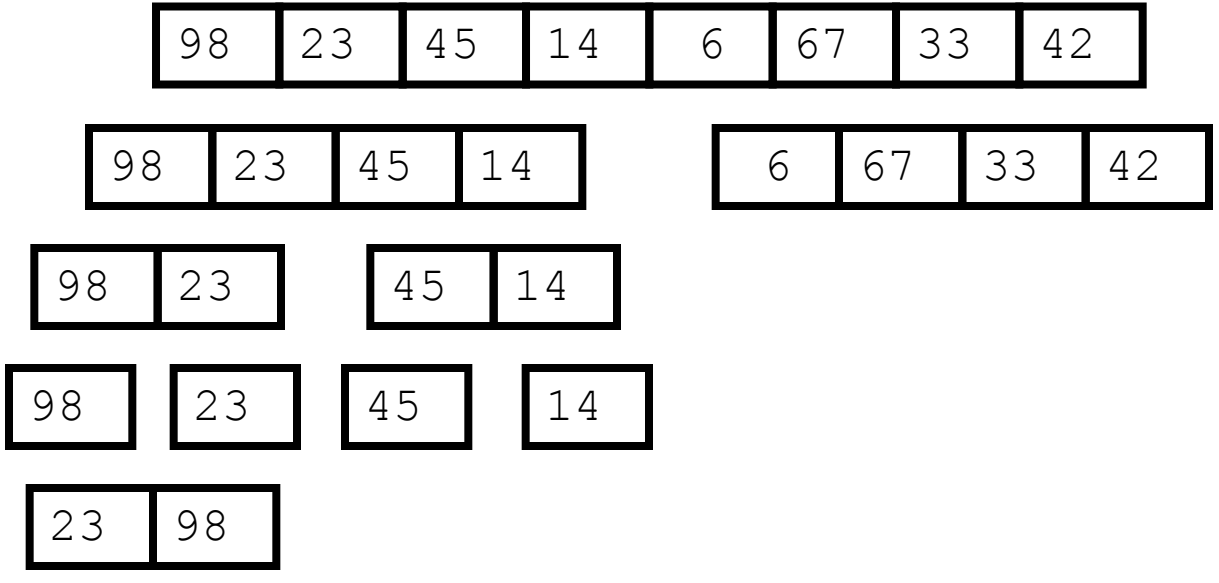
归并排序



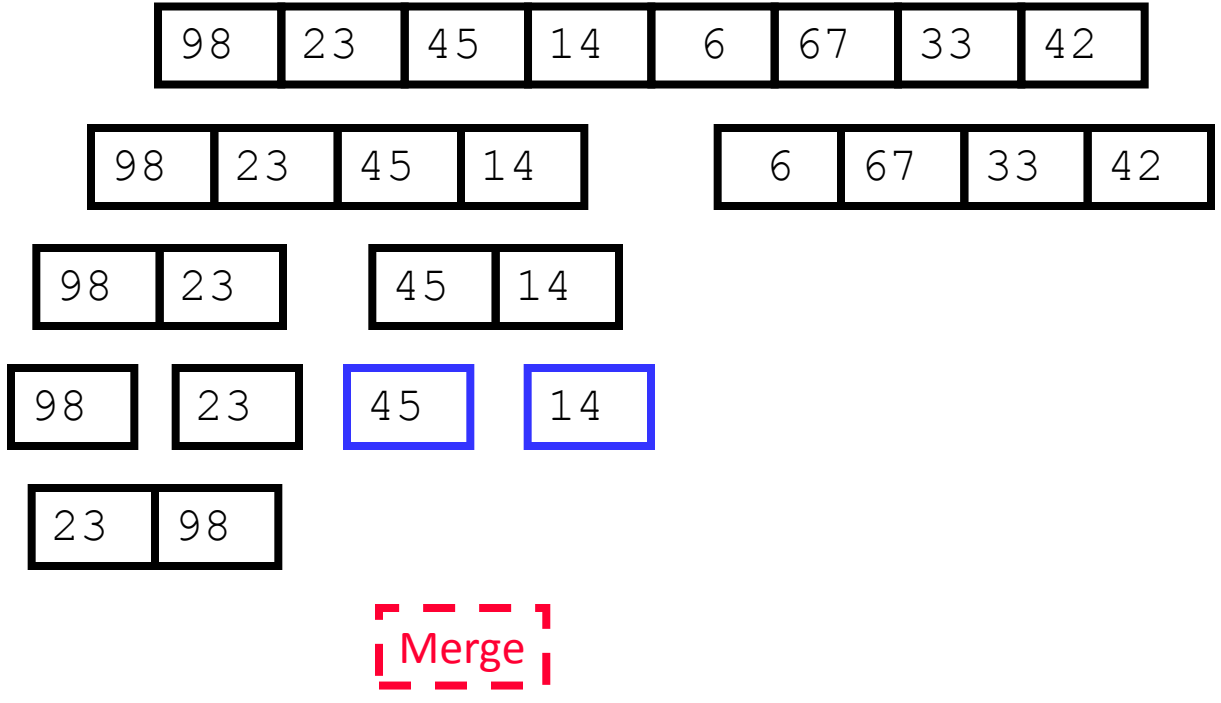
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
23 98
Merge

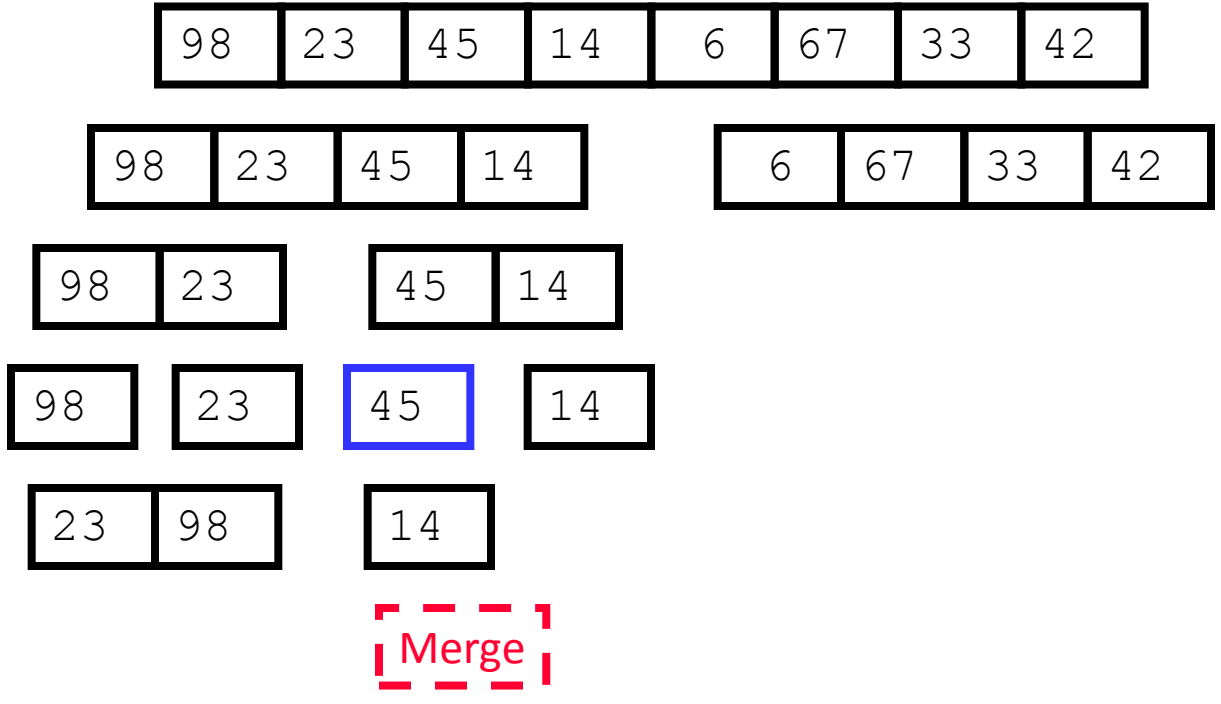
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98

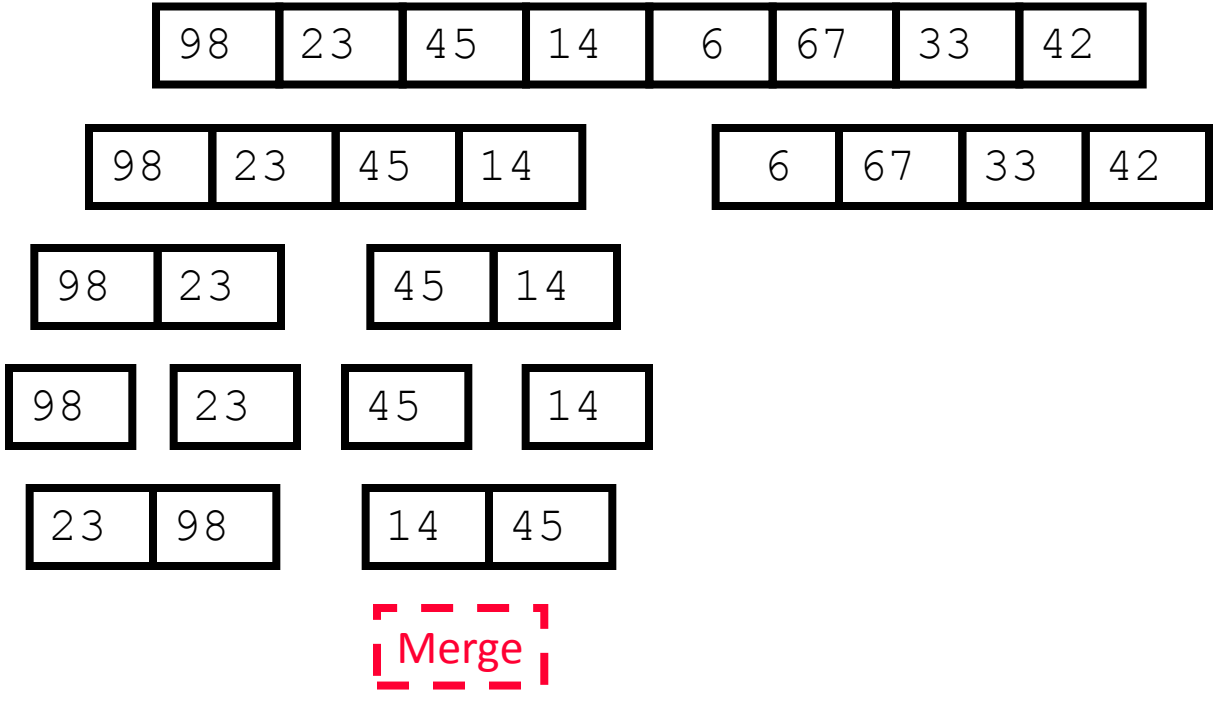
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
Merge

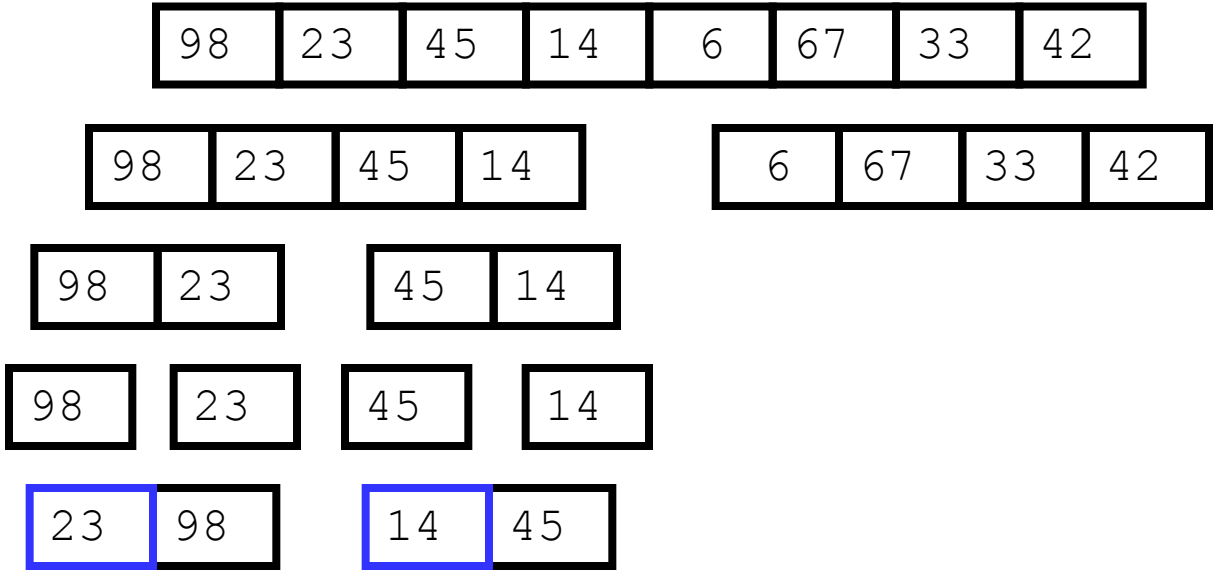
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14
Merge

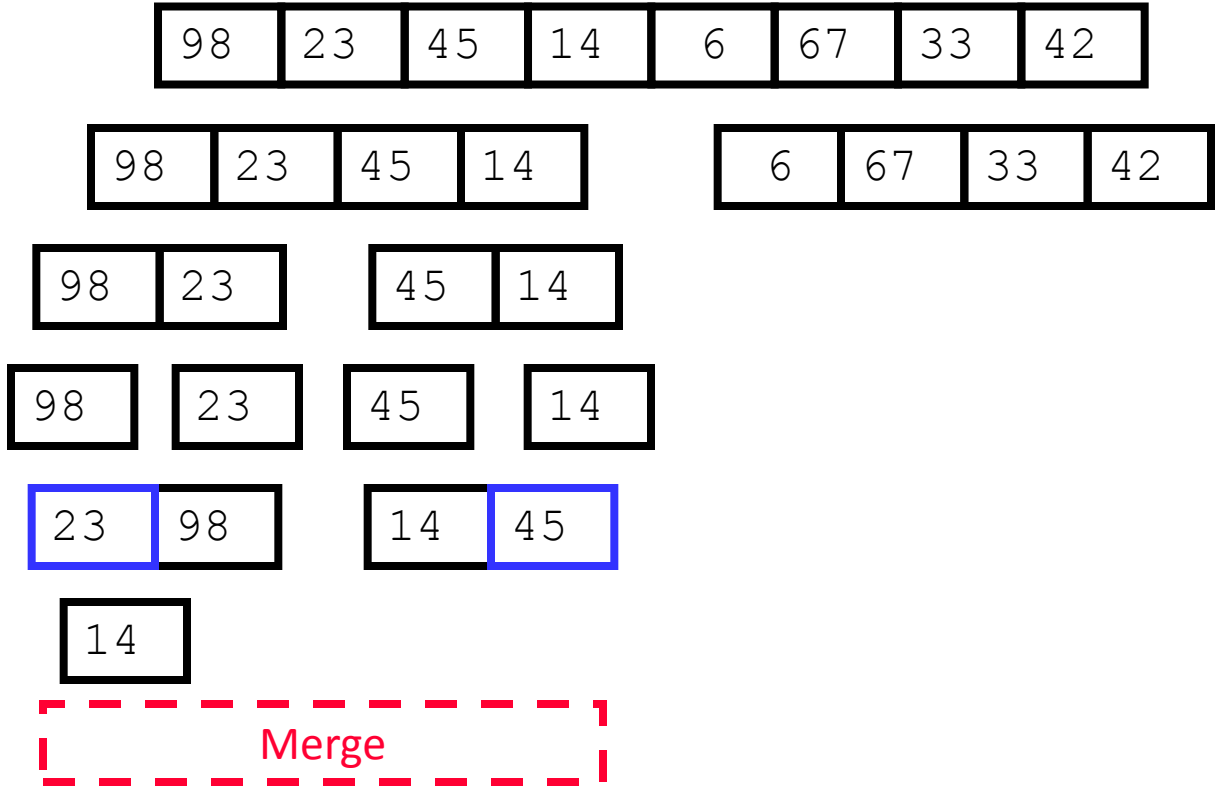
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
Merge

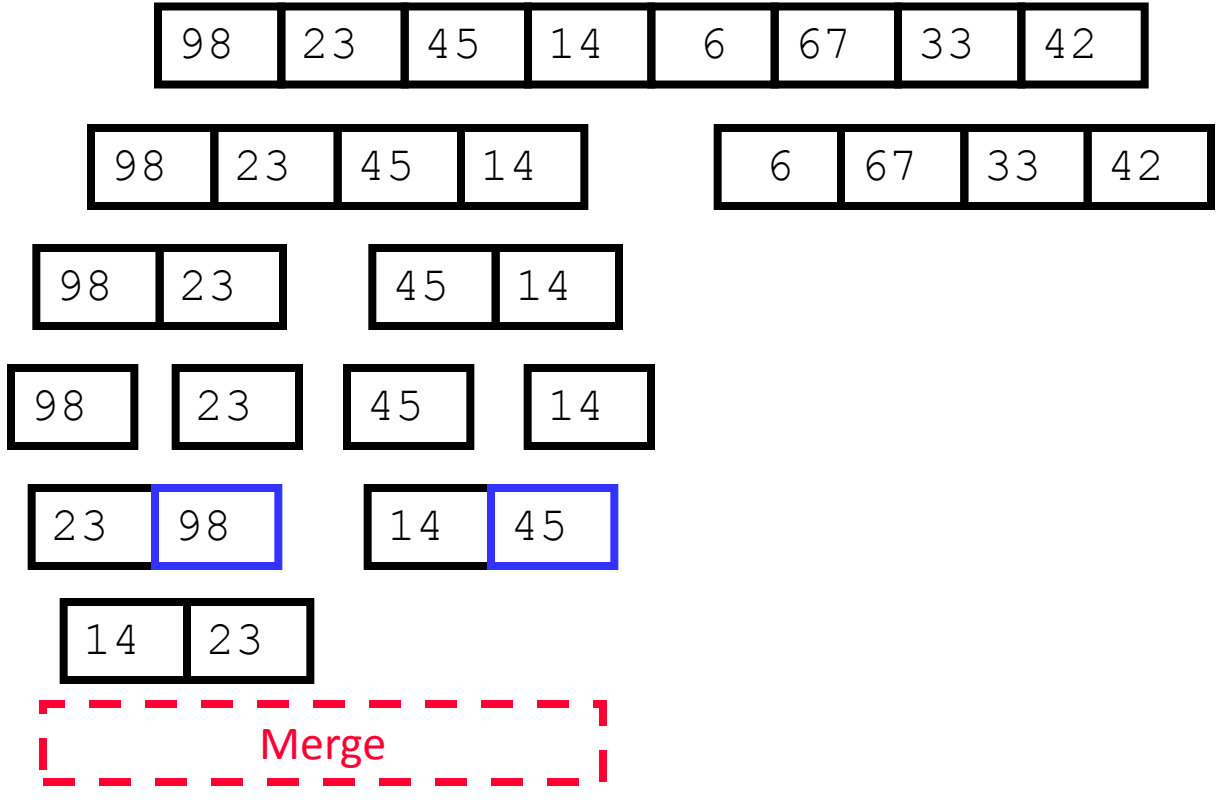
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
Merge

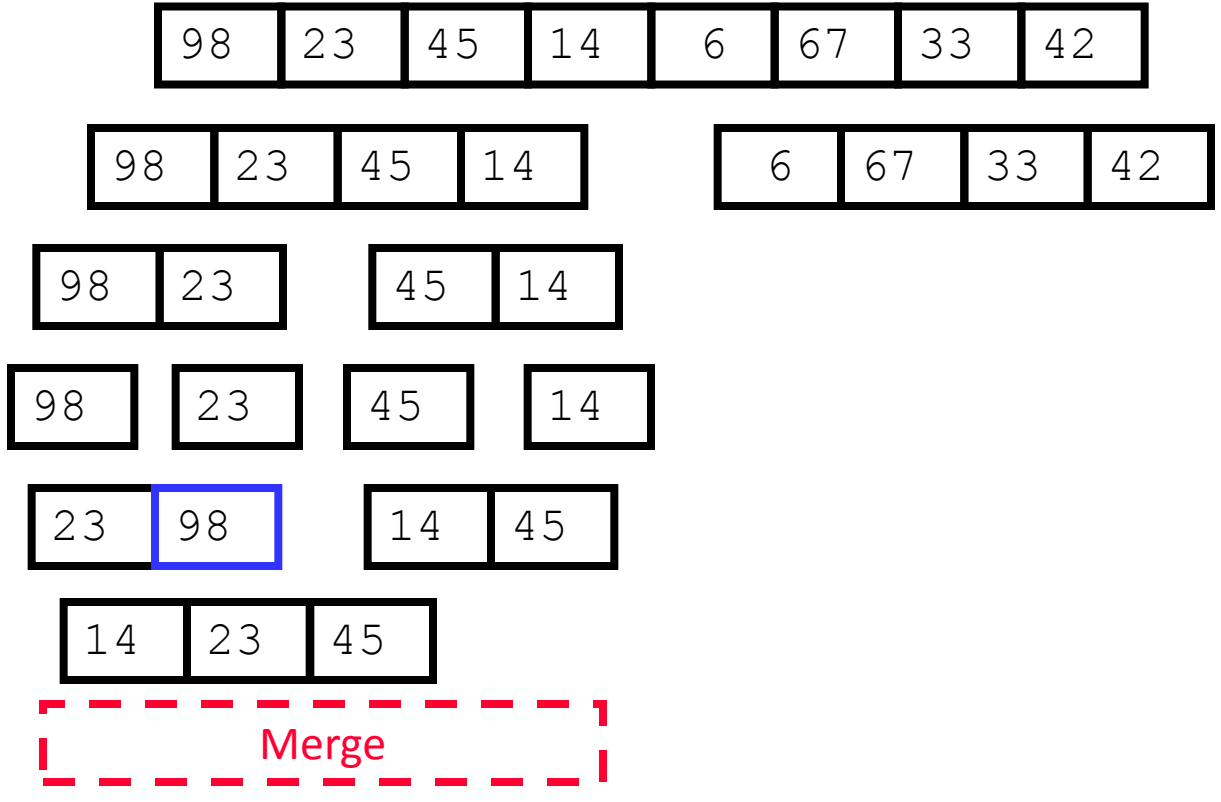
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14
Merge

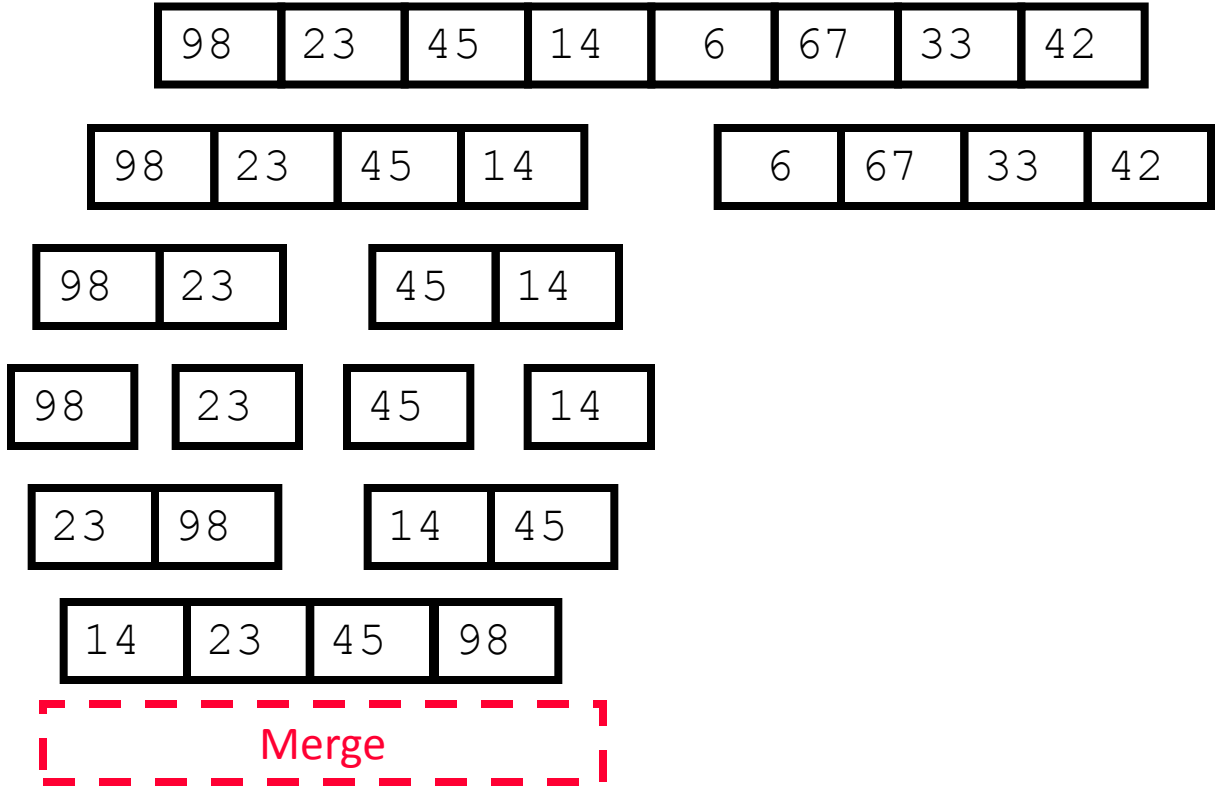
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23
Merge

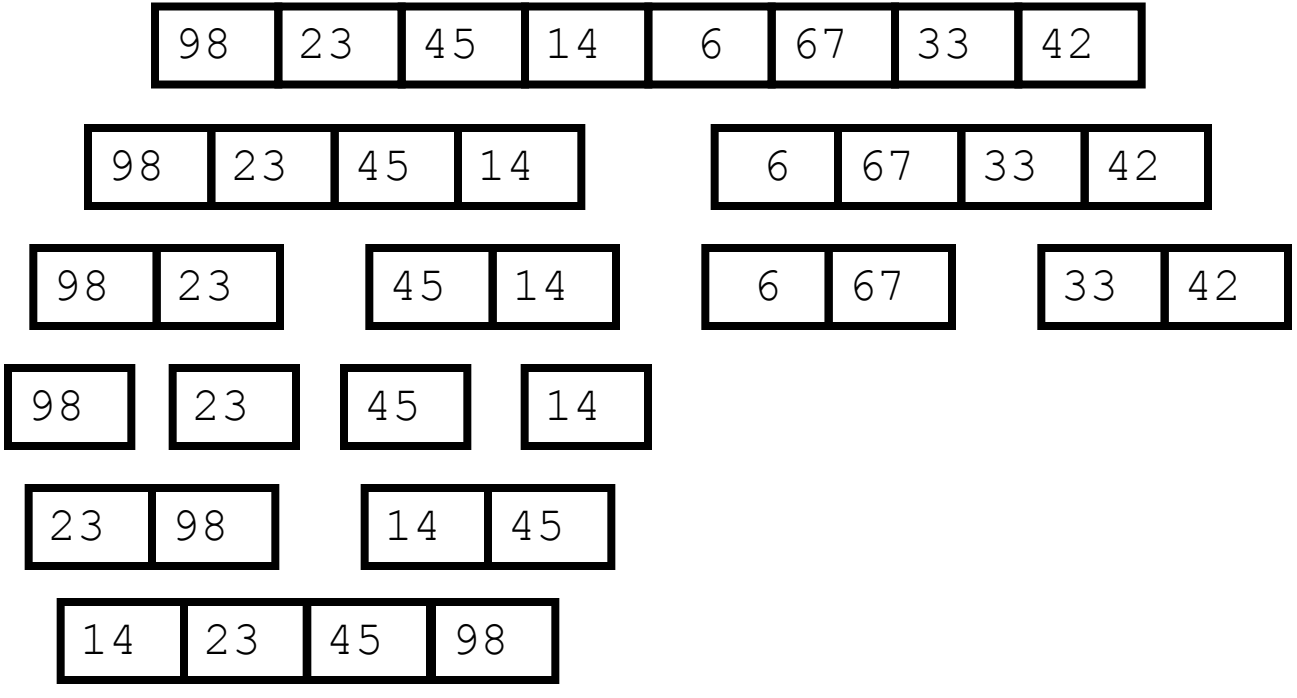
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23 45
Merge

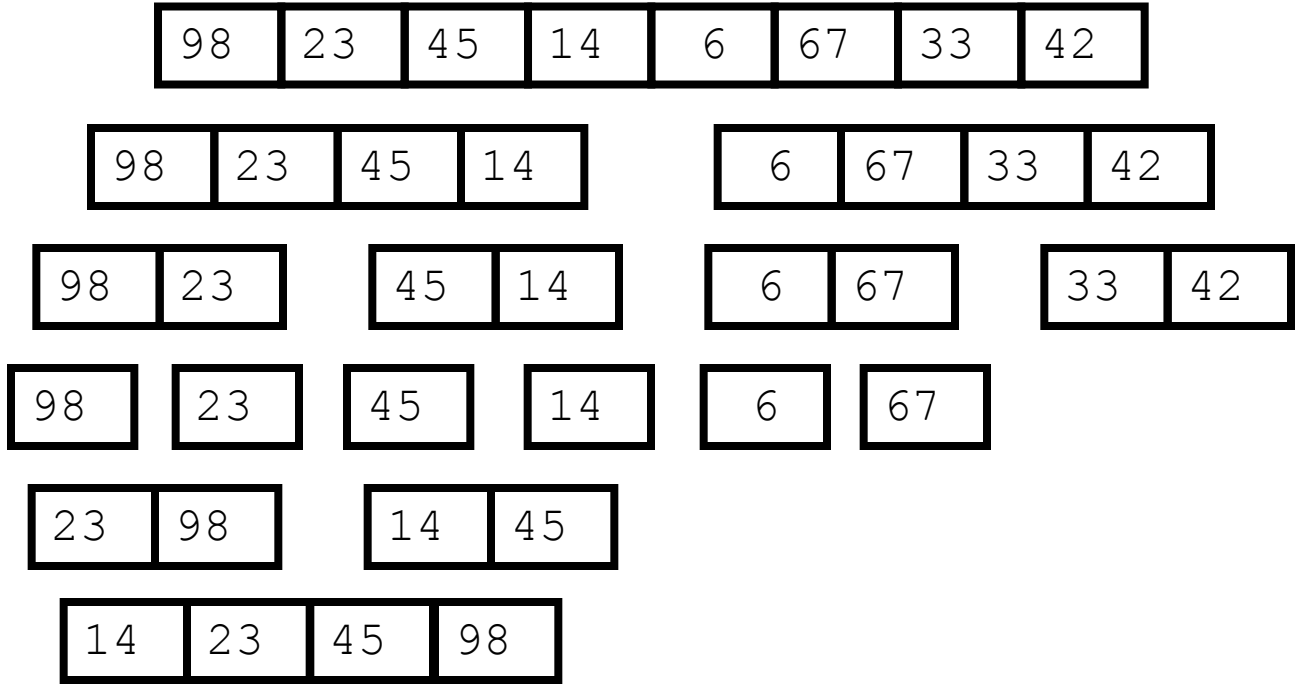
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23 45 98
Merge

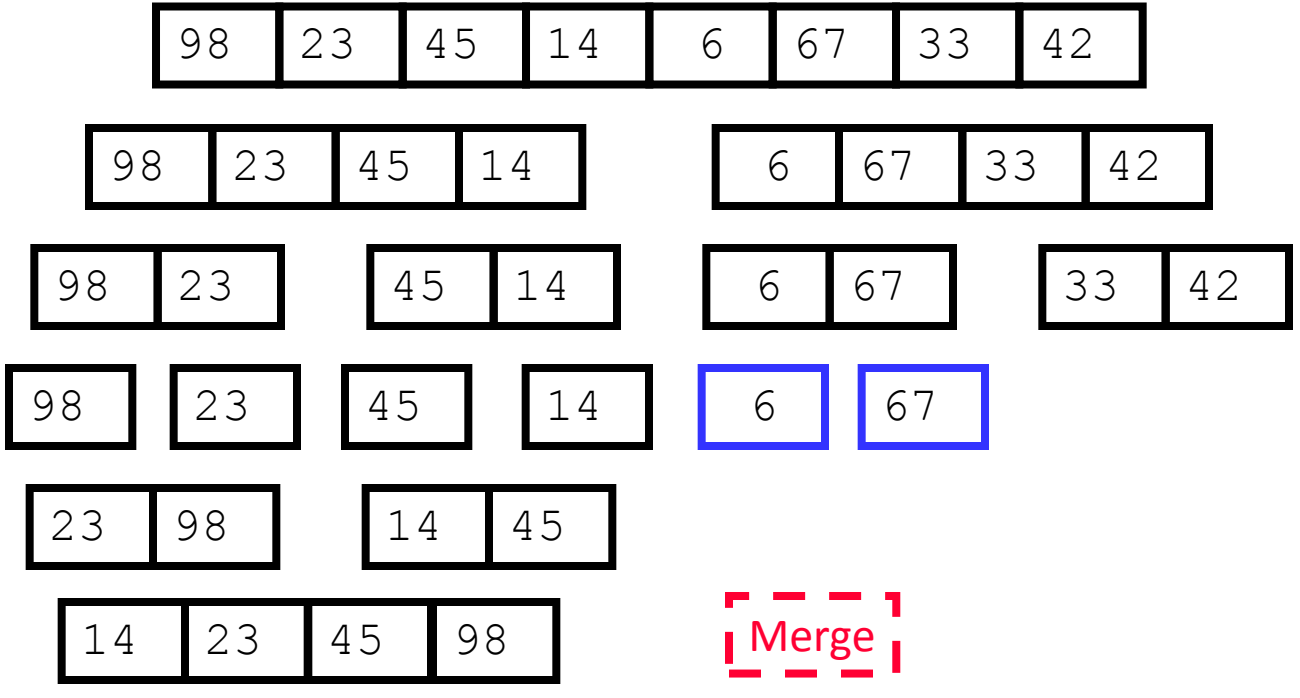
归并排序

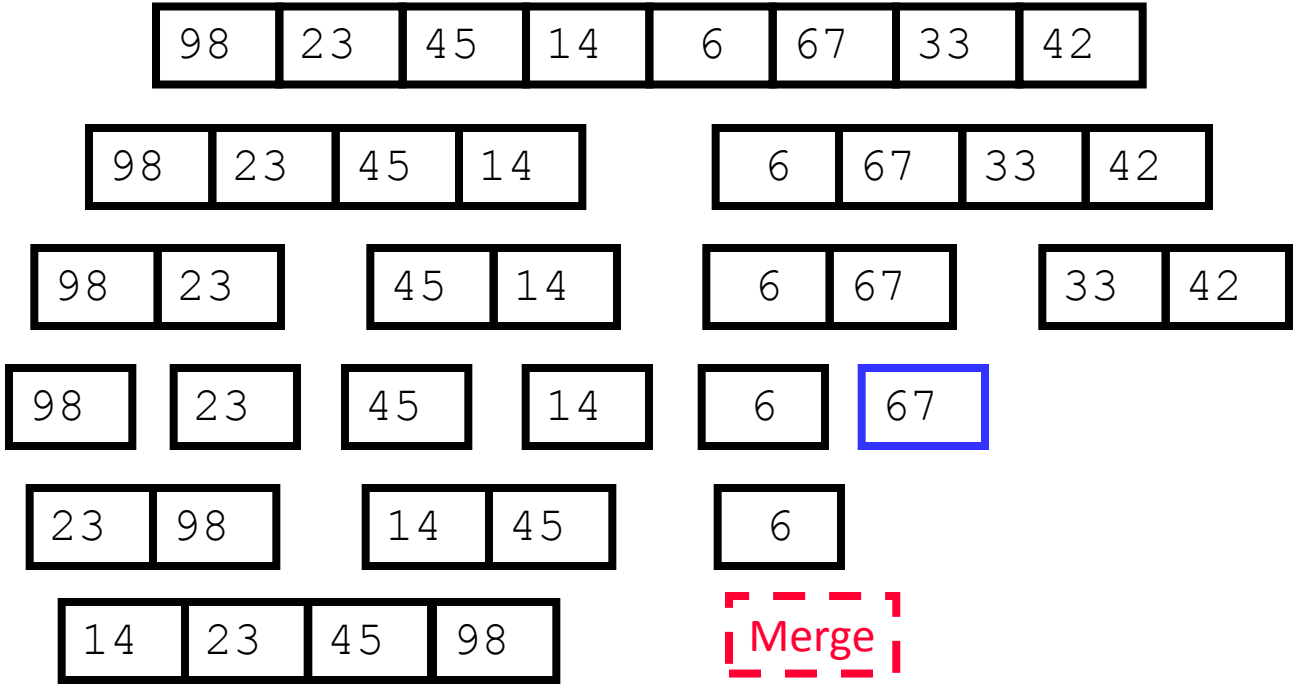

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
23 98
14 45
14 23 45 98

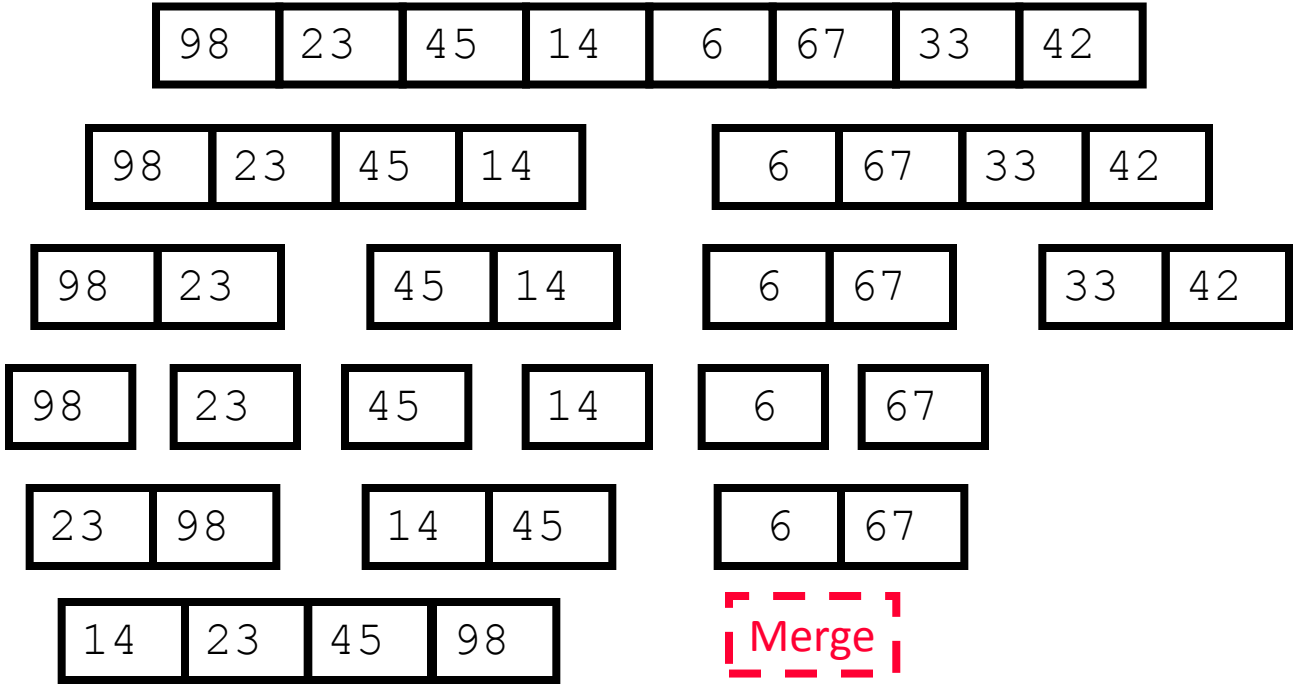
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
14 23 45 98

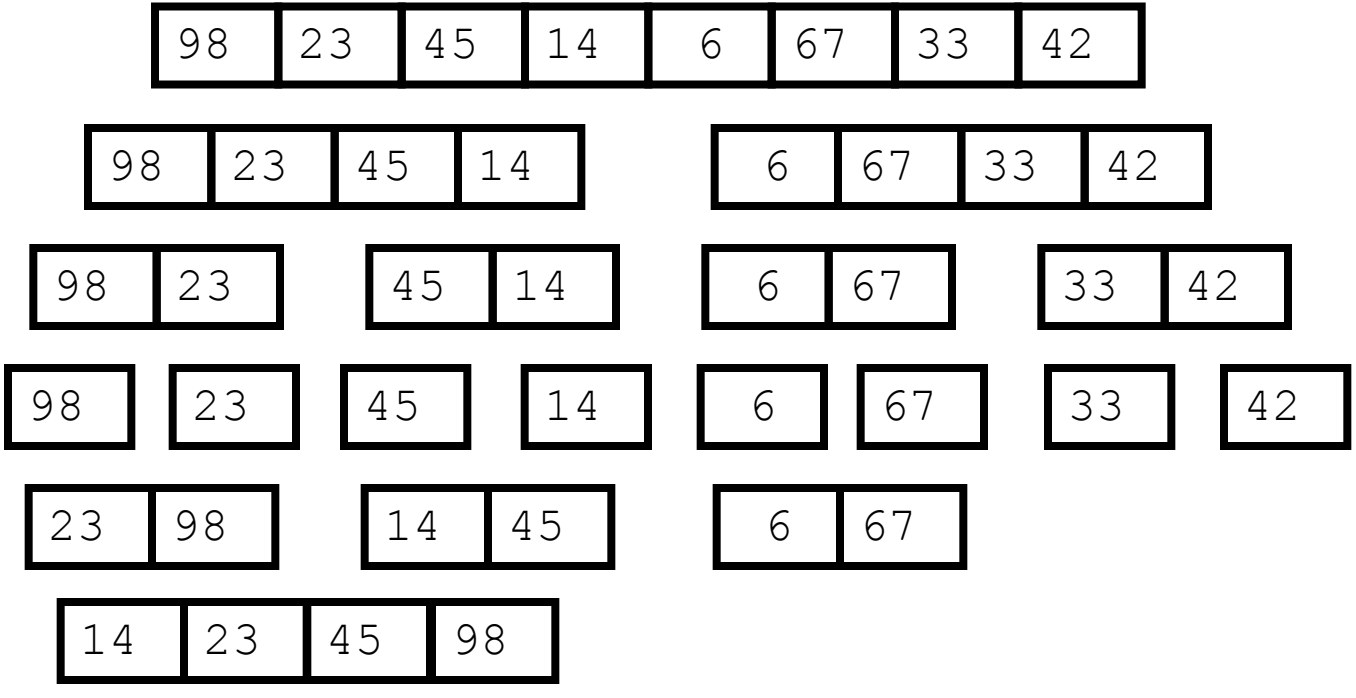
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
14 23 45 98
Merge

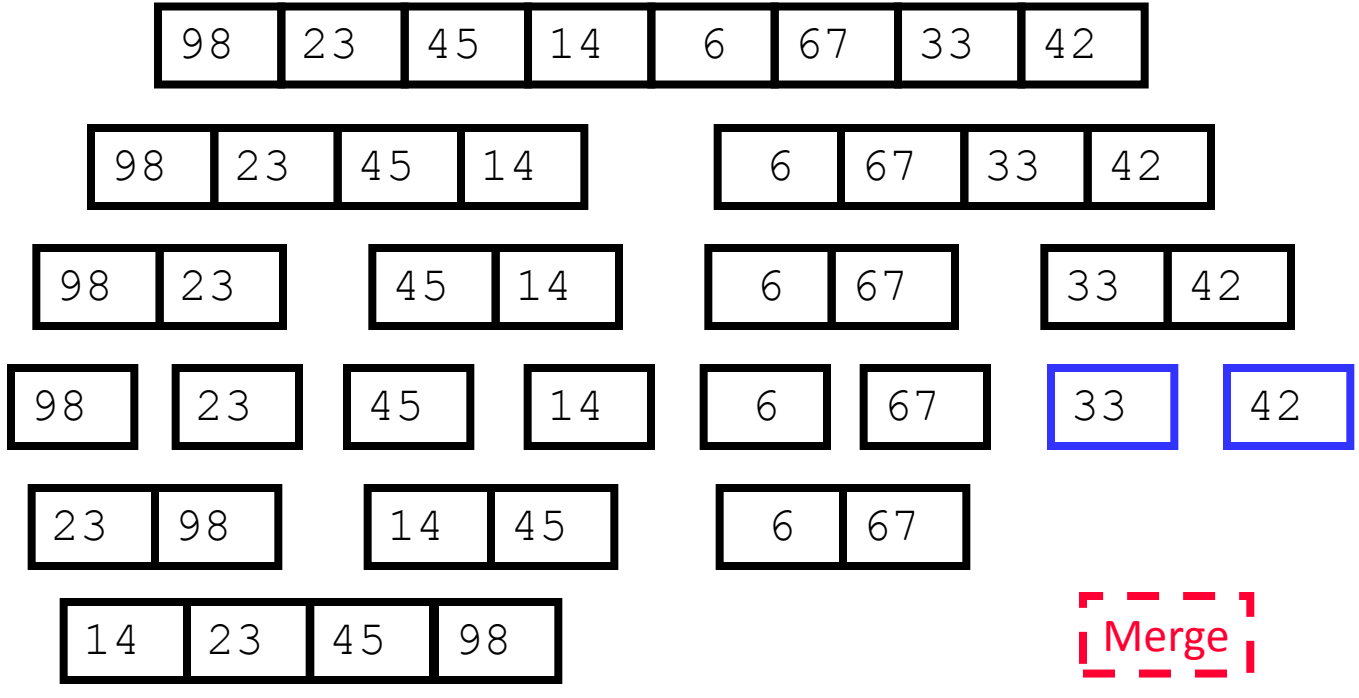

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
6
14 23 45 98
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
6 67
14 23 45 98
Merge

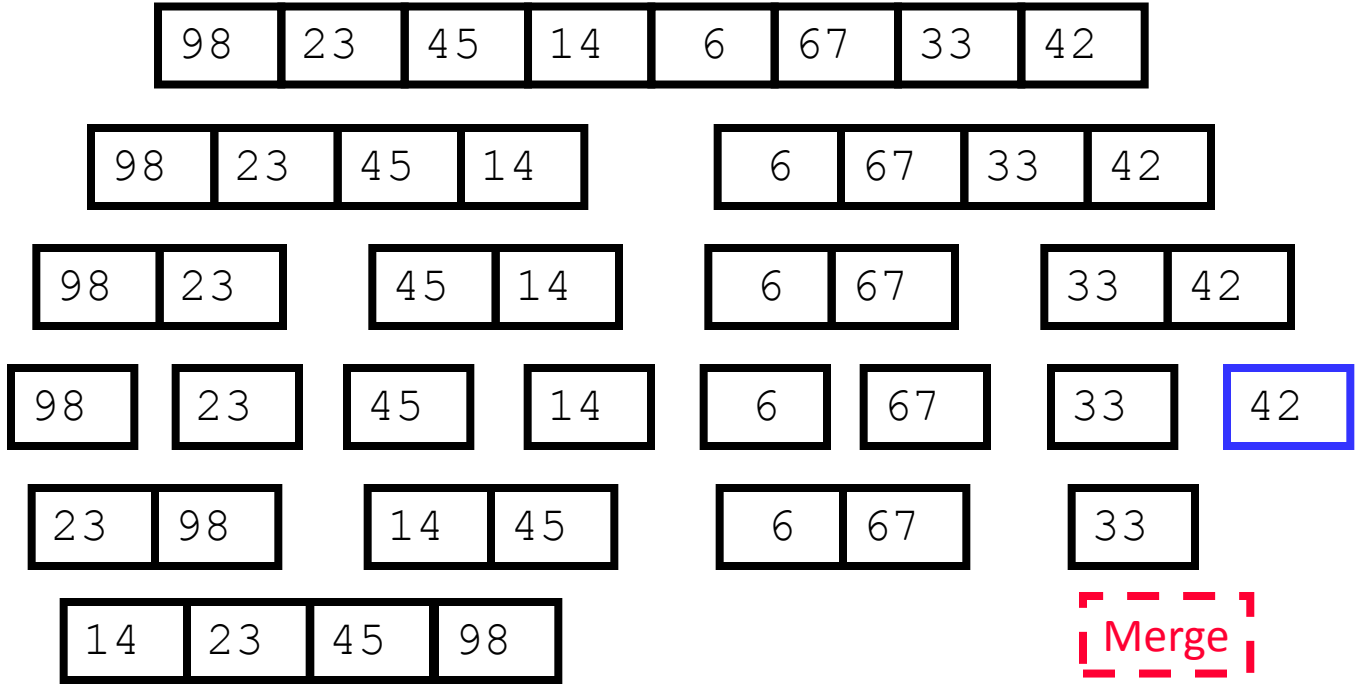
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
14 23 45 98

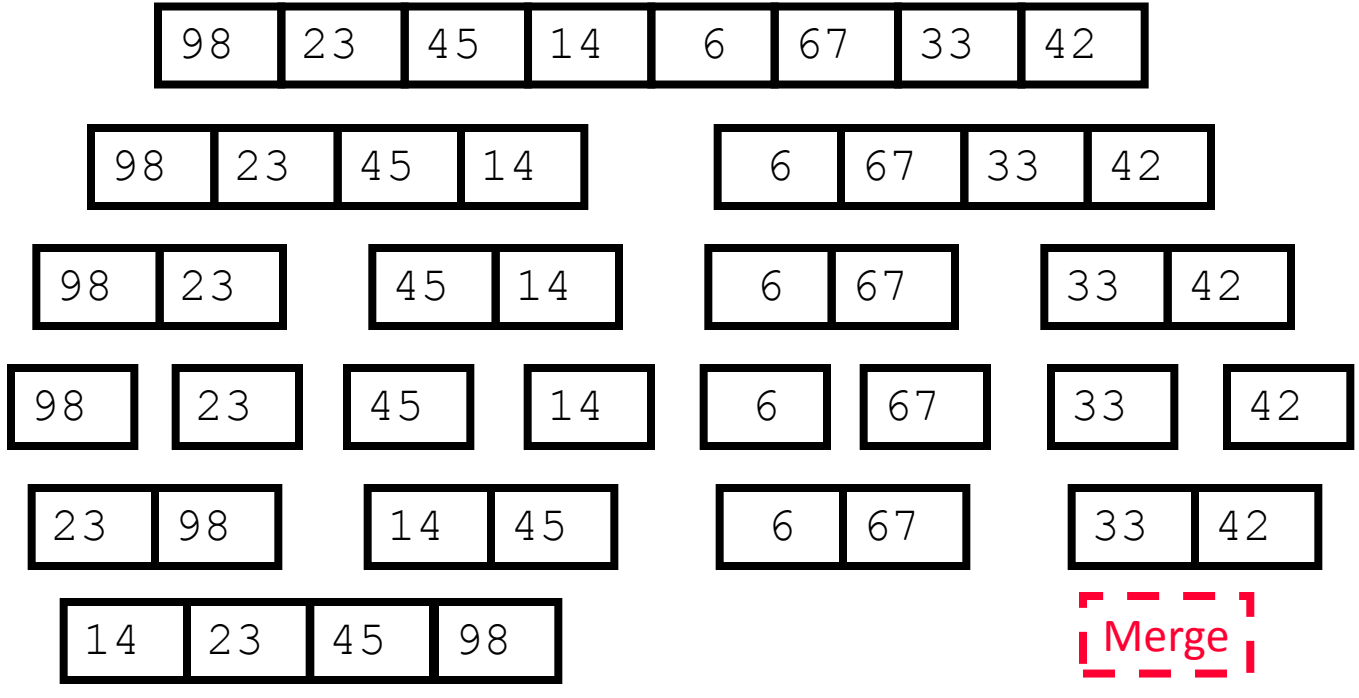
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
14 23 45 98
Merge

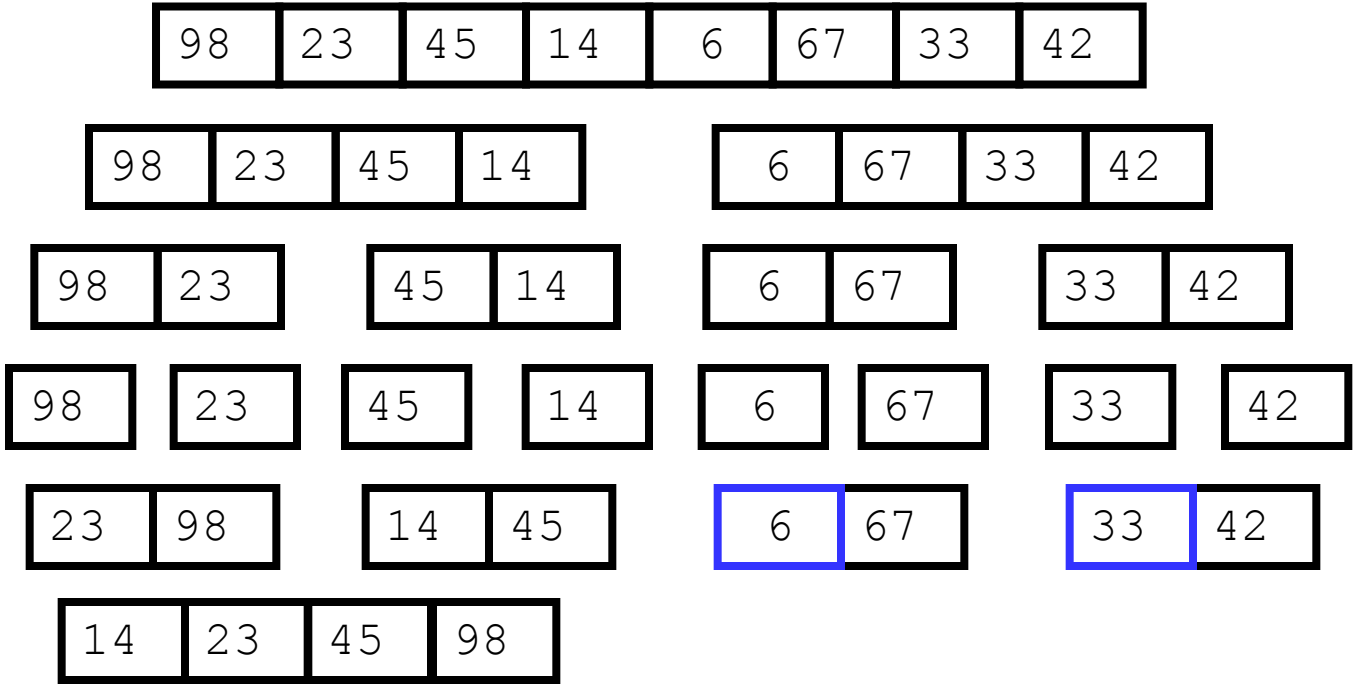
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33
14 23 45 98
Merge

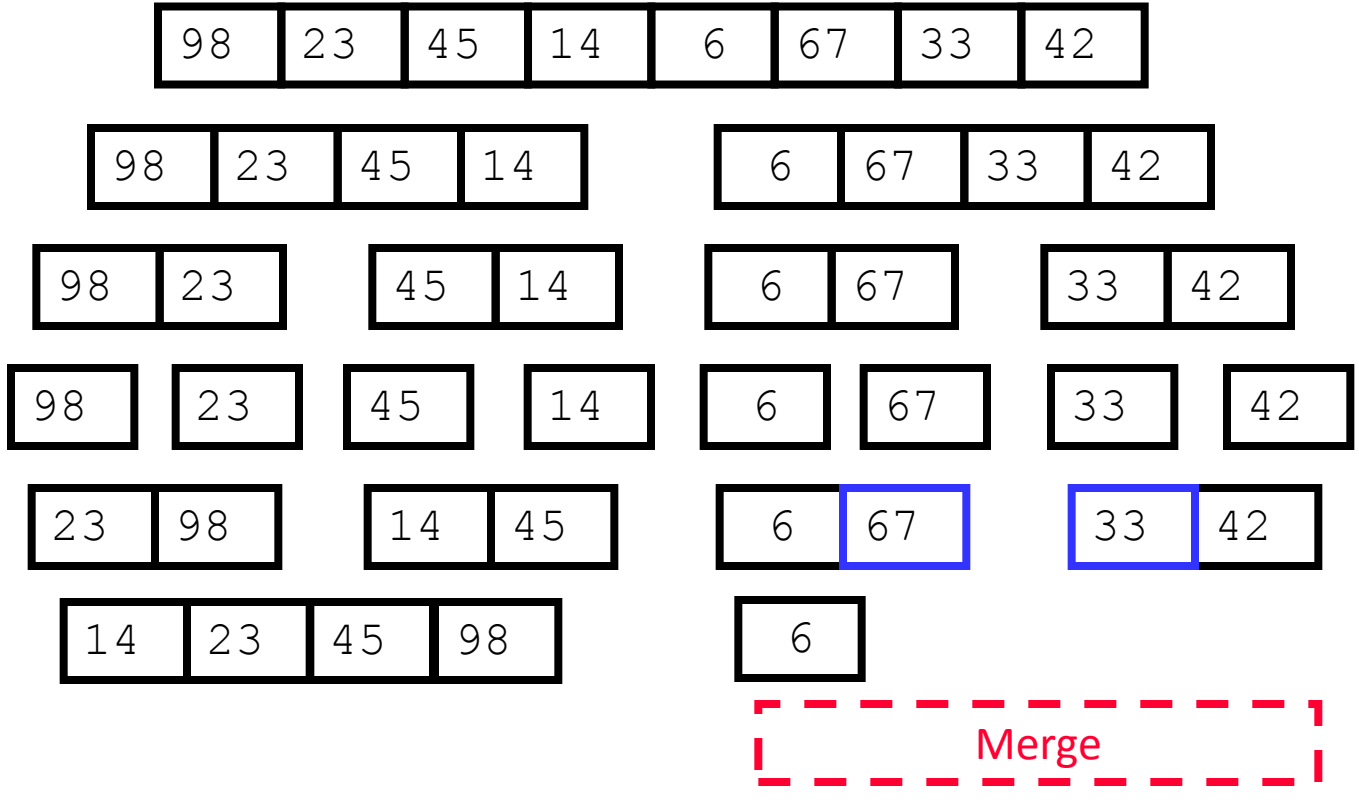
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
Merge

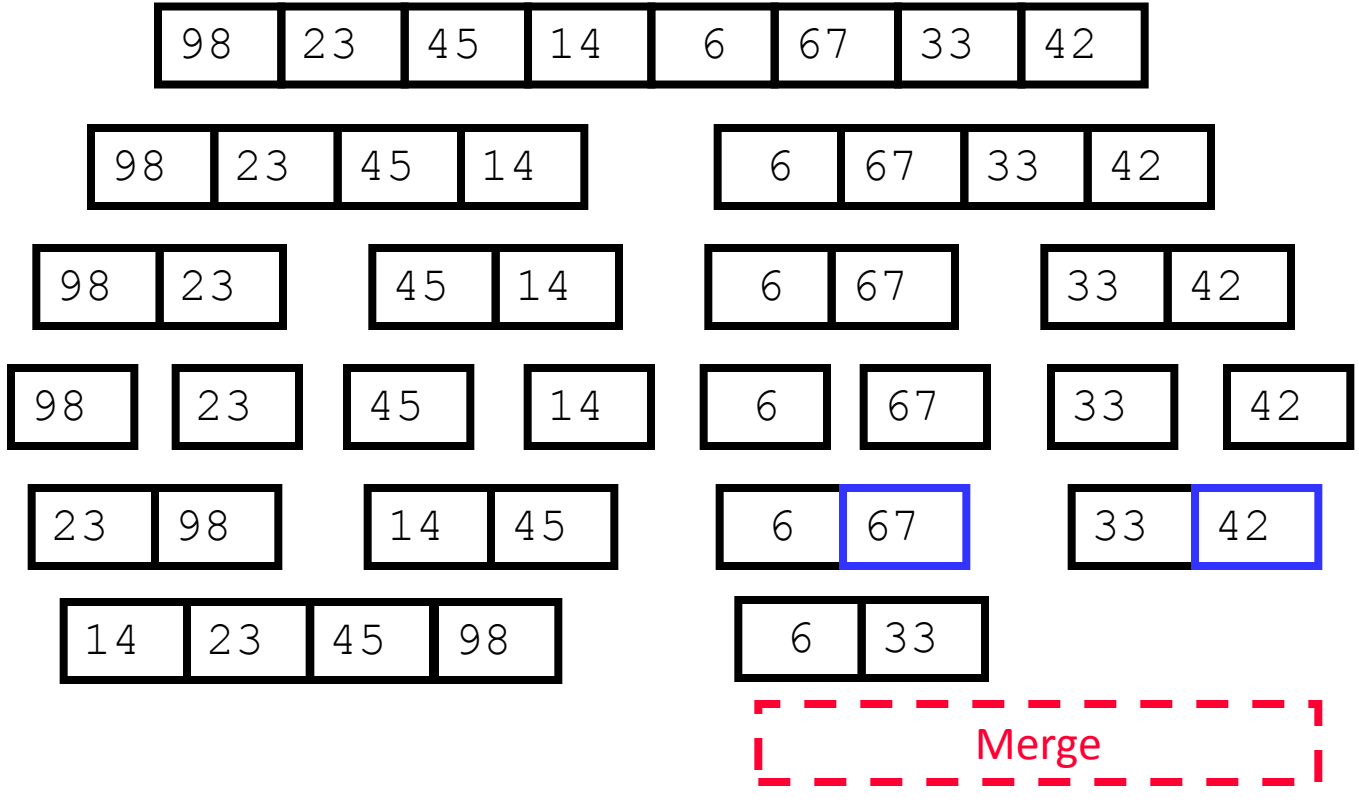
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
Merge

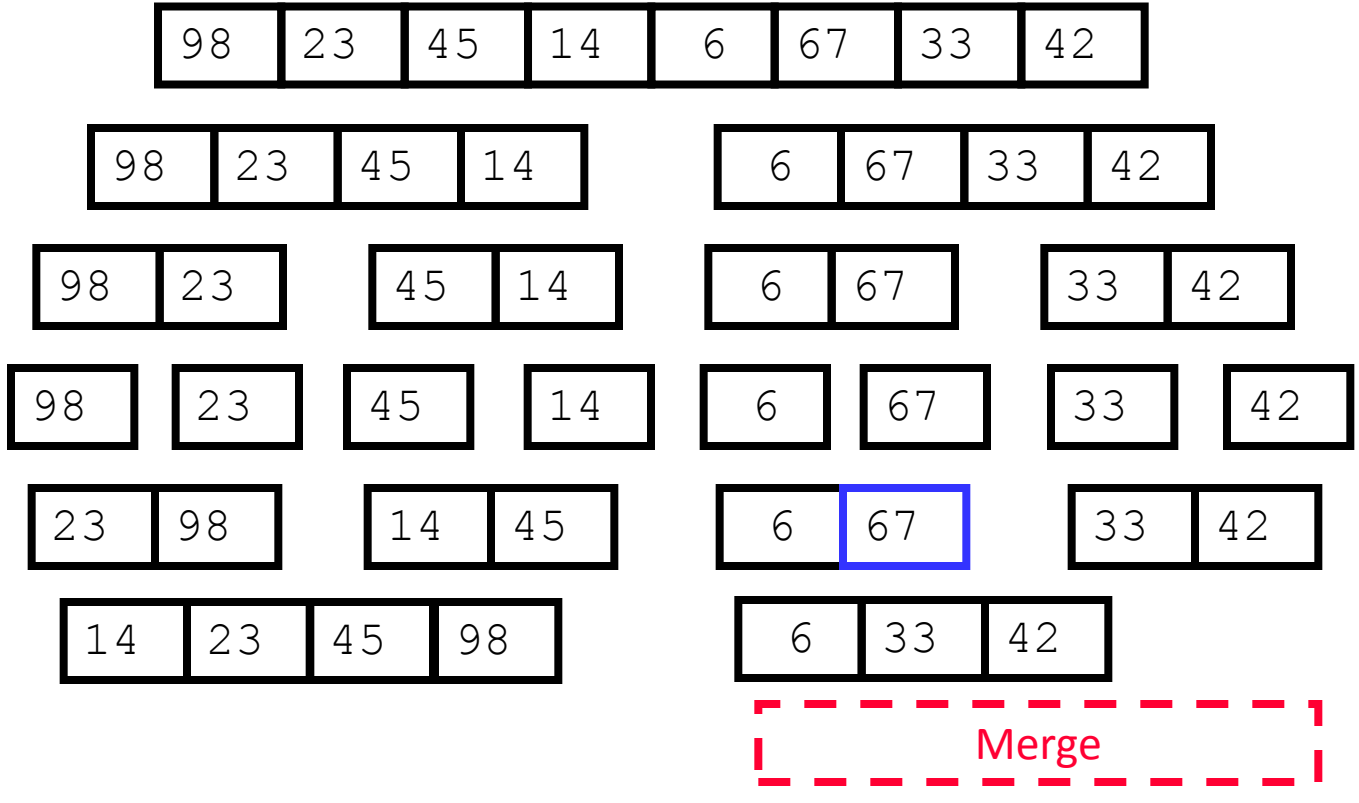
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6
Merge

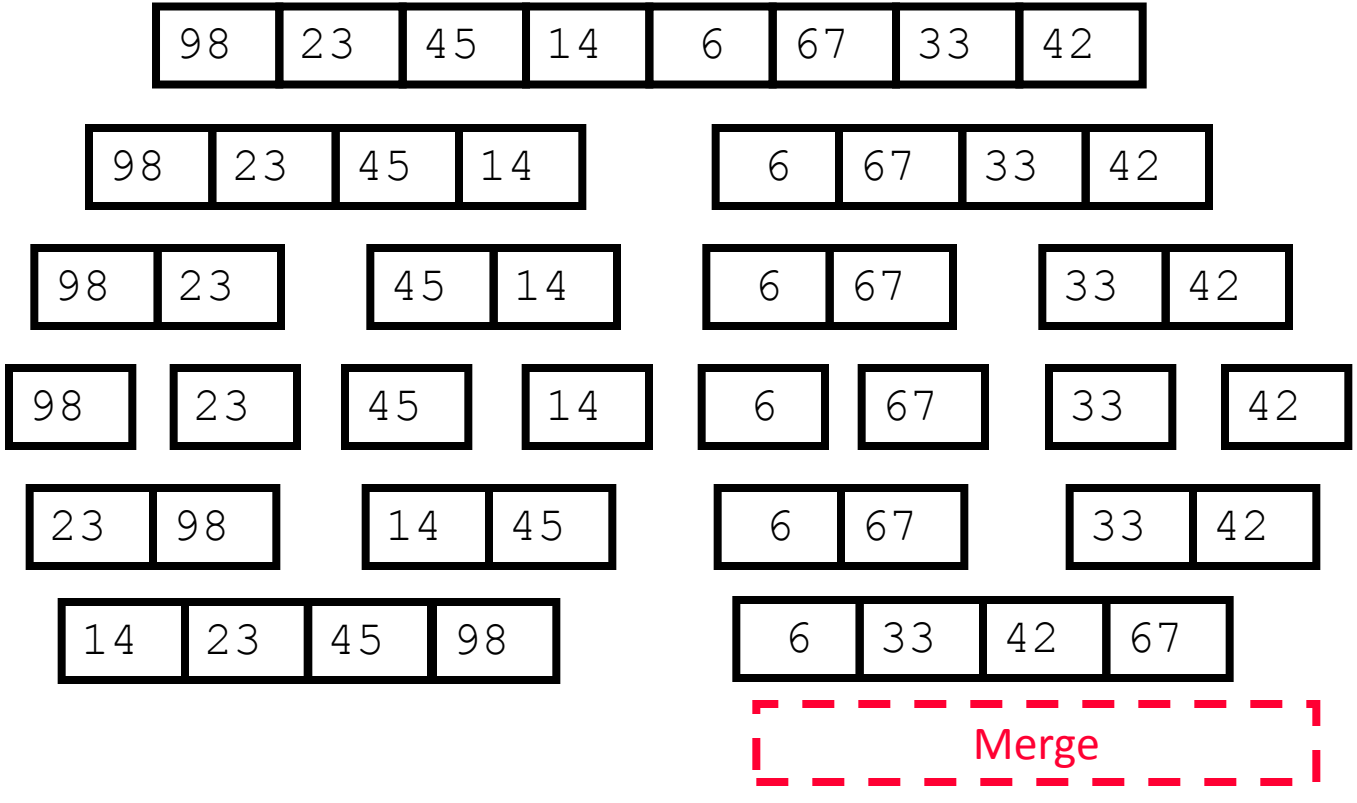
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33
Merge

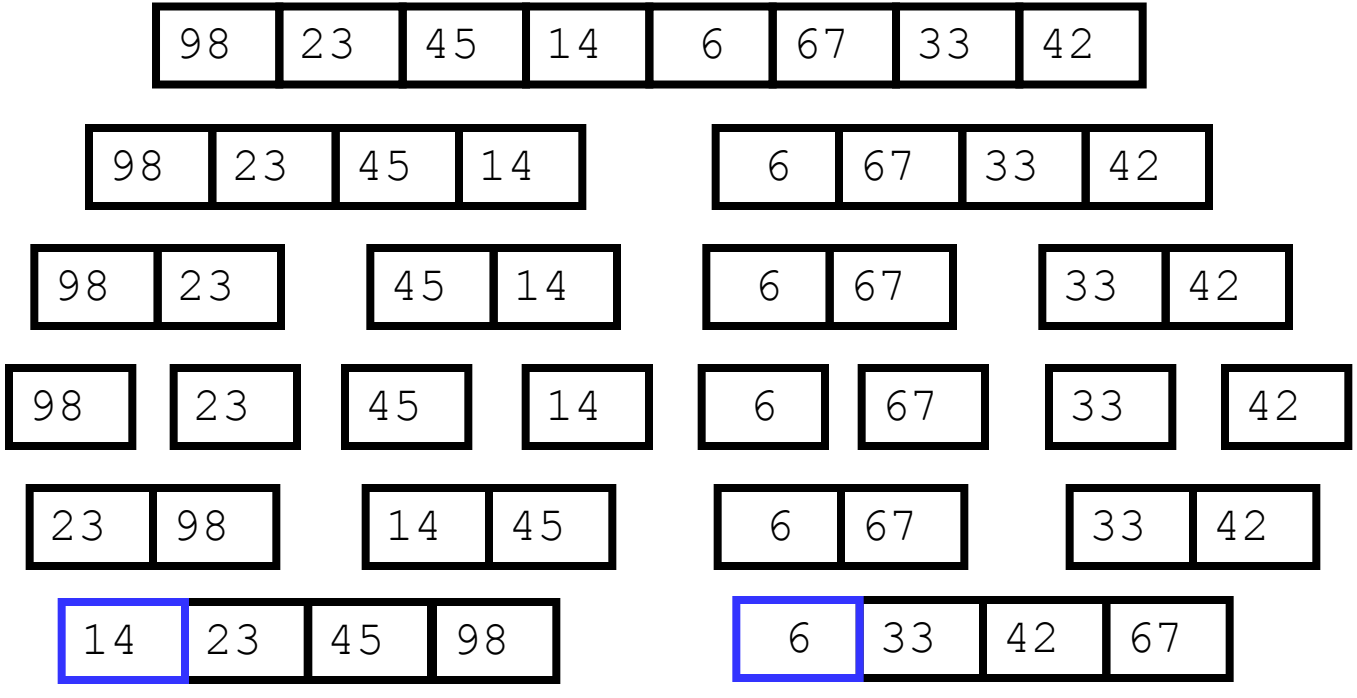
归并排序

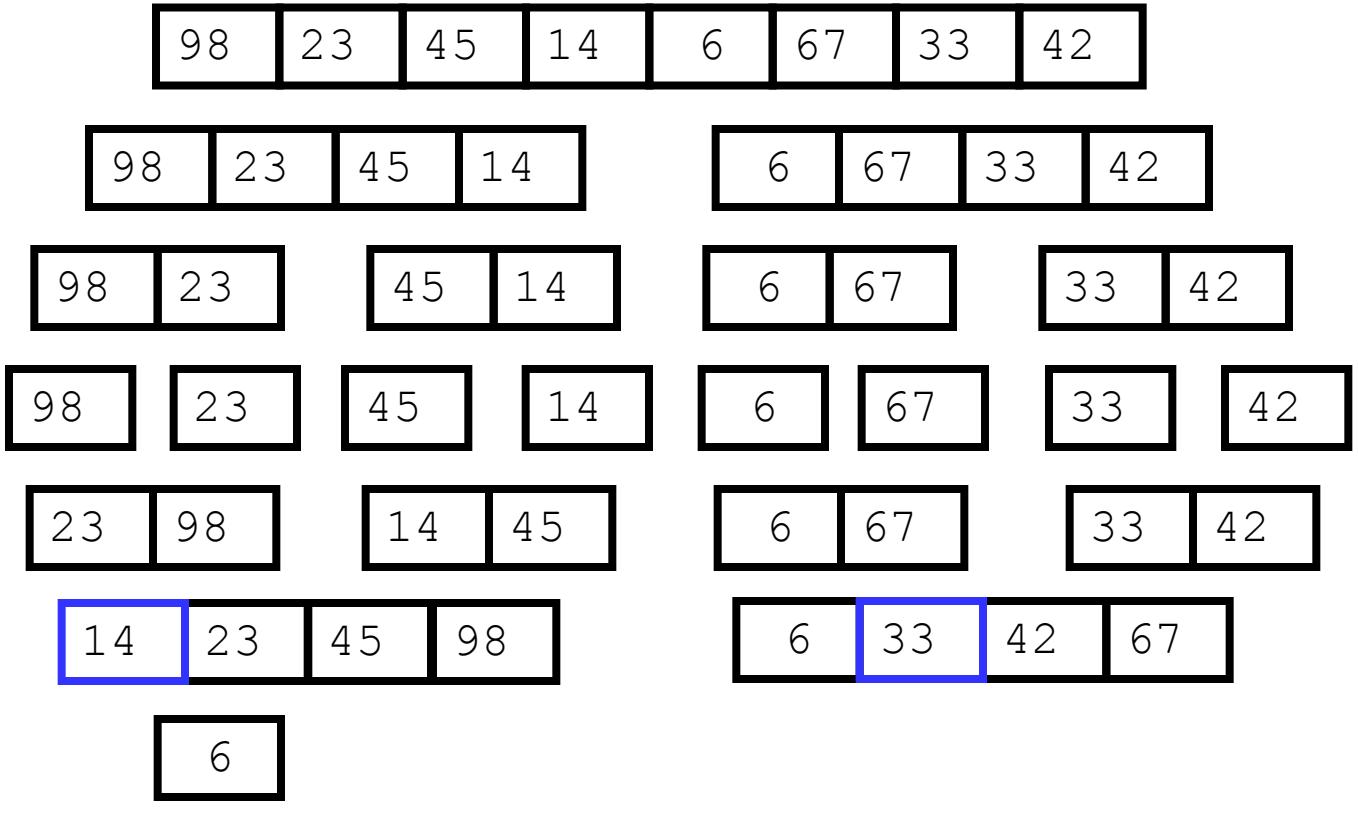

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42
Merge

归并排序

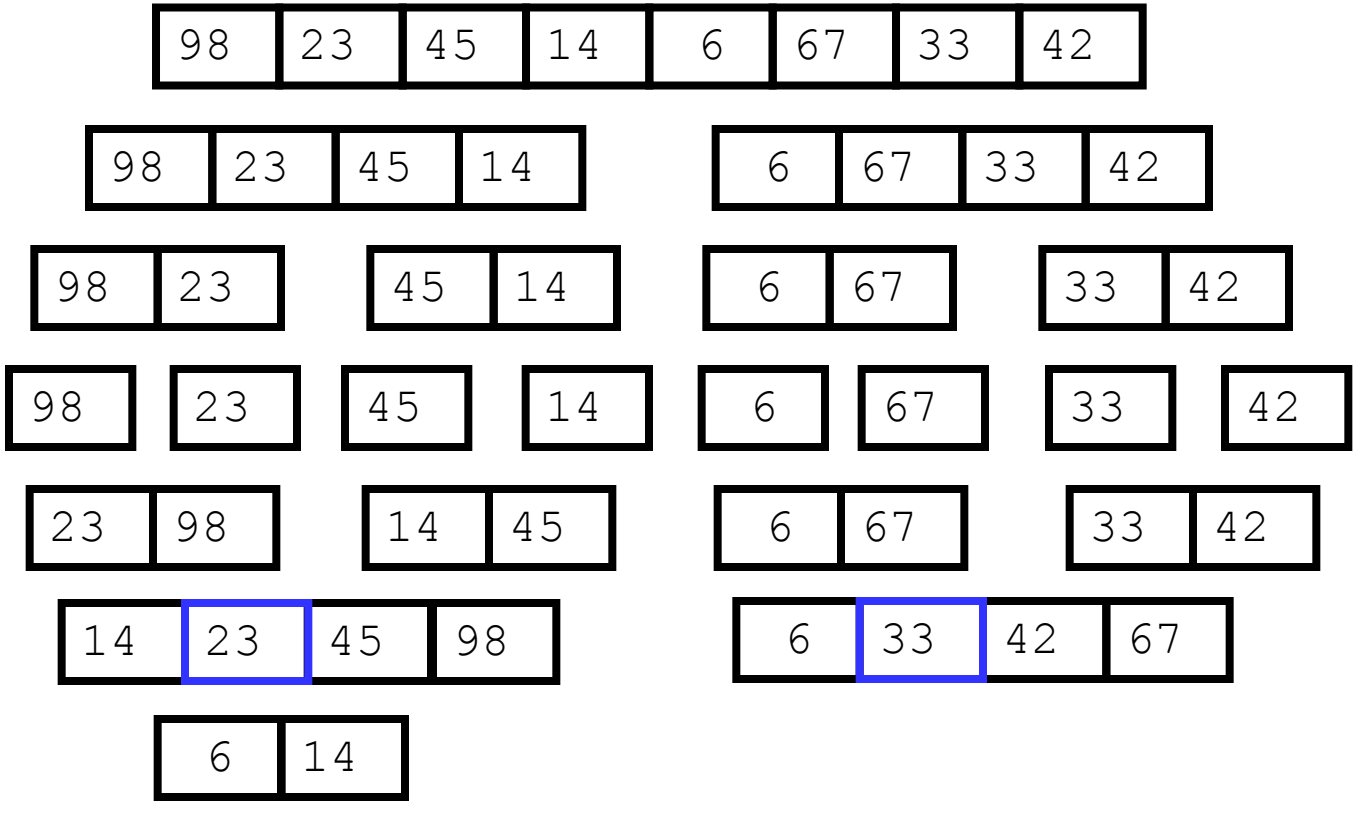

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
Merge

归并排序

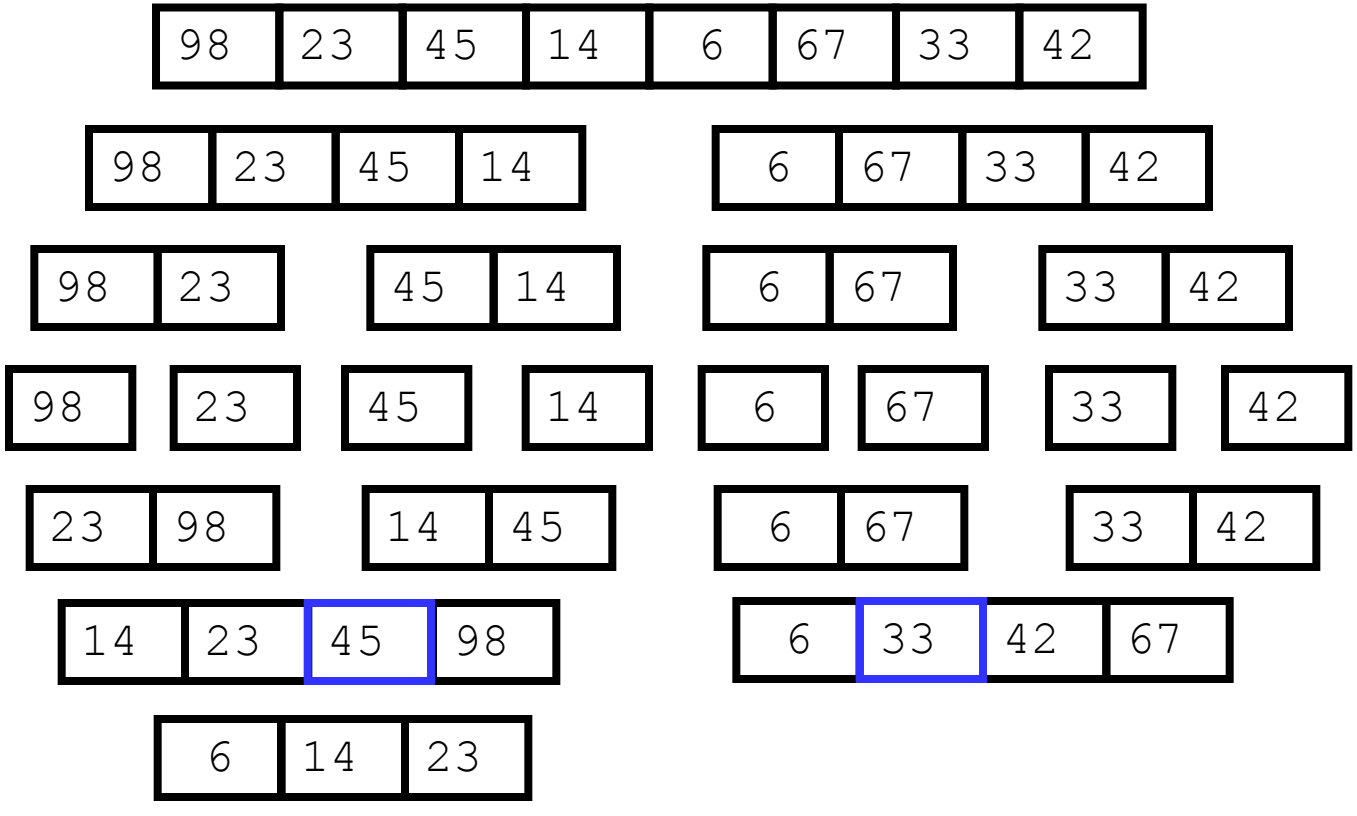

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6
Merge

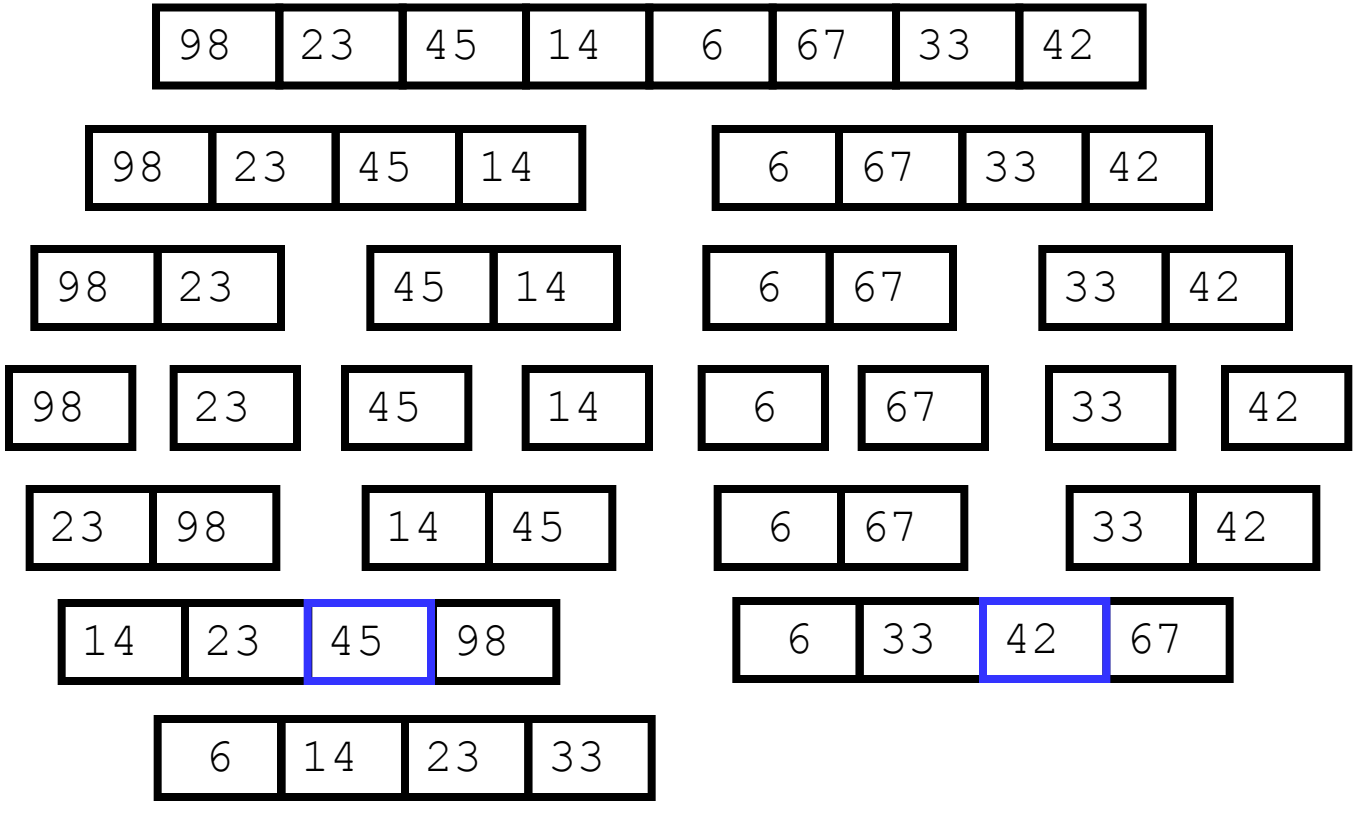
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14
Merge

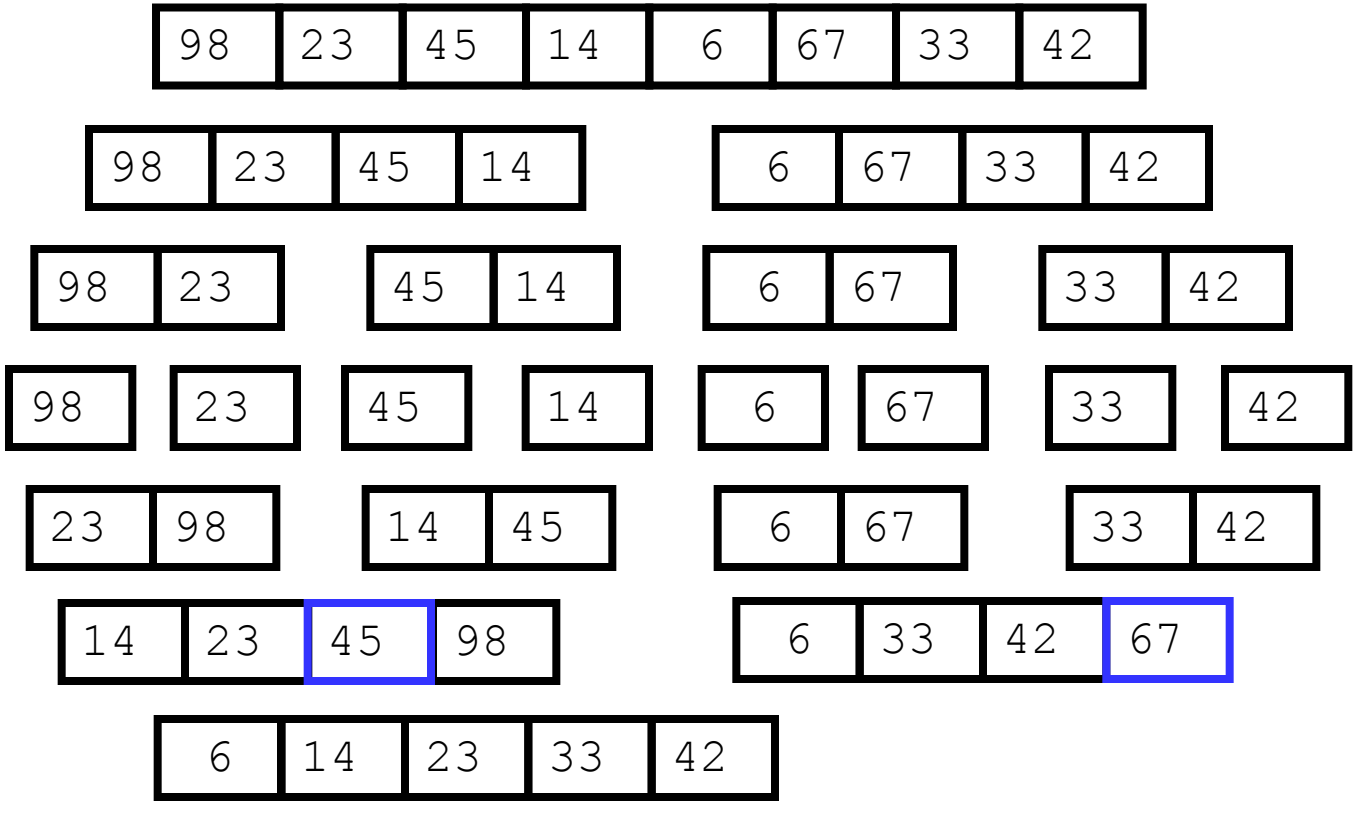
归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23
Merge

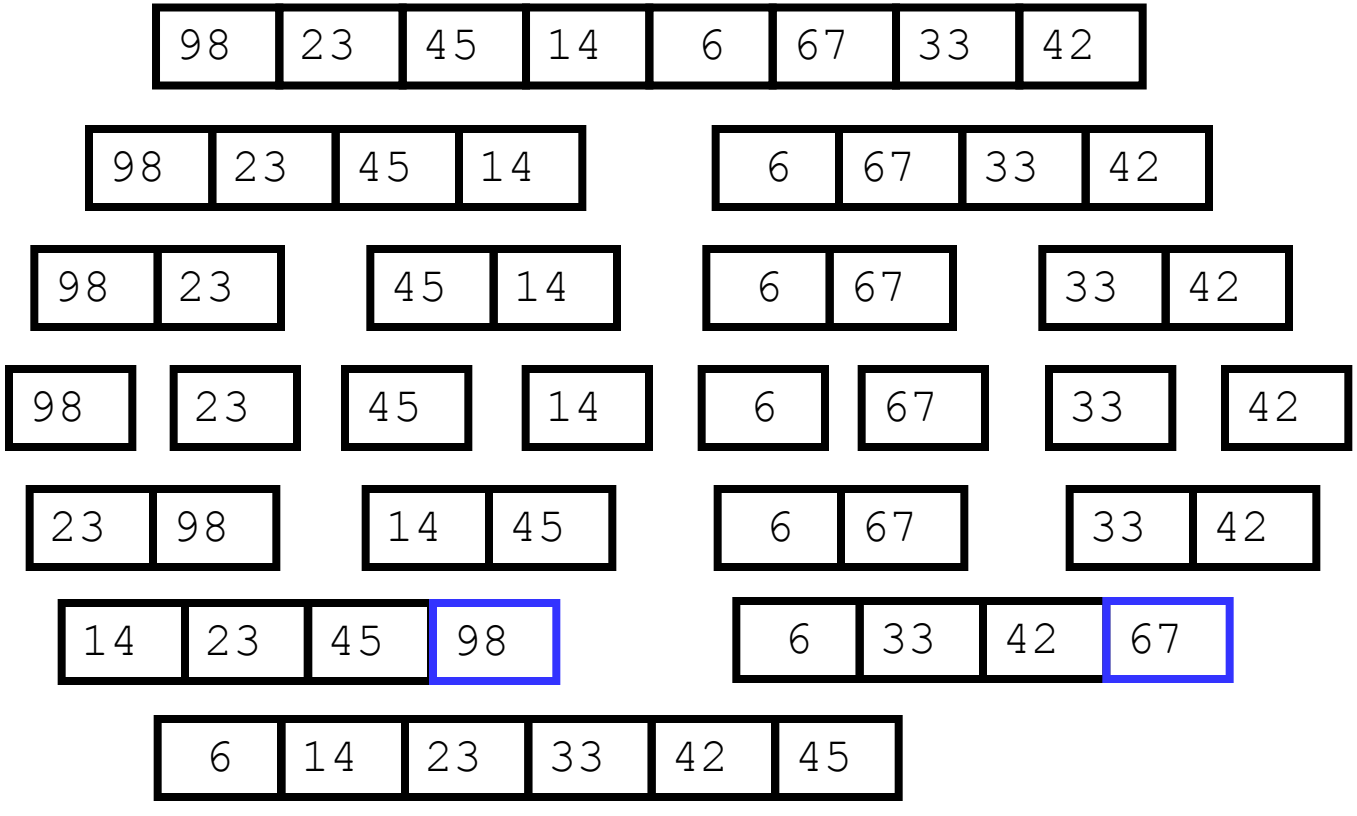
归并排序

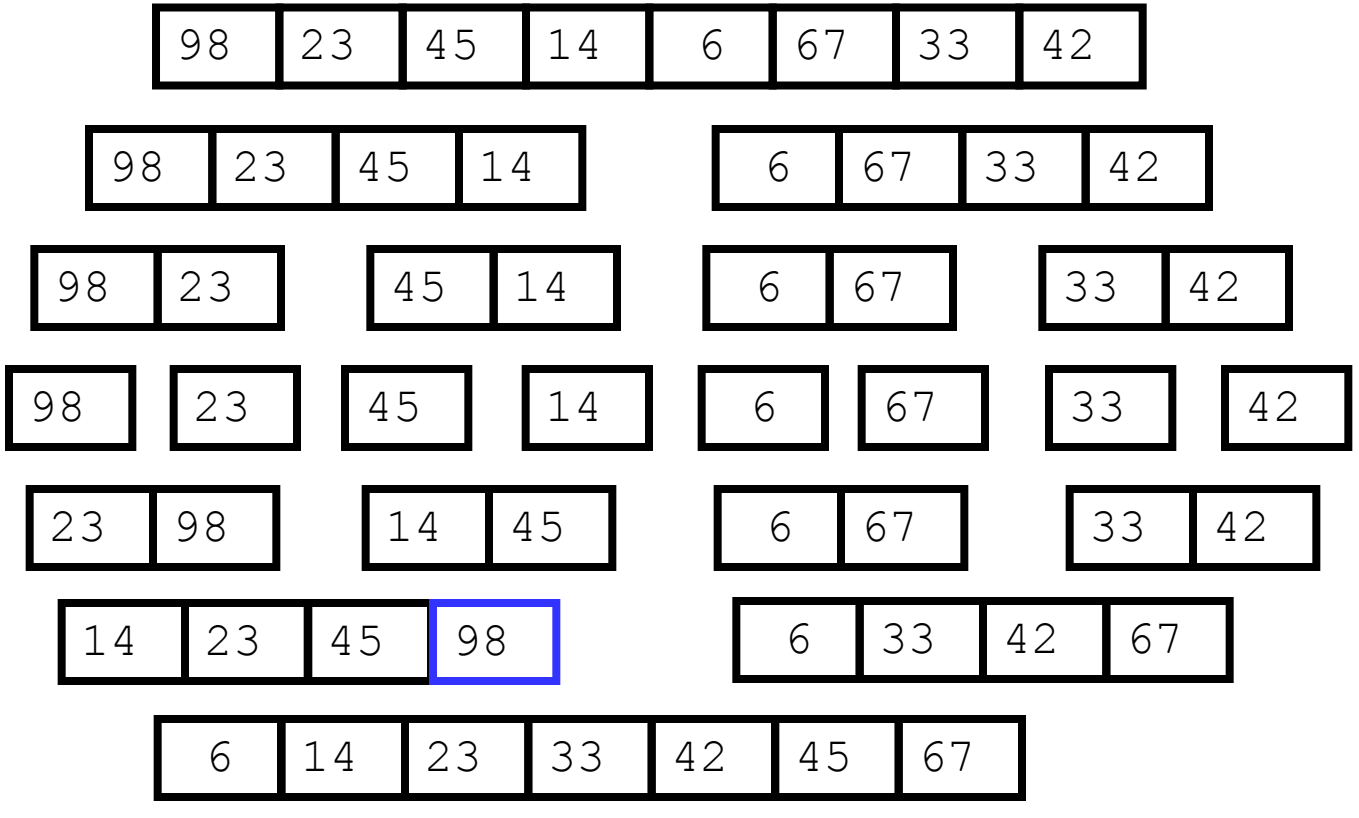

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33
Merge

归并排序

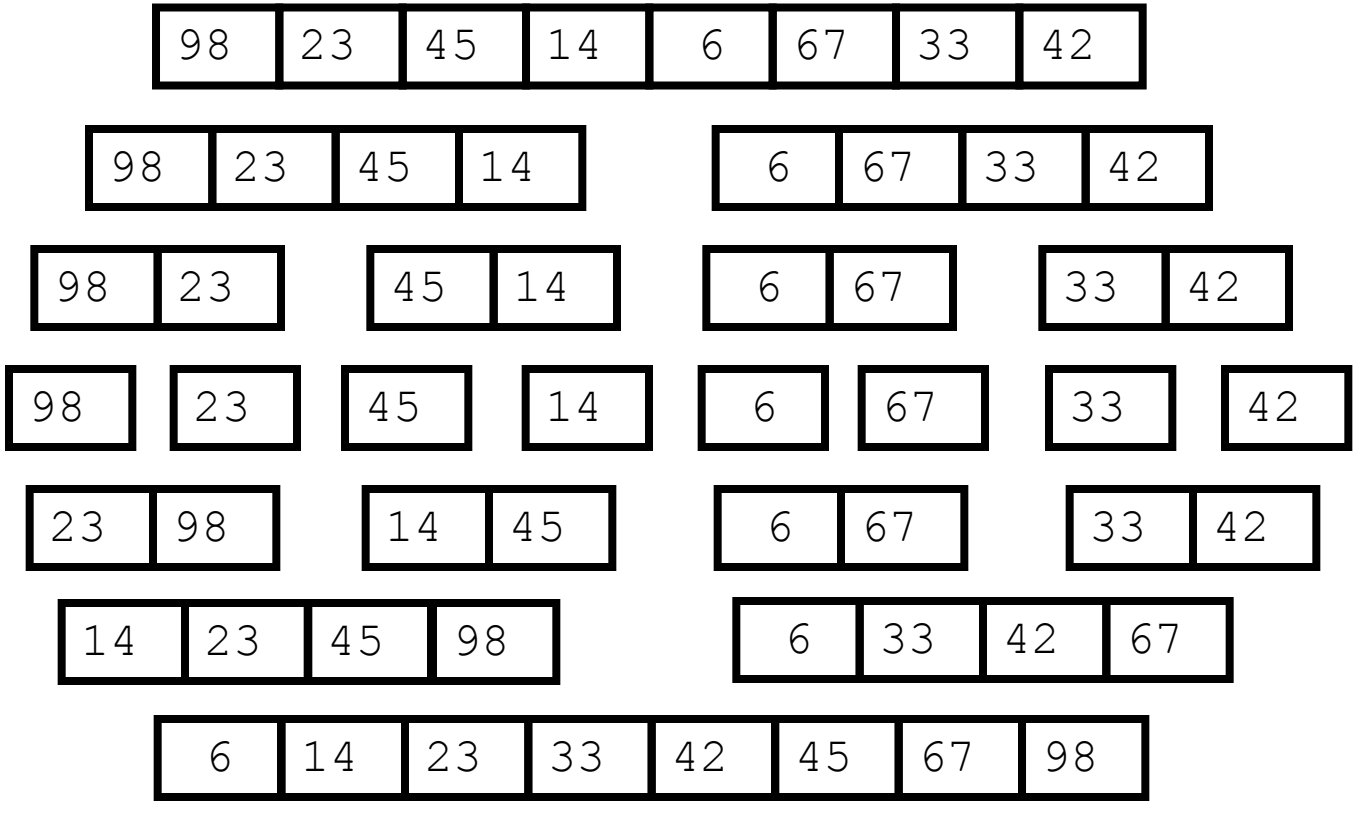

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42
Merge

归并排序

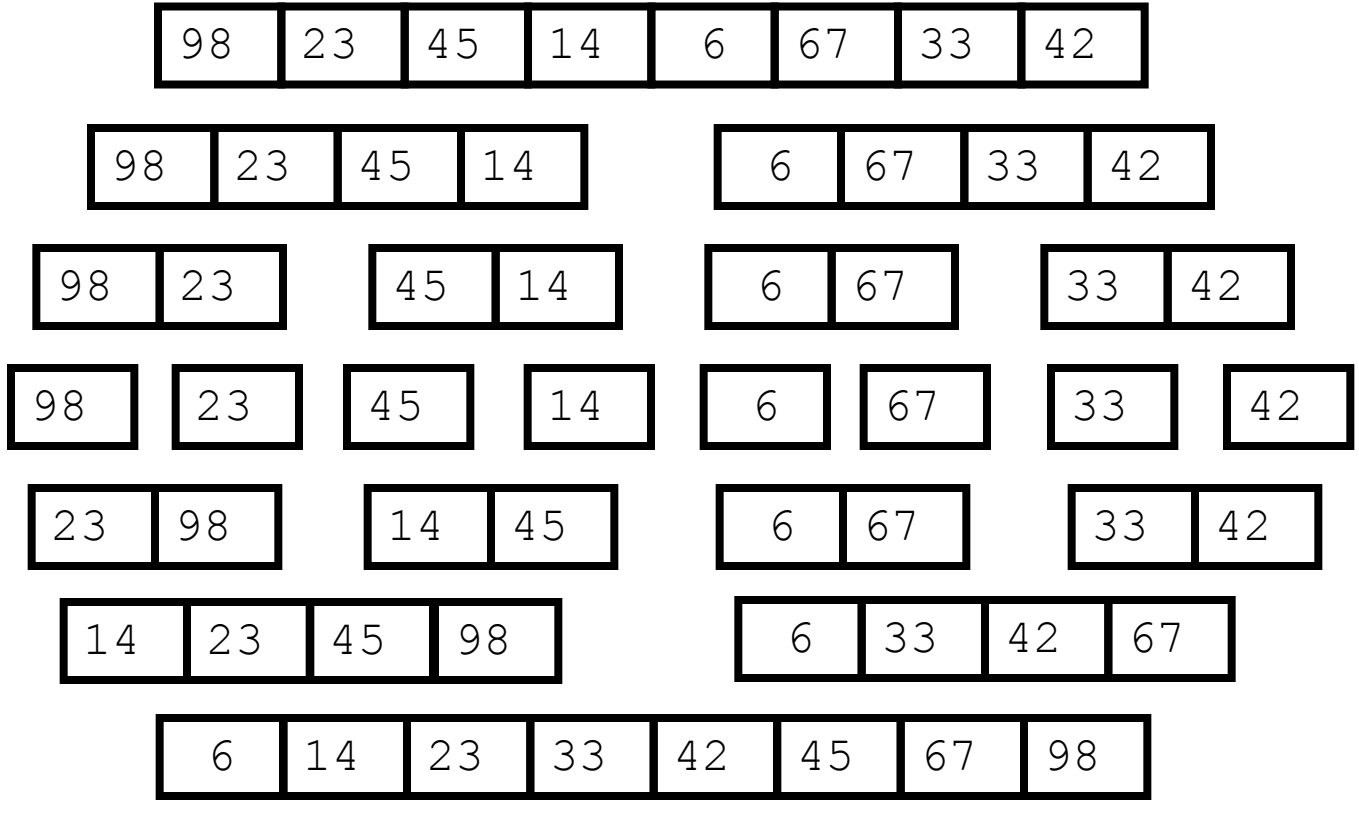

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45 67
Merge

归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45 67 98
Merge

归并排序


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45 67 98

归并排序算法的实现

```
def merge(left, right):                          #合并两个列表
    merged = []
    i, j = 0, 0                                  #i和j分别作为left和right的下标
    left_len, right_len = len(left), len(right)  #分别获取左右子列表的长度
    while i < left_len and j < right_len:        #循环归并左右子列表元素
        if left[i] <= right[j]:
            merged.append(left[i])               #归并左子列表元素
            i += 1
        else:
            merged.append(right[j])              #归并右子列表元素
            j += 1
    merged.extend(left[i:])                      #归并左子列表剩余元素
    merged.extend(right[j:])                     #归并右子列表剩余元素
    print(left,right,merged)                     #跟踪调试
    return merged                                #返回归并好的列表
```

归并排序算法的实现

```
def mergeSort(a):                    #归并排序
    if len(a) <= 1:                  #空或者只有1个元素，直接返回列表
        return a
    mid = len(a) // 2                #列表中间位置
    left = mergeSort(a[:mid])        #归并排序左子列表
    right = mergeSort(a[mid:])       #归并排序右子列表
    return merge(left, right)        #合并排好序的左右两个子列表

a = [98,23,45,14,6,67,33,42]
a1 = mergeSort(a)
print(a1)
```

- 归并排序算法的分析
  - 归并排序需要将列表一步步拆分成子列表，共$\log_2 N$步
  - 每一步都相当于需要合并N个元素的有序列表，最大比较次数是N次
  - 归并排序需要至多$N\log_2 N$次比较

- Python语言系统提供的排序算法：底层采用了一种归并排序算法实现
  - Python的列表类型提供sort()方法

```
a = [98,23,45,14,6,67,33,42]
a.sort()    #默认升序排序
print(a)    #输出[6, 14, 23, 33, 42, 45, 67, 98]
```

```
a = [98,23,45,14,6,67,33,42]
a.sort(reverse=True)            #降序排序
print(a)                        #输出[98, 67, 45, 42, 33, 23, 14, 6]
```

- Python语言系统提供的排序算法：底层采用了一种归并排序算法实现
  - Python的内置函数sorted

```
a = [98,23,45,14,6,67,33,42]
b = sorted(a)          #降序采用sorted(a,reverse=True)
print(a)    #输出[98, 23, 45, 14, 6, 67, 33, 42]不变
print(b)    #输出[6, 14, 23, 33, 42, 45, 67, 98]
```

大学计算机-Python算法实践