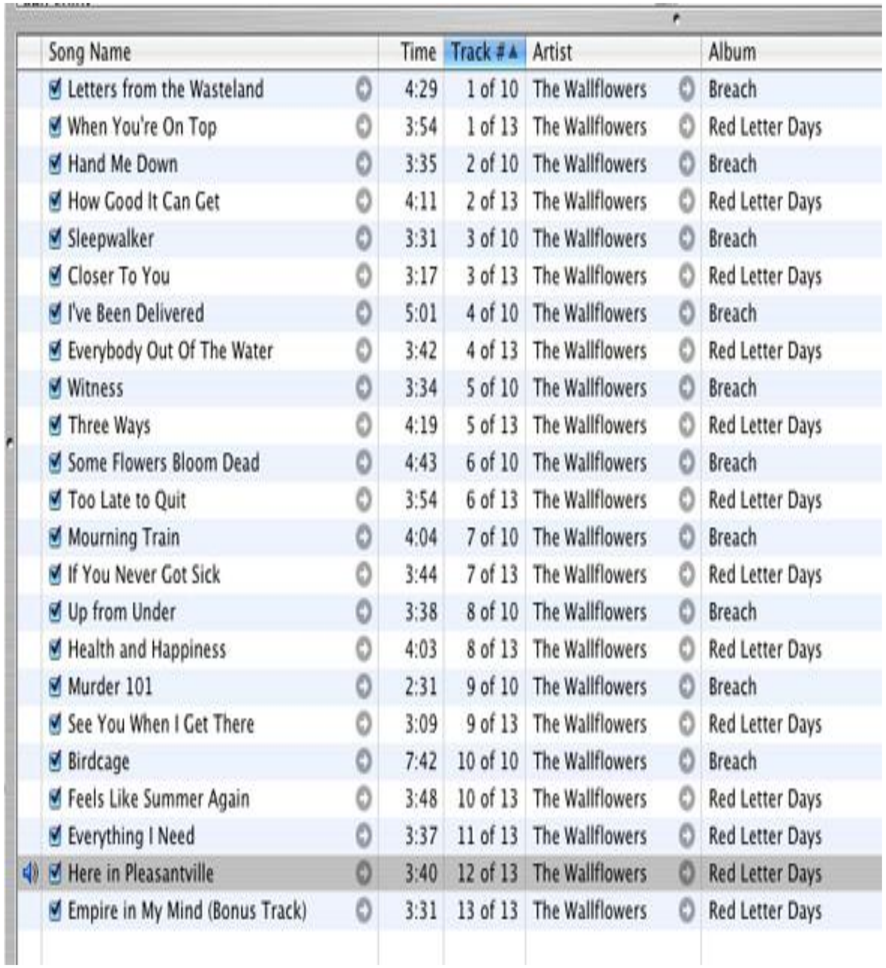
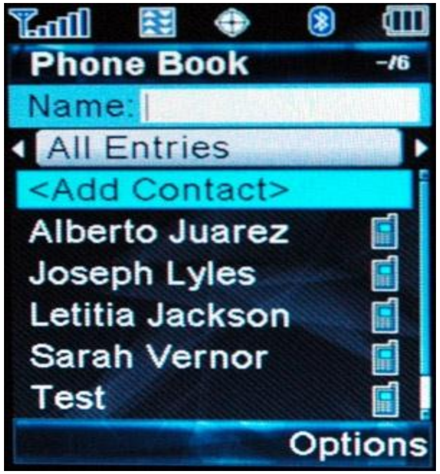
Sorting Algorithms

Sorting Algorithms



	Song Name	Time	Track # ▲	Artist	Album
☑	Letters from the Wasteland	4:29	1 of 10	The Wallflowers	Breach
☑	When You're On Top	3:54	1 of 13	The Wallflowers	Red Letter Days
☑	Hand Me Down	3:35	2 of 10	The Wallflowers	Breach
☑	How Good It Can Get	4:11	2 of 13	The Wallflowers	Red Letter Days
☑	Sleepwalker	3:31	3 of 10	The Wallflowers	Breach
☑	Closer To You	3:17	3 of 13	The Wallflowers	Red Letter Days
☑	I've Been Delivered	5:01	4 of 10	The Wallflowers	Breach
☑	Everybody Out Of The Water	3:42	4 of 13	The Wallflowers	Red Letter Days
☑	Witness	3:34	5 of 10	The Wallflowers	Breach
☑	Three Ways	4:19	5 of 13	The Wallflowers	Red Letter Days
☑	Some Flowers Bloom Dead	4:43	6 of 10	The Wallflowers	Breach
☑	Too Late to Quit	3:54	6 of 13	The Wallflowers	Red Letter Days
☑	Mourning Train	4:04	7 of 10	The Wallflowers	Breach
☑	If You Never Got Sick	3:44	7 of 13	The Wallflowers	Red Letter Days
☑	Up from Under	3:38	8 of 10	The Wallflowers	Breach
☑	Health and Happiness	4:03	8 of 13	The Wallflowers	Red Letter Days
☑	Murder 101	2:31	9 of 10	The Wallflowers	Breach
☑	See You When I Get There	3:09	9 of 13	The Wallflowers	Red Letter Days
☑	Birdcage	7:42	10 of 10	The Wallflowers	Breach
☑	Feels Like Summer Again	3:48	10 of 13	The Wallflowers	Red Letter Days
☑	Everything I Need	3:37	11 of 13	The Wallflowers	Red Letter Days
☑	Here in Pleasantville	3:40	12 of 13	The Wallflowers	Red Letter Days
☑	Empire in My Mind (Bonus Track)	3:31	13 of 13	The Wallflowers	Red Letter Days



Phone Book -/6

Name:

All Entries

<Add Contact>

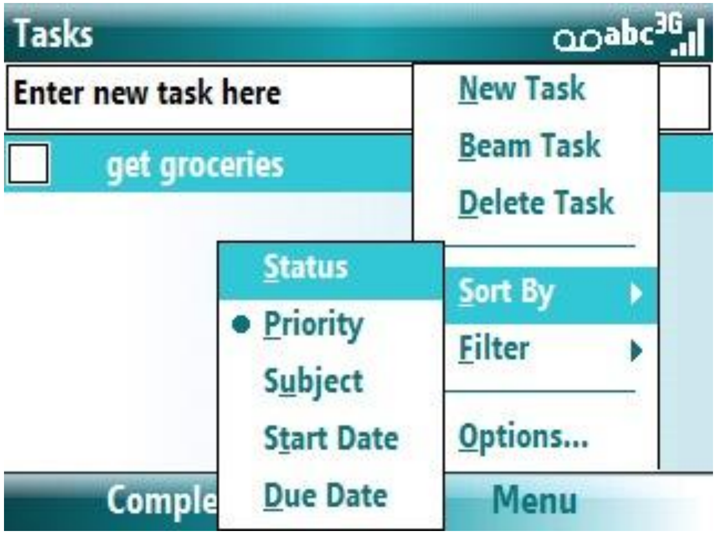
Alberto Juarez

Joseph Lyles

Letitia Jackson

Sarah Vernor

Test



Tasks

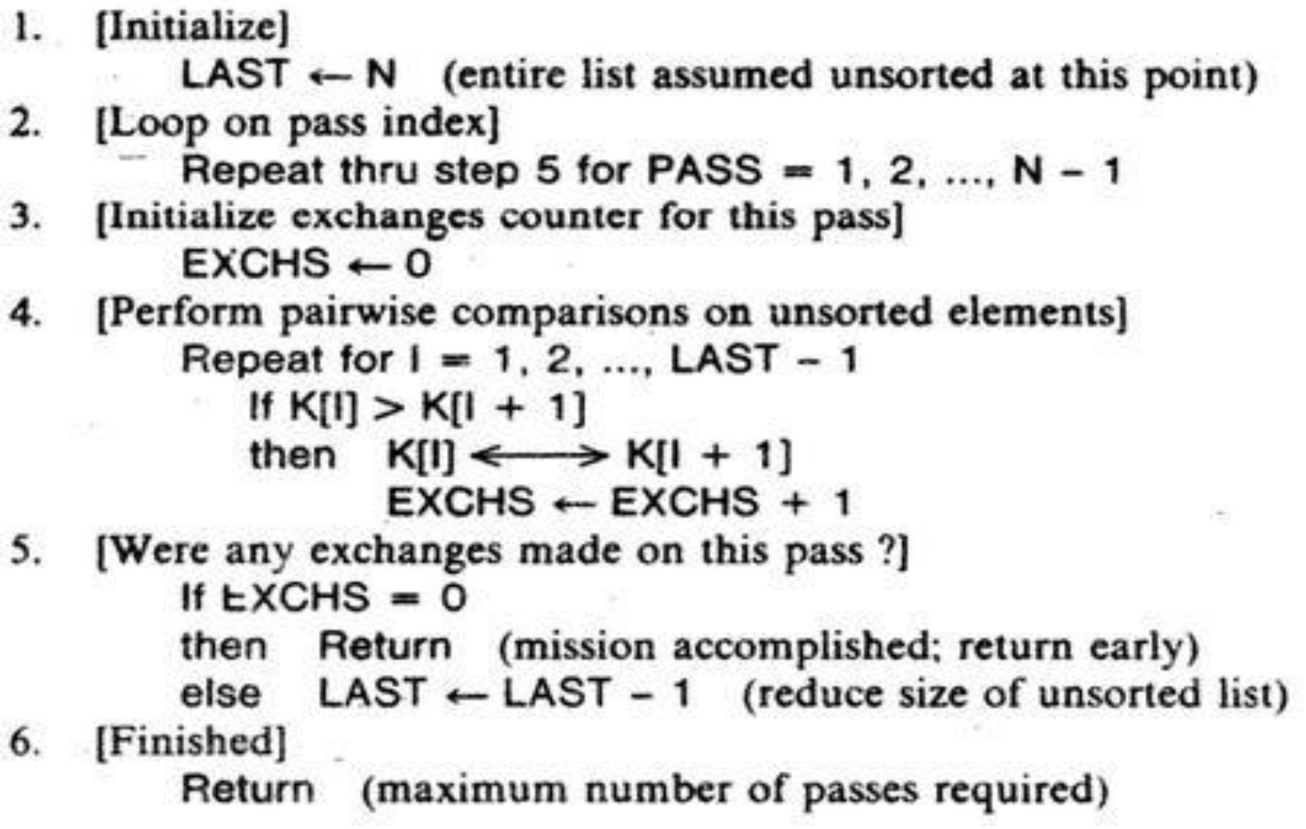
Enter new task here

□ get groceries

Bubble Sort

Bubble Sort

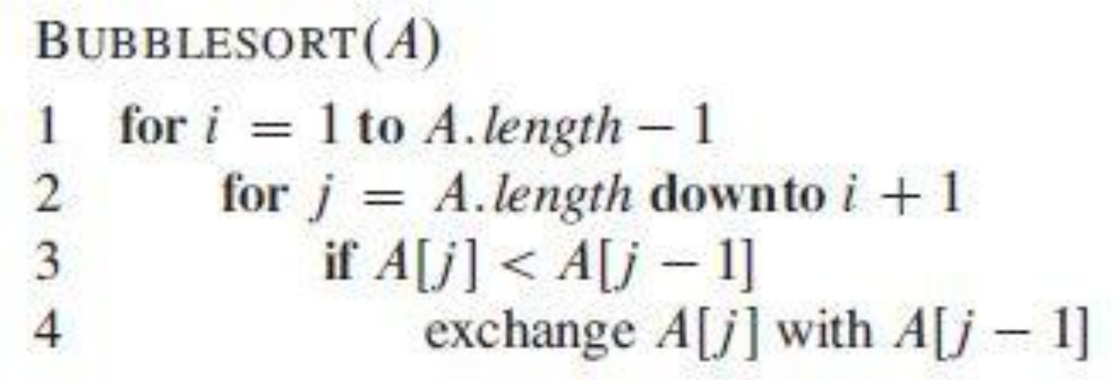
Function : BUBBLE_SORT (K,N)



```
1.  [Initialize]
        LAST ← N   (entire list assumed unsorted at this point)
2.  [Loop on pass index]
        Repeat thru step 5 for PASS = 1, 2, ..., N − 1
3.  [Initialize exchanges counter for this pass]
        EXCHS ← 0
4.  [Perform pairwise comparisons on unsorted elements]
        Repeat for I = 1, 2, ..., LAST − 1
            If K[I] > K[I + 1]
            then   K[I] ←——→ K[I + 1]
                   EXCHS ← EXCHS + 1
5.  [Were any exchanges made on this pass ?]
        If EXCHS = 0
        then   Return   (mission accomplished; return early)
        else   LAST ← LAST − 1   (reduce size of unsorted list)
6.  [Finished]
        Return   (maximum number of passes required)
```

Bubble Sort

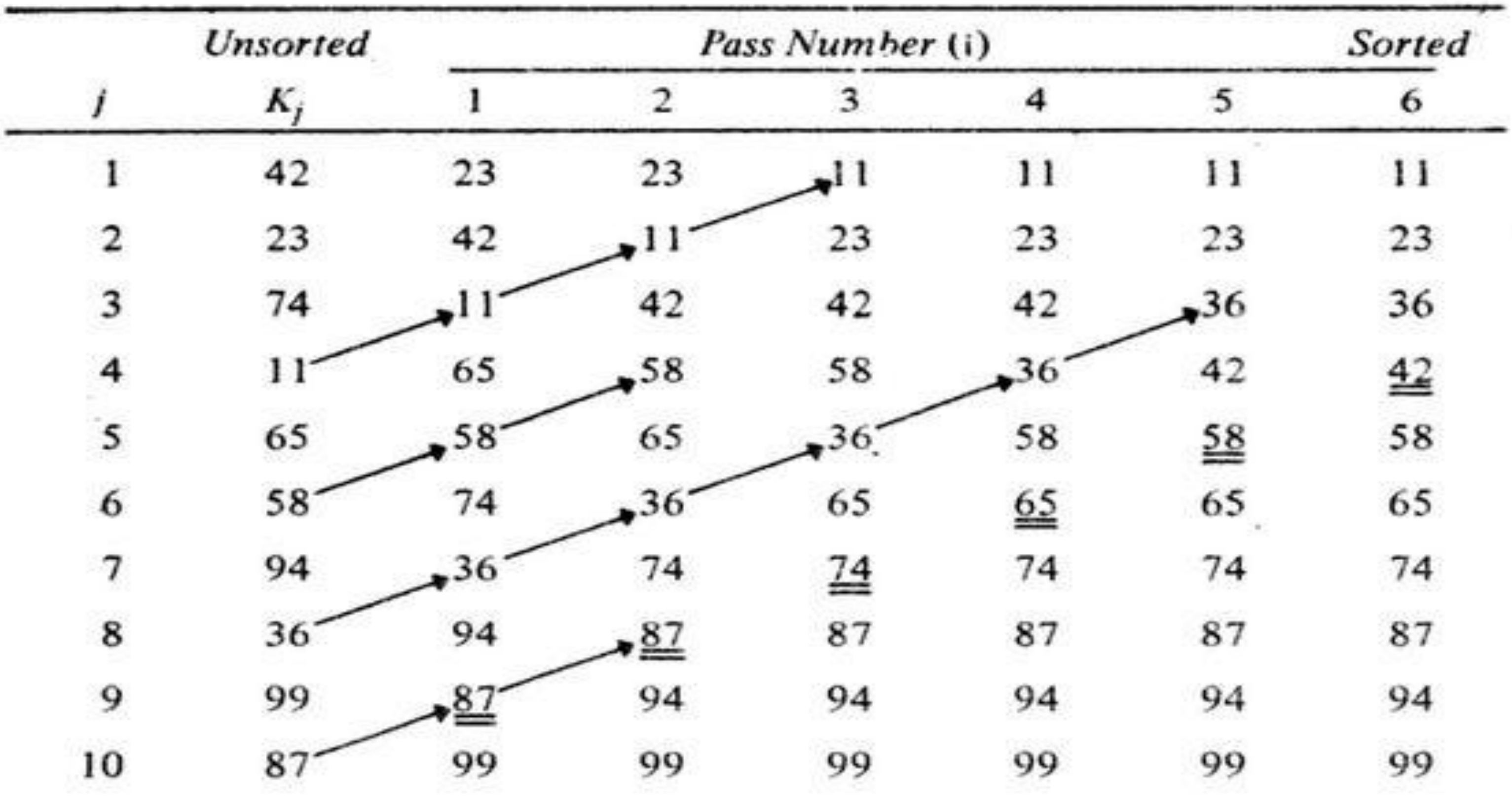
Function : BUBBLESORT (A)



```
BUBBLESORT($A$)
for $i = 1$ to $A.length - 1$
    for $j = A.length$ downto $i + 1$
        if $A[j] < A[j - 1]$
            exchange $A[j]$ with $A[j - 1]$
```

Bubble Sort

Function : BUBBLESORT (A)

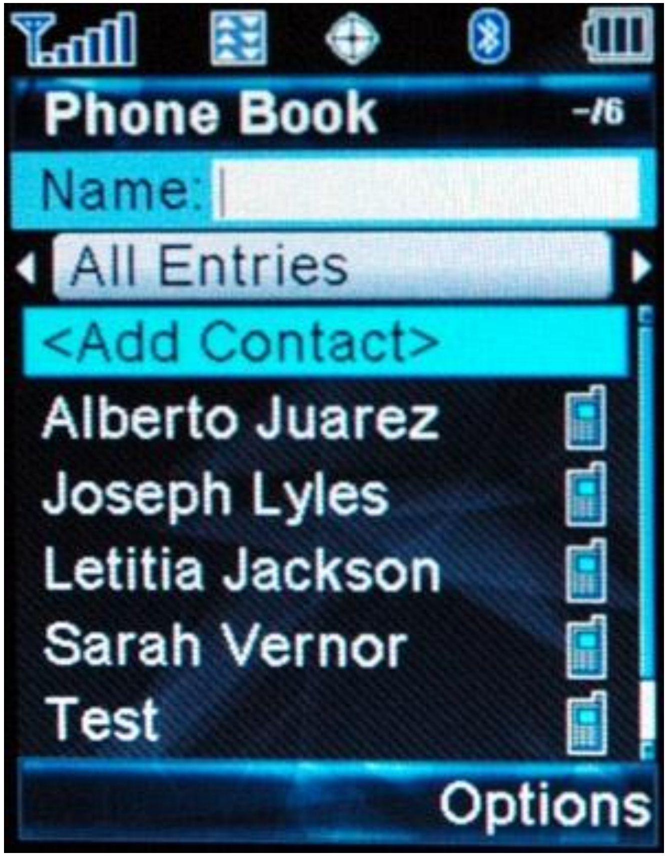


	Unsorted	*Pass Number (i)*					*Sorted*
j	K_j	1	2	3	4	5	6
1	42	23	23	11	11	11	11
2	23	42	11	23	23	23	23
3	74	11	42	42	42	36	36
4	11	65	58	58	36	42	42
5	65	58	65	36	58	58	58
6	58	74	36	65	65	65	65
7	94	36	74	74	74	74	74
8	36	94	87	87	87	87	87
9	99	87	94	94	94	94	94
10	87	99	99	99	99	99	99

FIGURE Trace of a bubble sort.

Insertion Sort

Insertion Sort



Phone Book –/6

Name: |

All Entries

<Add Contact>

Alberto Juarez

Joseph Lyles

Letitia Jackson

Sarah Vernor

Test

Options

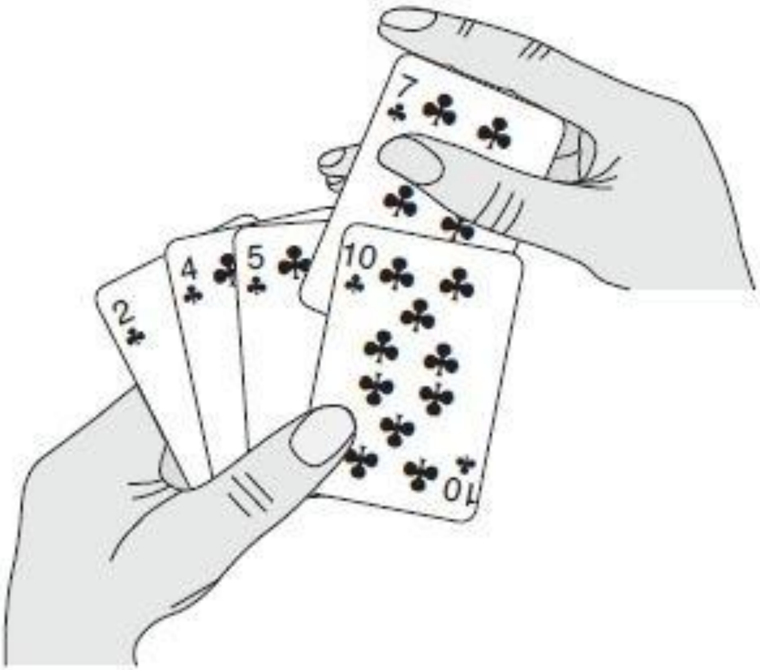
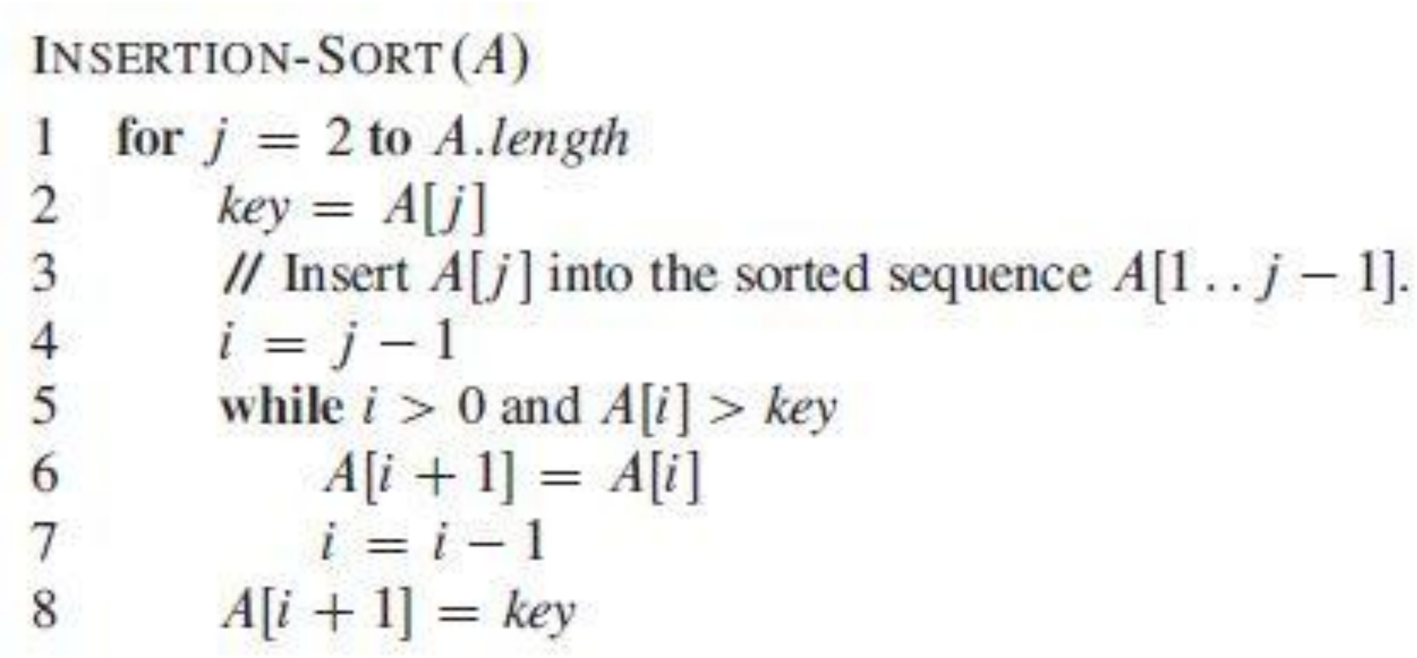


Figure Sorting a hand of cards using insertion sort.

Insertion Sort

Function : INSERTION-SORT (A)



```
INSERTION-SORT($A$)
for $j = 2$ to $A.length$
    $key = A[j]$
    // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.
    $i = j - 1$
    while $i > 0$ and $A[i] > key$
        $A[i+1] = A[i]$
        $i = i - 1$
    $A[i+1] = key$
```

Selection Sort

Selection Sort

- Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in A[1]. Then find the second smallest element of A, and exchange it with A[2]. Continue in this manner for the first n-1 elements of A. Write pseudo-code for this algorithm, which is known as selection sort.

Merge Sort

Merge Sort

```
MergeSort(low,high)
{
   if(low<high)
   {
          mid=(low+high)/2
          MergeSort(low,mid)
          MergeSort(mid+1,high)
          Merge(low,mid,high)
   }
}
```

Merge Sort

```
Merge(low,mid,high){
    h=low; i=low; j=mid+1;
    while ( (h<=mid) and (j<=high) )
    {       if(a[h]<=a[j])
            {           b[i]=a[h]
                        h=h+1
            }
            else
            {           b[i]=a[j]
                        j=j+1
            }
            i=i+1
    }//End of while
    If(h>mid)
    {       for k=j to high
            {           b[i]=a[k]
                        i=i+1
            }
    }
    else
    {       for k=h to mid
            {           b[i]=a[k]
                        i=i+1
            }
    }
    for k=low to high
            a[k]=b[k]
}//End of Merge
```

Merge Sort (Example)

98	23	45	14	6	67	33	42

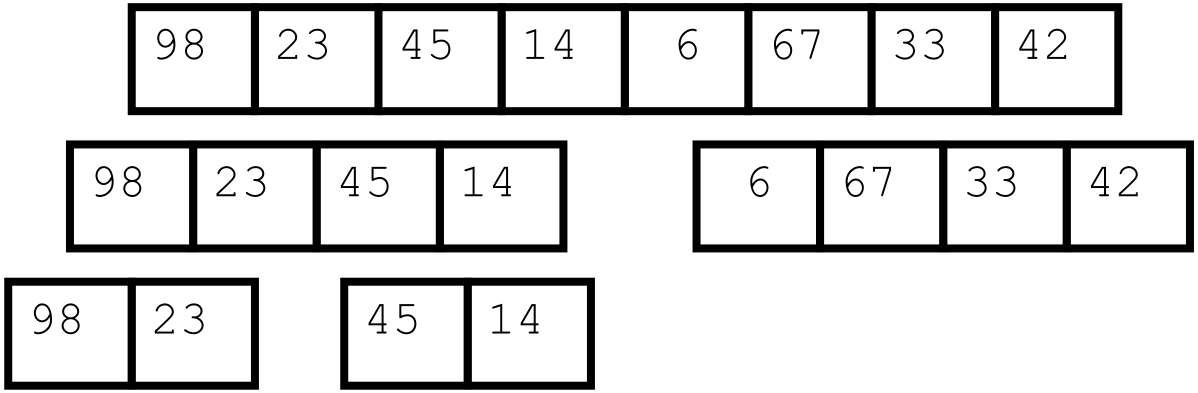
Merge Sort (Example)

98	23	45	14	6	67	33	42

98	23	45	14

6	67	33	42

Merge Sort (Example)

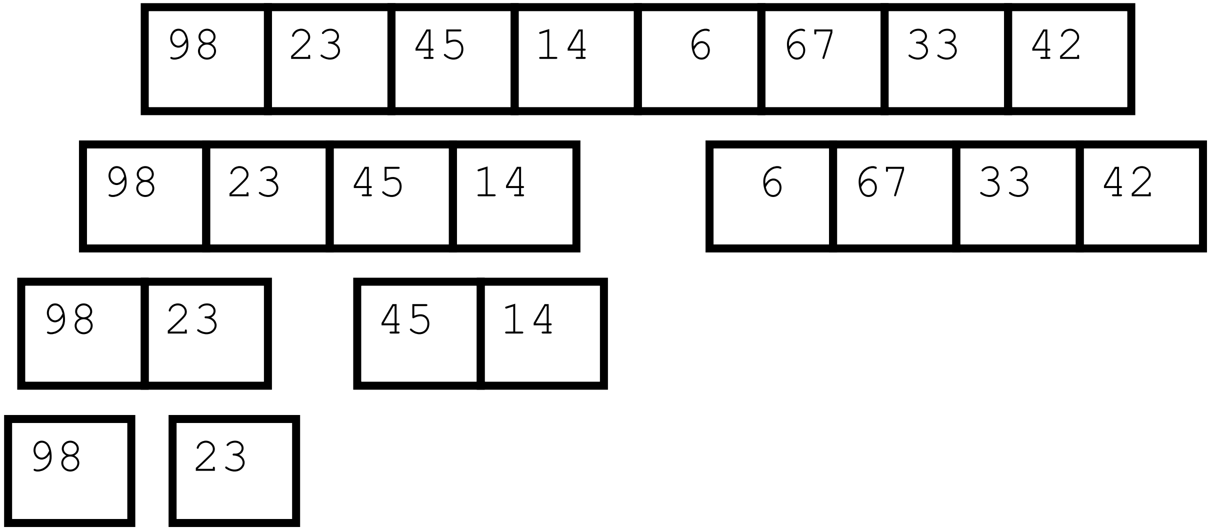


98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

98 23   45 14

Merge Sort (Example)



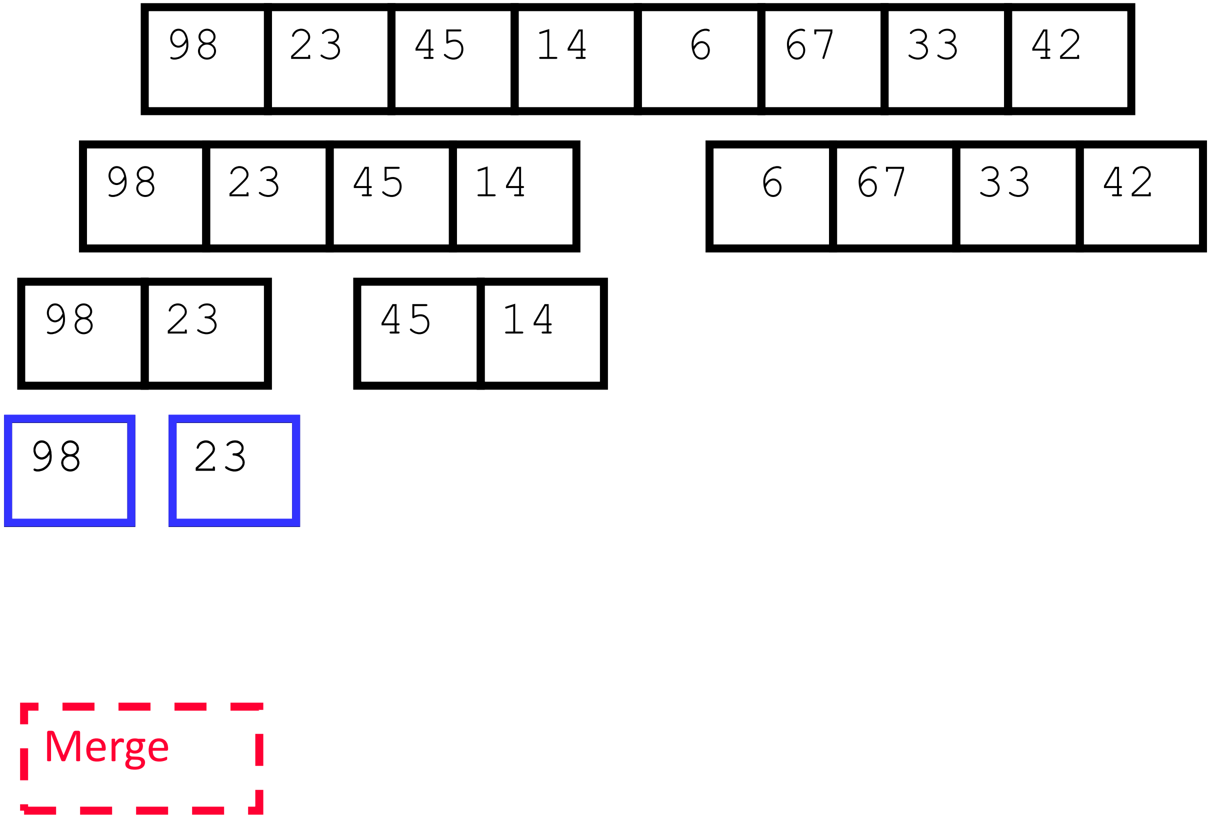
98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

98 23   45 14

98   23

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14

6 67 33 42

98 23

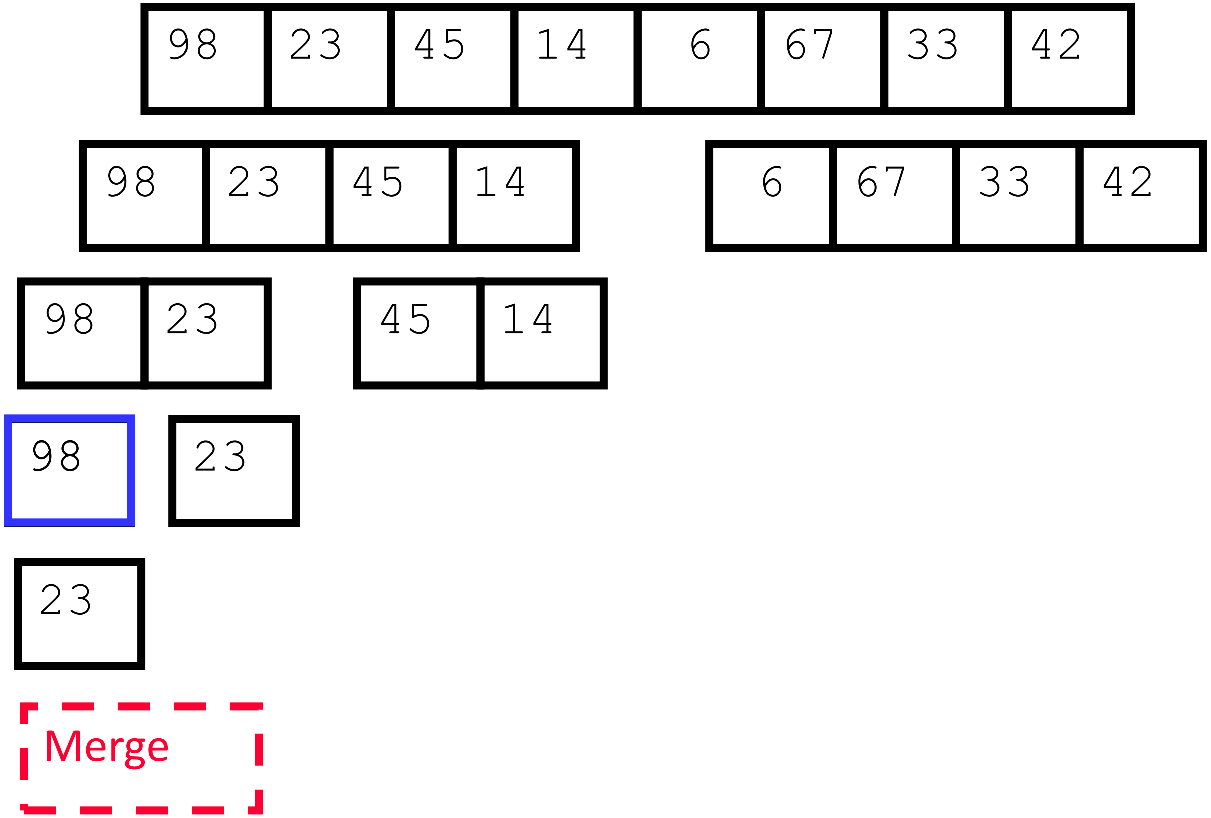
45 14

98

23

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14

6 67 33 42

98 23

45 14

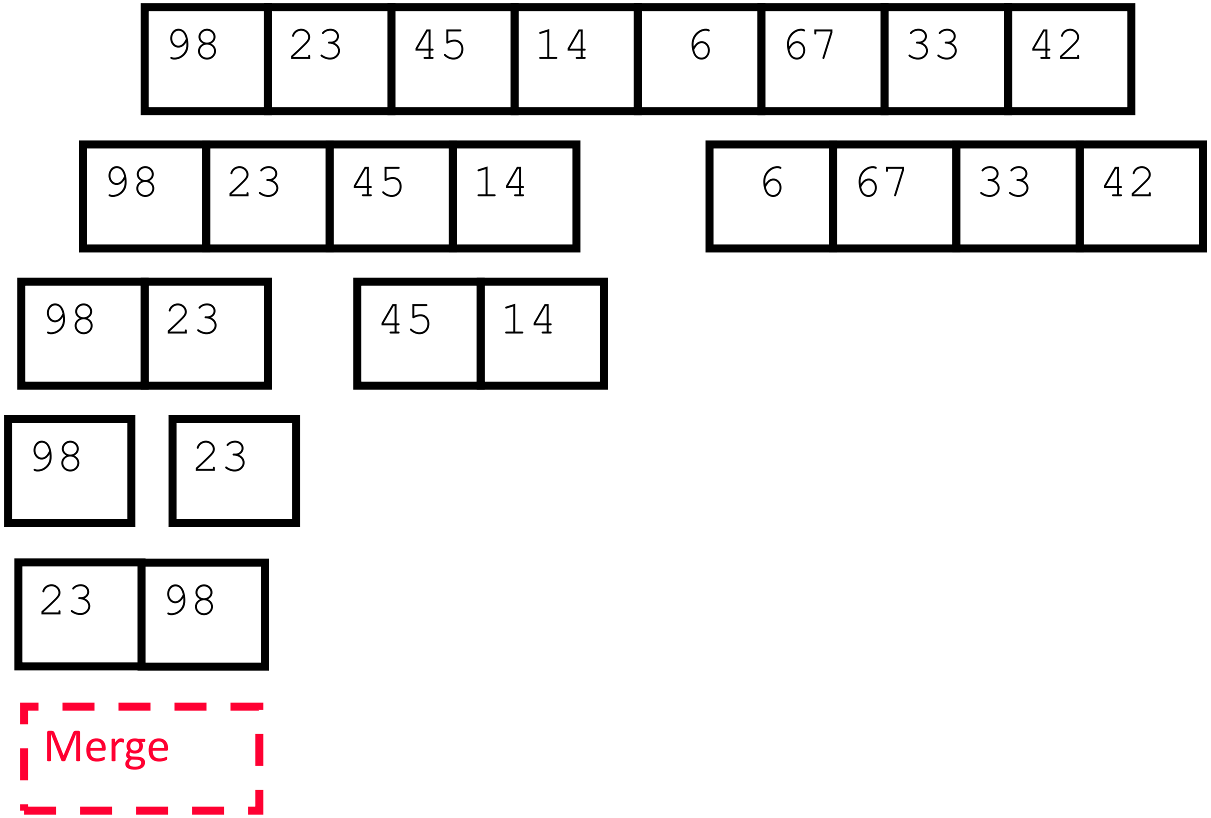
98

23

23

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

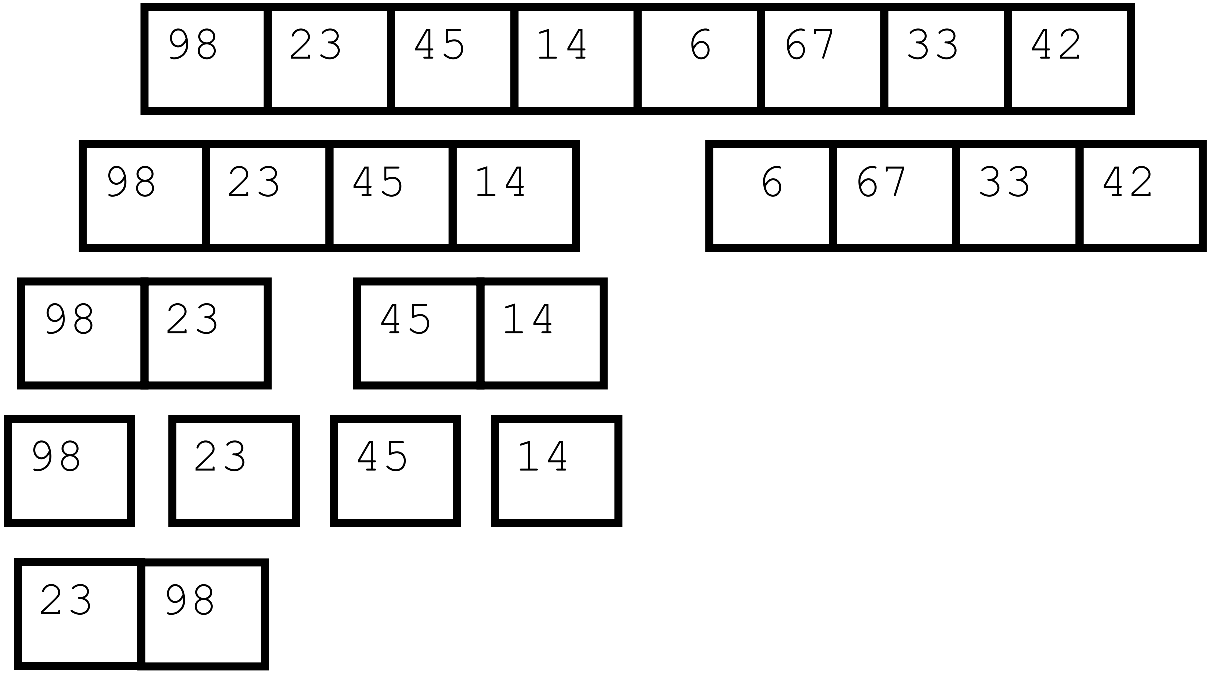
98 23   45 14

98   23

23 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14

6 67 33 42

98 23

45 14

98

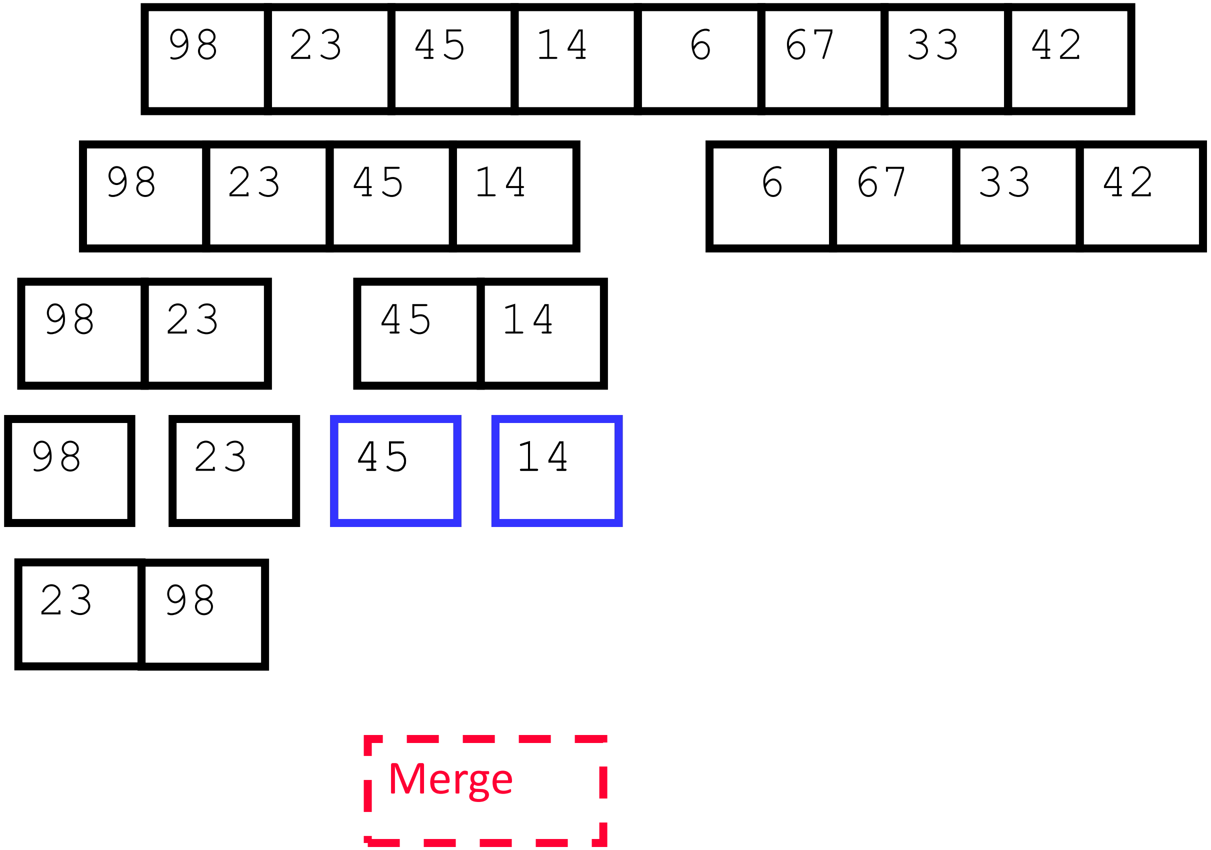
23

45

14

23 98

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

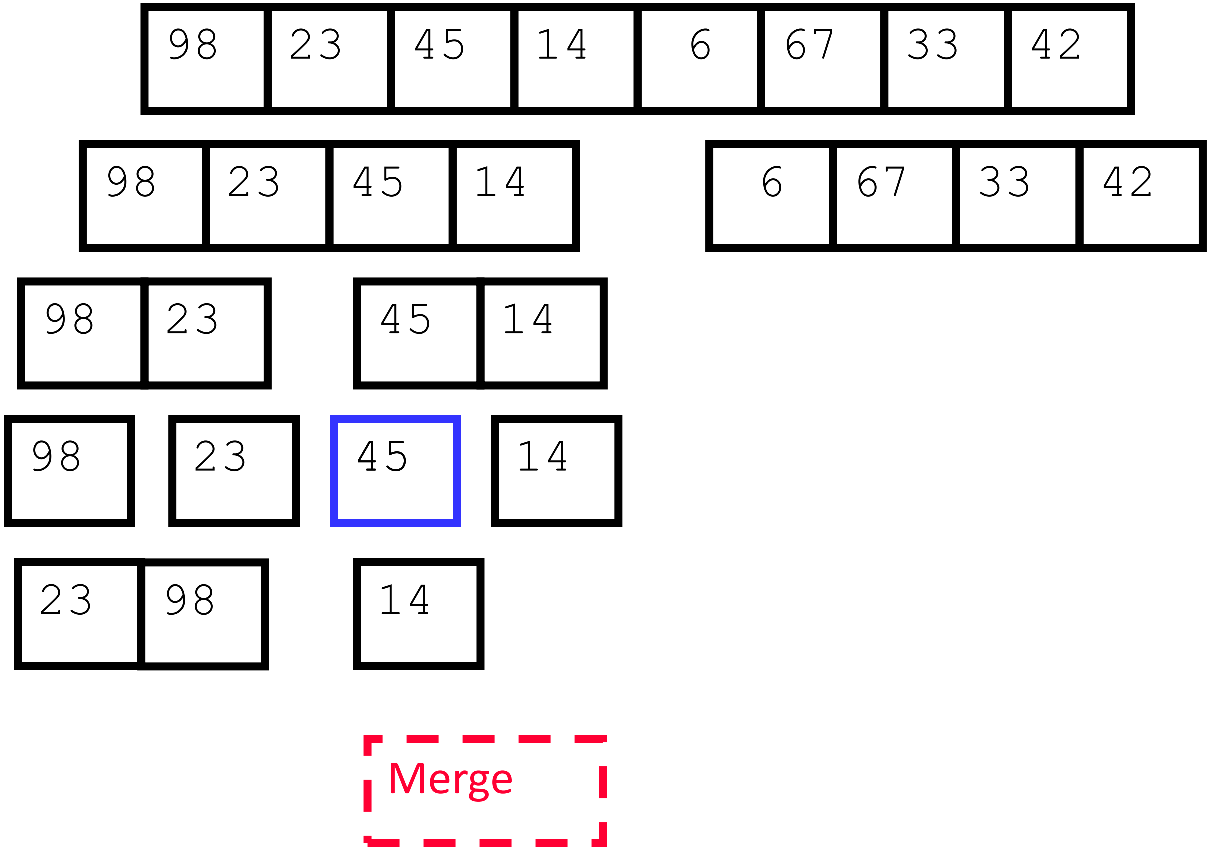
98 23   45 14

98   23   45   14

23 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

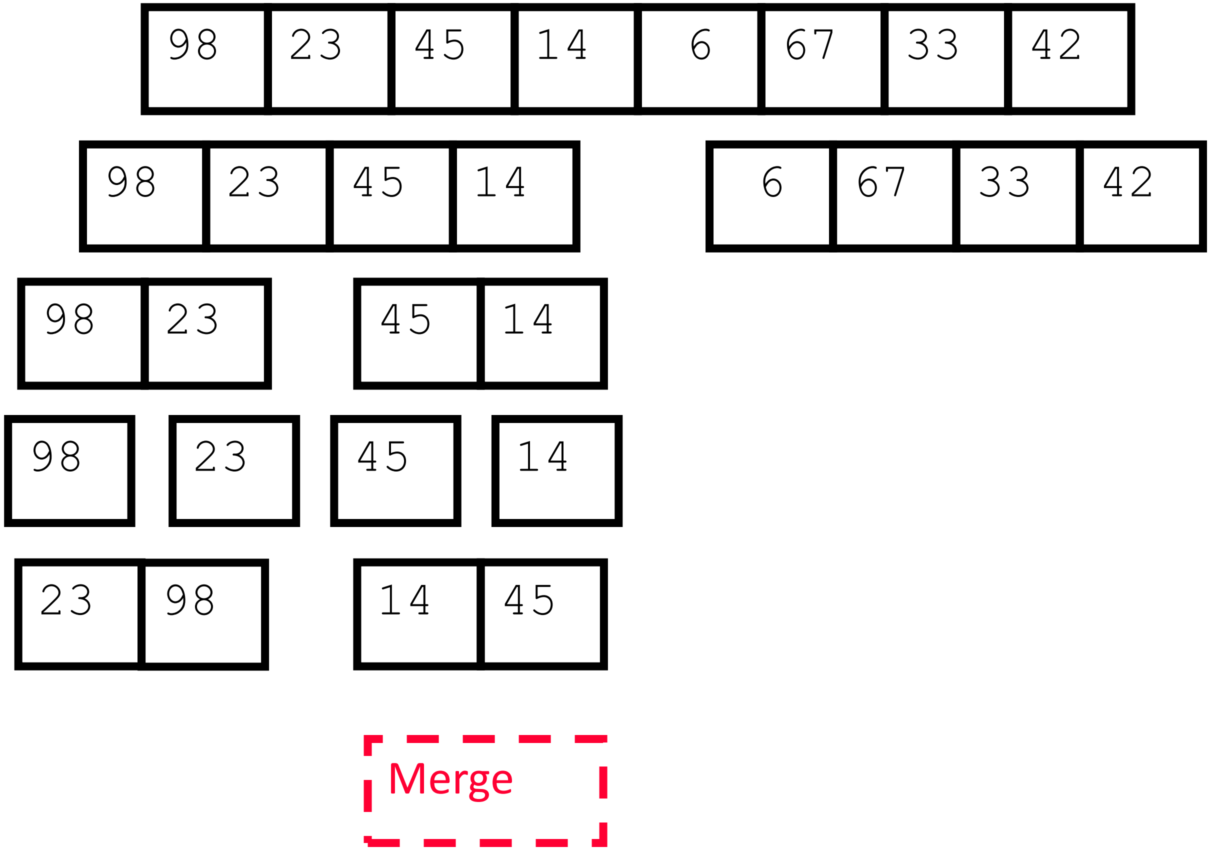
98 23   45 14

98   23   45   14

23 98   14

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

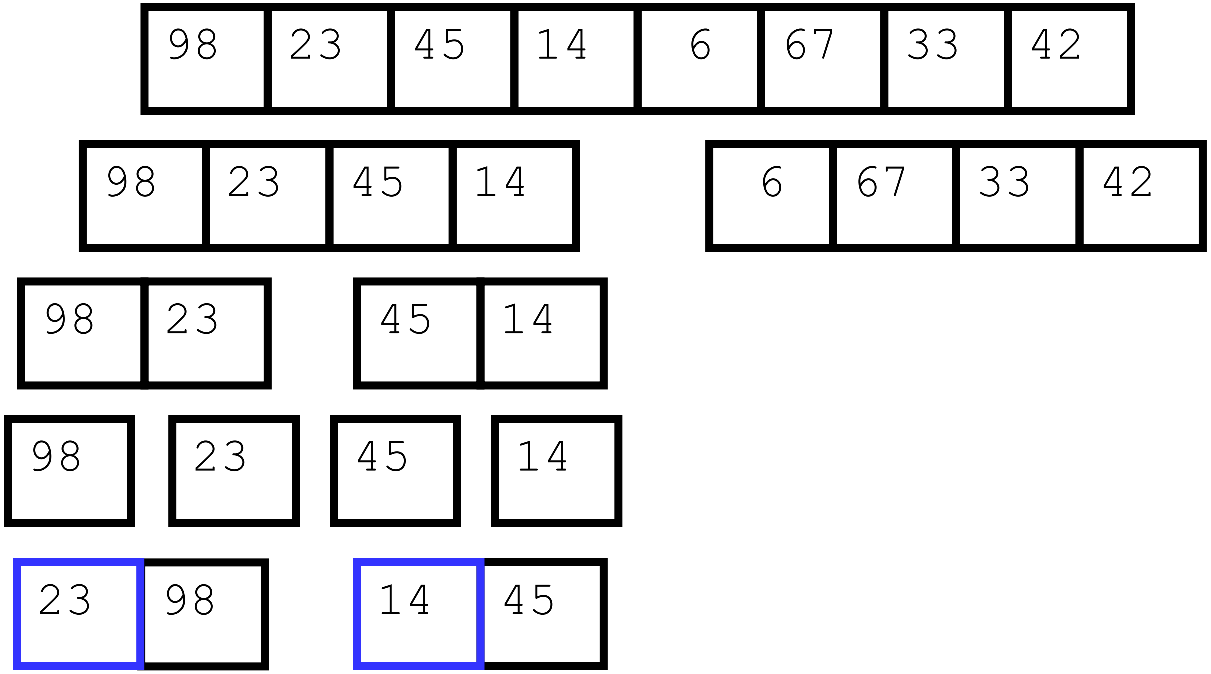
98 23   45 14

98   23   45   14

23 98   14 45

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

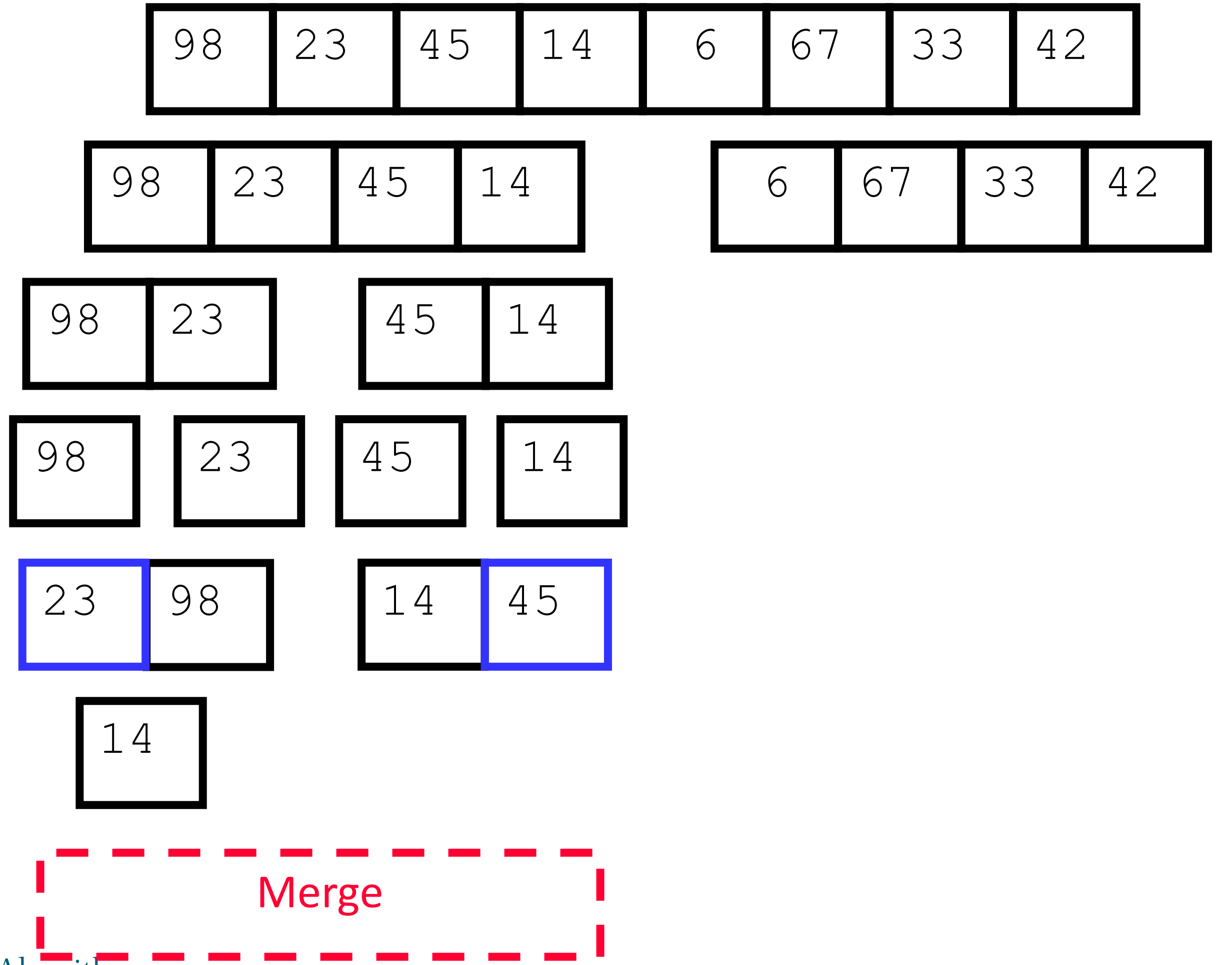
98 23 | 45 14

98 | 23 | 45 | 14

23 98 | 14 45

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 6 67 33 42

98 23 45 14

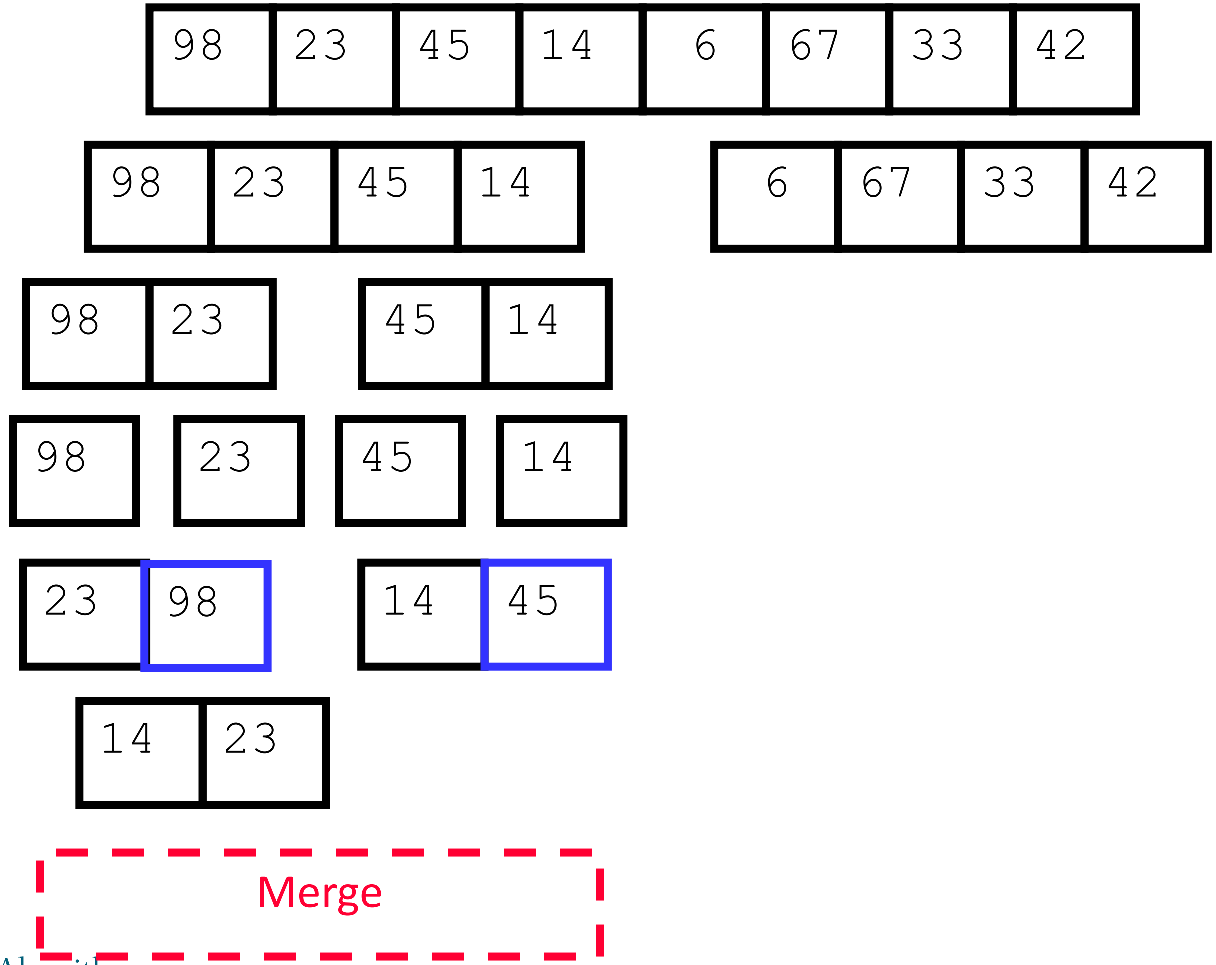
98 23 45 14

23 98 14 45

14

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14

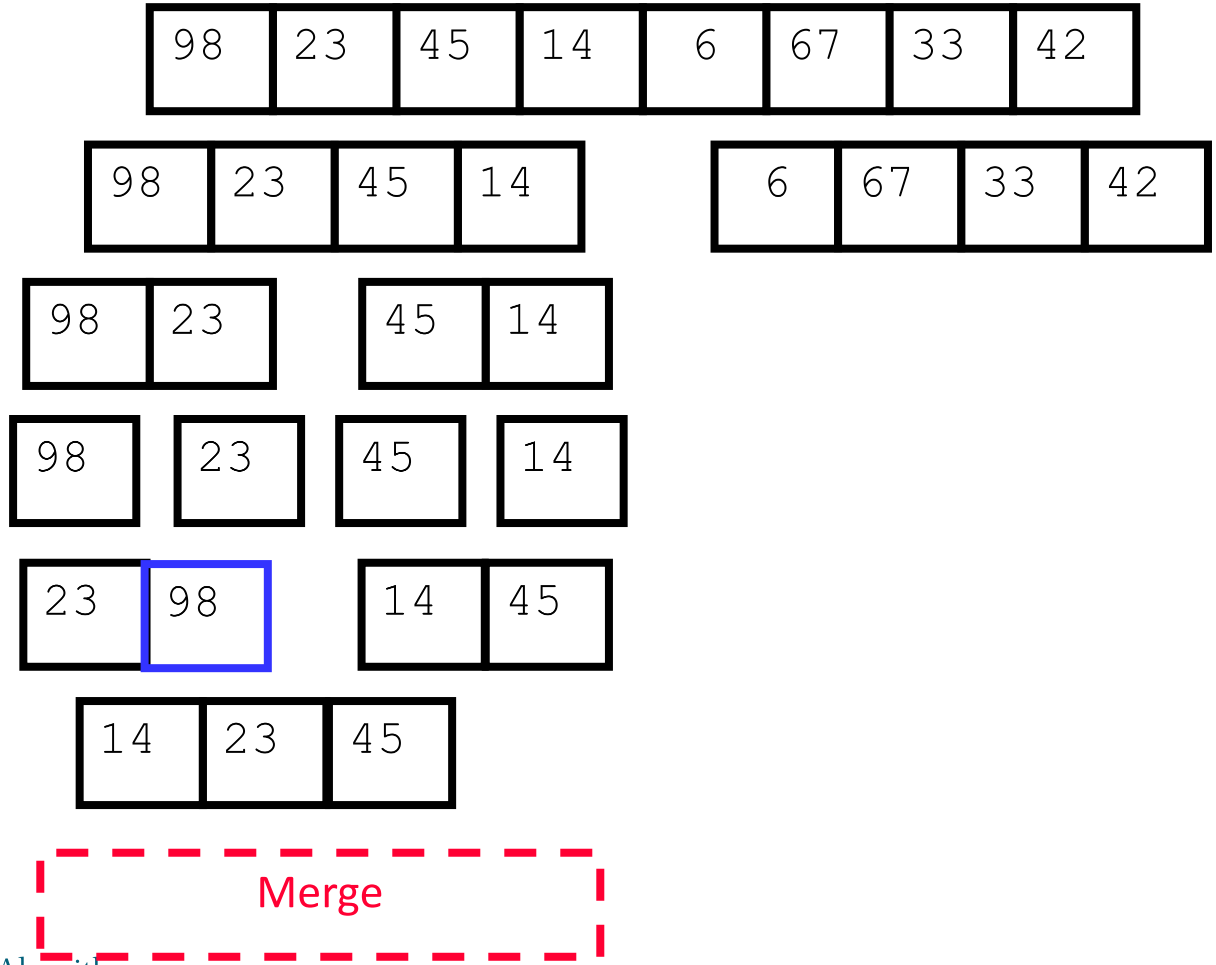
98 | 23 | 45 | 14

23 98 | 14 45

14 23

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

98 23   45 14

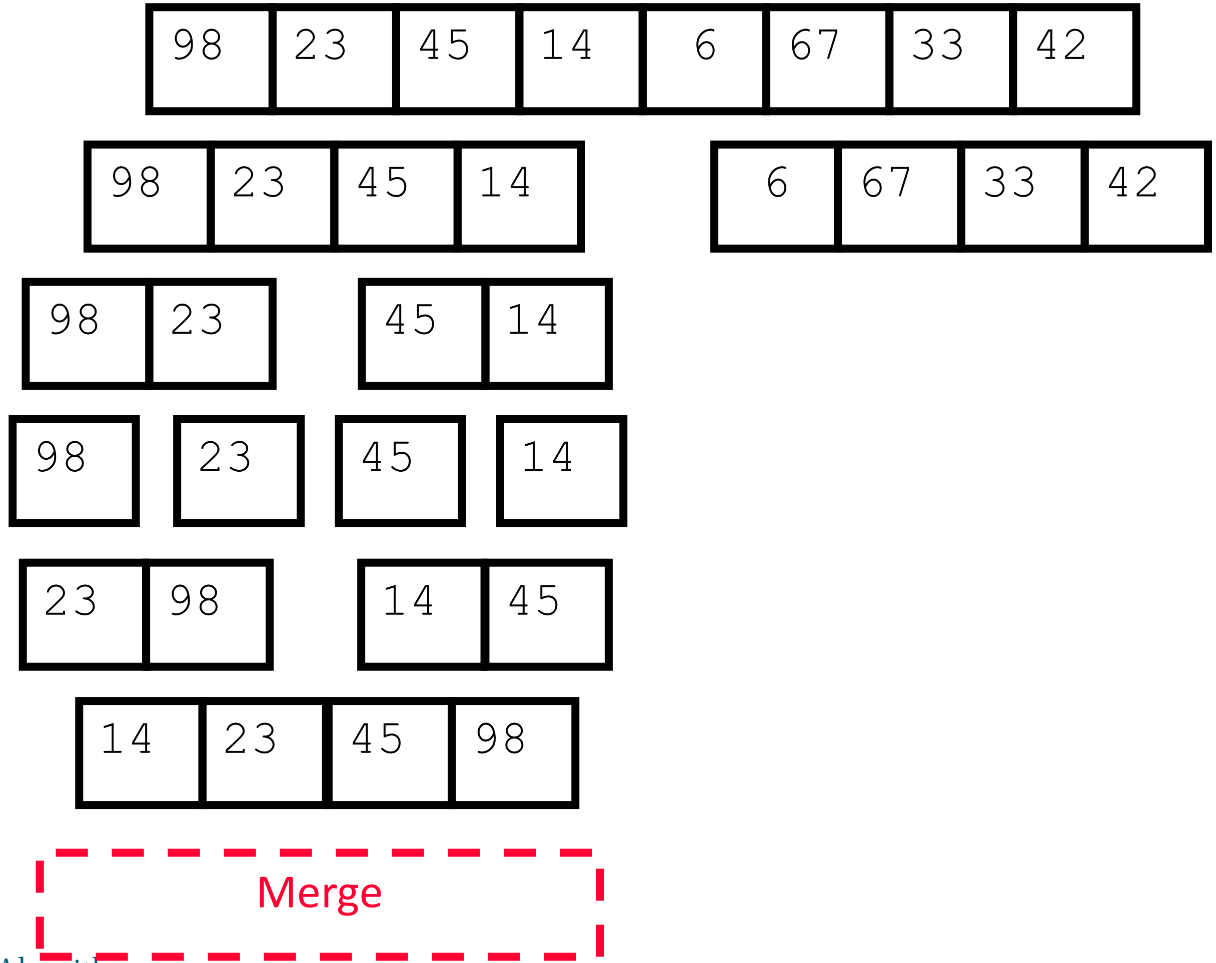
98   23   45   14

23 98   14 45

14 23 45

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

98 23   45 14

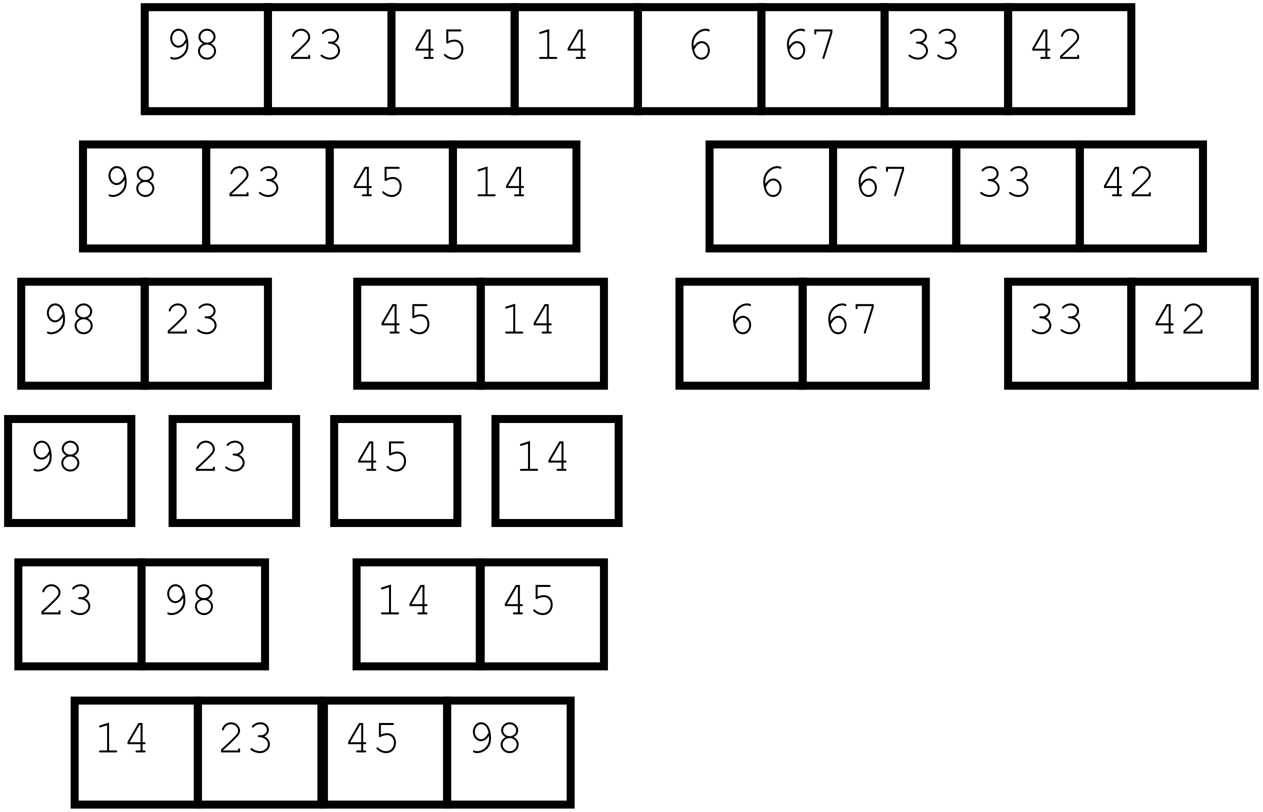
98   23   45   14

23 98   14 45

14 23 45 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

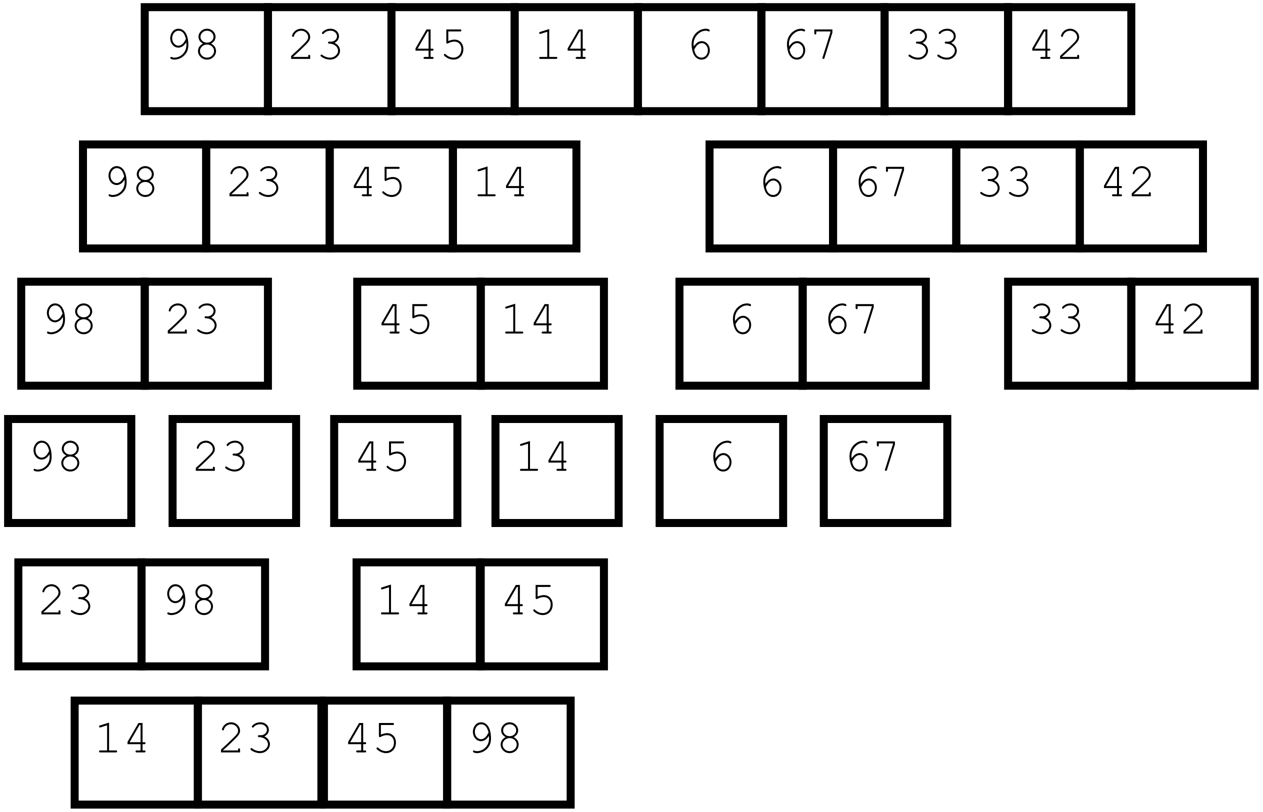
98 23   45 14   6 67   33 42

98   23   45   14

23 98   14 45

14 23 45 98

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

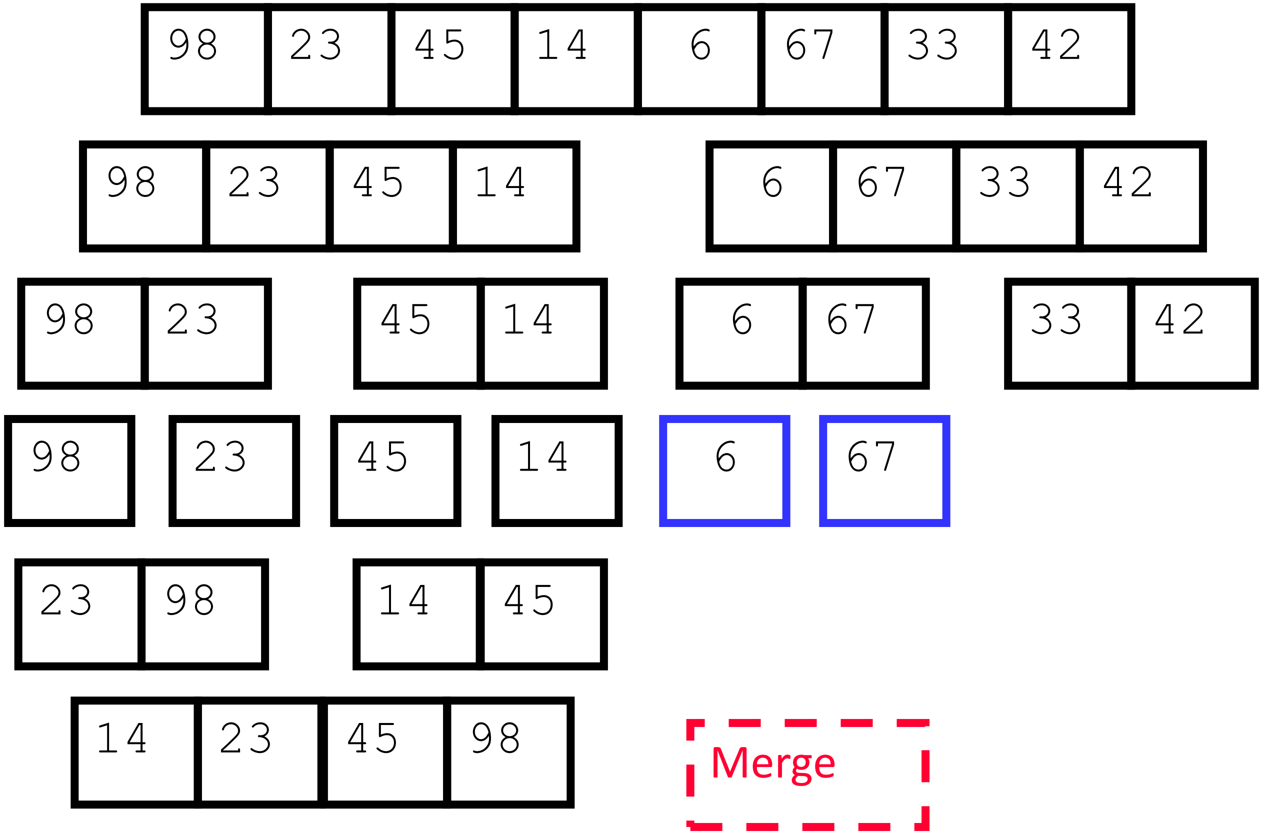
98 23 | 45 14 | 6 67 | 33 42

98 | 23 | 45 | 14 | 6 | 67

23 98 | 14 45

14 23 45 98

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

98 23   45 14   6 67   33 42

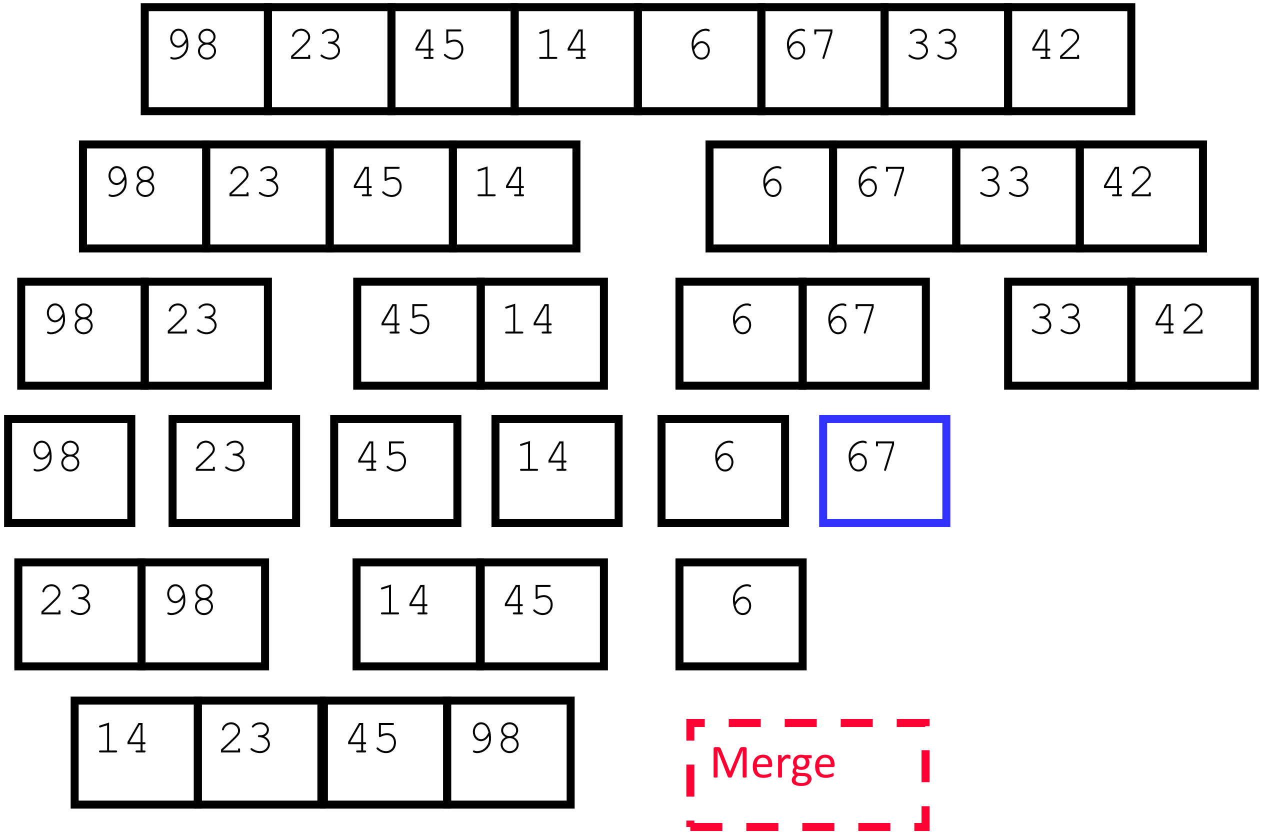
98   23   45   14   6   67

23 98   14 45

14 23 45 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

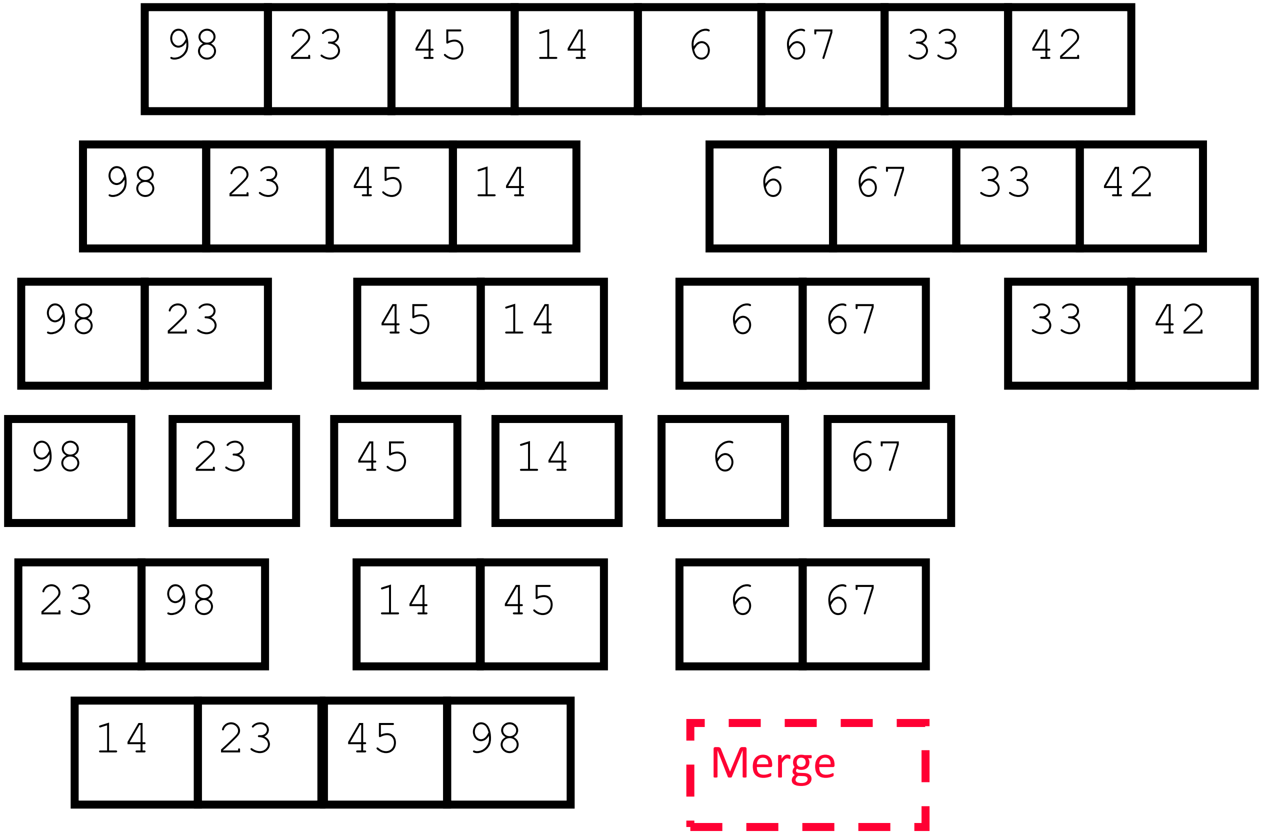
98 | 23 | 45 | 14 | 6 | 67

23 98 | 14 45 | 6

14 23 45 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 6 67 33 42

98 23 45 14 6 67 33 42

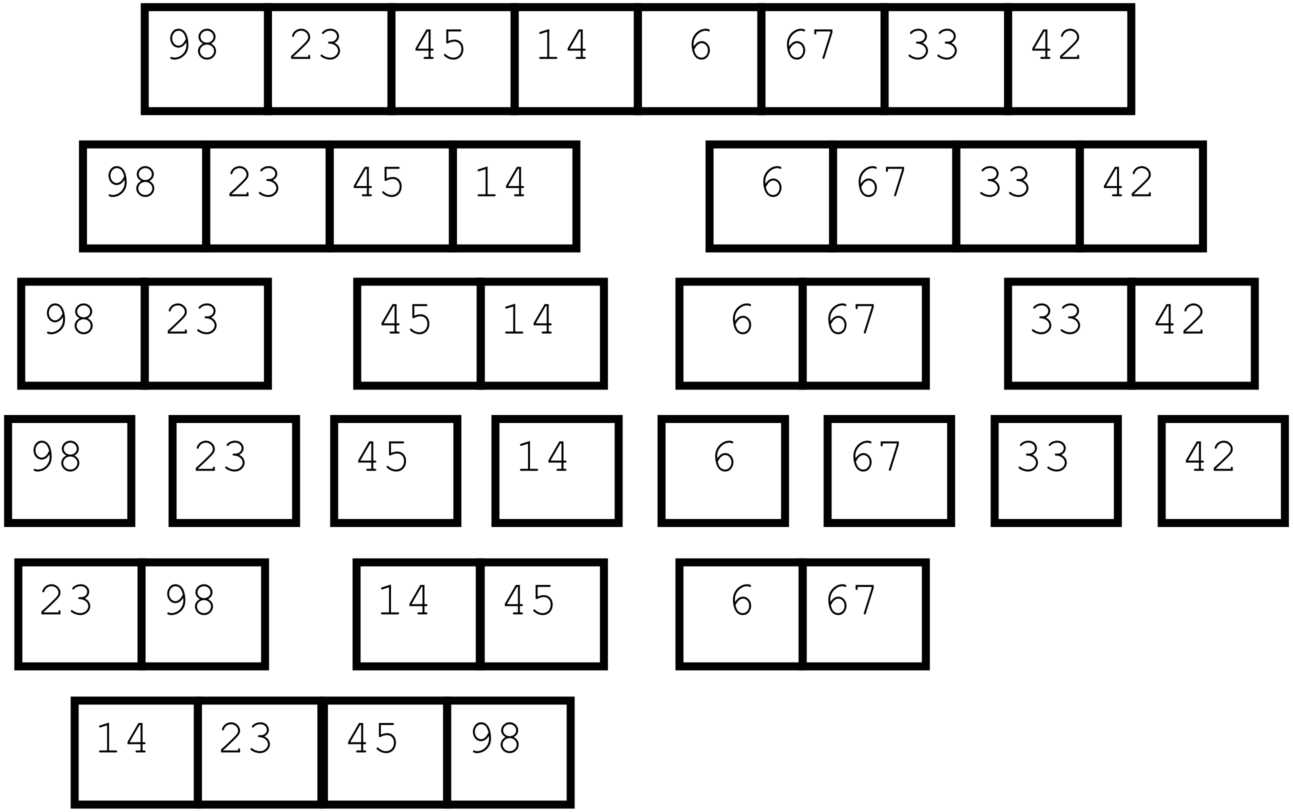
98 23 45 14 6 67

23 98 14 45 6 67

14 23 45 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

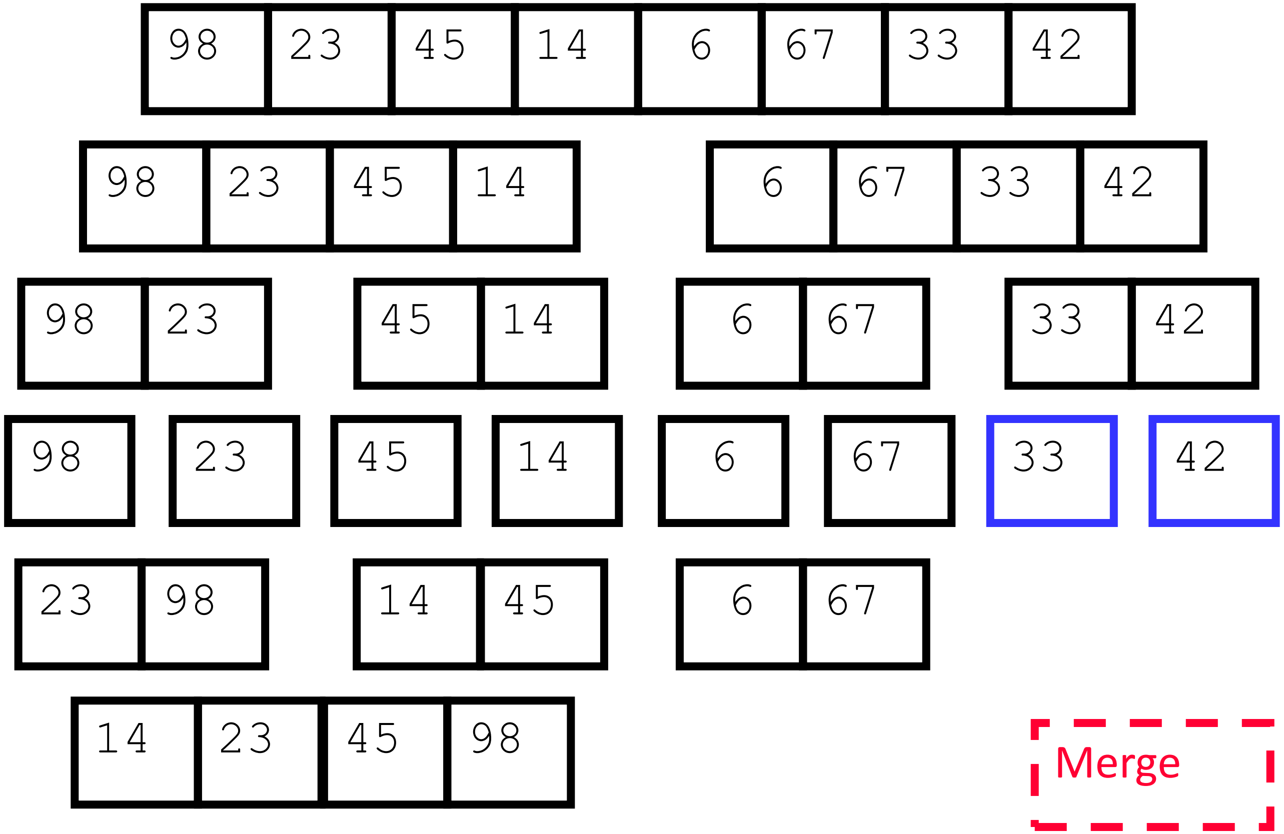
98 23 | 45 14 | 6 67 | 33 42

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

23 98 | 14 45 | 6 67

14 23 45 98

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

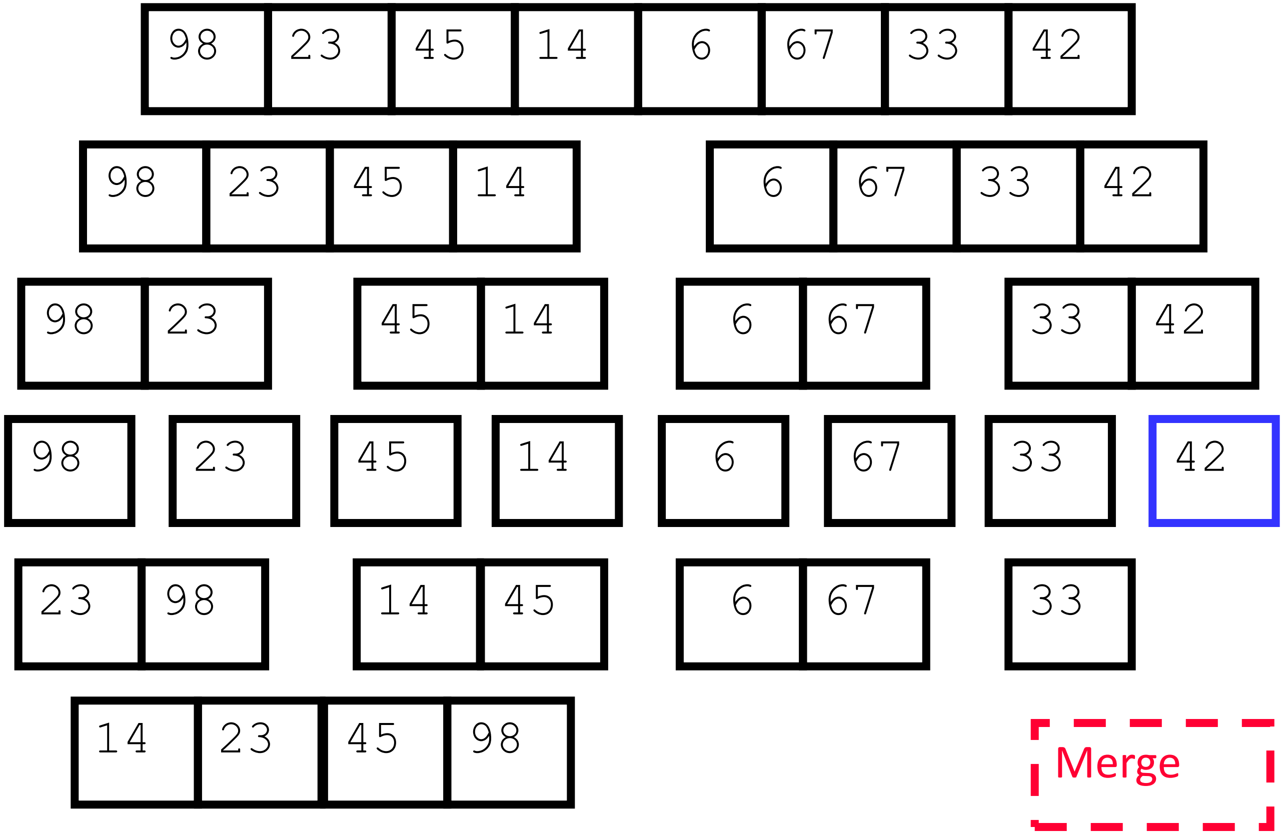
98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67

14 23 45 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

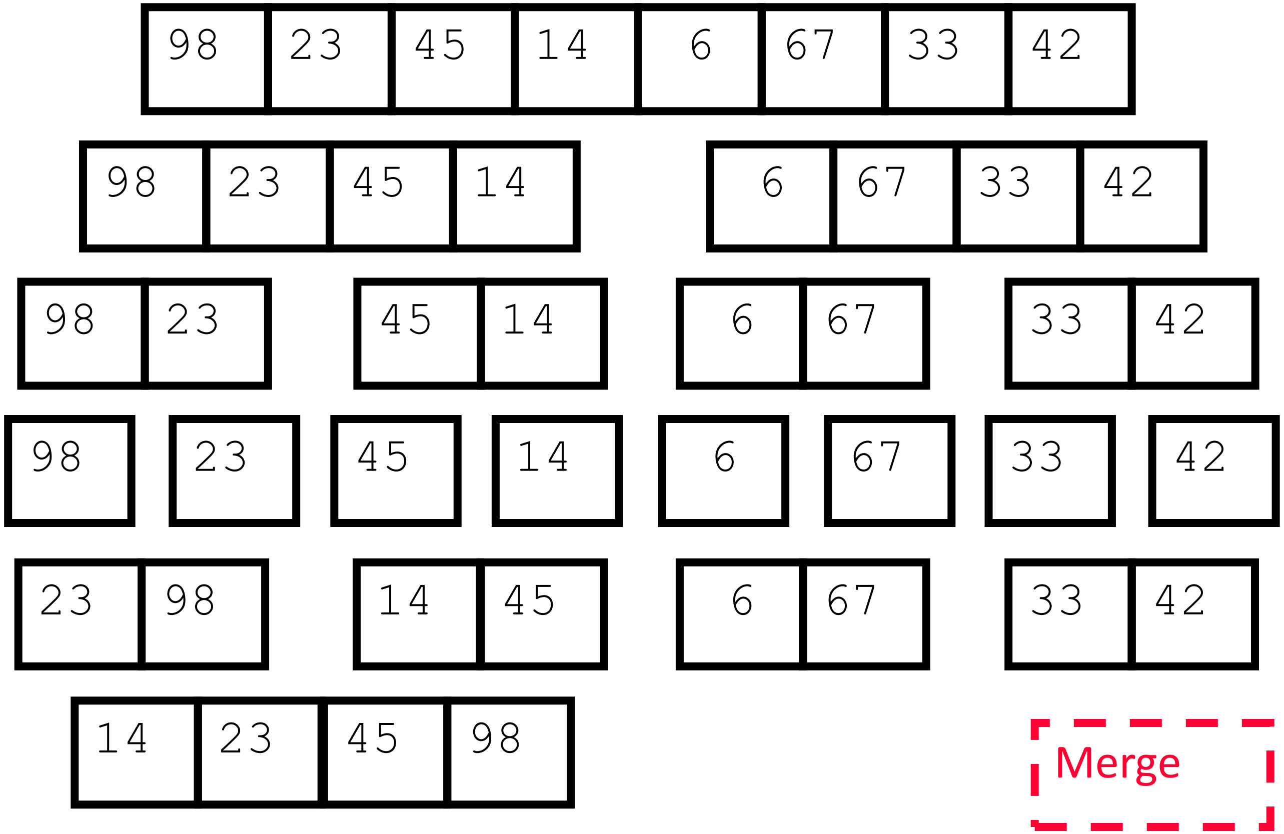
98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67 | 33

14 23 45 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

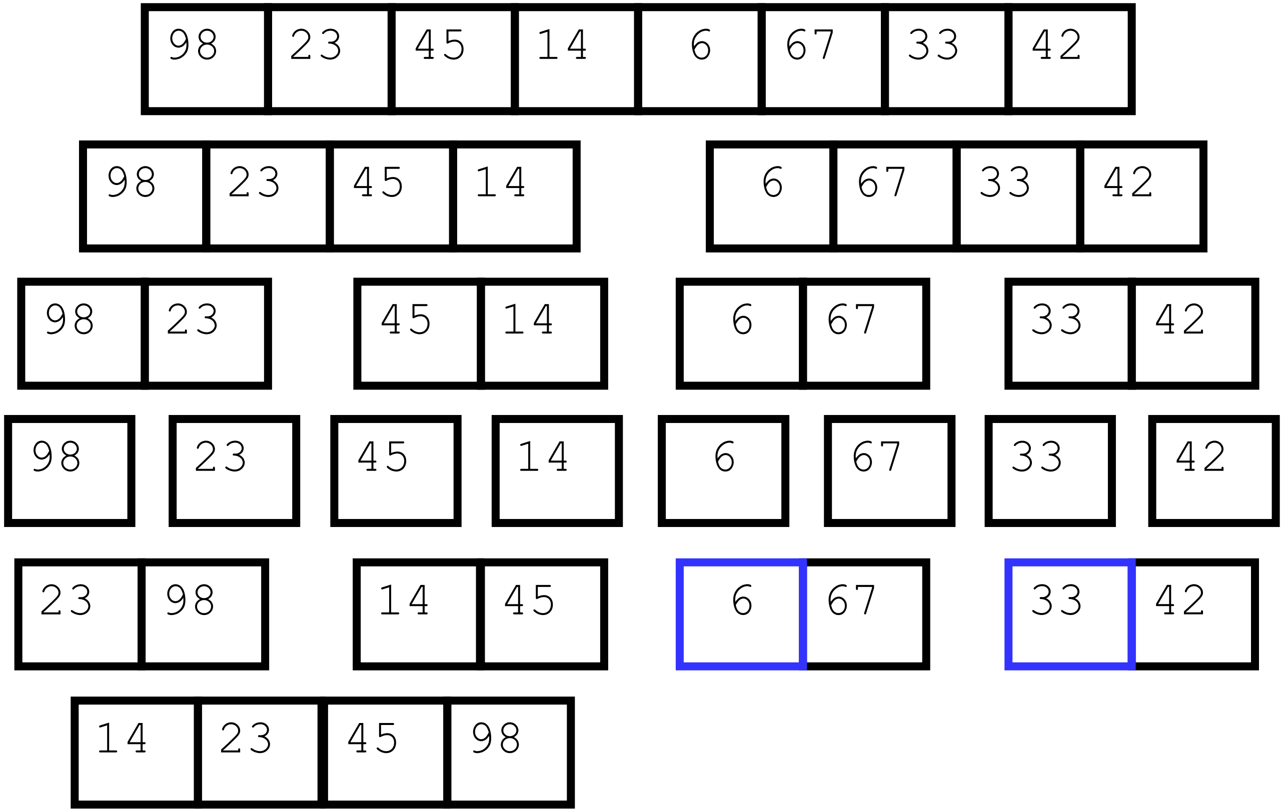
98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67 | 33 42

14 23 45 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

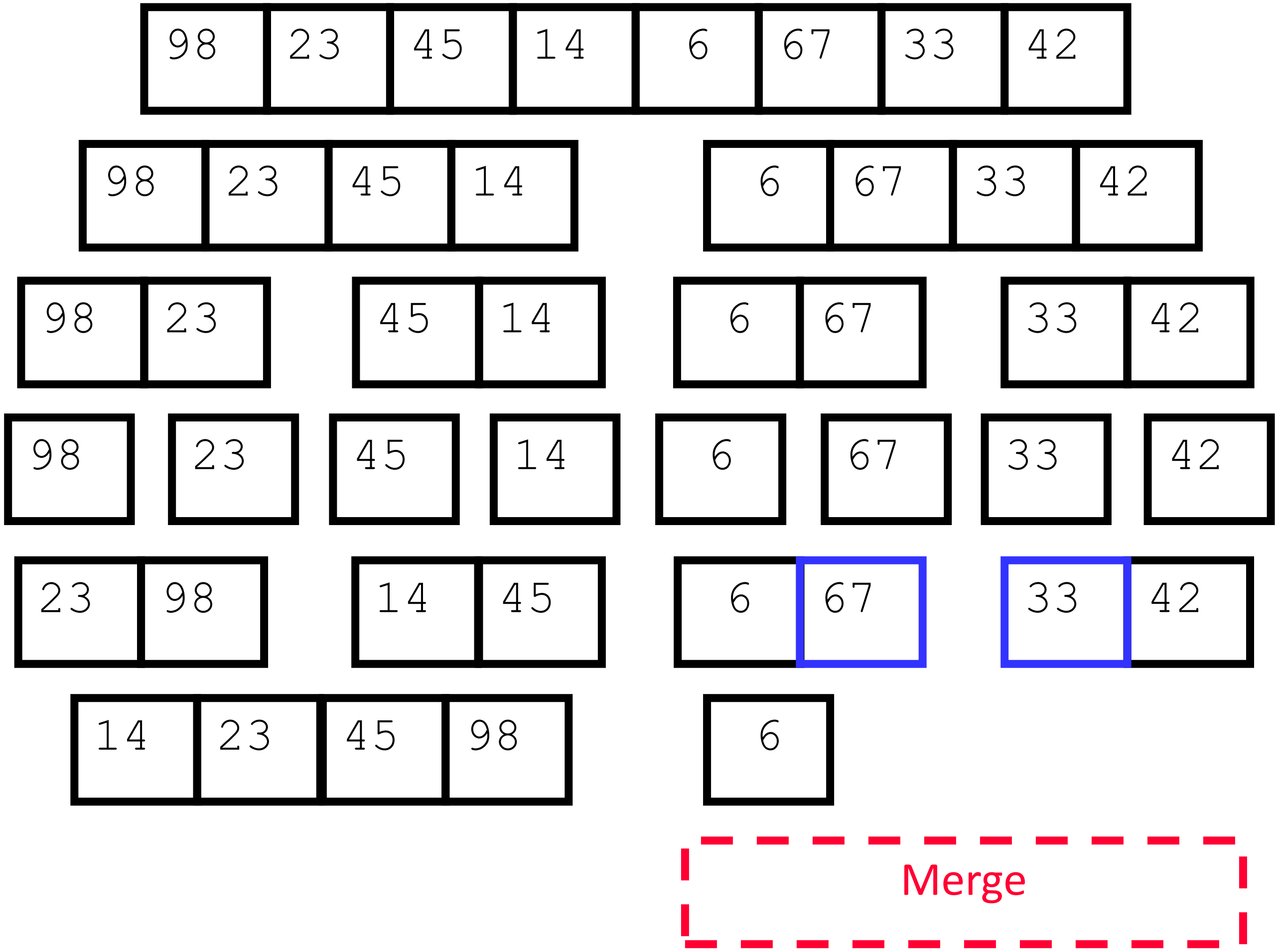
98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67 | 33 42

14 23 45 98

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

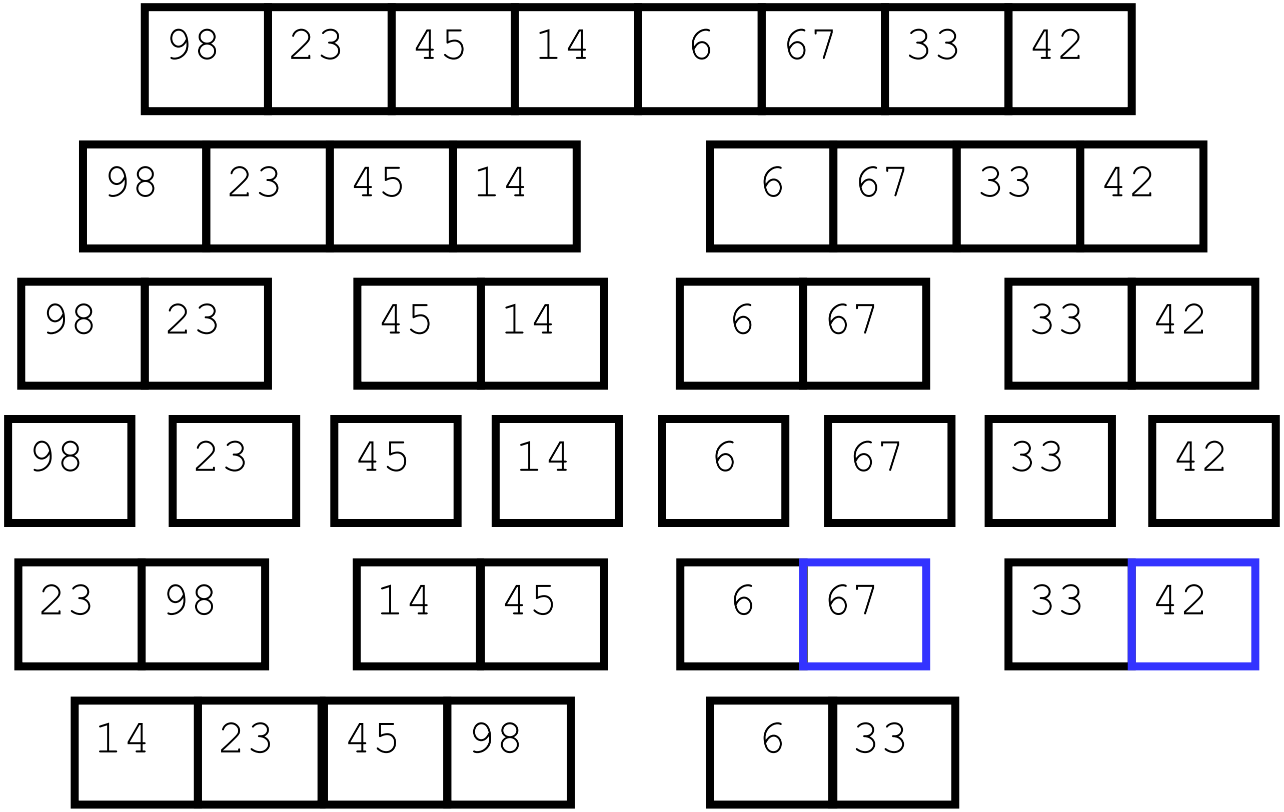
98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67 | 33 42

14 23 45 98 | 6

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

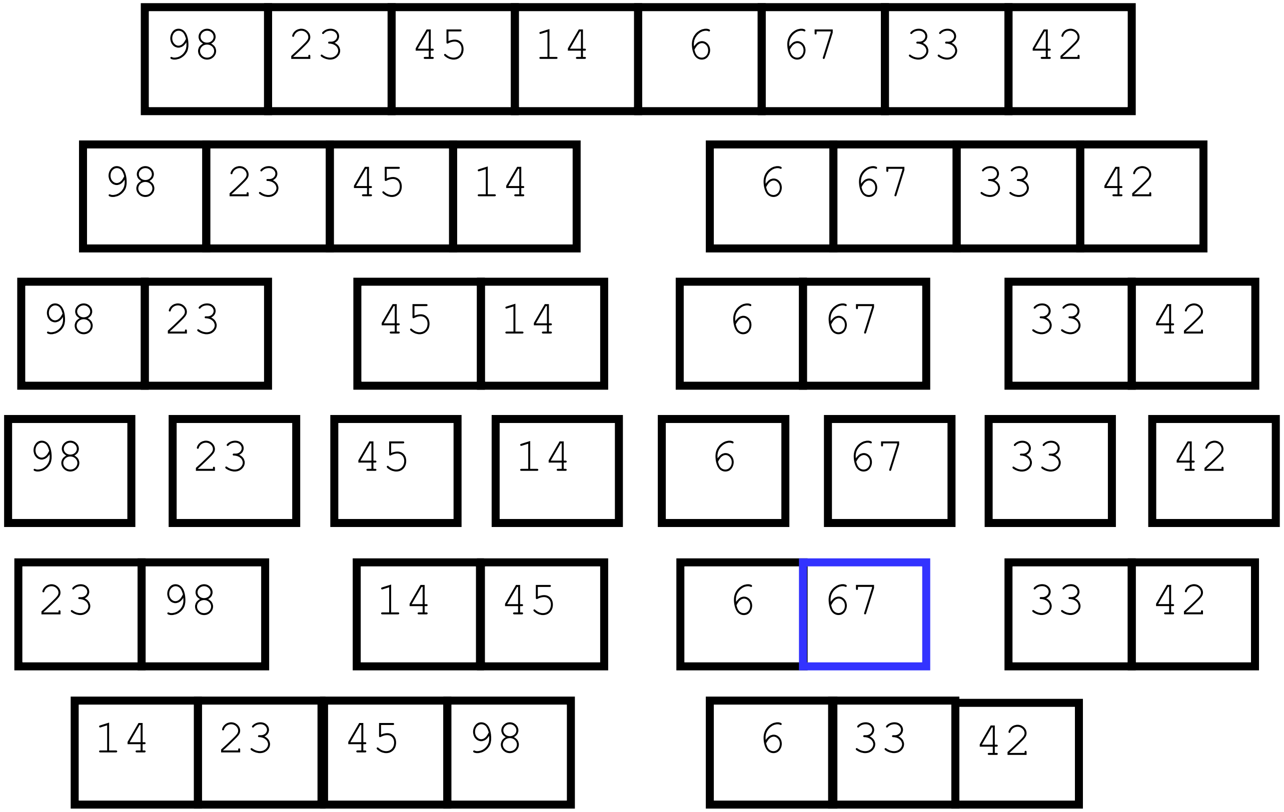
98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67 | 33 42

14 23 45 98 | 6 33

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

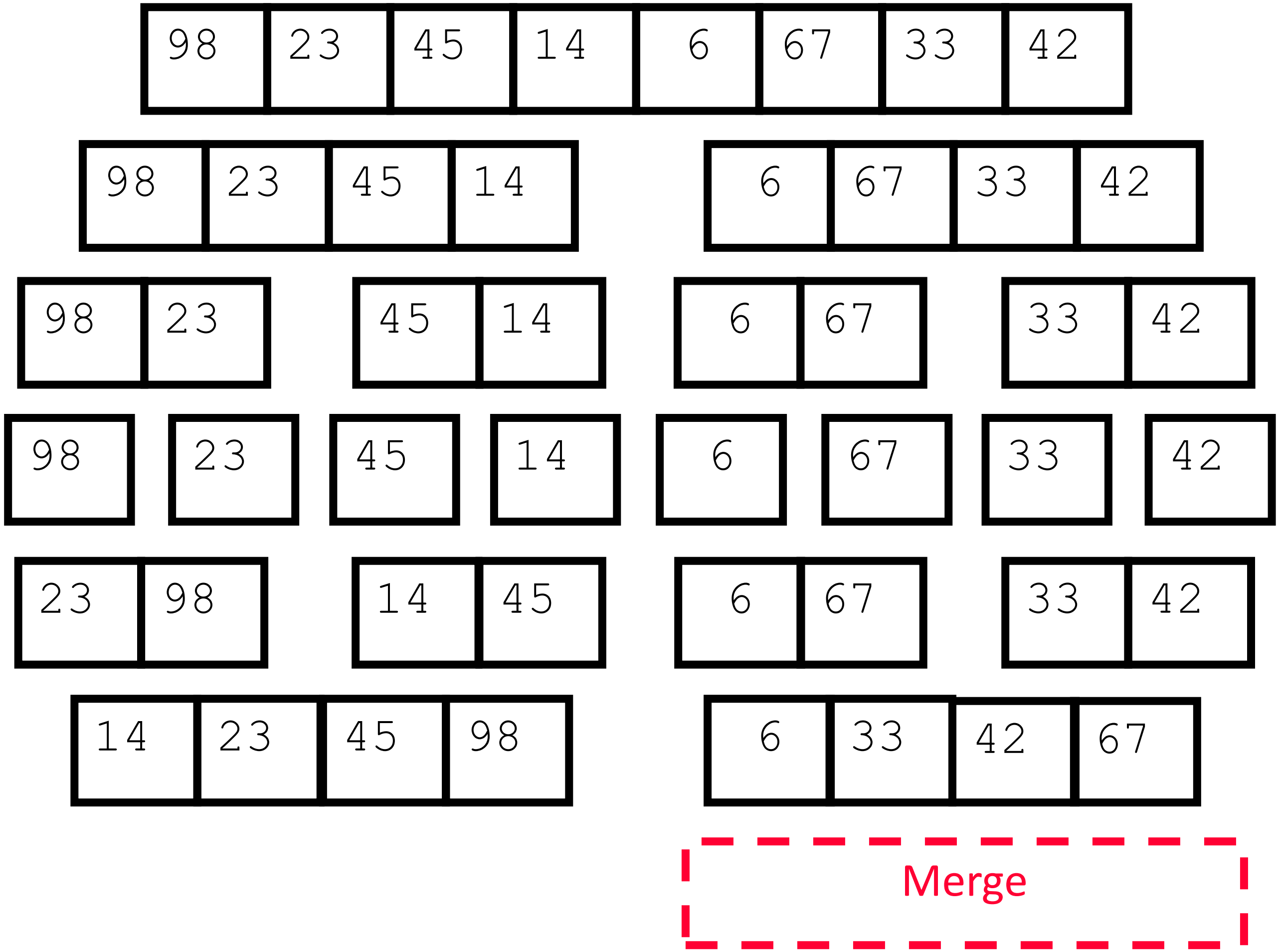
98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67 | 33 42

14 23 45 98 | 6 33 42

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14   6 67 33 42

98 23   45 14   6 67   33 42

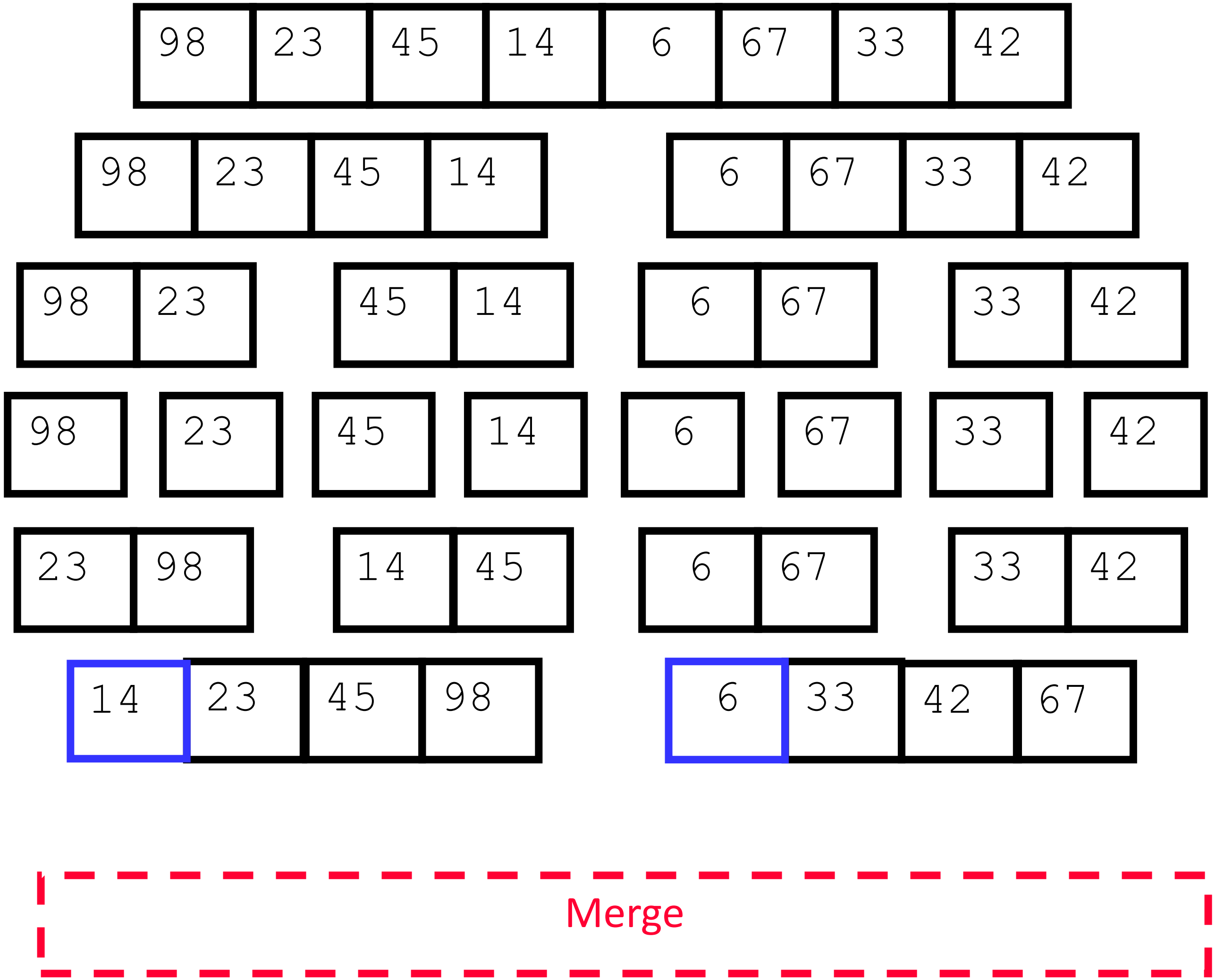
98   23   45   14   6   67   33   42

23 98   14 45   6 67   33 42

14 23 45 98   6 33 42 67

Merge

Merge Sort (Example)



98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

98 23 | 45 14 | 6 67 | 33 42

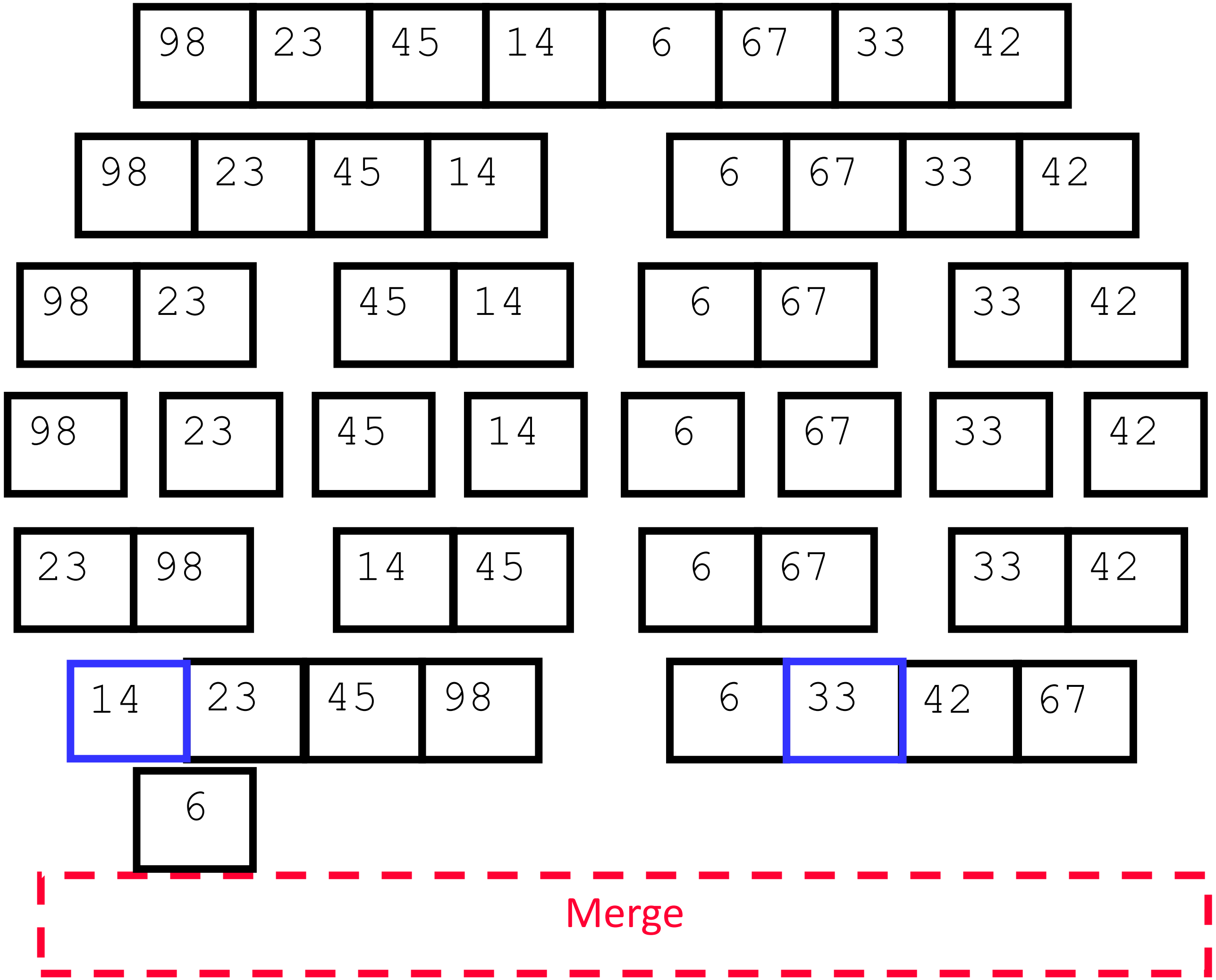
98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67 | 33 42

14 23 45 98 | 6 33 42 67

Merge

Merge Sort (Example)



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

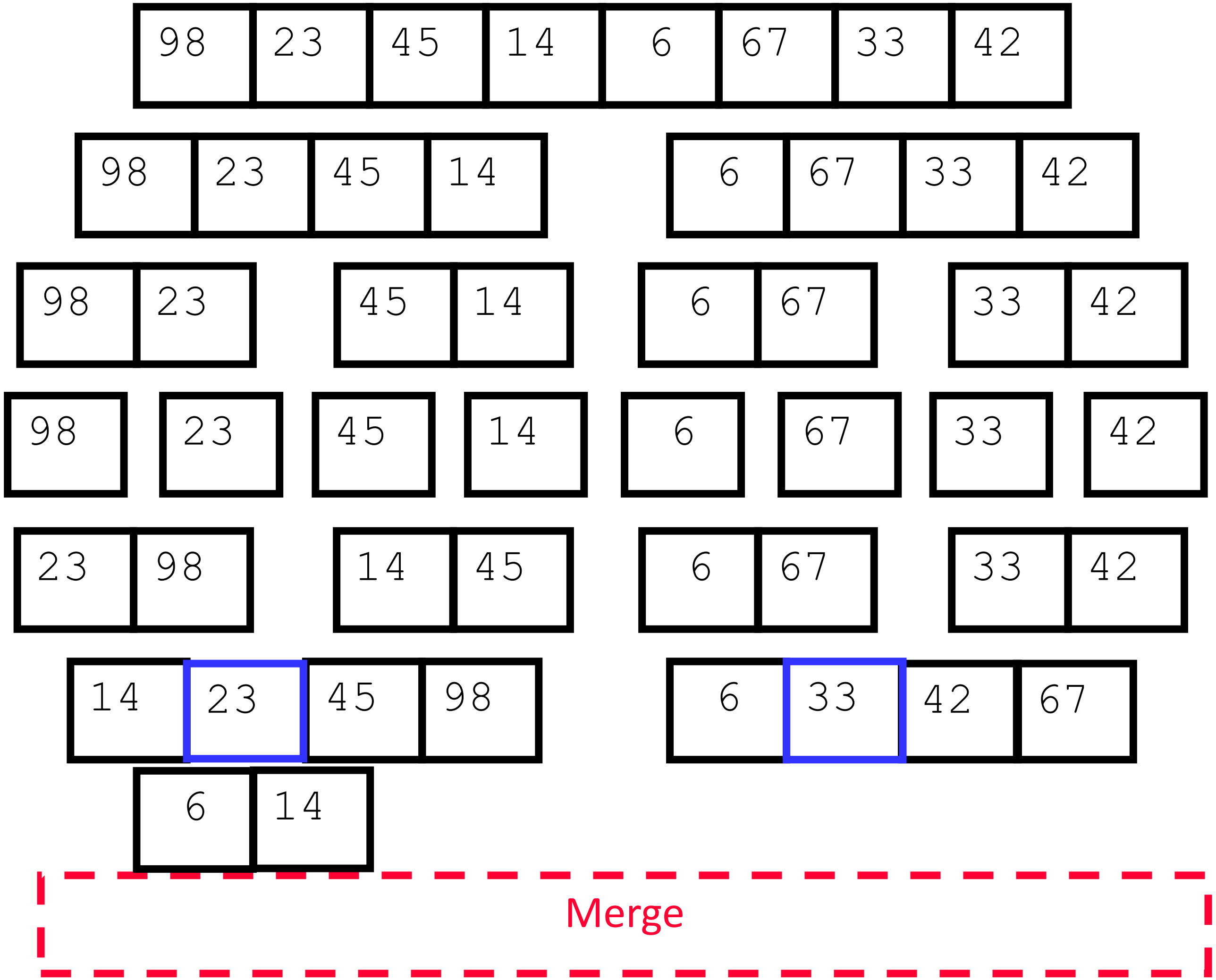
23 98 14 45 6 67 33 42

14 23 45 98 6 33 42 67

6

Merge

Merge Sort (Example)



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

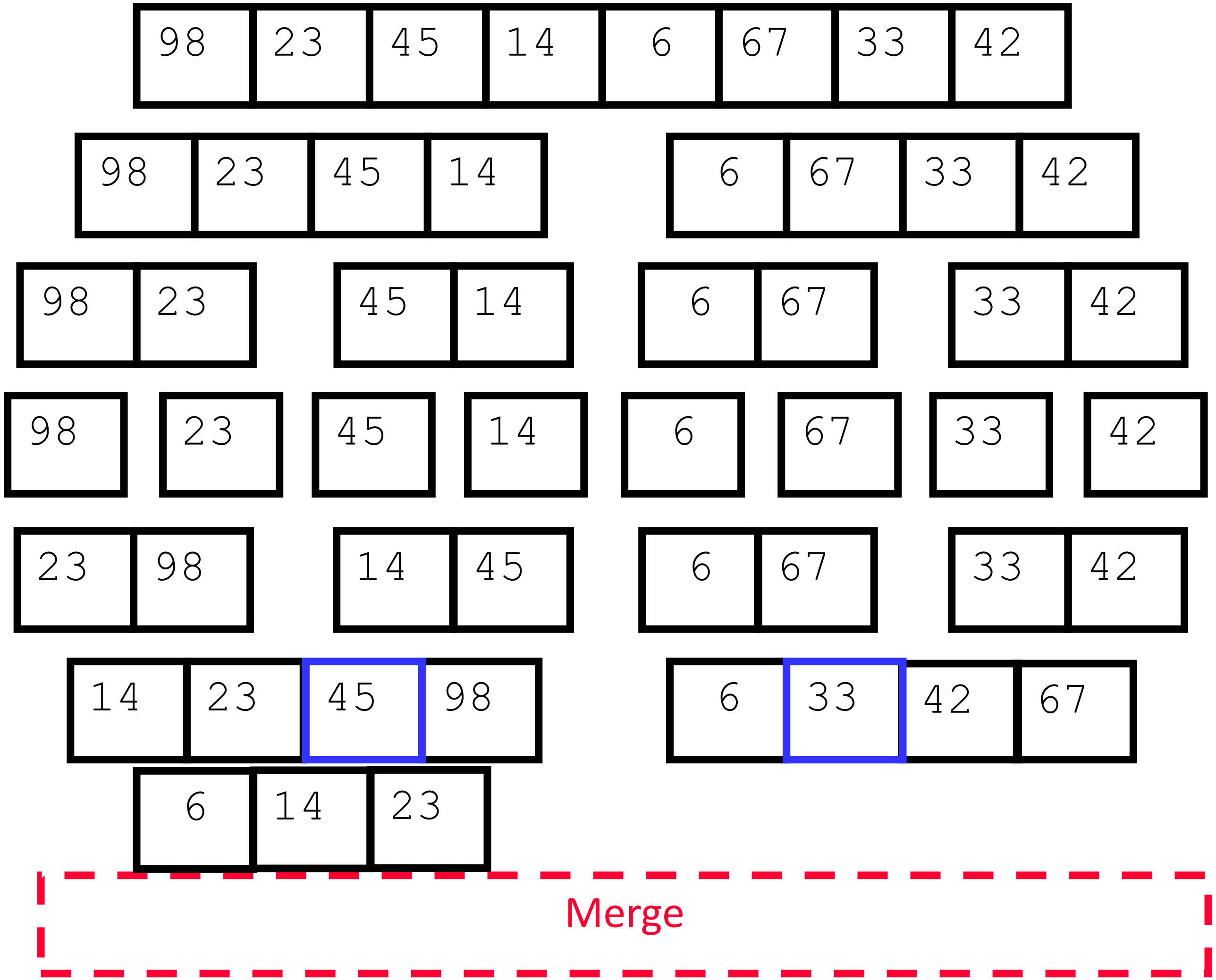
23 98   14 45   6 67   33 42

14 23 45 98   6 33 42 67

6 14

Merge

Merge Sort (Example)



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

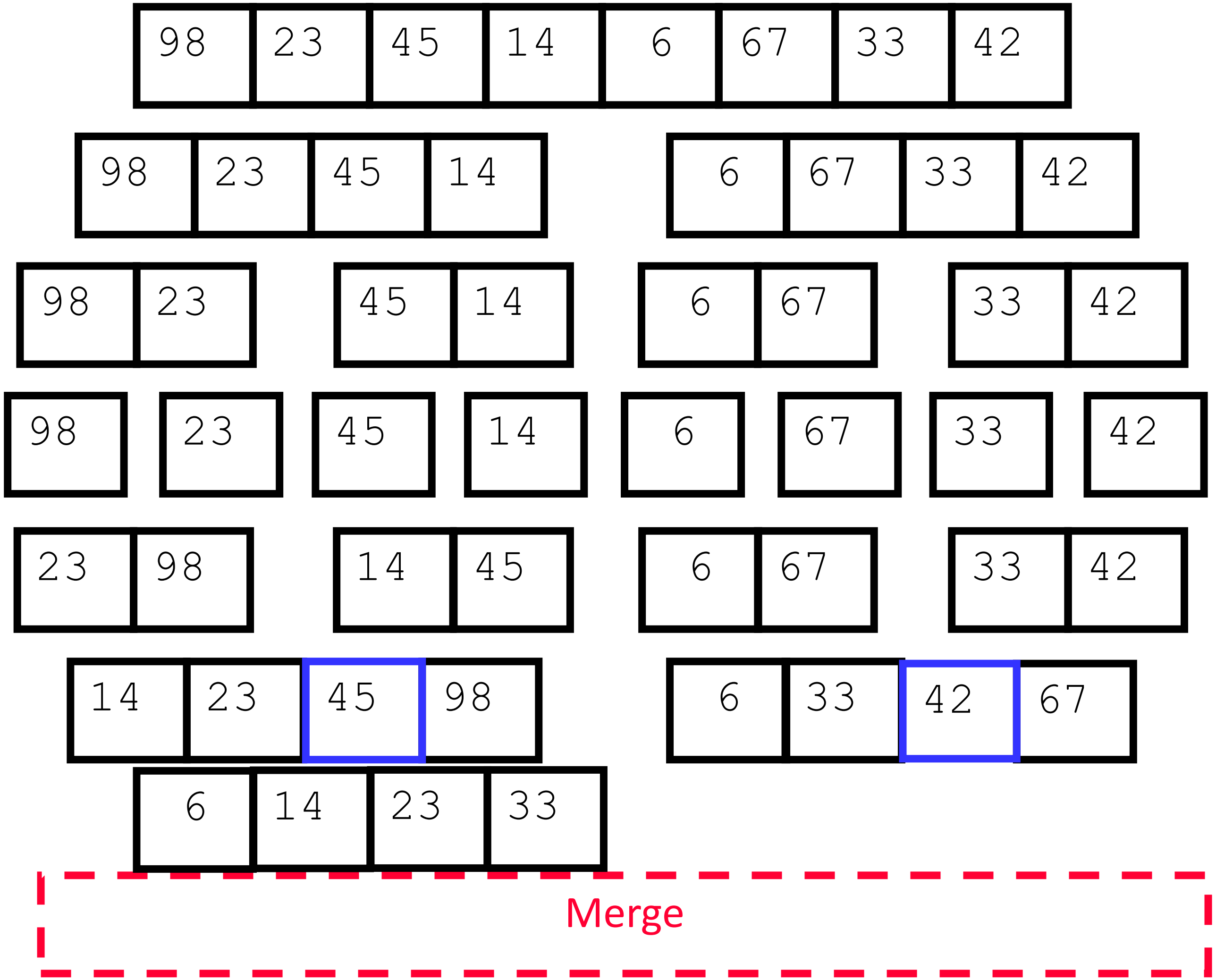
23 98 | 14 45 | 6 67 | 33 42

14 23 45 98 | 6 33 42 67

6 14 23

Merge

Merge Sort (Example)



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

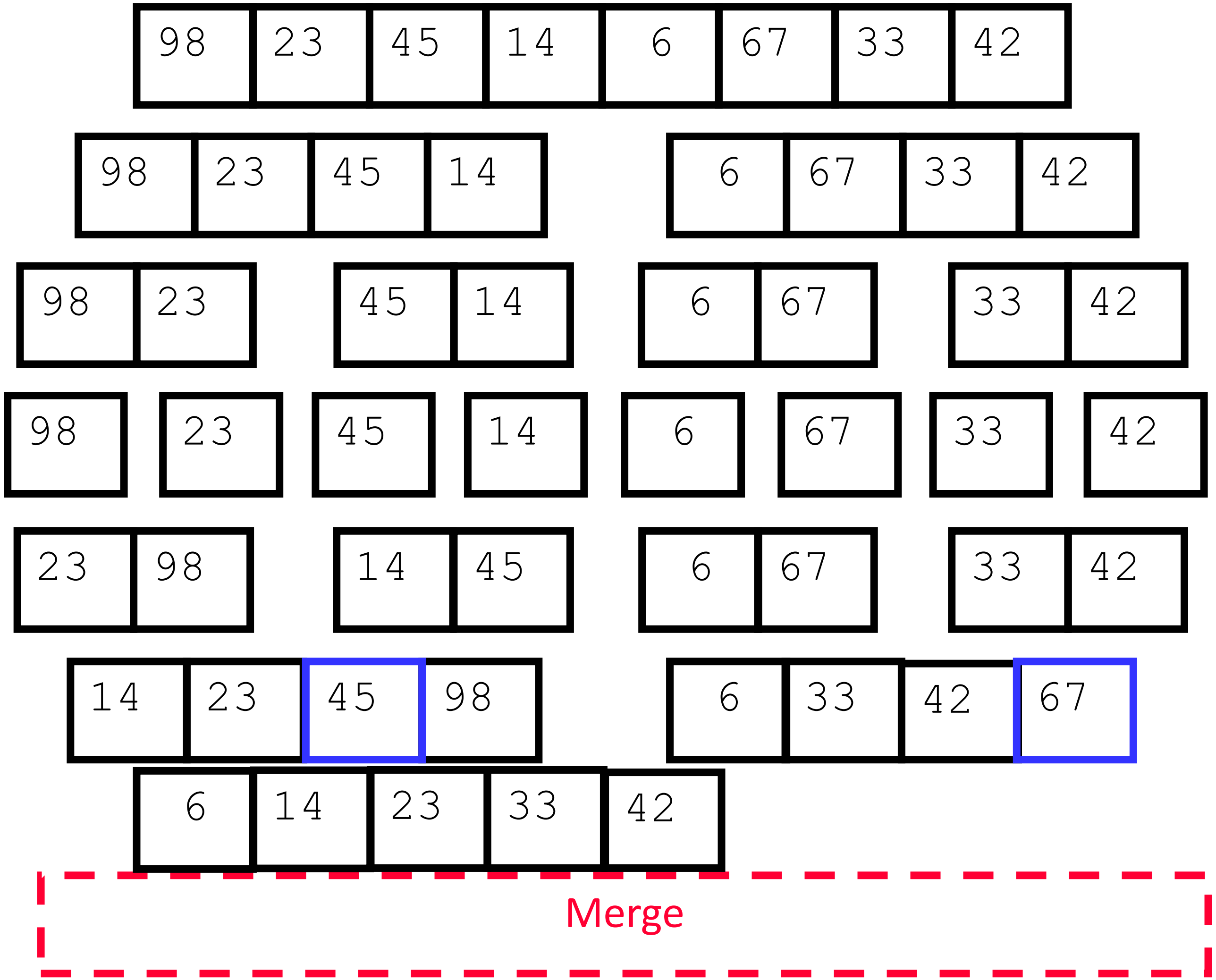
23 98   14 45   6 67   33 42

14 23 45 98   6 33 42 67

6 14 23 33

Merge

Merge Sort (Example)



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

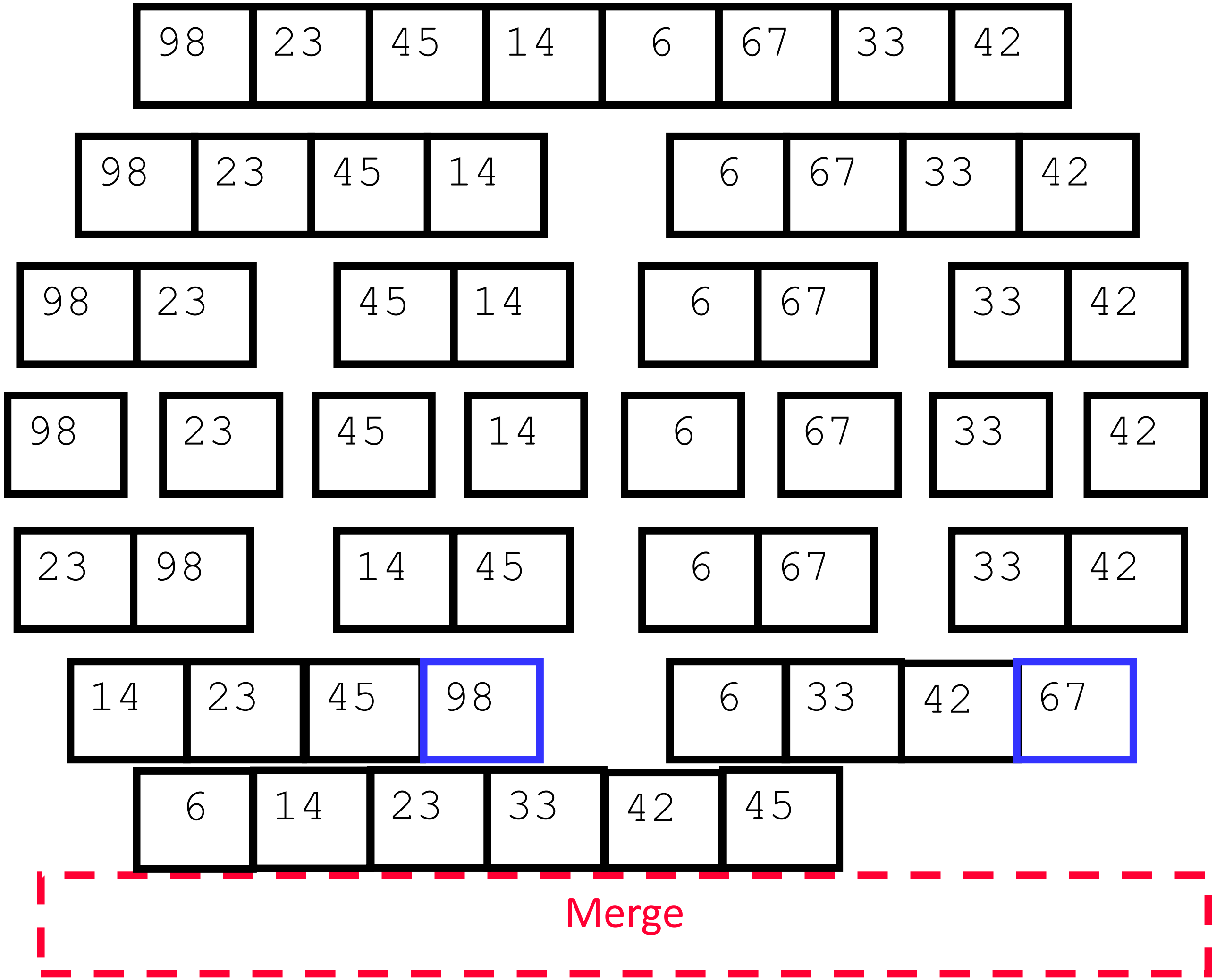
23 98 14 45 6 67 33 42

14 23 45 98 6 33 42 67

6 14 23 33 42

Merge

Merge Sort (Example)



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

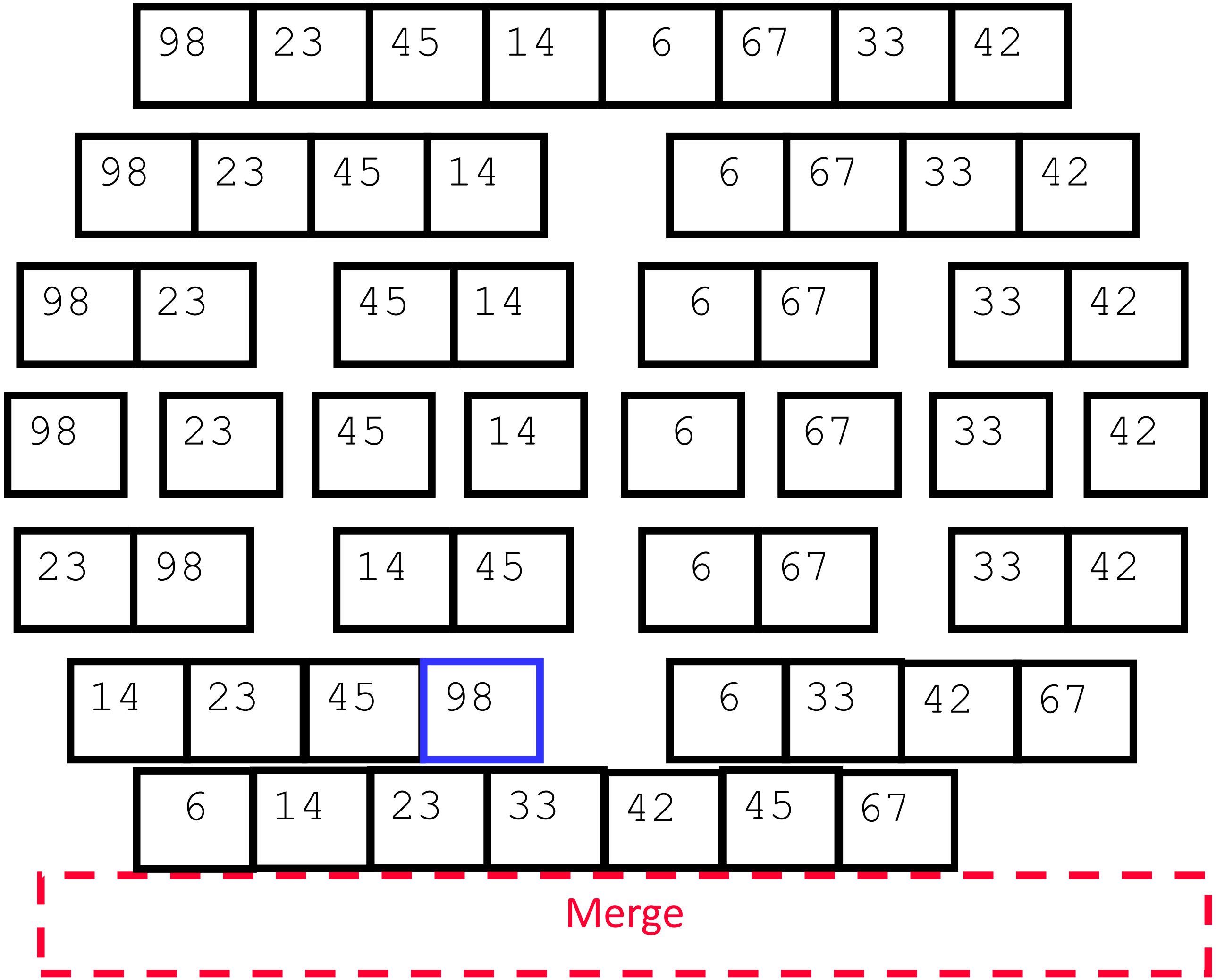
23 98 | 14 45 | 6 67 | 33 42

14 23 45 98 | 6 33 42 67

6 14 23 33 42 45

Merge

Merge Sort (Example)



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

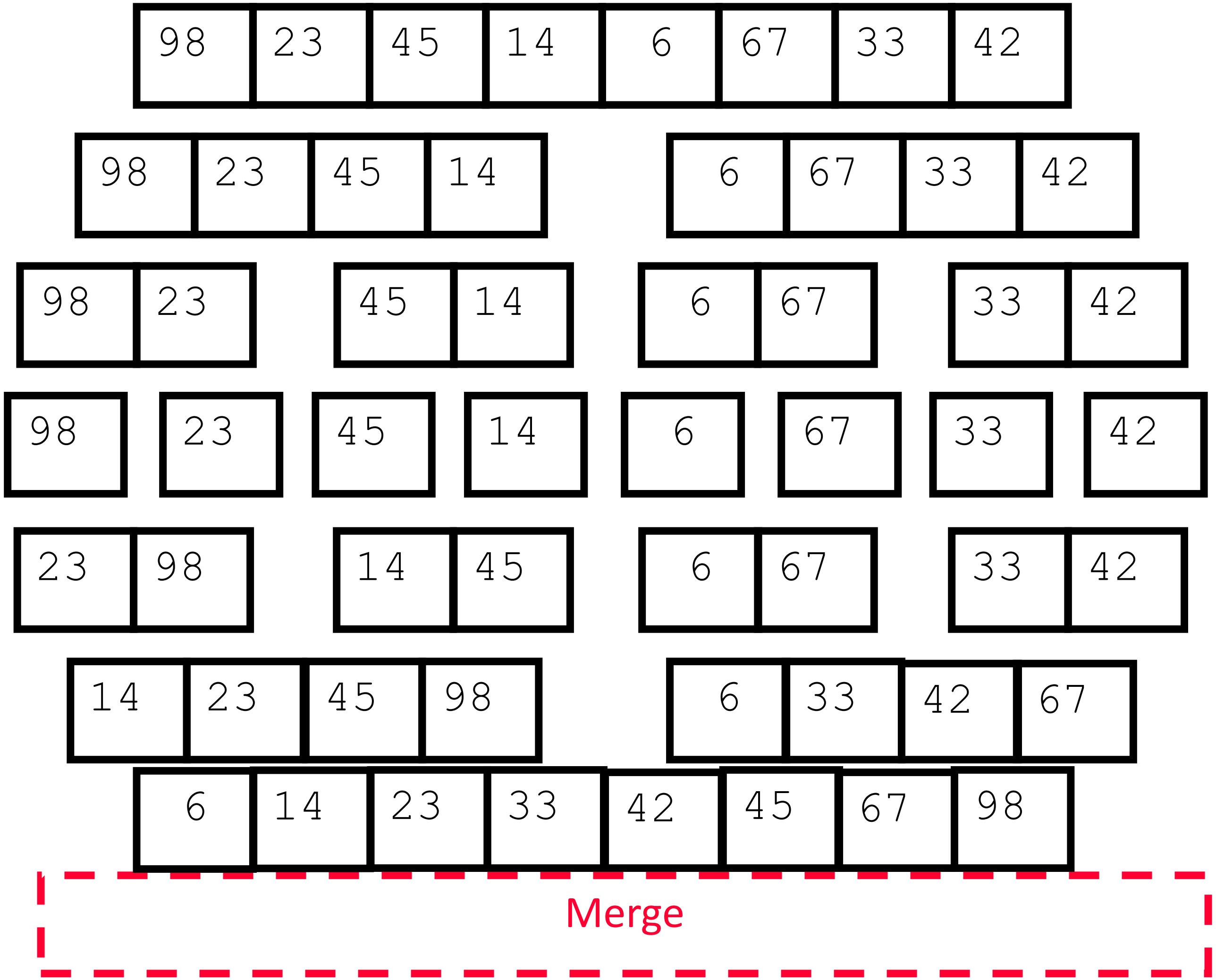
23 98 | 14 45 | 6 67 | 33 42

14 23 45 98 | 6 33 42 67

6 14 23 33 42 45 67

Merge

Merge Sort (Example)



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

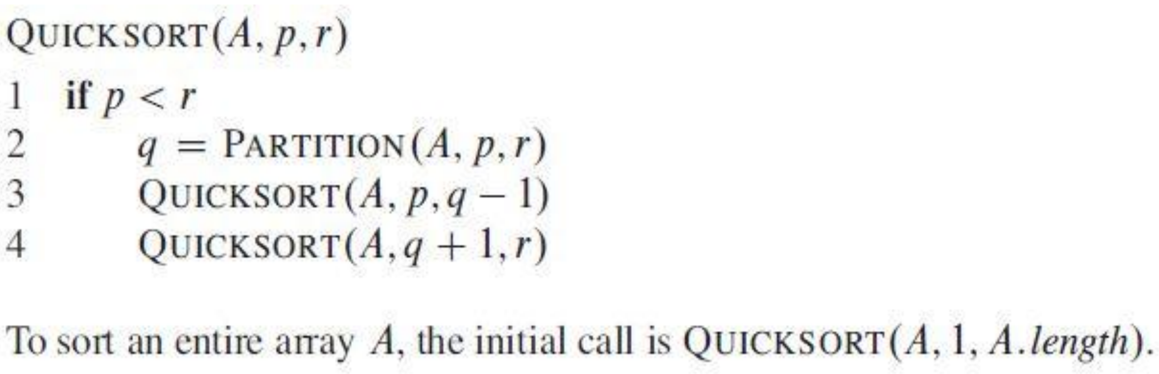
14 23 45 98   6 33 42 67

6 14 23 33 42 45 67 98

Merge

Quicksort

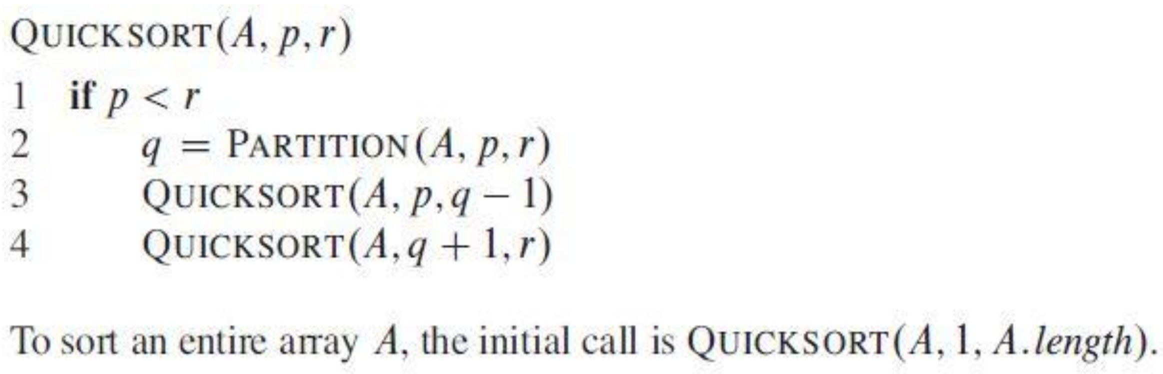
Quicksort



```
QUICKSORT($A, p, r$)
if $p < r$
    $q$ = PARTITION($A, p, r$)
    QUICKSORT($A, p, q - 1$)
    QUICKSORT($A, q + 1, r$)
```

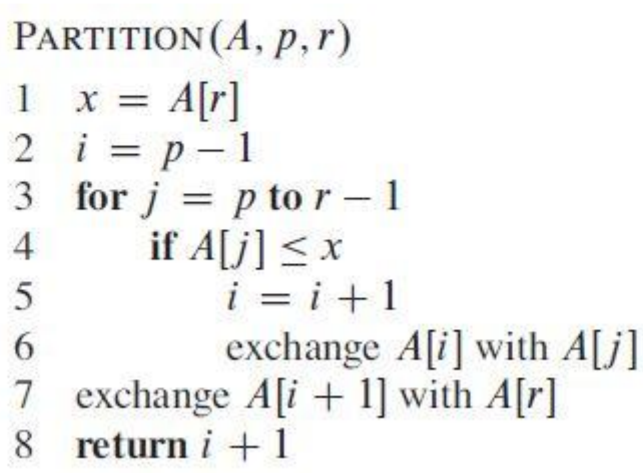
To sort an entire array A, the initial call is QUICKSORT($A, 1, A.length$).

Quicksort



```
QUICKSORT($A, p, r$)
if $p < r$
    $q$ = PARTITION($A, p, r$)
    QUICKSORT($A, p, q - 1$)
    QUICKSORT($A, q + 1, r$)
```

To sort an entire array A, the initial call is QUICKSORT(A, 1, $A.length$).



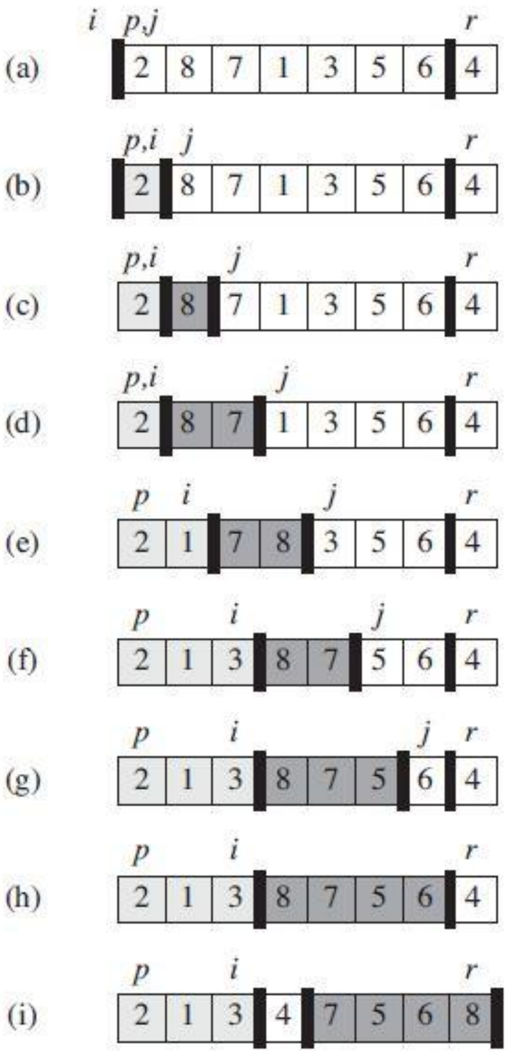
```
PARTITION($A, p, r$)
$x = A[r]$
$i = p - 1$
for $j = p$ to $r - 1$
    if $A[j] \leq x$
        $i = i + 1$
        exchange $A[i]$ with $A[j]$
exchange $A[i + 1]$ with $A[r]$
return $i + 1$
```

Quicksort

```
QUICKSORT($A, p, r$)
if $p < r$
    $q$ = PARTITION($A, p, r$)
    QUICKSORT($A, p, q - 1$)
    QUICKSORT($A, q + 1, r$)
```

To sort an entire array A, the initial call is QUICKSORT(A, 1, A.*length*).

```
PARTITION($A, p, r$)
$x = A[r]$
$i = p - 1$
for $j = p$ to $r - 1$
    if $A[j] \le x$
        $i = i + 1$
        exchange $A[i]$ with $A[j]$
exchange $A[i + 1]$ with $A[r]$
return $i + 1$
```



(a) i p,j r 2 8 7 1 3 5 6 4
(b) p,i j r 2 8 7 1 3 5 6 4
(c) p,i j r 2 8 7 1 3 5 6 4
(d) p,i j r 2 8 7 1 3 5 6 4
(e) p i j r 2 1 7 8 3 5 6 4
(f) p i j r 2 1 3 8 7 5 6 4
(g) p i j r 2 1 3 8 7 5 6 4
(h) p i r 2 1 3 8 7 5 6 4
(i) p i r 2 1 3 4 7 5 6 8