Algoritma dan Struktur Data

Merge Sort

Umi Sa'adah
Tita Karlita
Entin Martiana Kusumaningtyas
Arna Fariza
2021

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Sorting Algorithms

1. Selection
2. Insertion
3. Bubble
4. Shell
5. Merge
6. Quick

Merge Sort

- Merupakan algoritma divide-and-conquer (membagi dan menyelesaikan).
- Membagi array menjadi dua bagian sampai sub-array hanya berisi satu elemen.
- Menyelesaikan dengan cara menggabungkan solusi sub-problem :
  - Membandingkan elemen pertama sub-array
  - Memindahkan elemen terkecil dan meletakkannya ke array hasil
  - Lanjutkan proses sampai semua elemen berada pada array hasil

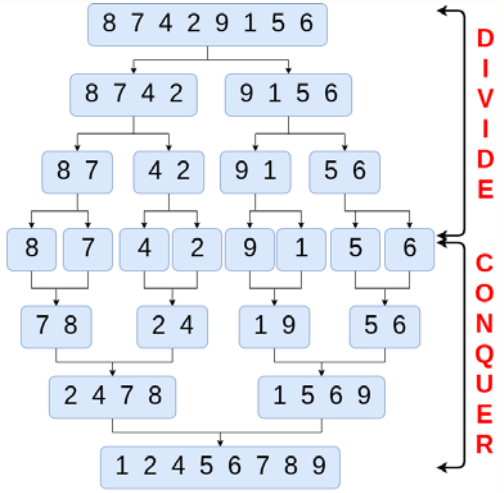
Merge Sort

Dibawah ini adalah data yang akan dilakukan proses Merge Sort

37	23	6	89	15	12	2	19

Merge Sort Algorithm

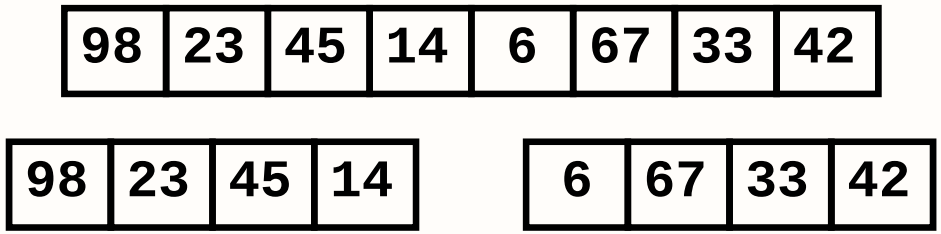
Mergesort (Passed an array)
  Jika ukuran array > 1
   Bagi array menjadi dua
   Panggil fungsi **Mergesort** untuk bagian pertama
   Panggil fungsi **Mergesort** untuk bagian kedua
   Merge dua bagian tersebut.

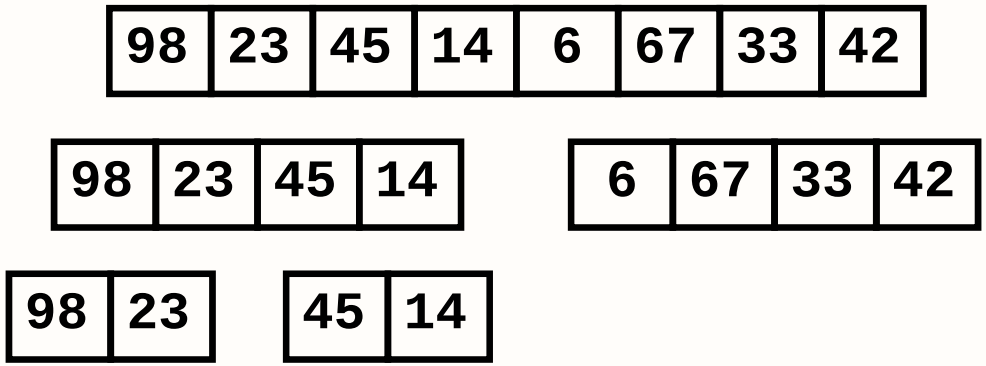


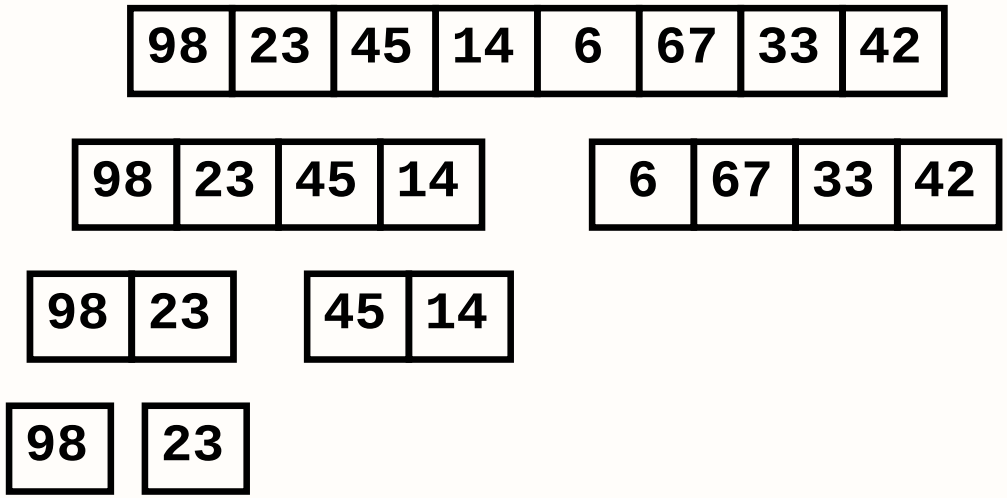
Merge Sort

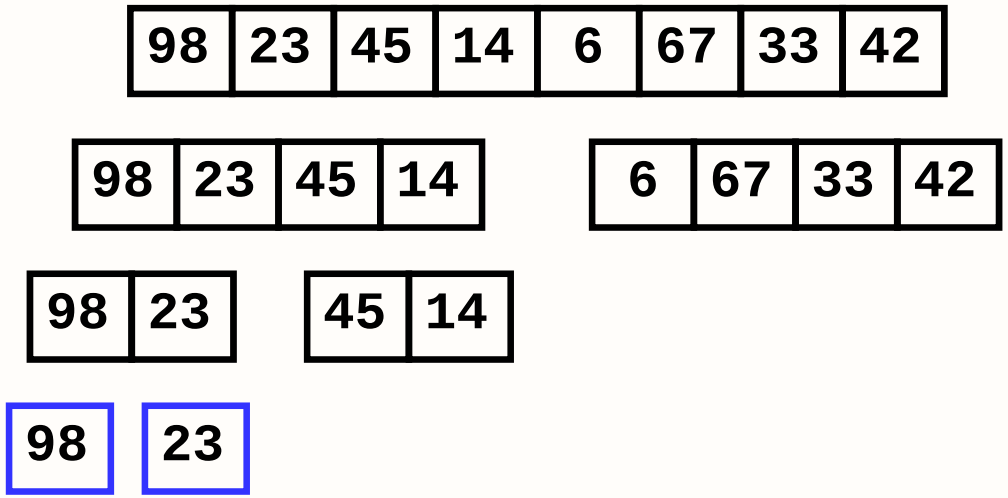
Merge (Passed two arrays)
  Bandingkan elemen pertama dari kedua array
  Pilih yang lebih kecil dan tempatkan pada array hasil, update posisi elemen pertama pd array yang telah diambil elemennya
  (Jika salah satu array input telah kosong, maka letakkan elemen yang tersisa dari array lainnya ke array hasil)

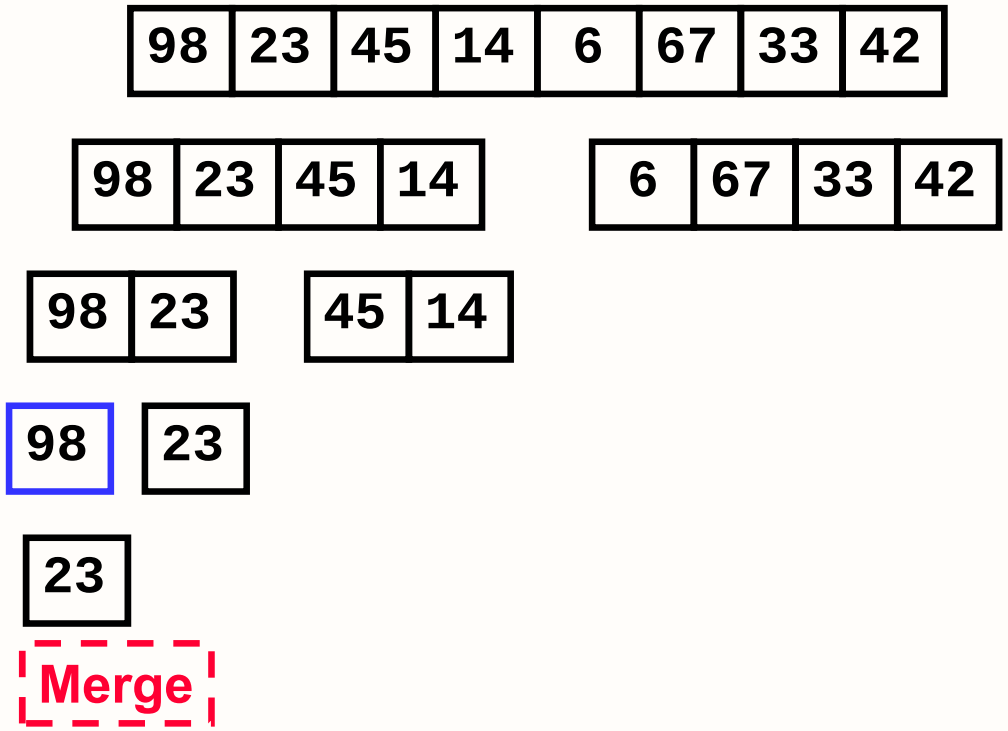
98	23	45	14	6	67	33	42

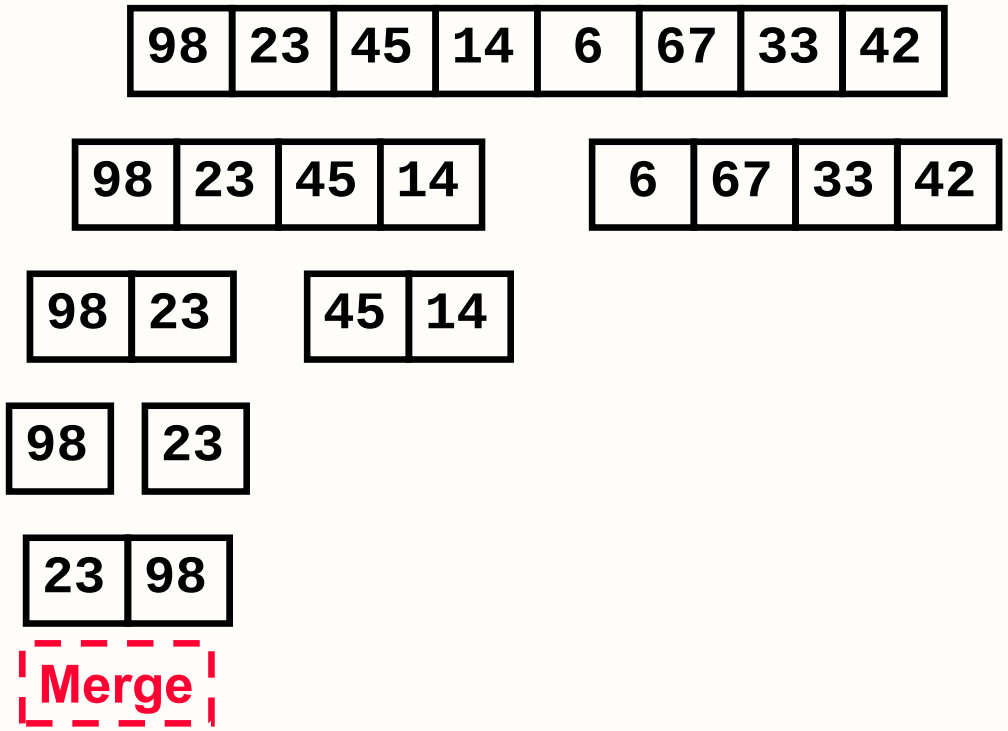

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42

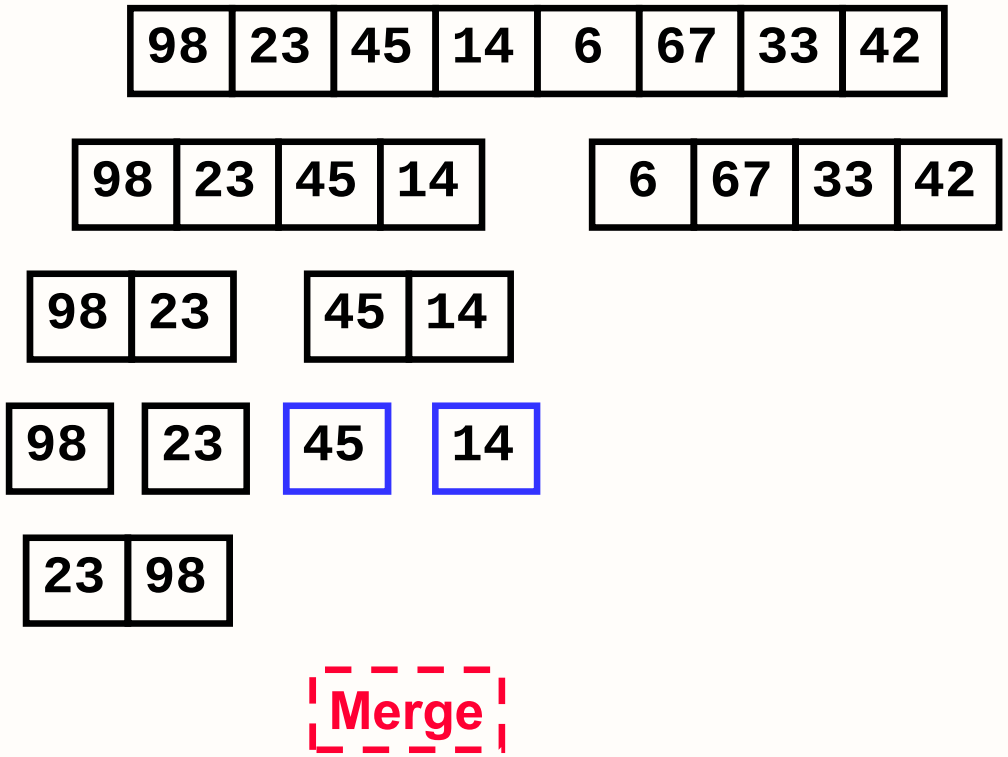

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14

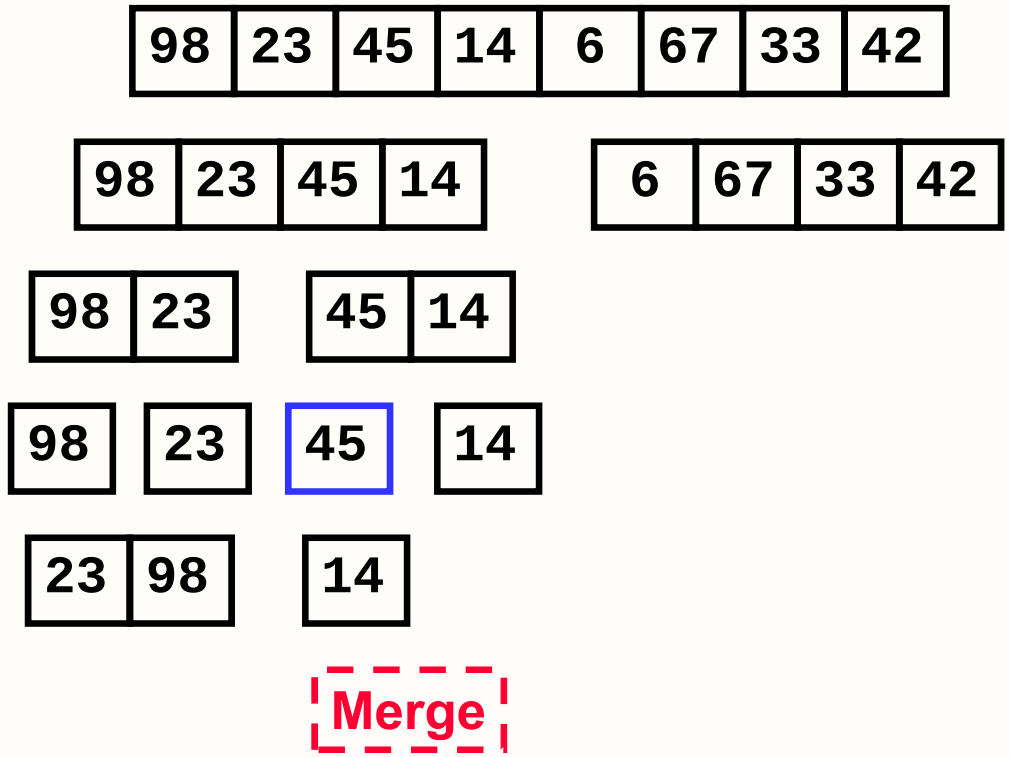

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23

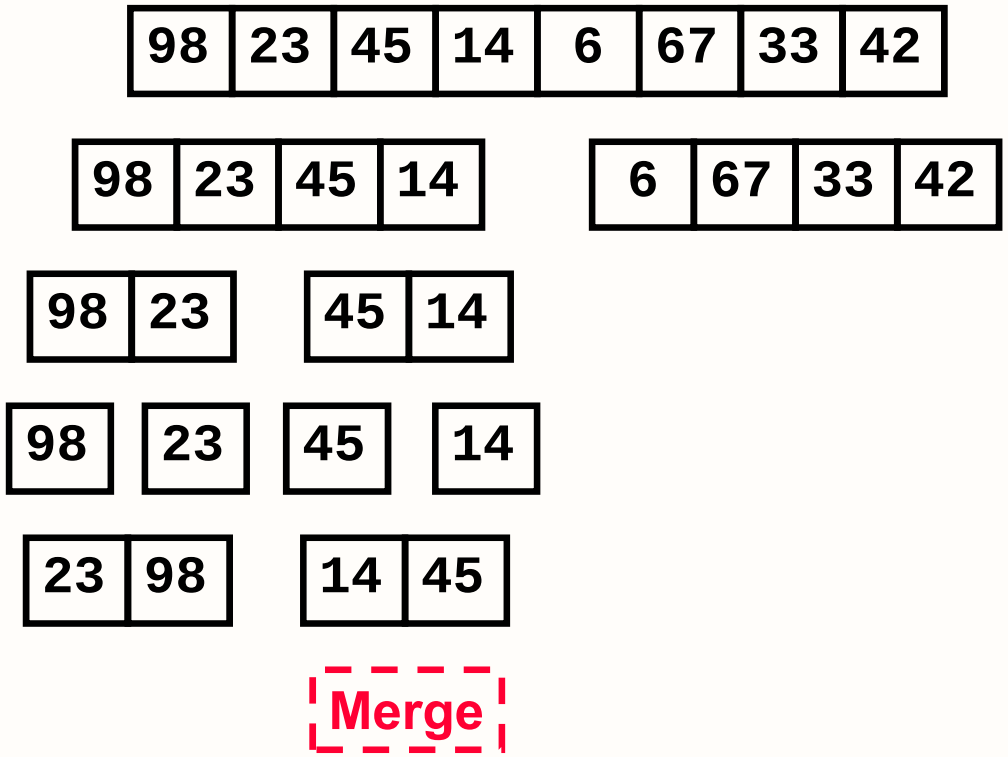

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
Merge

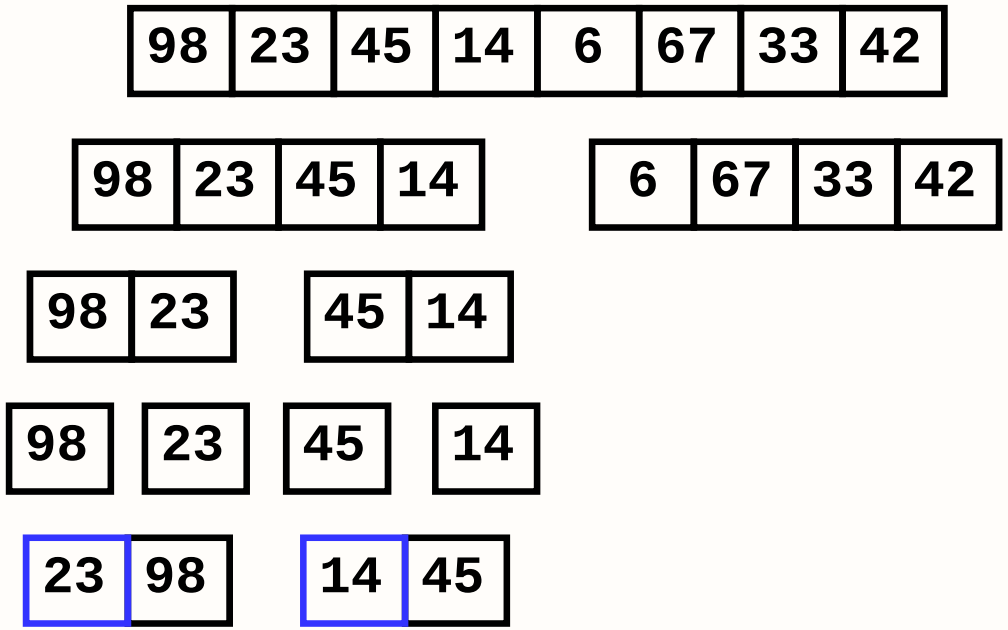

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
23
Merge

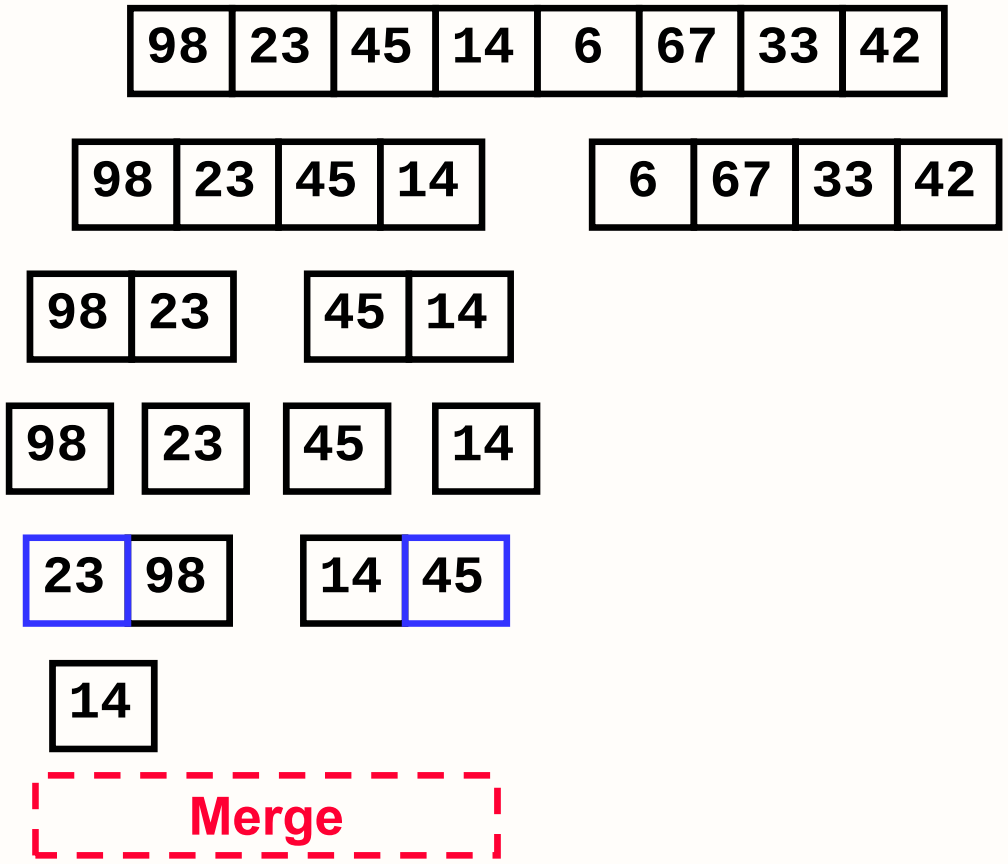

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
23 98
Merge

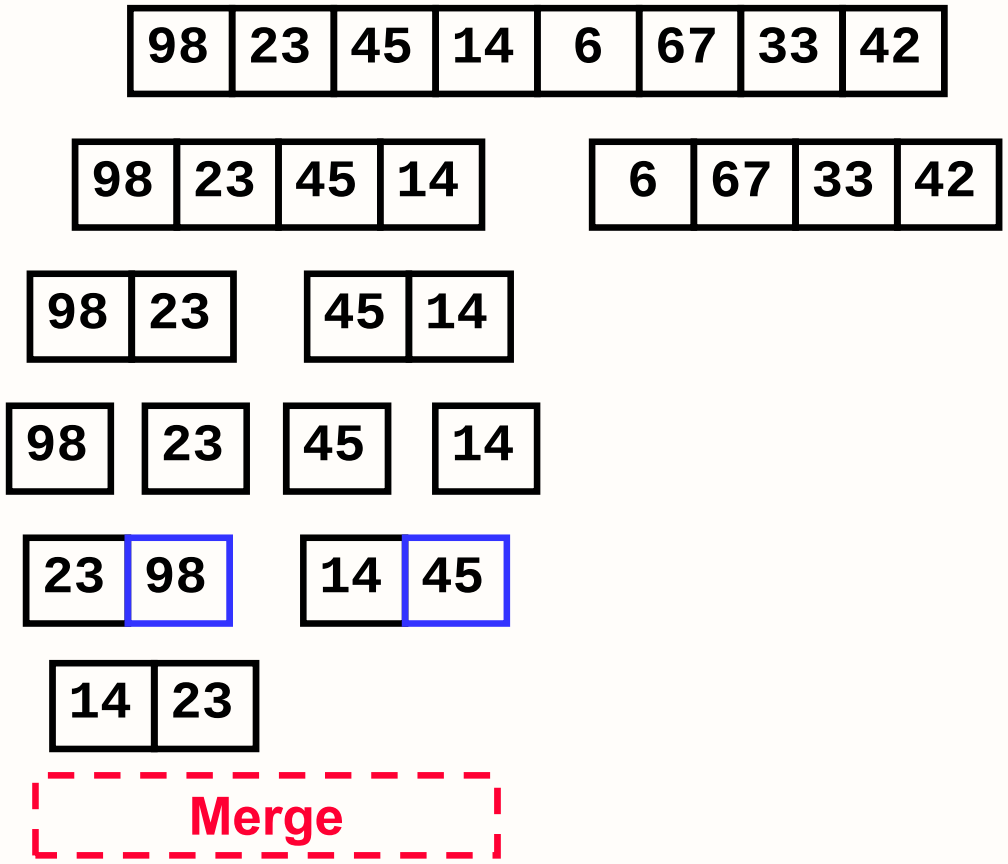

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
Merge

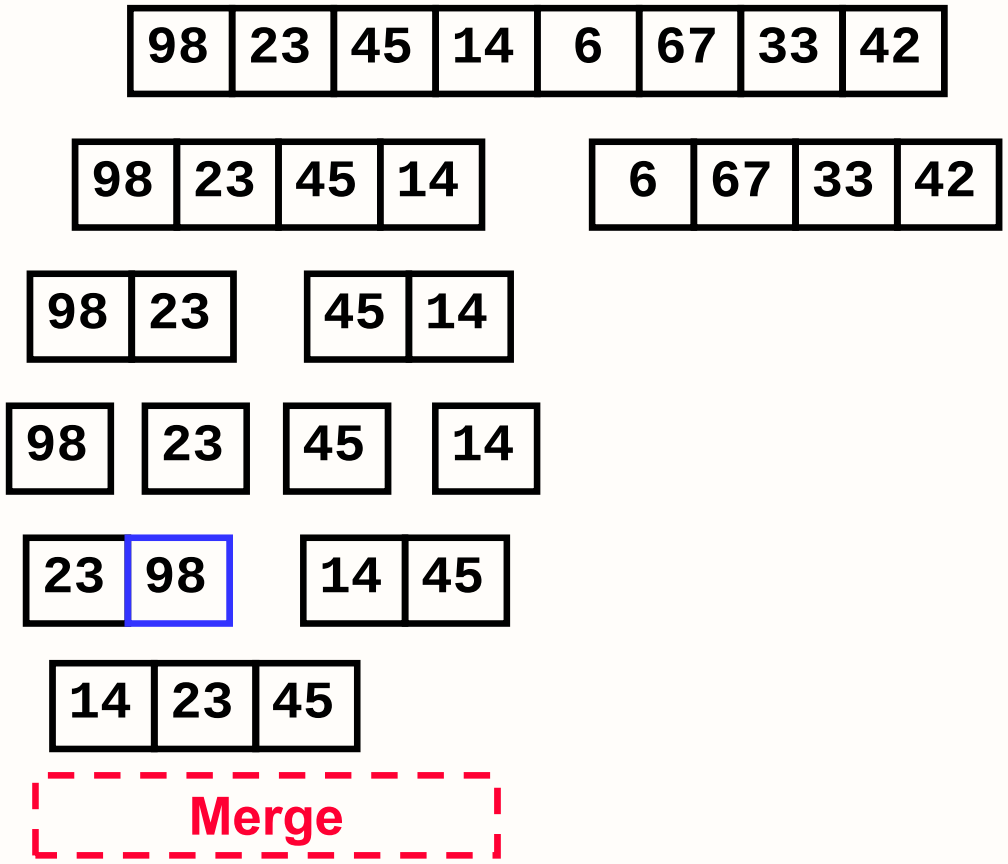

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14
Merge

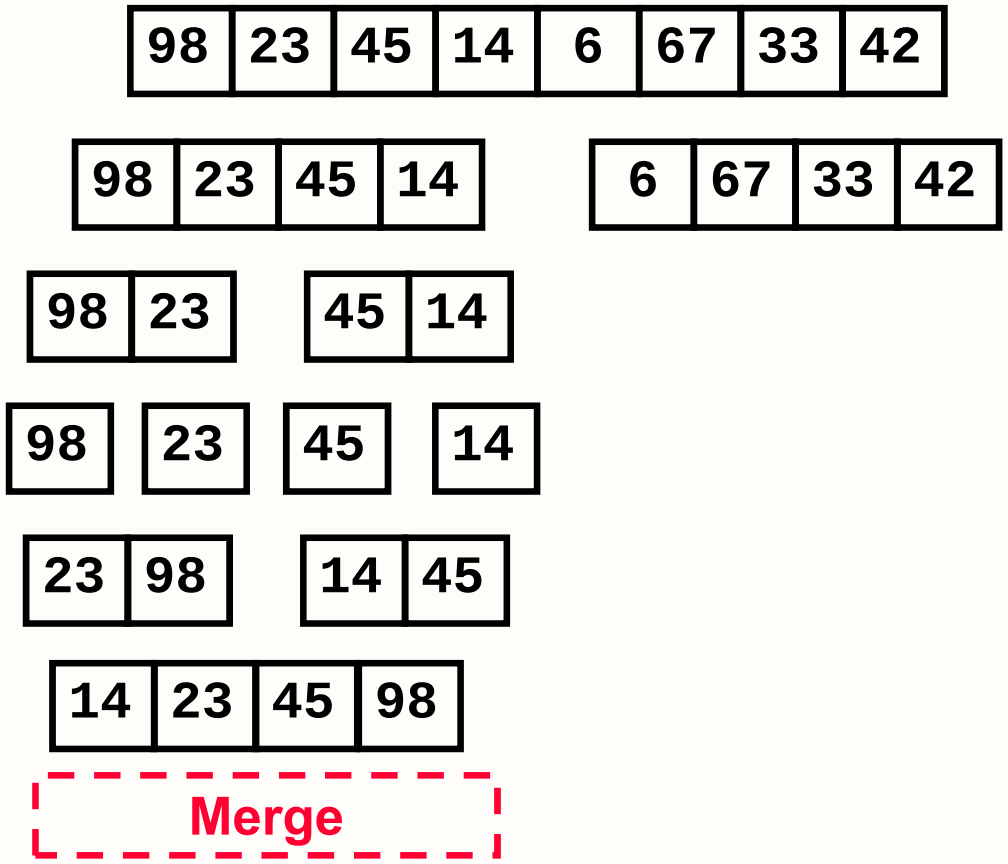

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
Merge

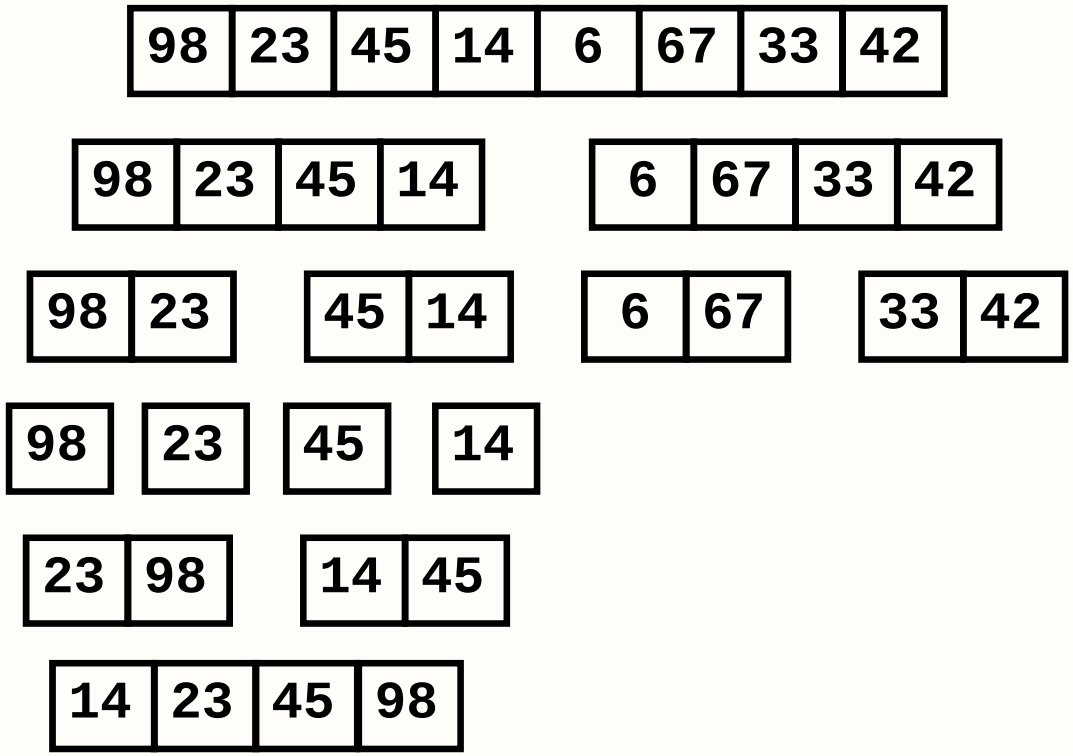

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
Merge

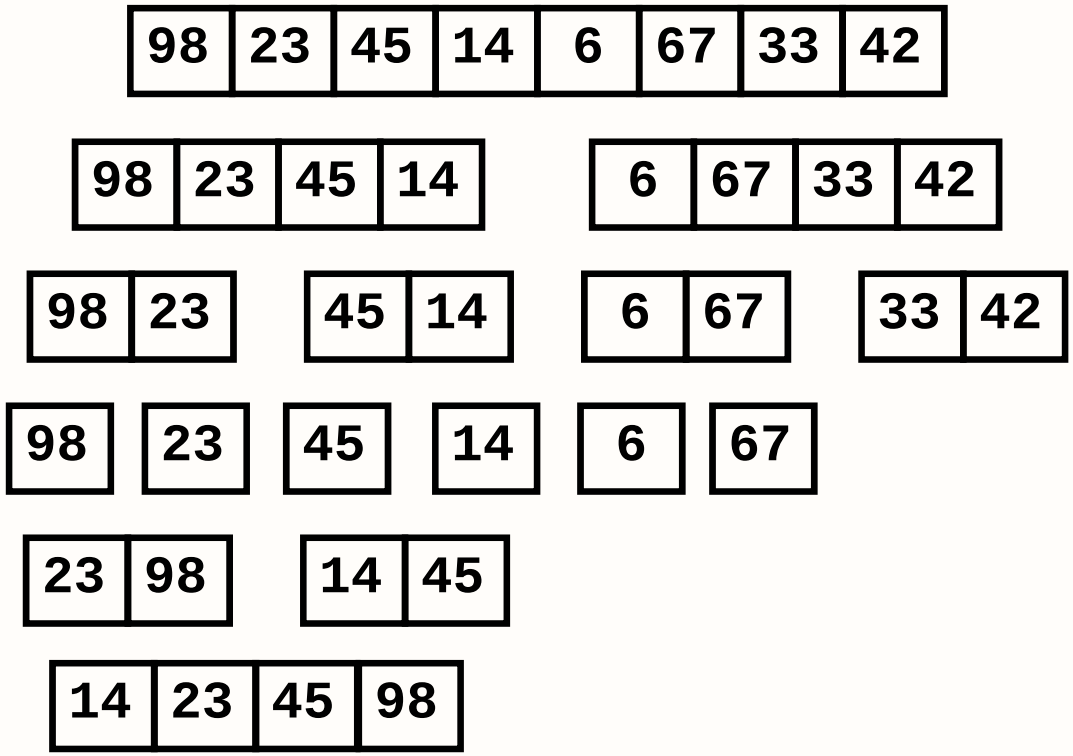

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14
Merge

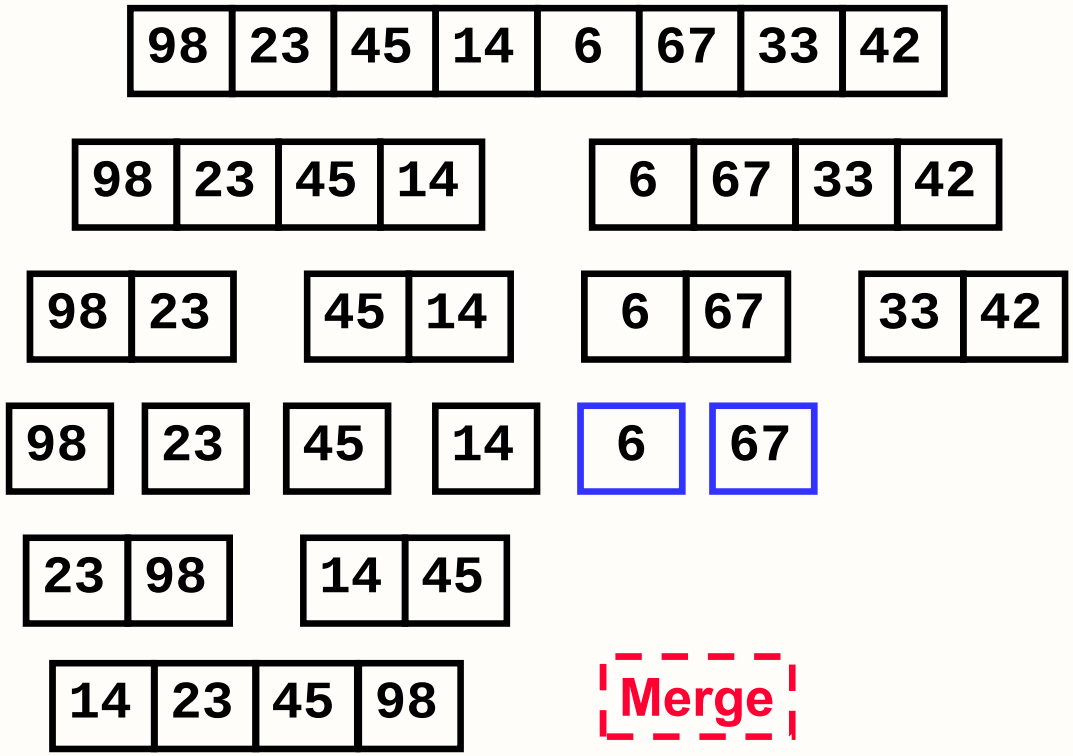

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23
Merge

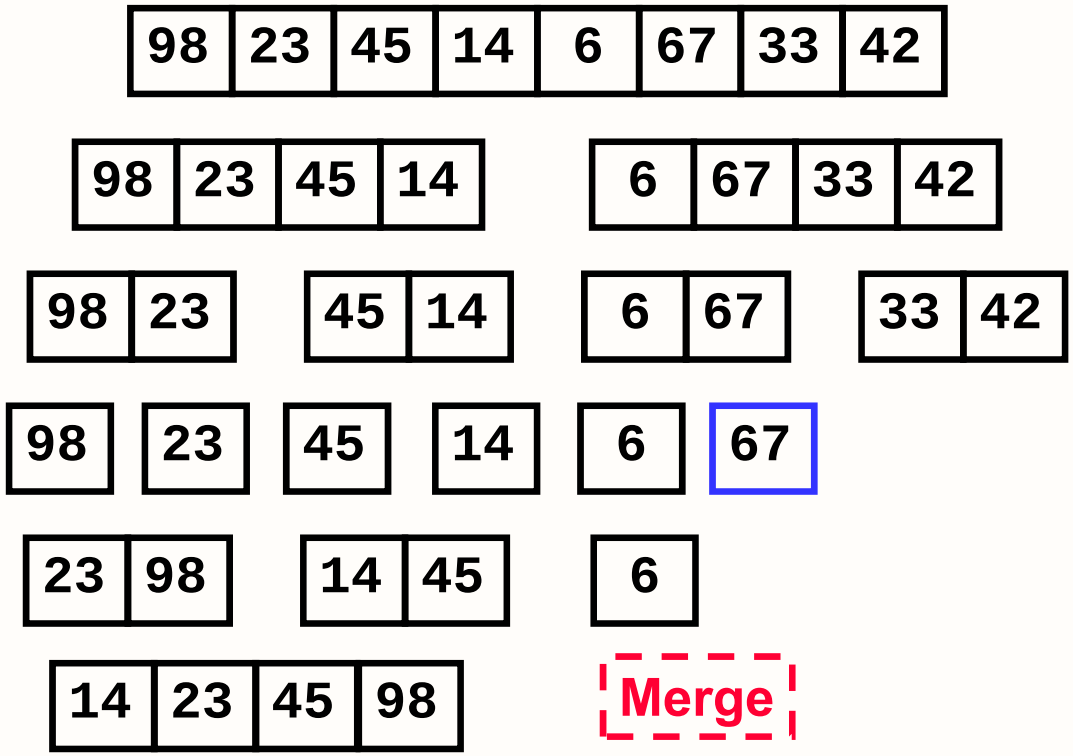

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23 45
Merge

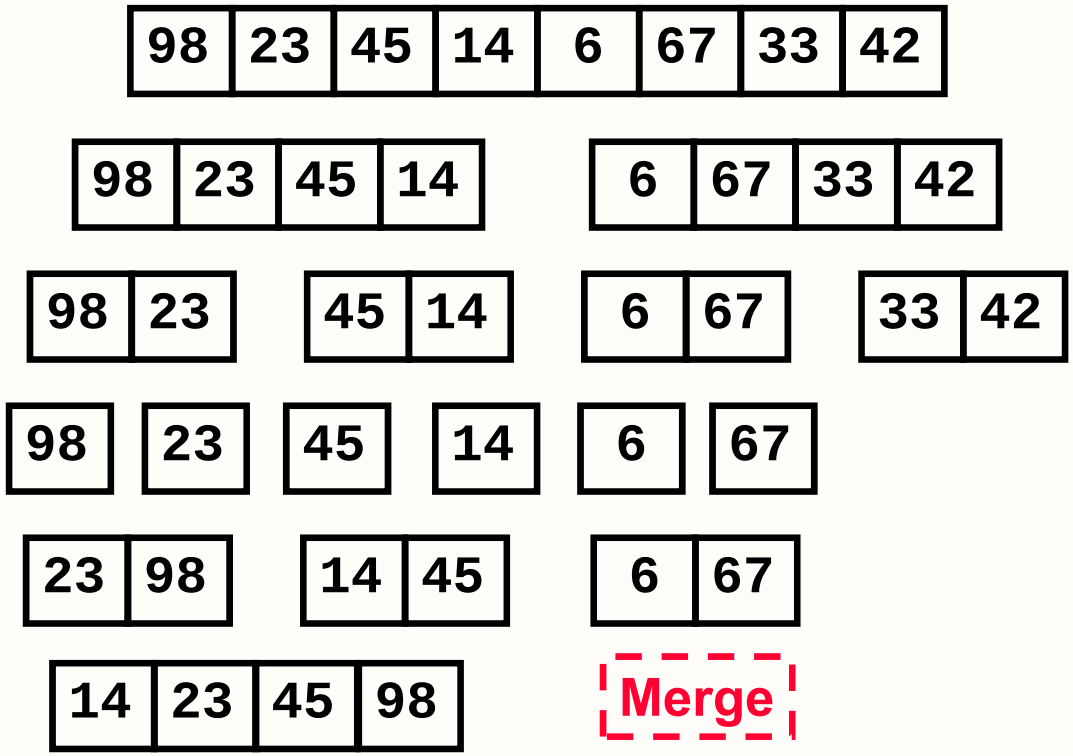

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23 45 98
Merge

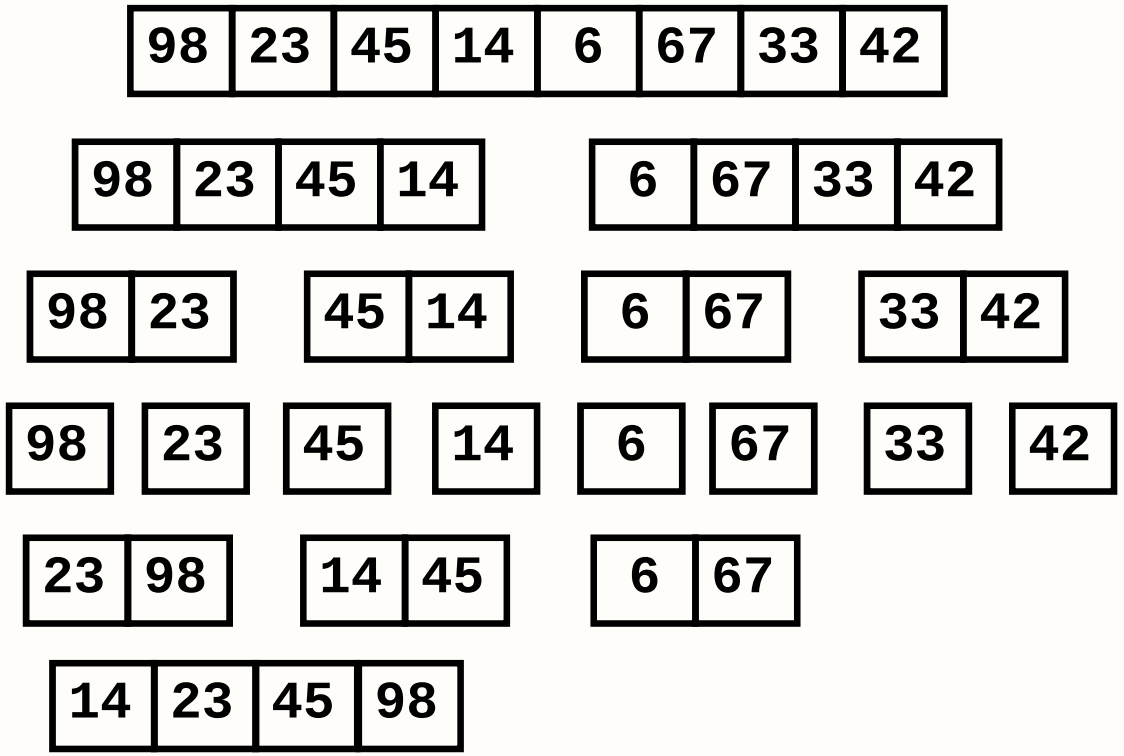

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
23 98
14 45
14 23 45 98

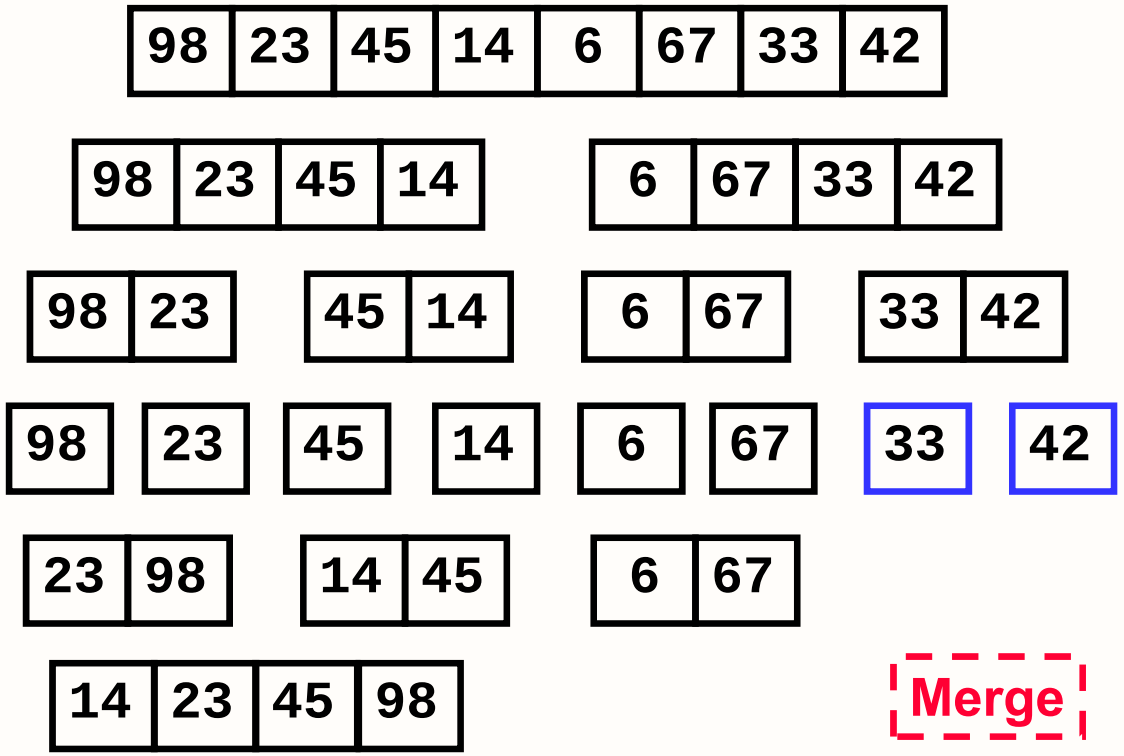

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
14 23 45 98

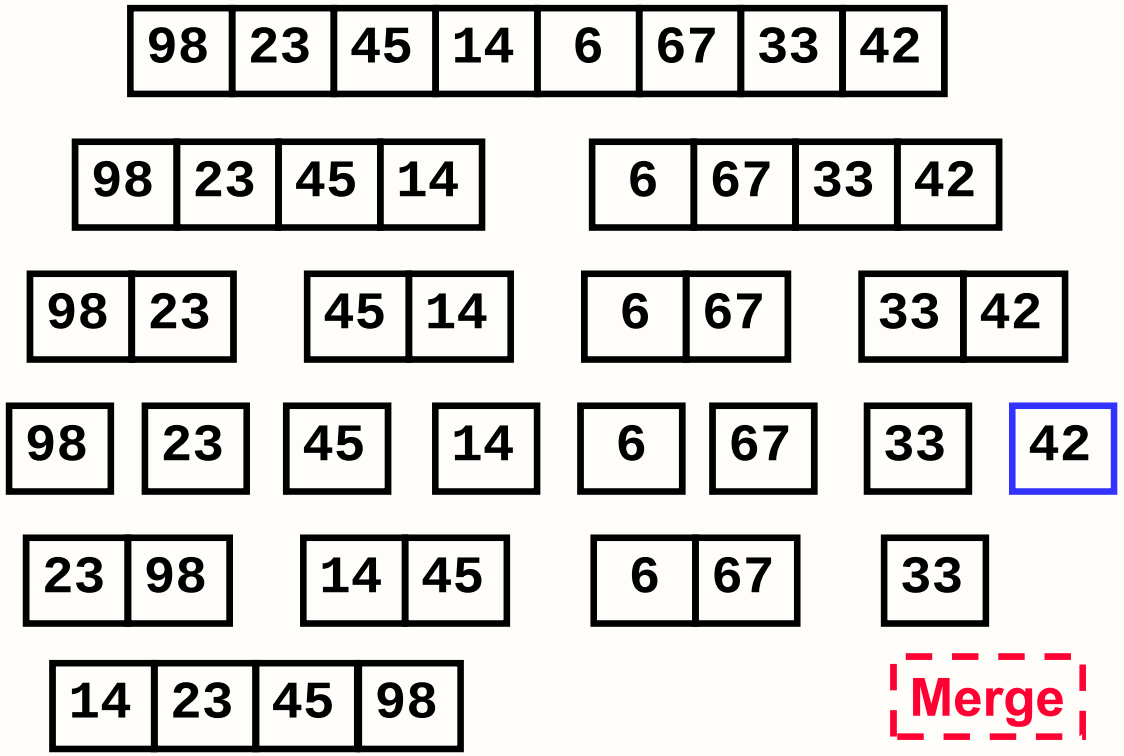

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
14 23 45 98
Merge

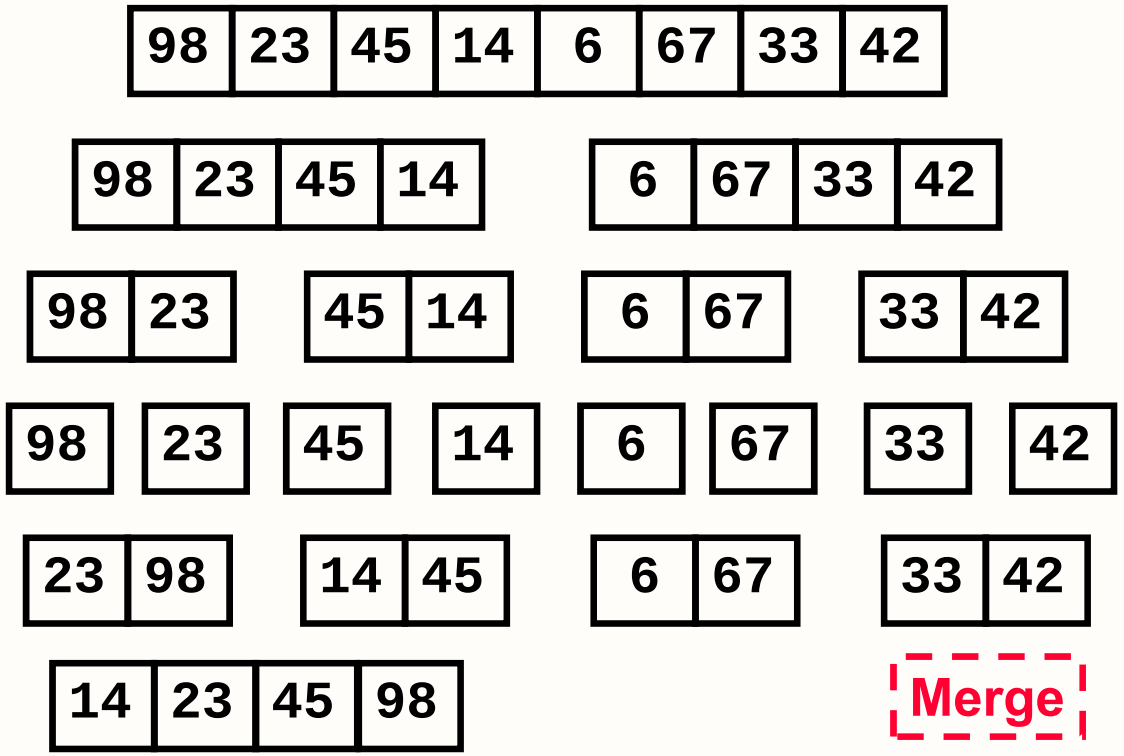

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
6
14 23 45 98
Merge

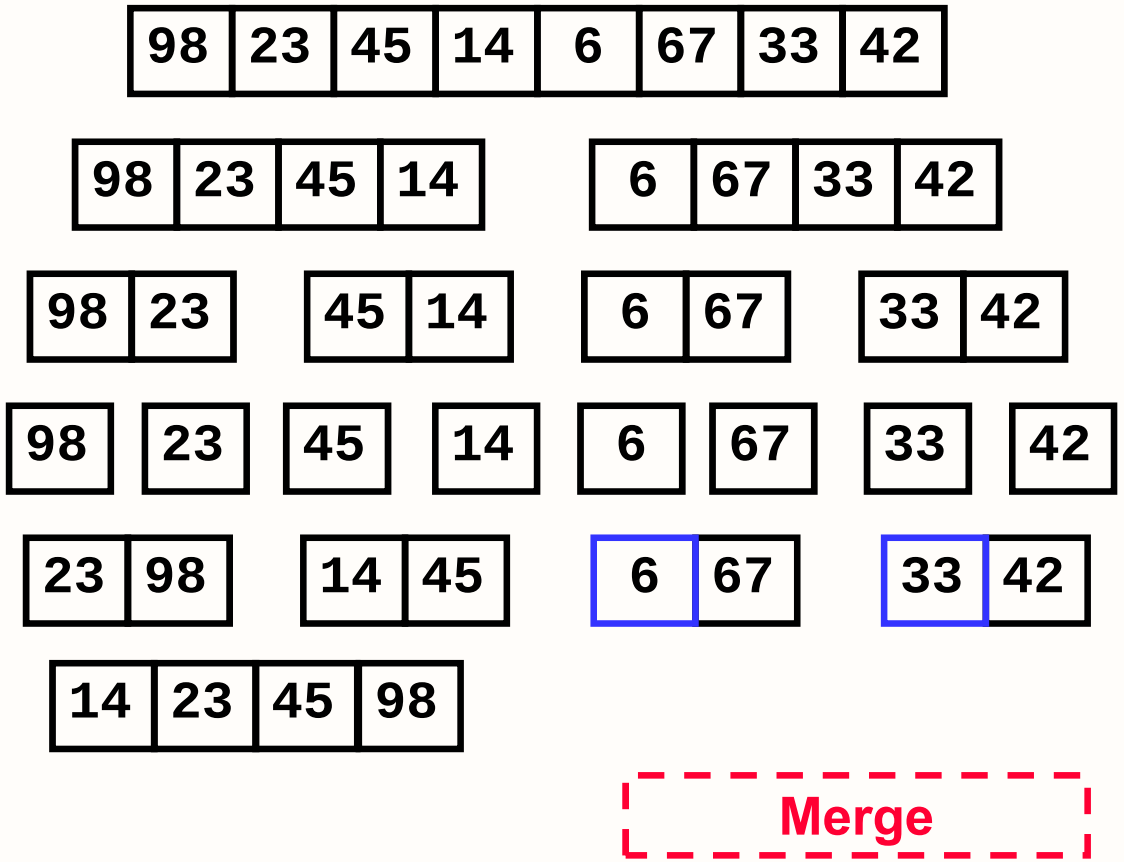

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
6 67
14 23 45 98
Merge

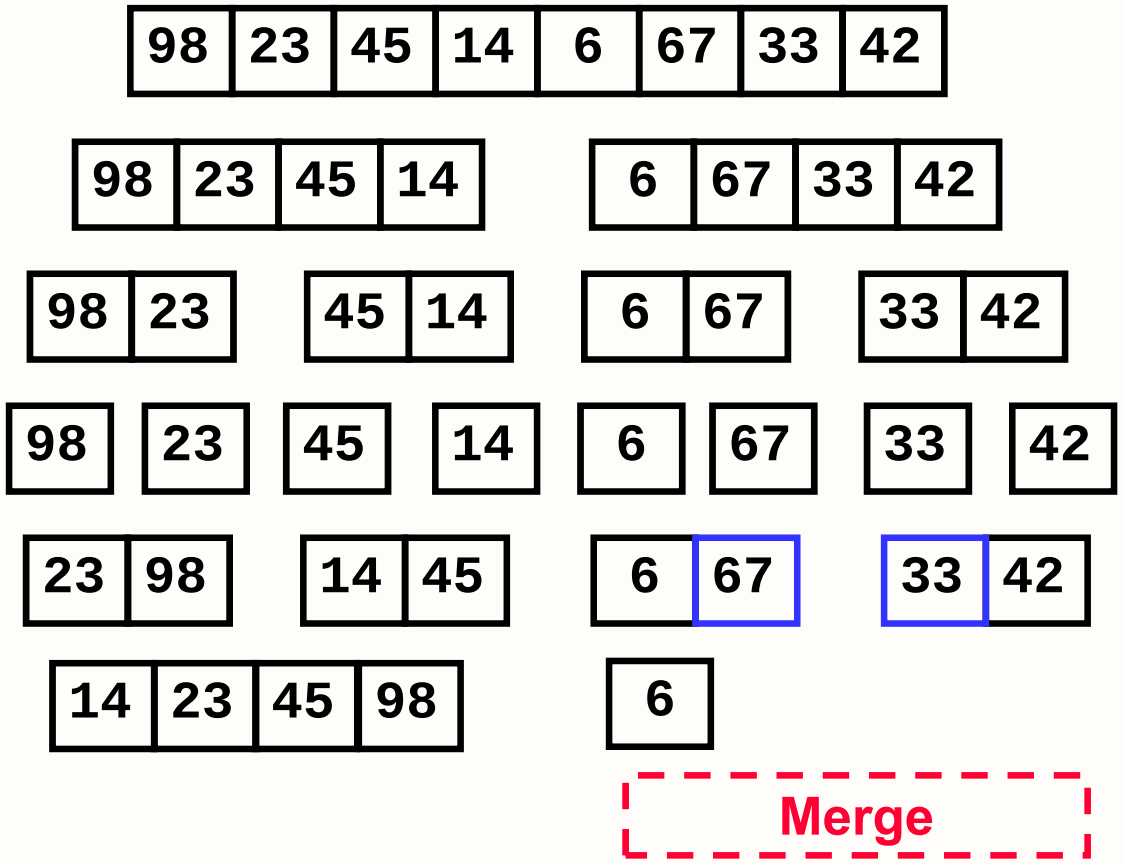

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
14 23 45 98

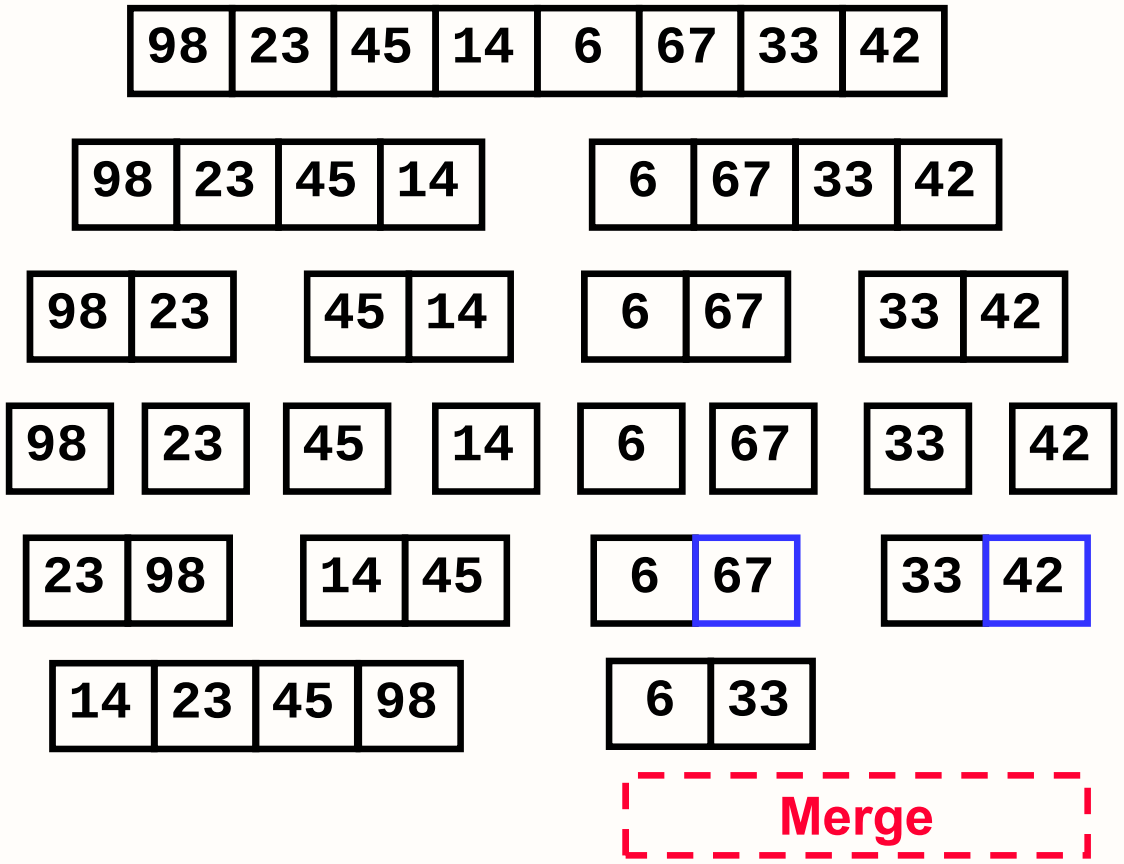

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
14 23 45 98
Merge

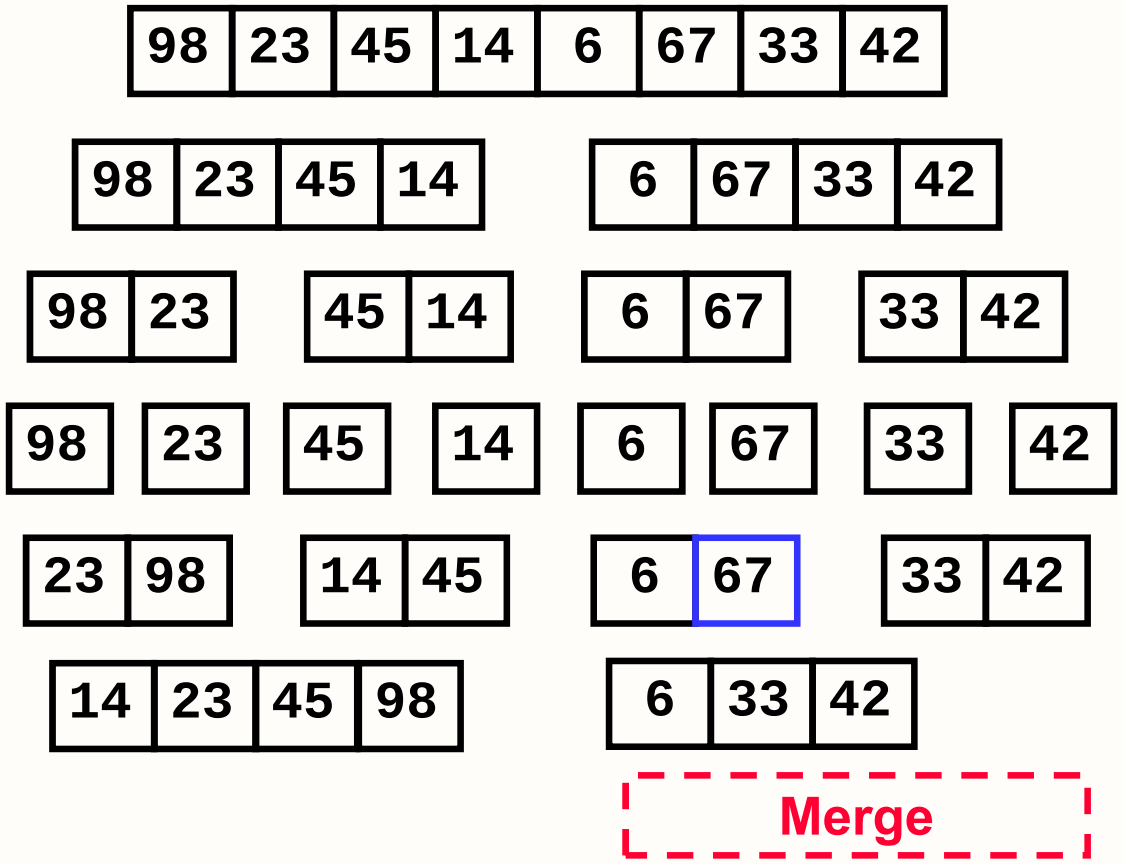

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33
14 23 45 98
Merge

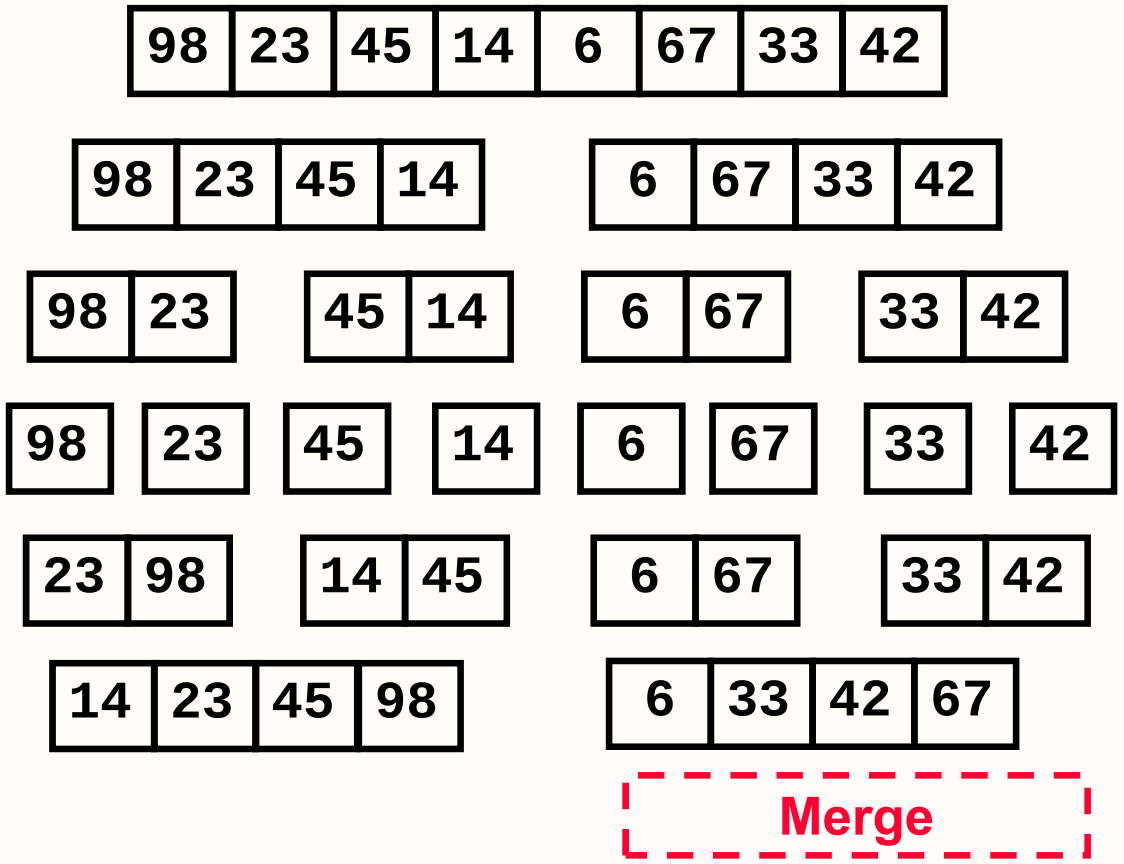

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
Merge

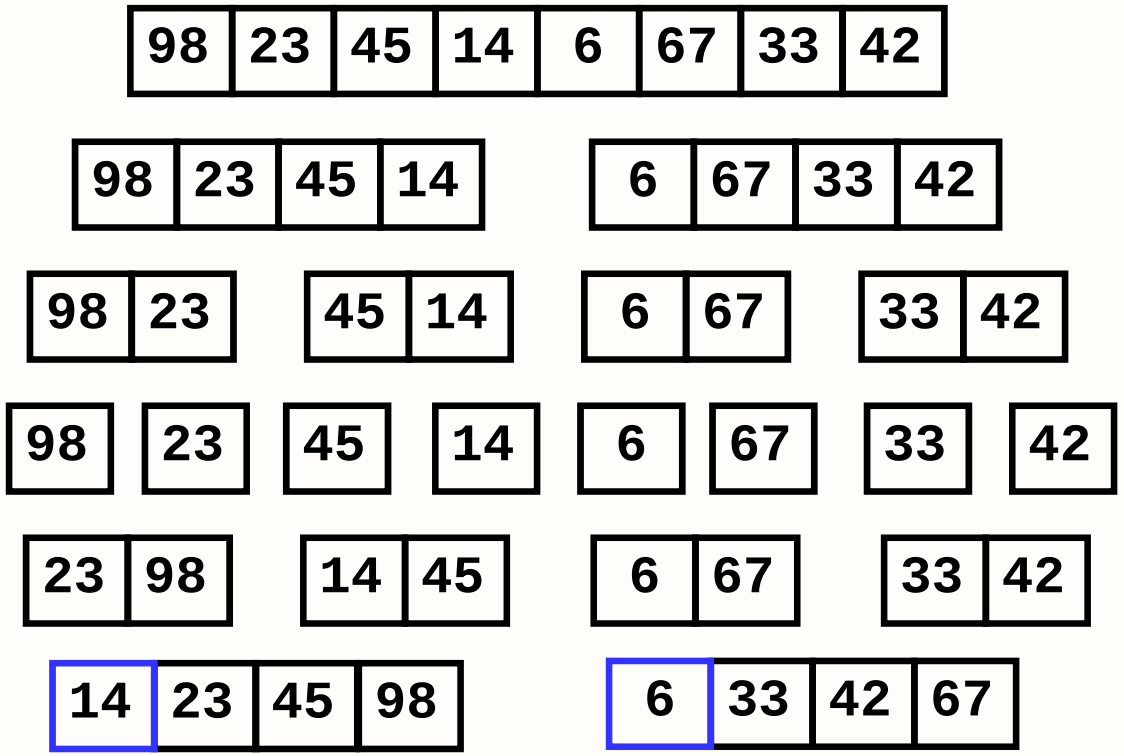

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
Merge


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

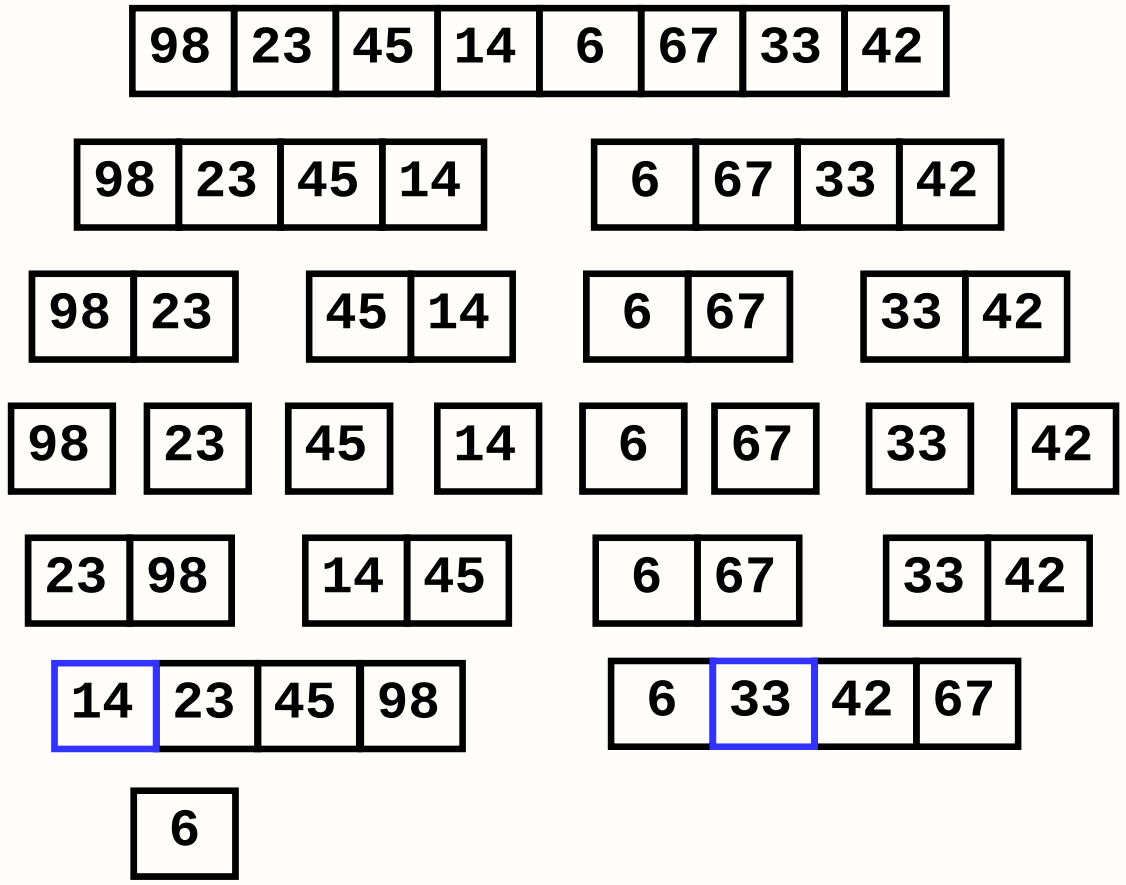

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33
Merge

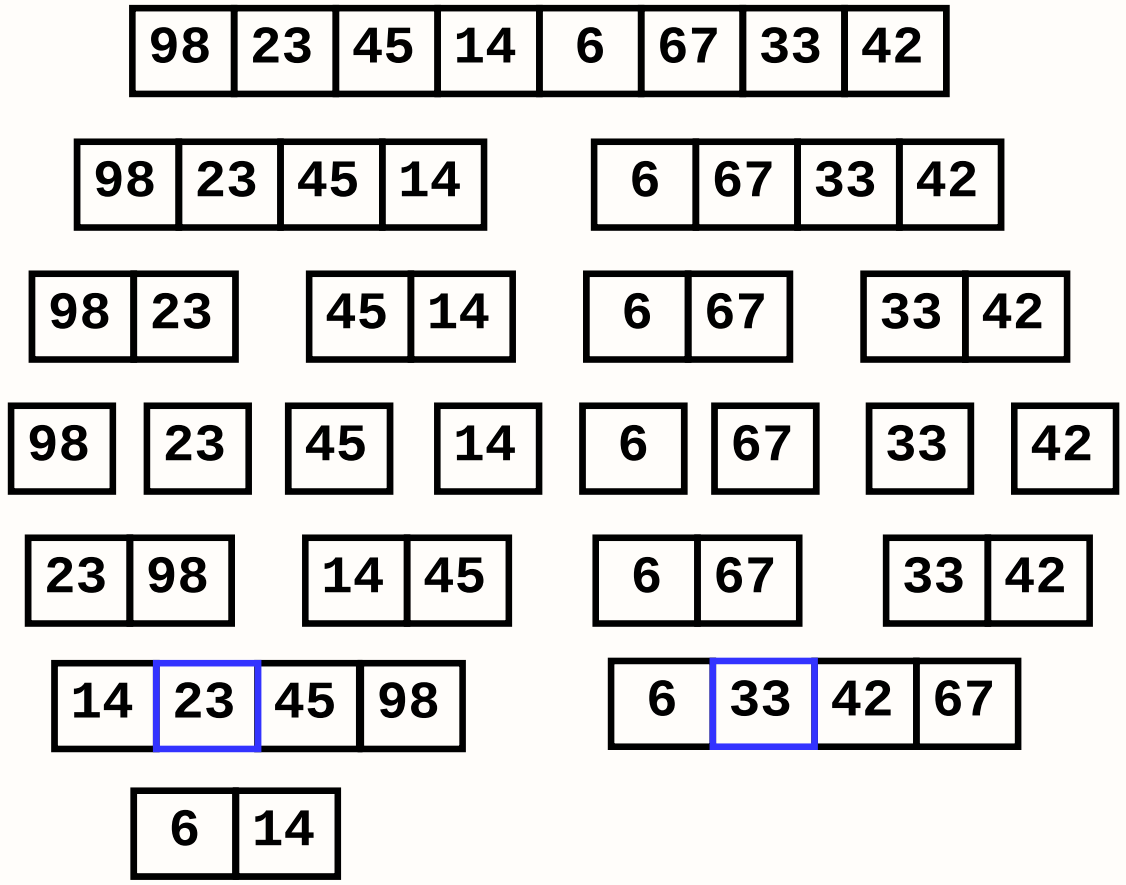

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42
Merge

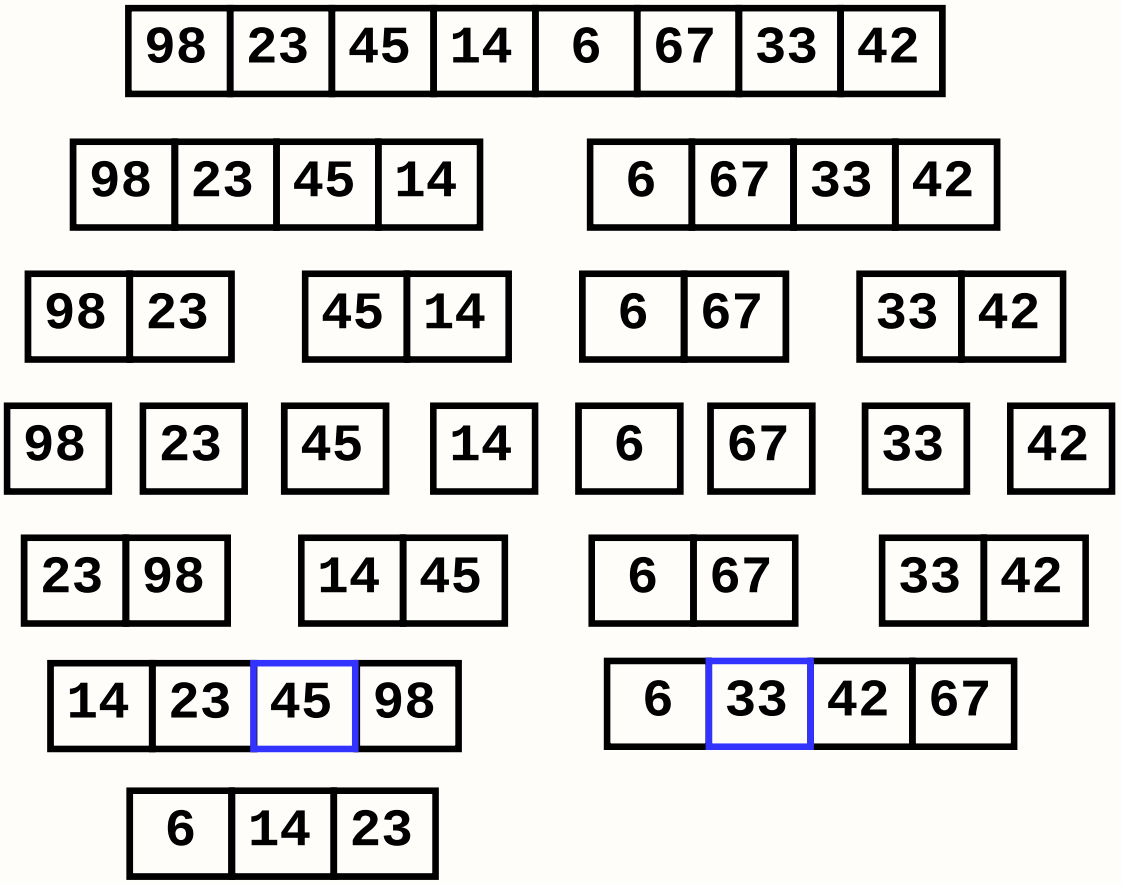

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
Merge

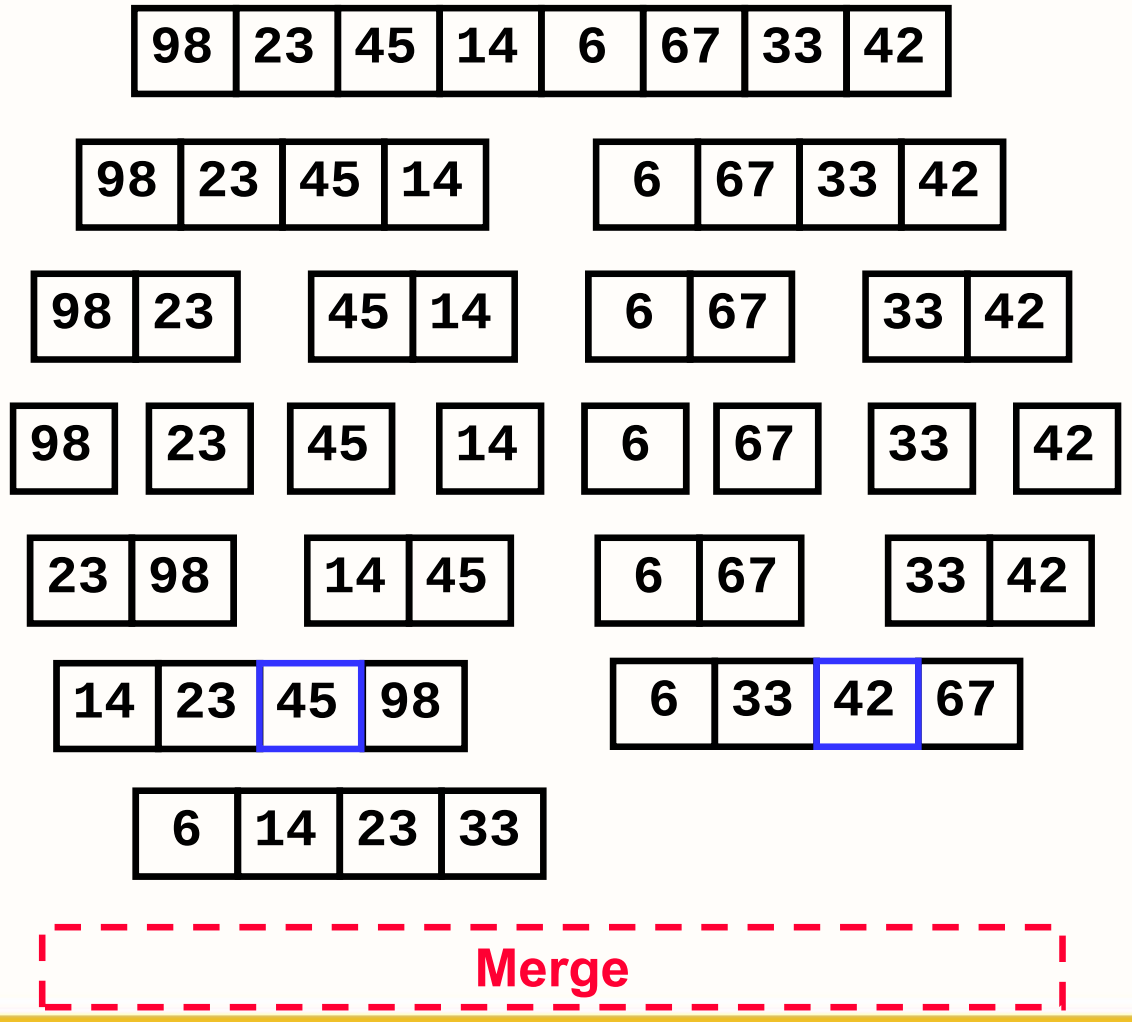

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67

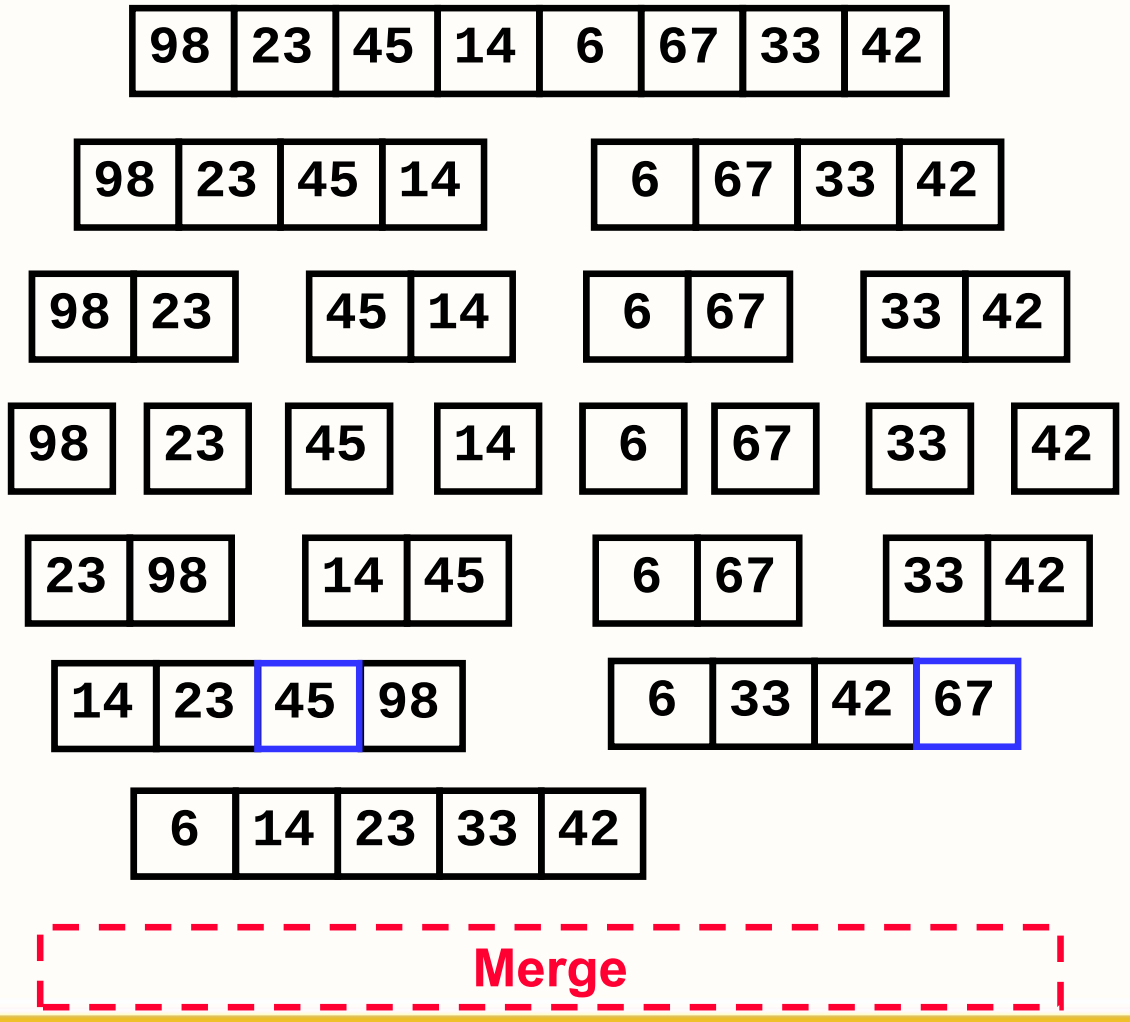
Merge

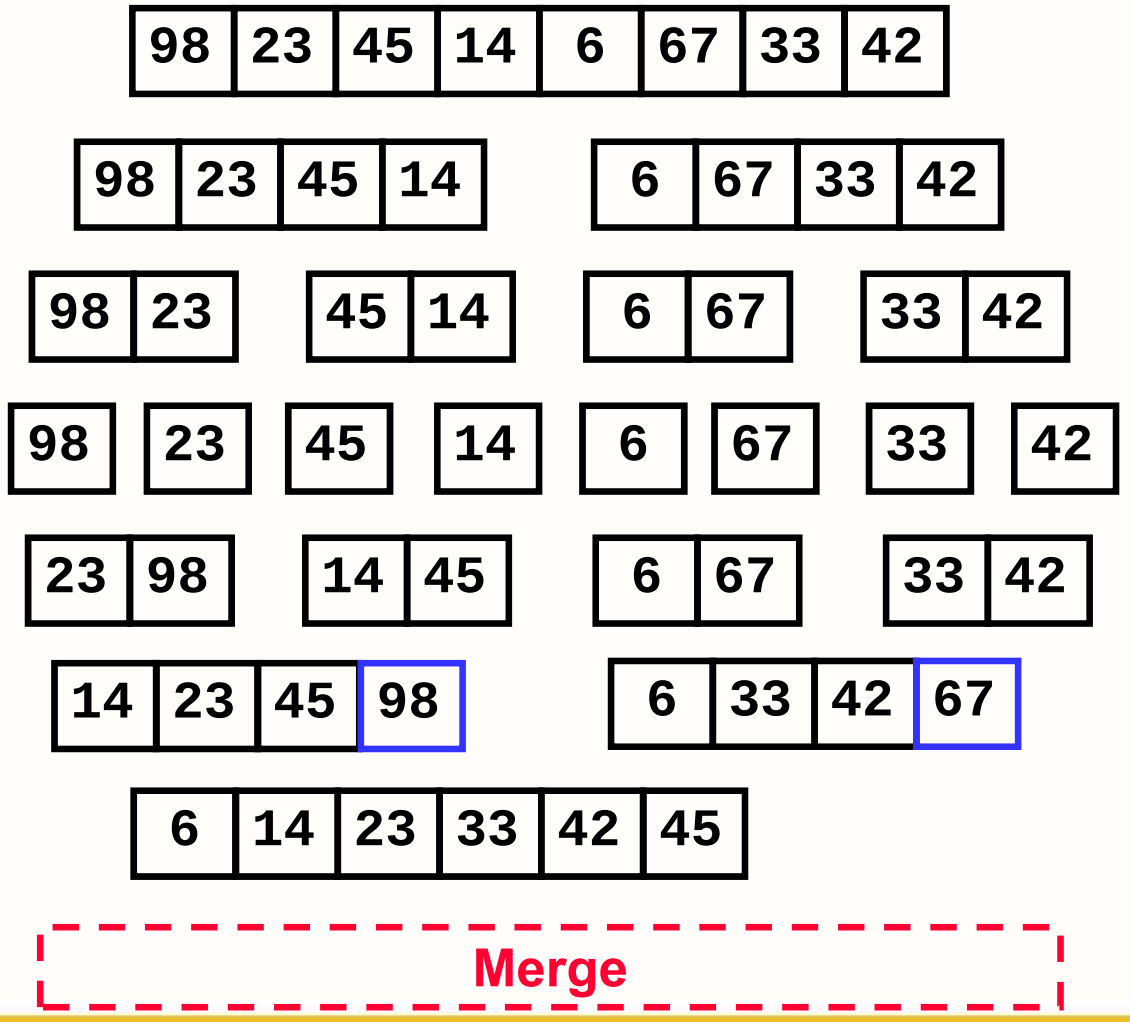

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6
Merge

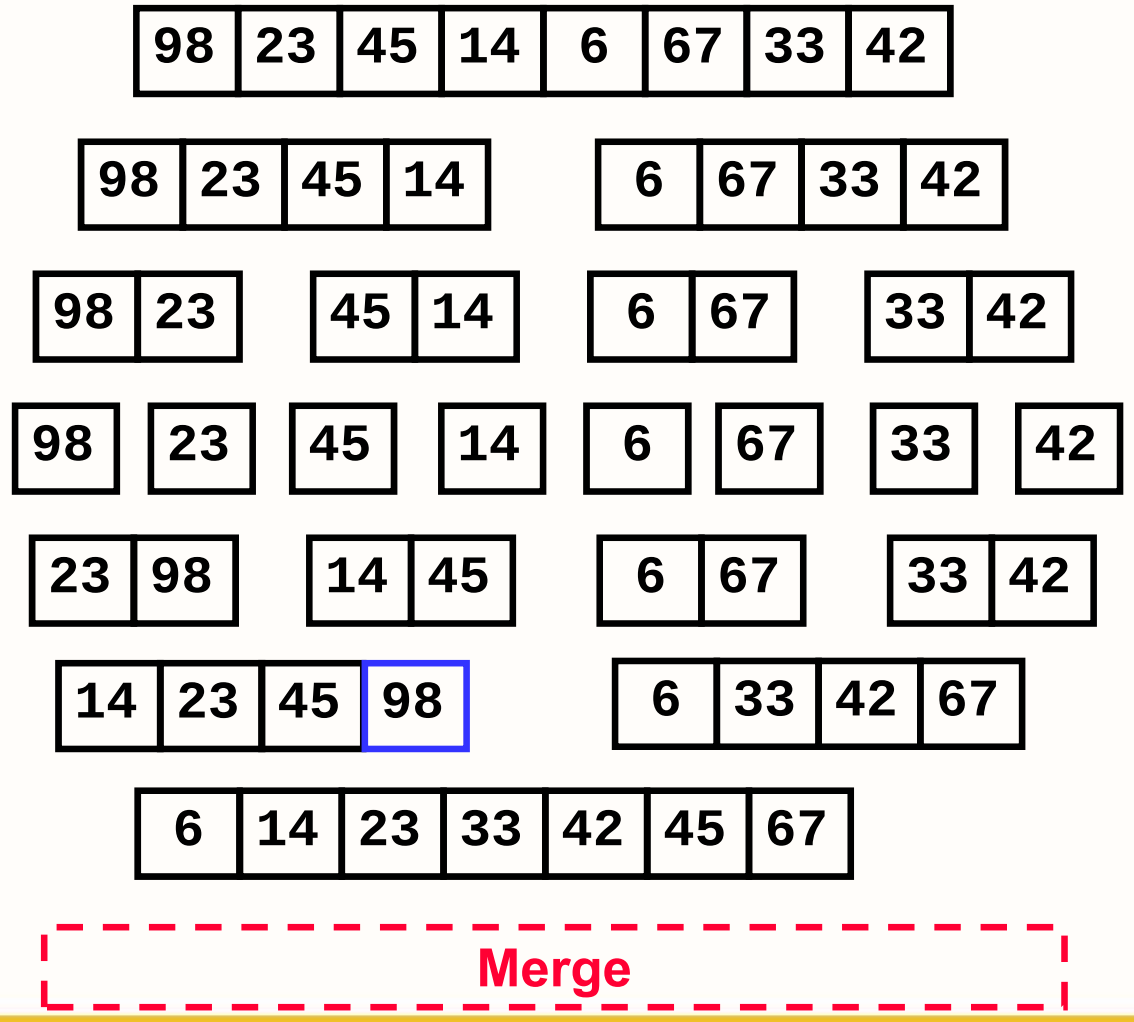

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14
Merge

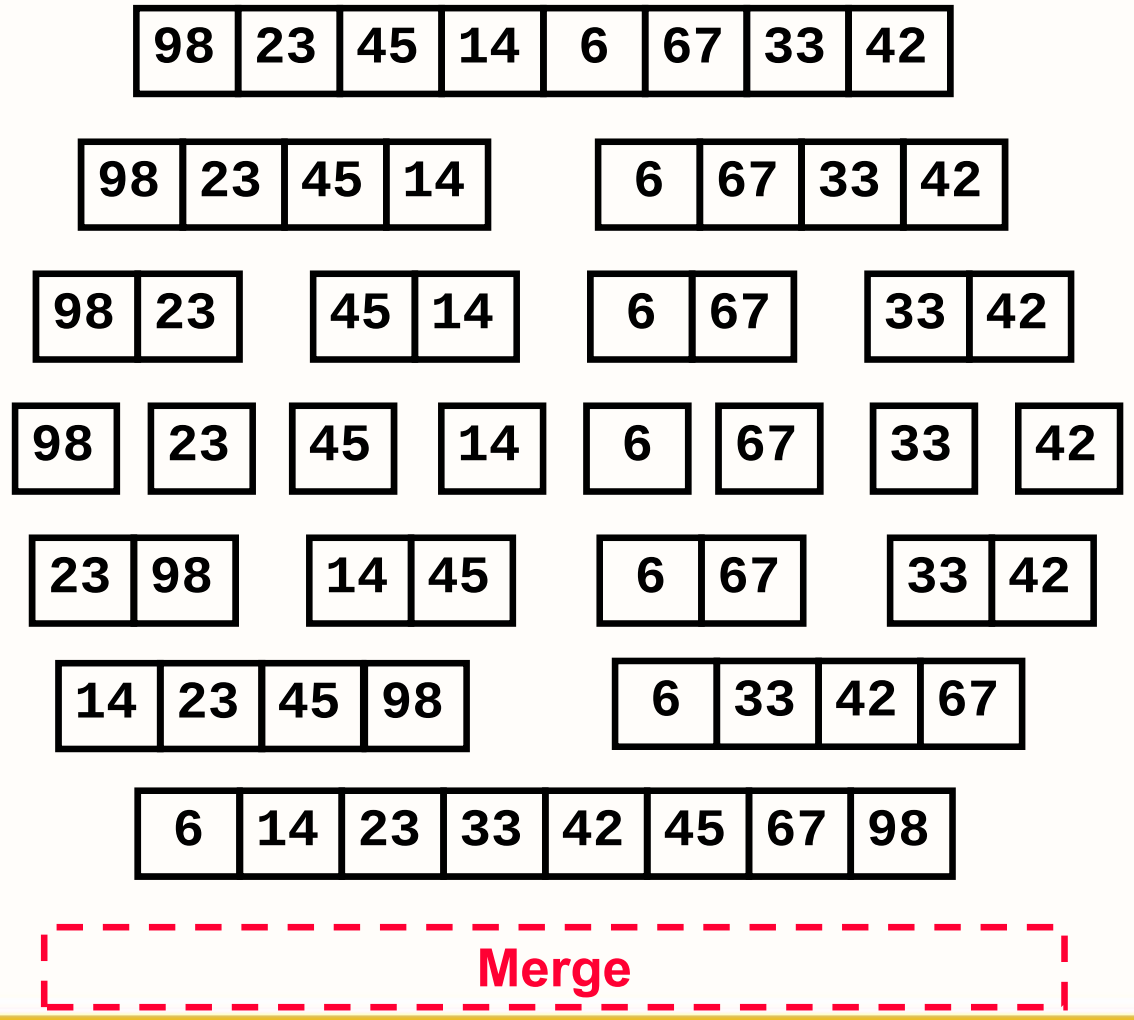

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23
Merge

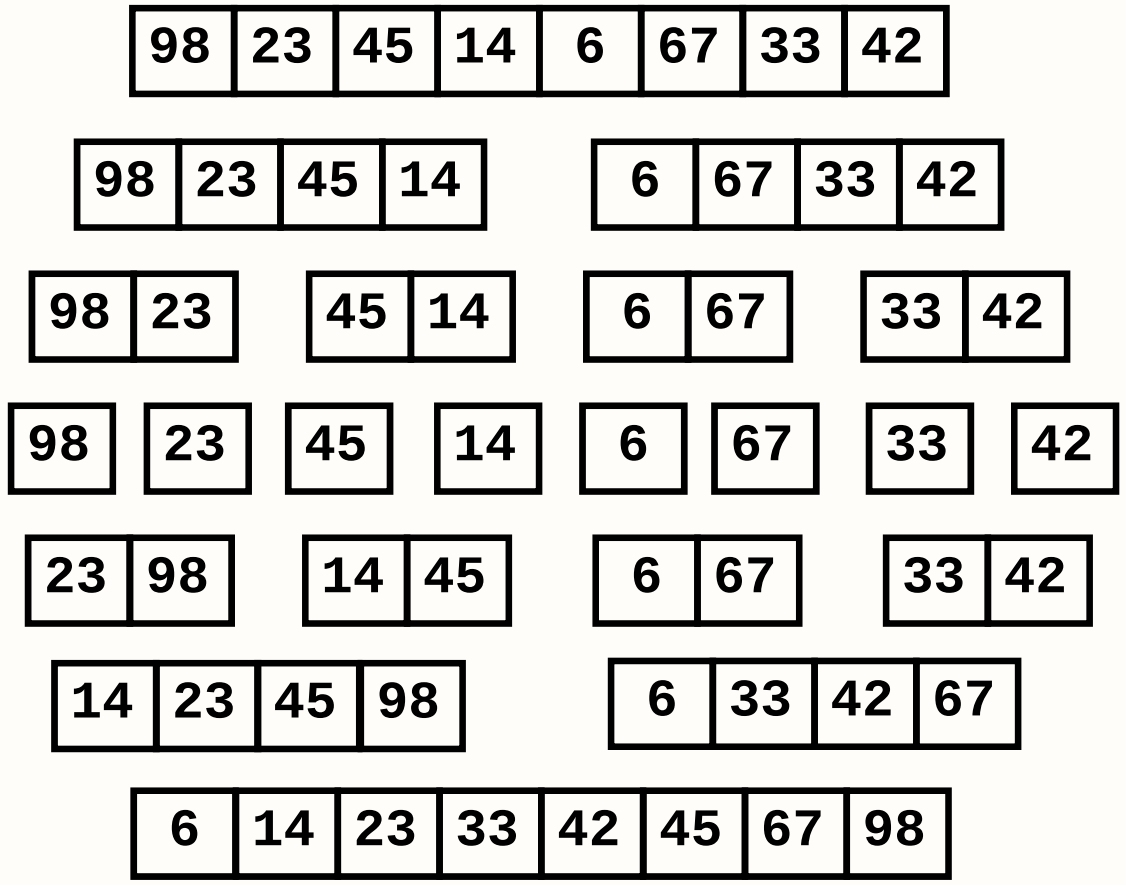

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33
Merge

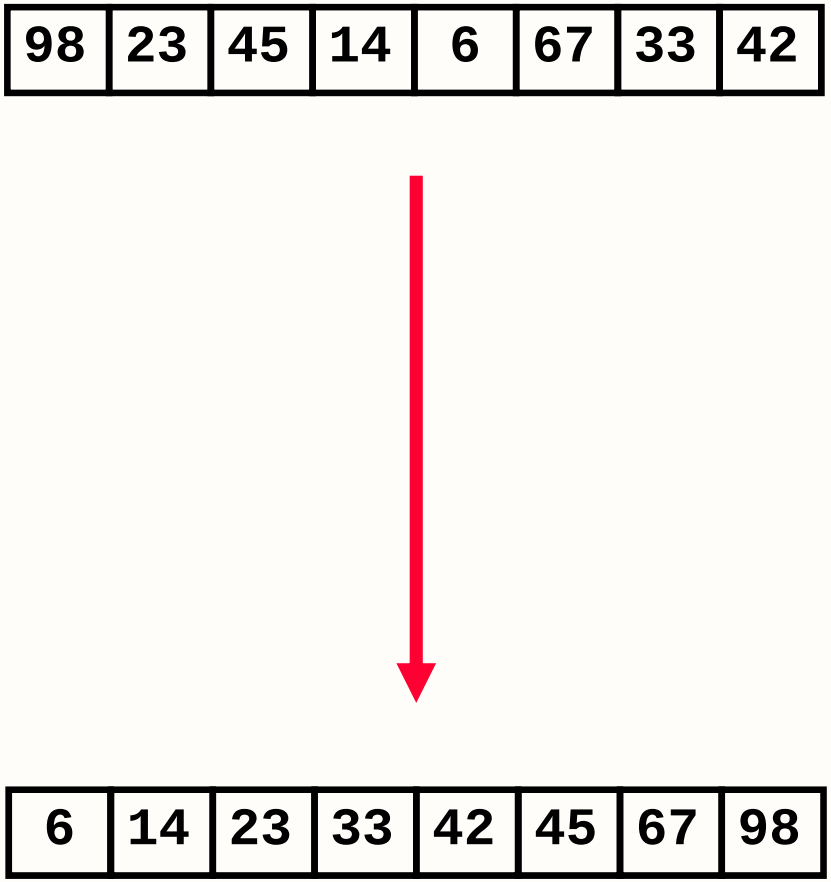

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42
Merge

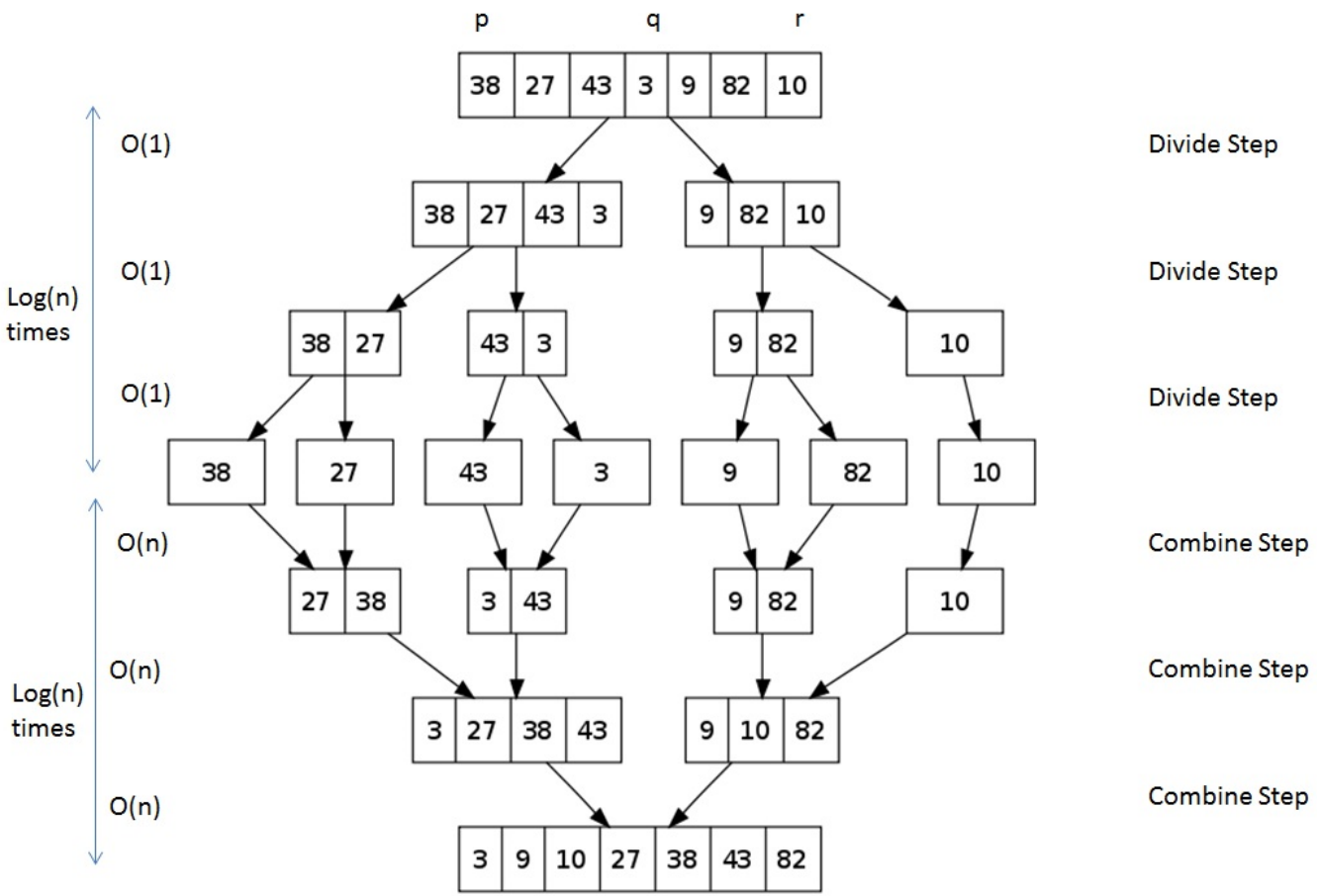

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45
Merge


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

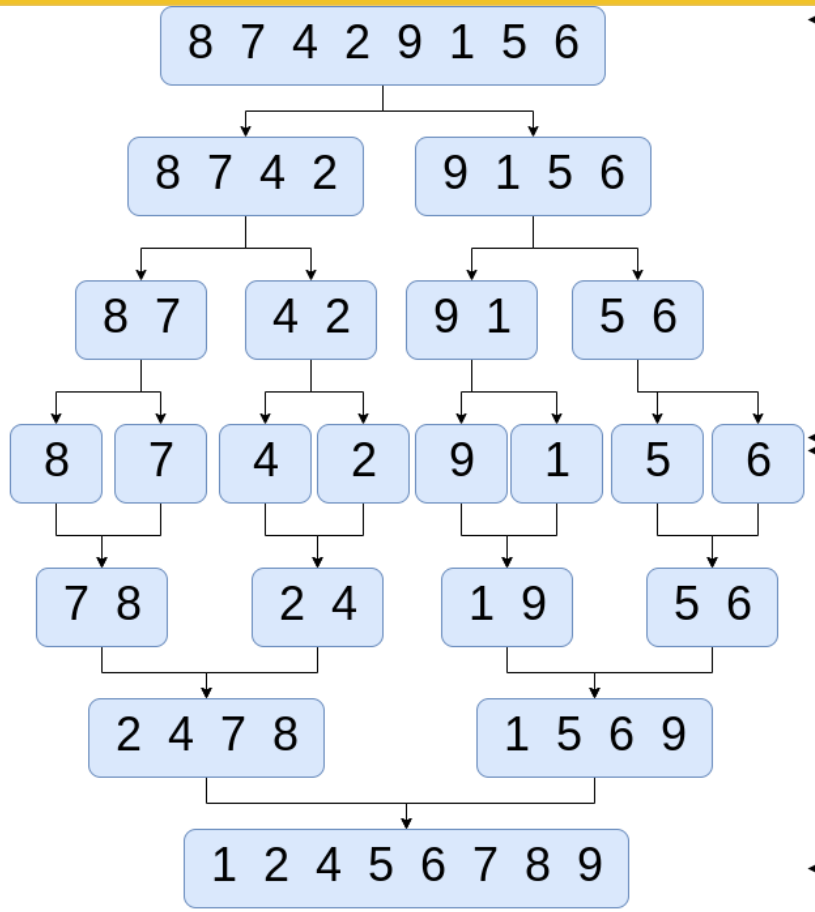

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45 67 98
Merge


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


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



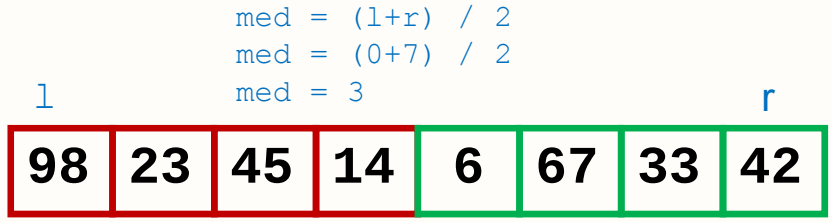
Total Runtime = Total time required in Divide + Total time required in Combine
= 1 * Log(n) + n * Log(n) = n Log(n).


8 7 4 2 9 1 5 6
8 7 4 2
9 1 5 6
8 7
4 2
9 1
5 6
8
7
4
2
9
1
5
6
7 8
2 4
1 9
5 6
2 4 7 8
1 5 6 9
1 2 4 5 6 7 8 9
DIVIDE
CONQUER

Merge Sort

Algoritma Merge Sort

```
1. void MergeSortRekursif(l, r)
2. jika (l < r) maka kerjakan baris 3-6
3.          med = (l+r) / 2 ;
4.          MergeSortRekursif(l,med);
5.          MergeSortRekursif(med+1,r);
6.          Merge(l,med,r);
```



Fungsi Merge

```
void Merge(left, median, right)
1. kiri1 ← left
2. kanan1 ← median
3. kiri2 ← median+1
4. kanan2 ← right
5. i ← left;
   //selama jumlah data kelompok sebelah kiri dan kanan (masing2 kelompok array)
   // >= 1 buah
6. selama (kiri1<=kanan1) dan (kiri2<=kanan2) kerjakan 7-13
7.    jika (Data[kiri1] <= Data[kiri2]) kerjakan 8-9
8.          hasil[i] = Data[kiri1];
9.          kiri1++
10.   jika tidak kerjakan baris 11-12
11.         hasil[i] = Data[kiri2];
12.         kiri2++
13.   i++
```

Menempatkan data yang lebih kecil ke array hasil

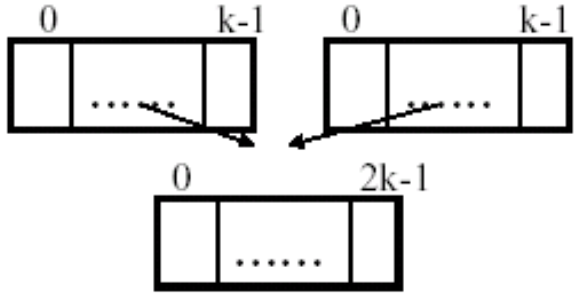
```
//selama masih ada data kelompok sebelah kiri yang belum di cek
14. selama (kiri1<=kanan1) kerjakan baris 15-17
15.       hasil[i] = Data[kiri1]
16.       kiri1++
17.       i++

//selama masih ada data kelompok sebelah kanan yang belum di cek
18. selama (kiri2<=kanan2) kerjakan baris 19-21
19.       hasil[i] = Data[kiri2]
20.       i++
21.       kiri2++

//memindahkan hasil data terurut yang tersimpan di array hasil ke
//array asal yaitu array Data
22. j ← left
23. selama (j <=right) kerjakan baris 24-25
24.       Data[j] = hasil[j]
25.       j++
```

Mergesort – Analysis of Merge (cont.)

Merging two sorted arrays of size ***k***



- ***Best-case:***
  - All the elements in the first array are smaller (or larger) than all the elements in the second array.
  - The number of moves: 2k + 2k
  - The number of key comparisons: k
- ***Worst-case:***
  - The number of moves: 2k + 2k
  - The number of key comparisons: 2k-1

Waktu Kompleksitas Mergesort

- Kompleksitas waktu dari proses Rekursif.
- **T(n) adalah** running time untuk worst-case untuk mengurutkan **n** data/bilangan.
- Diasumsikan $n=2^k$, for some integer k.

```
void mergesort(vector<int> & A, int left, int right)
{
    if (left < right) {
        int center = (left + right)/2;
        mergesort(A,left,center);
        mergesort(A,center+1,right);
        merge(A,left,center+1,right);
    }
}
```

T(n/2)

T(n/2)

Terdapat 2 rekursif merge sort, kompleksitas waktunya @ T(n/2)

Proses Merge memerlukan waktu **O(n)** untuk menggabungkan hasil dari merge sort rekursif

O(n/2+n/2=n)

$$T(n) = \begin{cases} 2T(n/2) + O(n) & n>1 \\ O(1) & n=1 \end{cases}$$

Waktu Kompleksitas Mergesort

$$
\begin{aligned}
& T(n) \\
&= 2T(n/2) + O(n) \\
&= 2(2T(n/4) + O(n/2)) + O(n) \\
&= 4T(n/4) + 2 \cdot O(n/2)) + O(n) \\
&= 4T(n/4) + O(2n/2)) + O(n) \\
&= 4T(n/4) + O(n) + O(n) \\
&= 4(2T(n/8) + O(n/4)) + O(n) + O(n) \\
&= 8T(n/8) + 4 \cdot O(n/4) + O(n) + O(n) \\
&= 8T(n/8) + O(4n/4) + O(n) + O(n) \\
&= 8T(n/8) + O(n) + O(n) + O(n)
\end{aligned}
$$

Recursive step

Recursive step

Collect terms

Recursive step

Collect terms

$$T(n) = 2^k T(n/2^k) + k \cdot O(n)$$

Setelah level ke - k

Kompleksitas Waktu Mergesort

$$T(n) = 2^k T(n/2^k) + k \cdot O(n)$$

Karena $n=2^k$, setelah level ke- k ($=\log_2 n$) pemanggilan rekursif, bertemu dengan (n=1)

Put $k = \log_2 n$, $(n=2^k)$

$$\begin{aligned} T(n) &= 2^k T(n/2^k) + kO(n) \\ &= nT(n/n) + kO(n) \\ &= nT(1) + \log_2 nO(n) \\ &= nO(1) + O(n \log_2 n) \\ &= O(n) + O(n \log_2 n) \\ &= O(n \log_2 n) \\ &= O(n \log n) \end{aligned}$$

$$T(n) = O(n \log n)$$

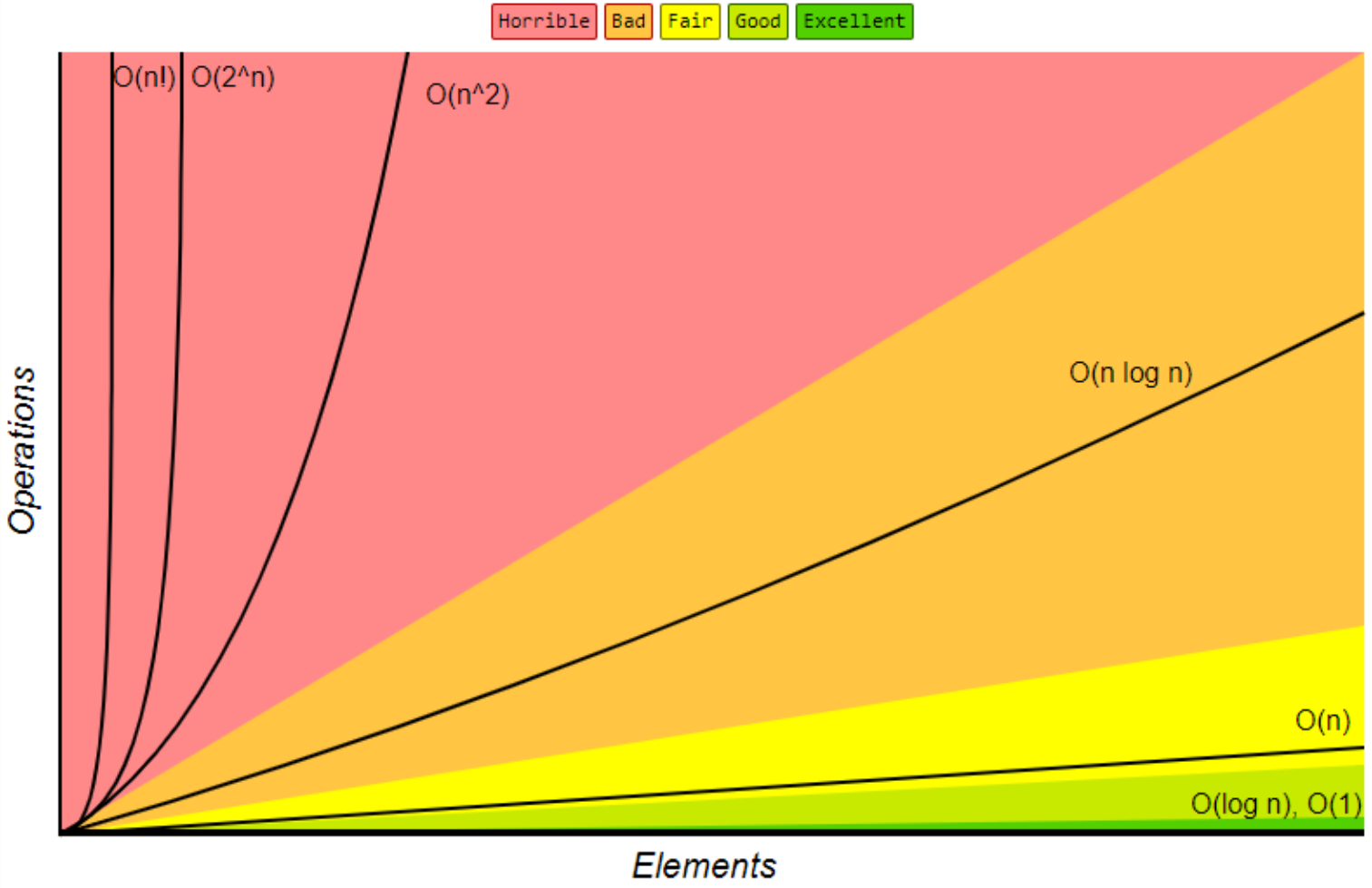
Perbandingan insertion sort dan merge sort (dalam detik)

n	Insertion sort	Merge sort	Ratio
100	0.01	0.01	1
1000	0.18	0.01	18
2000	0.76	0.04	19
3000	1.67	0.05	33
4000	2.90	0.07	41
5000	4.66	0.09	52
6000	6.75	0.10	67
7000	9.39	0.14	67
8000	11.93	0.14	85

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Big-O Complexity Chart



Kesimpulan

- Merupakan algoritma divide-and-conquer (membagi dan menyelesaikan)
- Membagi array menjadi dua bagian sampai subarray hanya berisi satu elemen
- Mengabungkan solusi sub-problem:
  - Membandingkan elemen pertama subarray
  - Memindahkan elemen terkecil dan meletakkannya ke array hasil
  - Lanjutkan Proses sampai semua elemen berada pada array hasil

Latihan Soal

- Urutkan data di bawah ini dengan Algoritma Merge Sort dan Quick Sort, jelaskan pula langkah-langkahnya !
- **9 1 2 5 6 4**