NPTEL MOOC,JAN-FEB 2015
Week 2, Module 5

DESIGN AND ANALYSIS OF ALGORITHMS

Merge Sort

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

$O(n^2)$ sorting algorithms

- Selection sort and insertion sort are both $O(n^2)$
- $O(n^2)$ sorting is infeasible for n over 100000

A different strategy?

- Divide array in two equal parts
- Separately sort left and right half
- Combine the two sorted halves to get the full array sorted

Combining sorted lists

- Given two sorted lists A and B, combine into a sorted list C
  - Compare first element of A and B
  - Move it into C
  - Repeat until all elements in A and B are over
- Merging A and B

Merging two sorted lists

32 74 89

21 55 64

Merging two sorted lists

32 74 89

~~21~~ 55 64

21

Merging two sorted lists

~~32~~ 74 89

~~21~~ 55 64

21 32

Merging two sorted lists

~~32~~ 74 89

~~21~~ ~~55~~ 64

21 32 55

Merging two sorted lists

~~32~~ 74 89

~~21~~ ~~55~~ ~~64~~

21 32 55 64

Merging two sorted lists

~~32~~ ~~74~~ 89

~~21~~ ~~55~~ ~~64~~

21 32 55 64 74

Merging two sorted lists

~~32~~ ~~74~~ ~~89~~

~~21~~ ~~55~~ ~~64~~

21 32 55 64 74 89

Merge Sort

- Sort A[0] to A[n/2-1]
- Sort A[n/2] to A[n-1]
- Merge sorted halves into B[0..n-1]
- How do we sort the halves?
  - Recursively, using the same strategy!

Merge Sort

43	32	22	78	63	57	91	13

Merge Sort

43	32	22	78	63	57	91	13

43	32	22	78

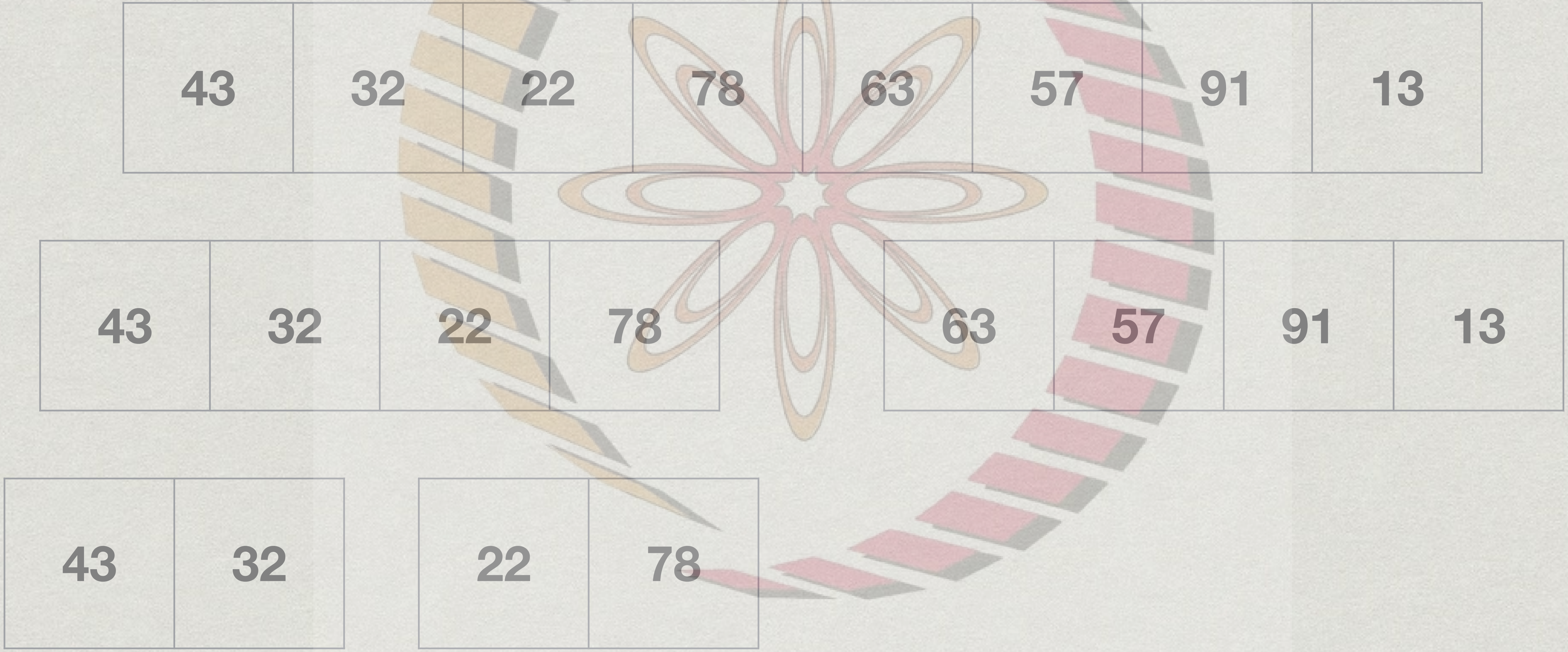
Merge Sort

43	32	22	78	63	57	91	13

43	32	22	78

63	57	91	13

Merge Sort



43 32 22 78 63 57 91 13

43 32 22 78

63 57 91 13

43 32

22 78

Merge Sort

43	32	22	78	63	57	91	13

43	32	22	78

63	57	91	13

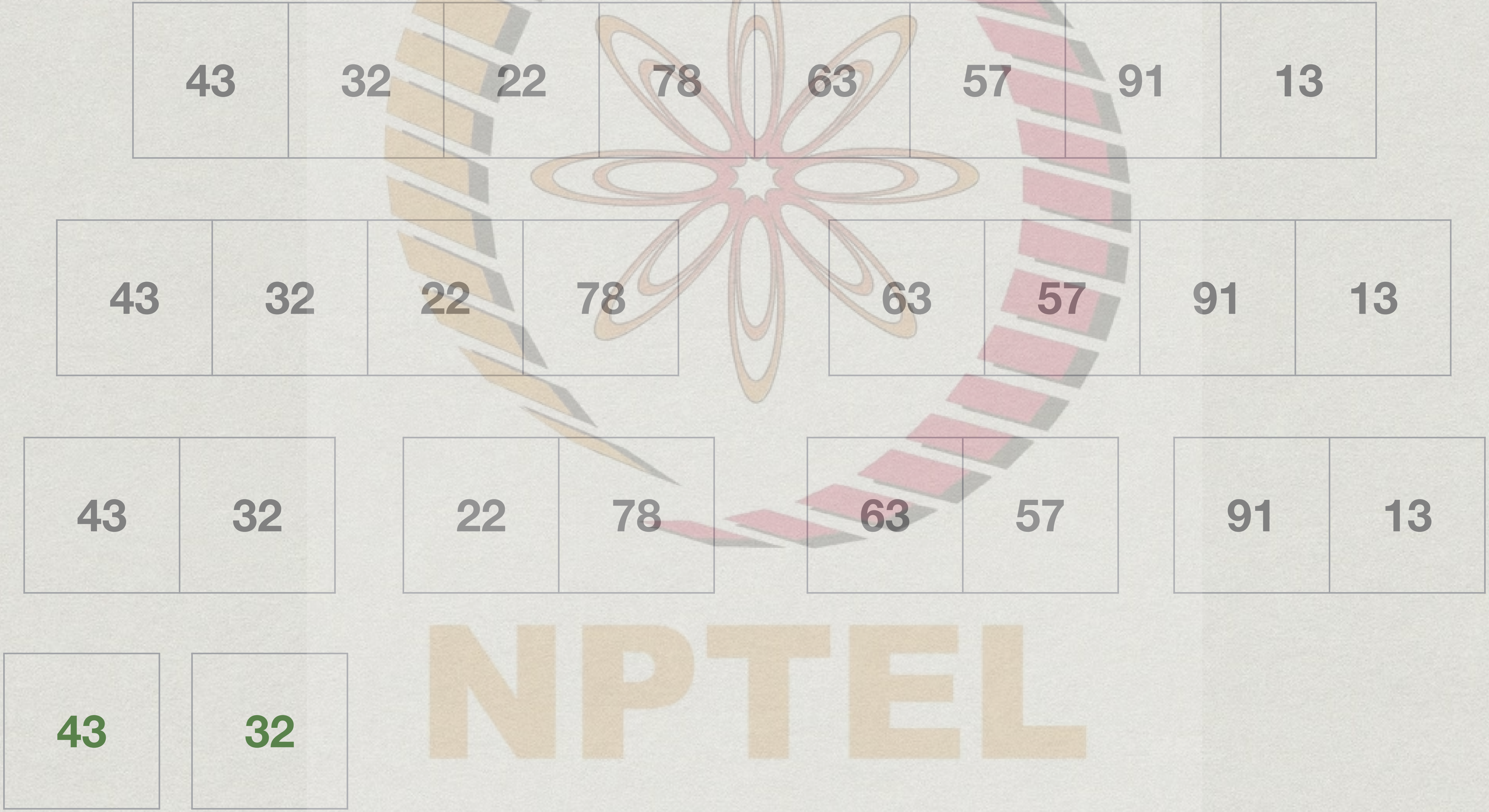
43	32

22	78

63	57

91	13

Merge Sort



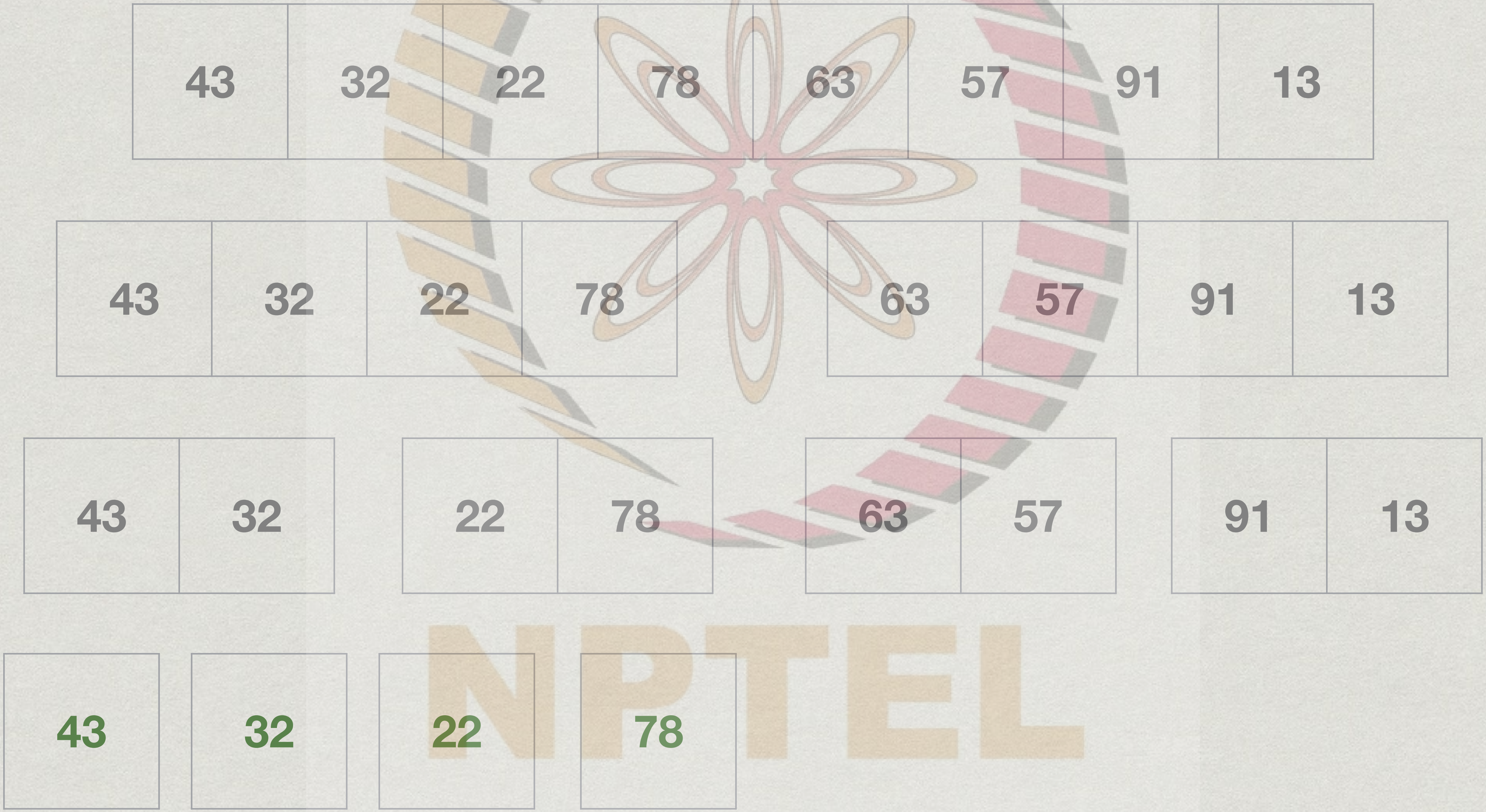
43 32 22 78 63 57 91 13

43 32 22 78   63 57 91 13

43 32   22 78   63 57   91 13

43   32

Merge Sort



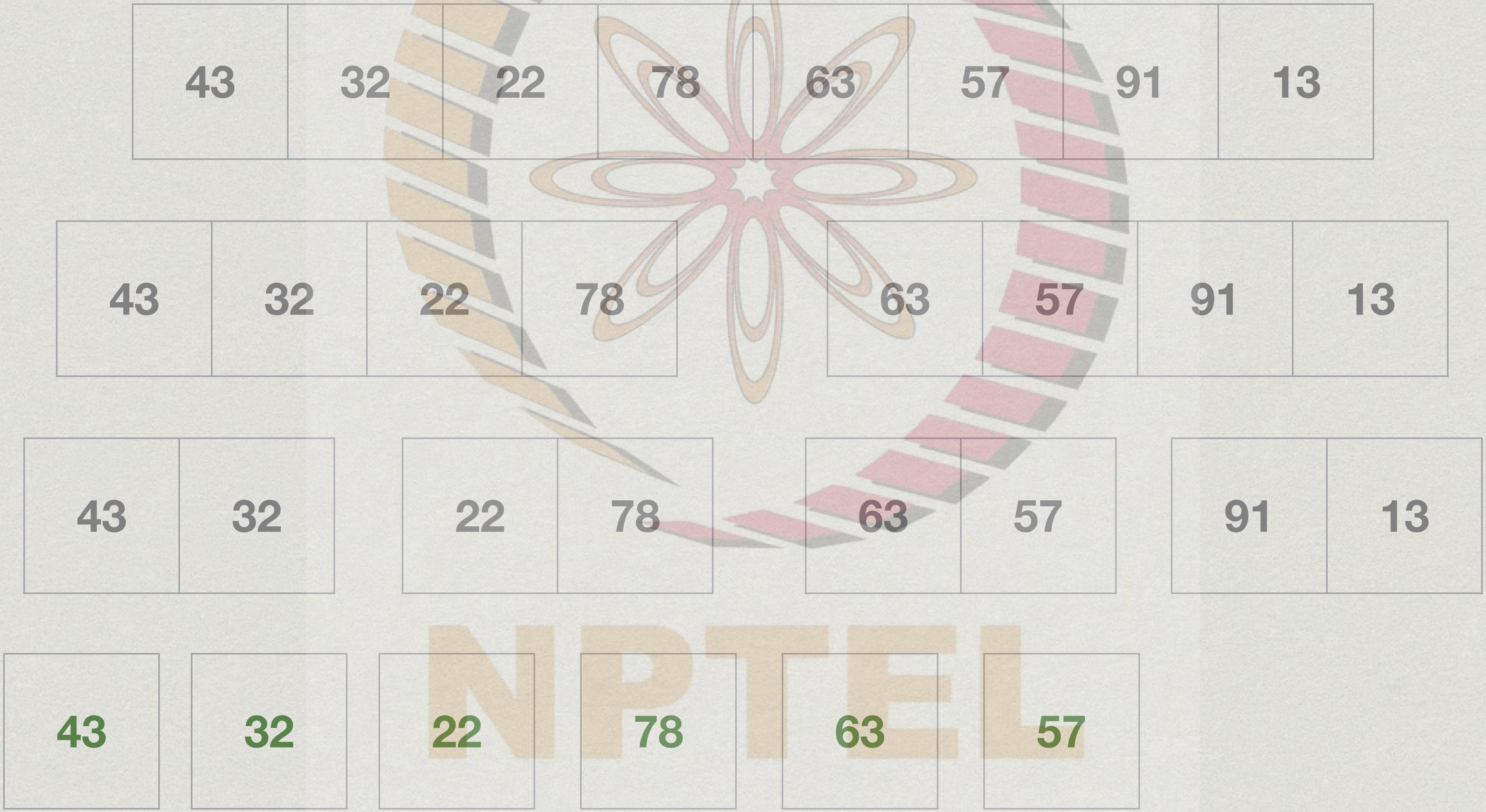
43 32 22 78 63 57 91 13

43 32 22 78   63 57 91 13

43 32   22 78   63 57   91 13

43   32   22   78

Merge Sort



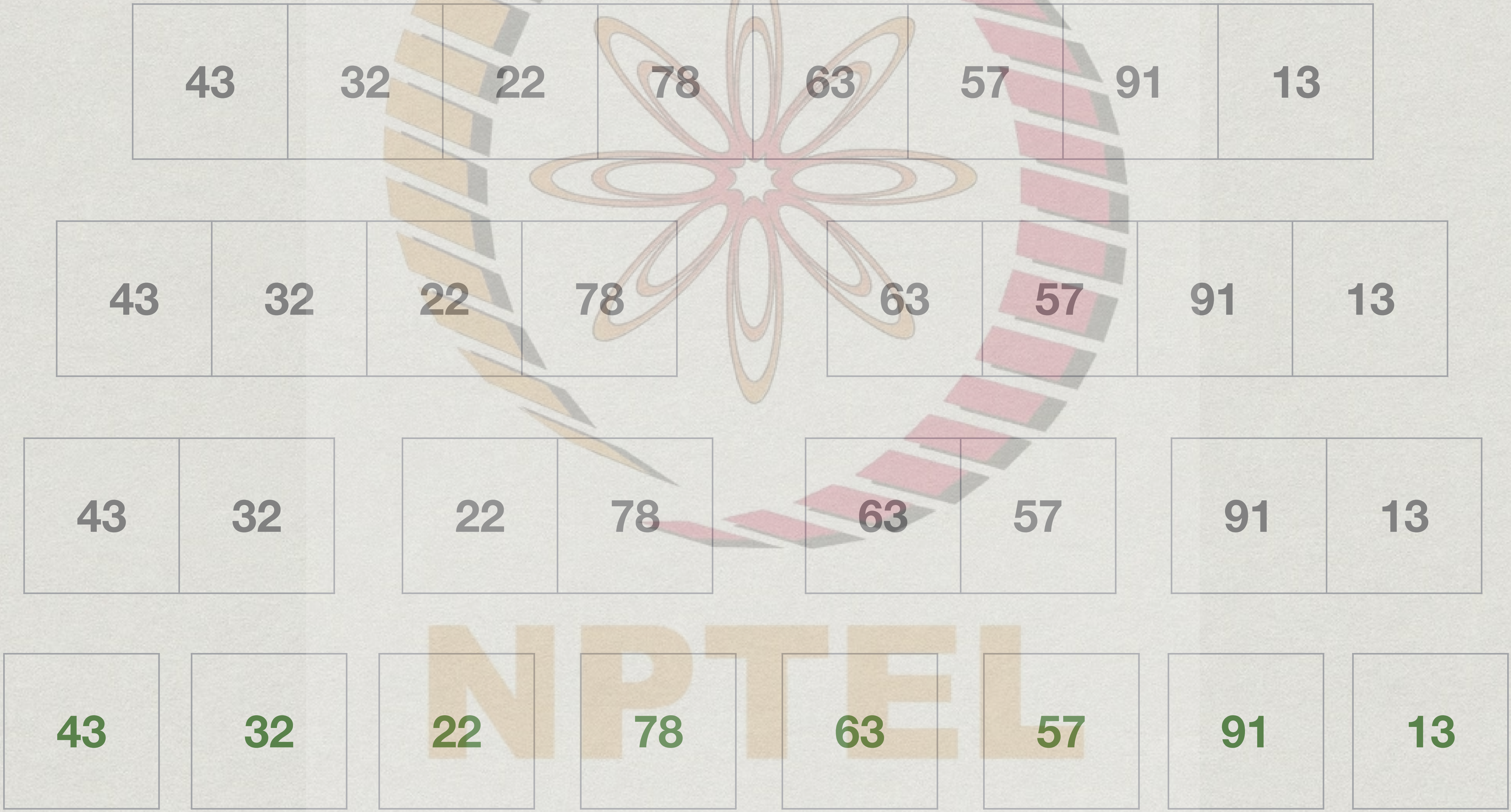
43 32 22 78 63 57 91 13

43 32 22 78 | 63 57 91 13

43 32 | 22 78 | 63 57 | 91 13

43 | 32 | 22 | 78 | 63 | 57

Merge Sort



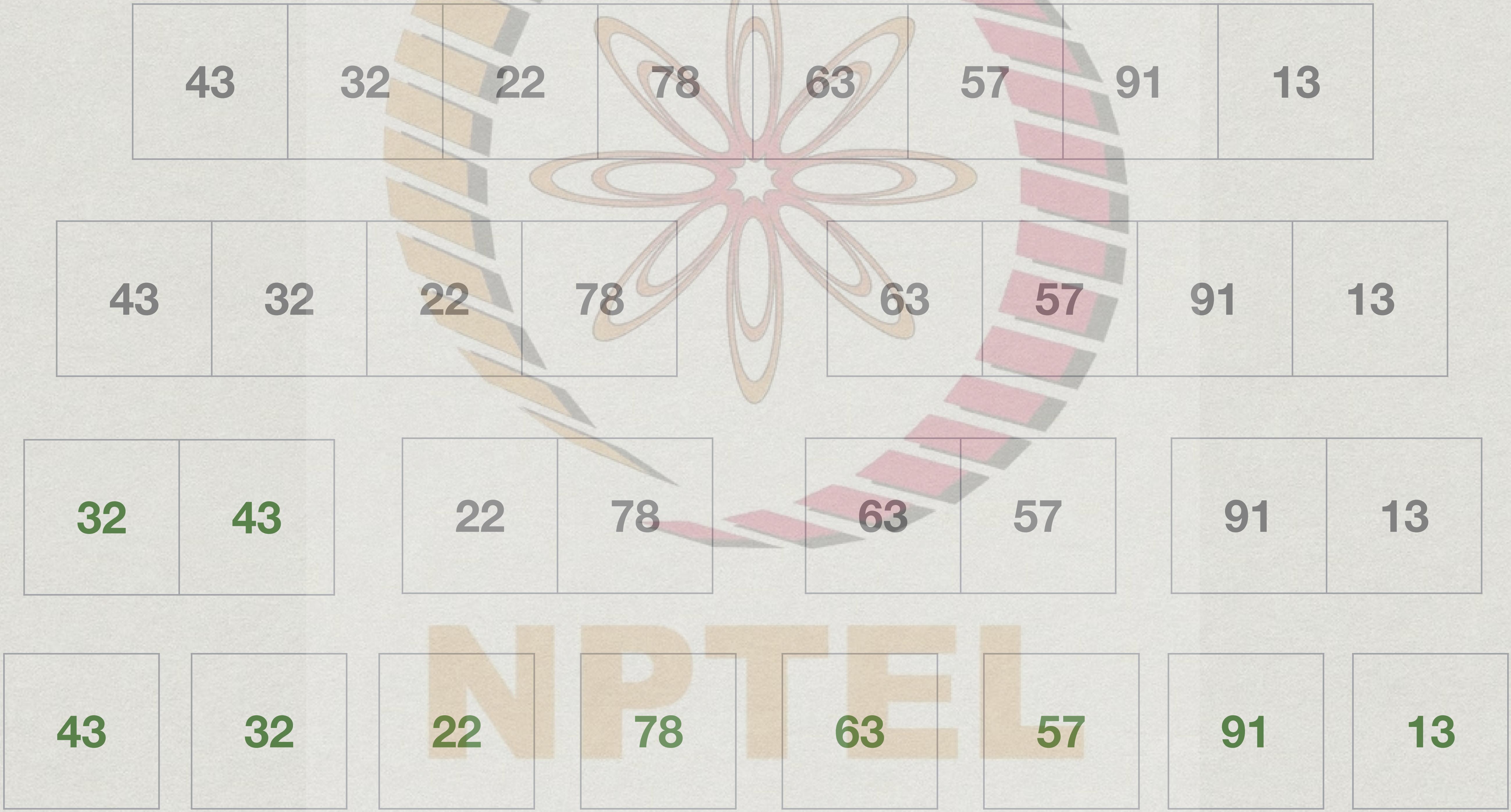
43 32 22 78 63 57 91 13

43 32 22 78   63 57 91 13

43 32   22 78   63 57   91 13

43   32   22   78   63   57   91   13

Merge Sort



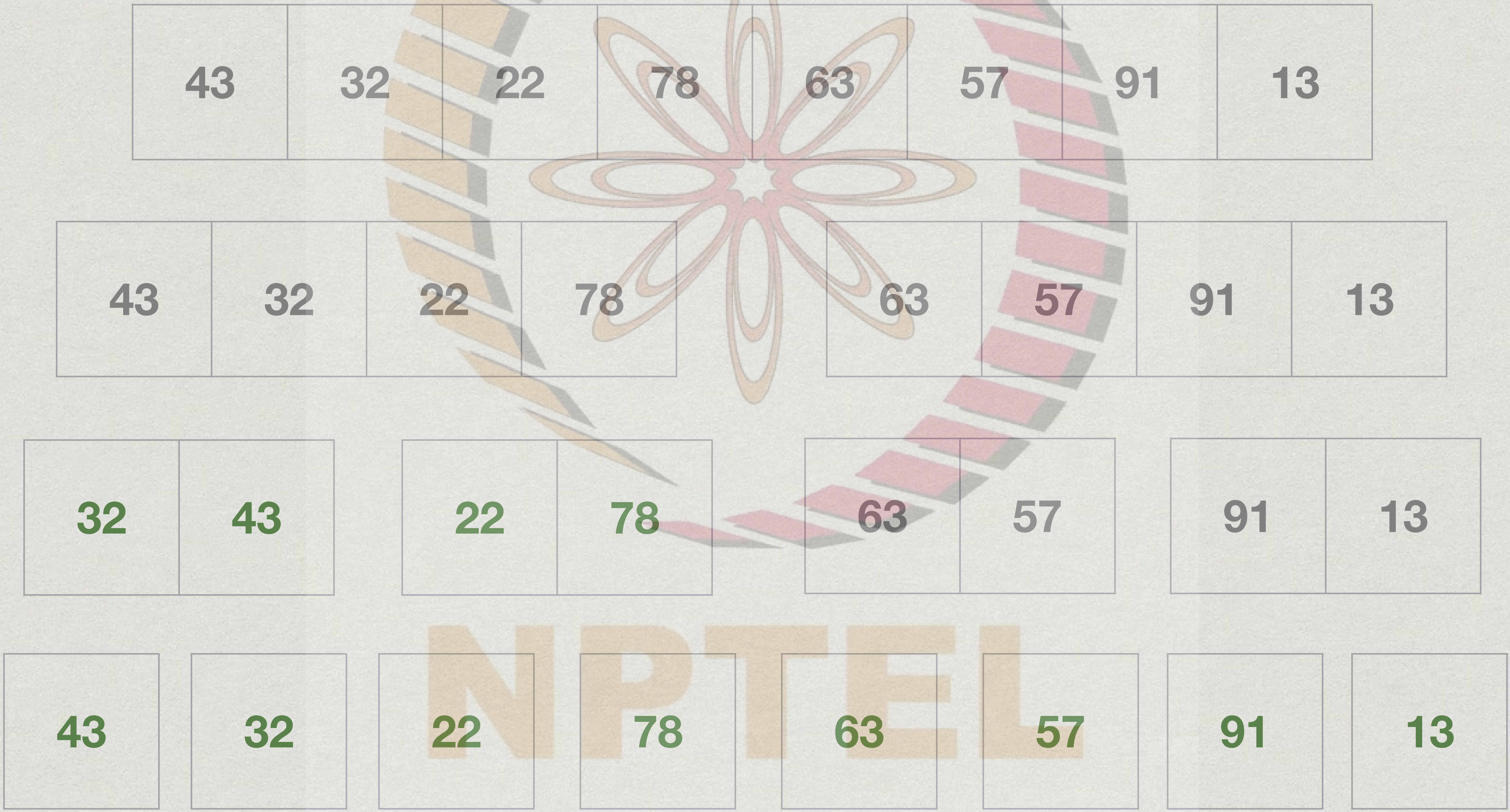
43 32 22 78 63 57 91 13

43 32 22 78   63 57 91 13

32 43   22 78   63 57   91 13

43   32   22   78   63   57   91   13

Merge Sort



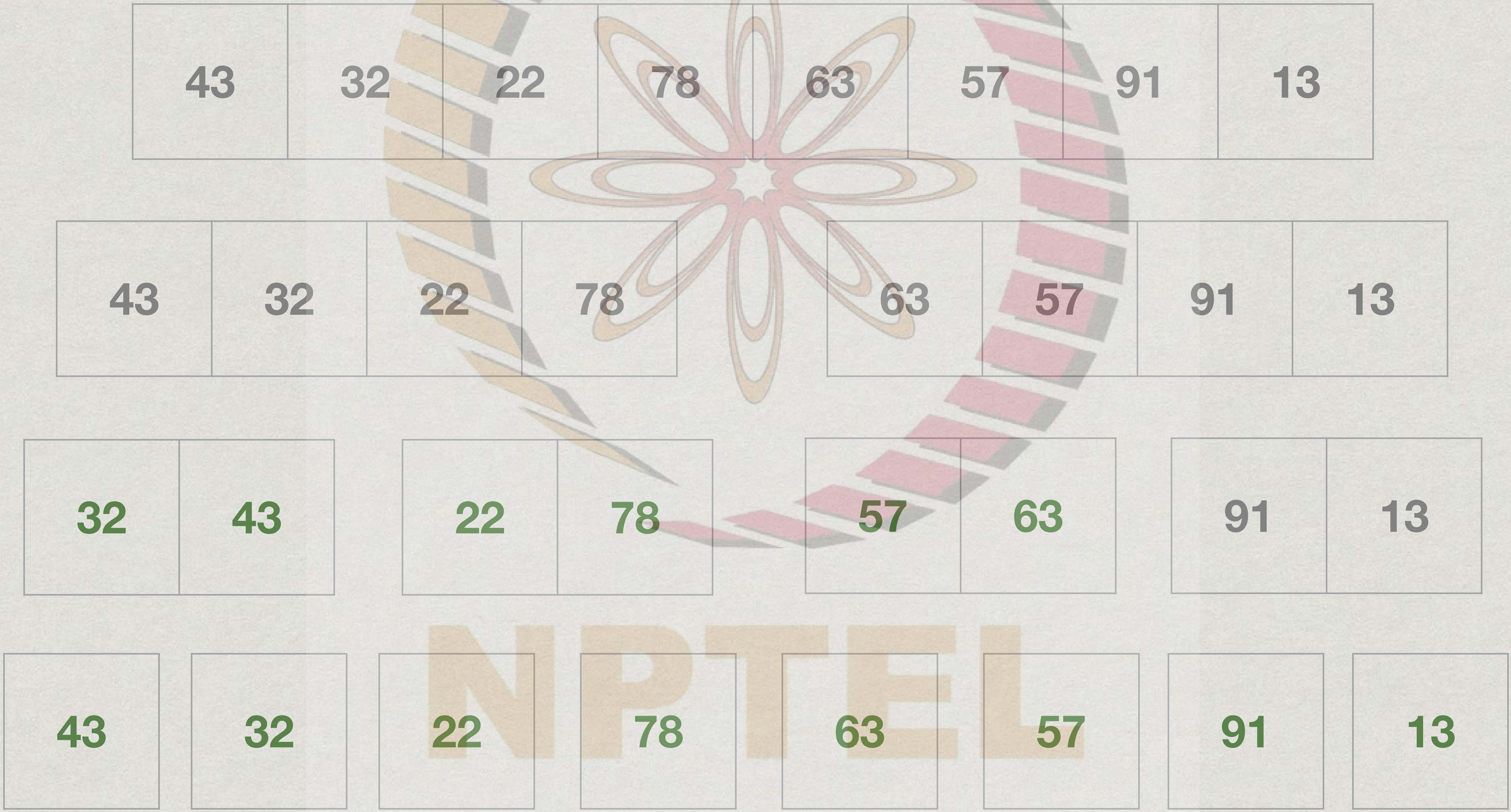
43 32 22 78 63 57 91 13

43 32 22 78 | 63 57 91 13

32 43 | 22 78 | 63 57 | 91 13

43 | 32 | 22 | 78 | 63 | 57 | 91 | 13

Merge Sort



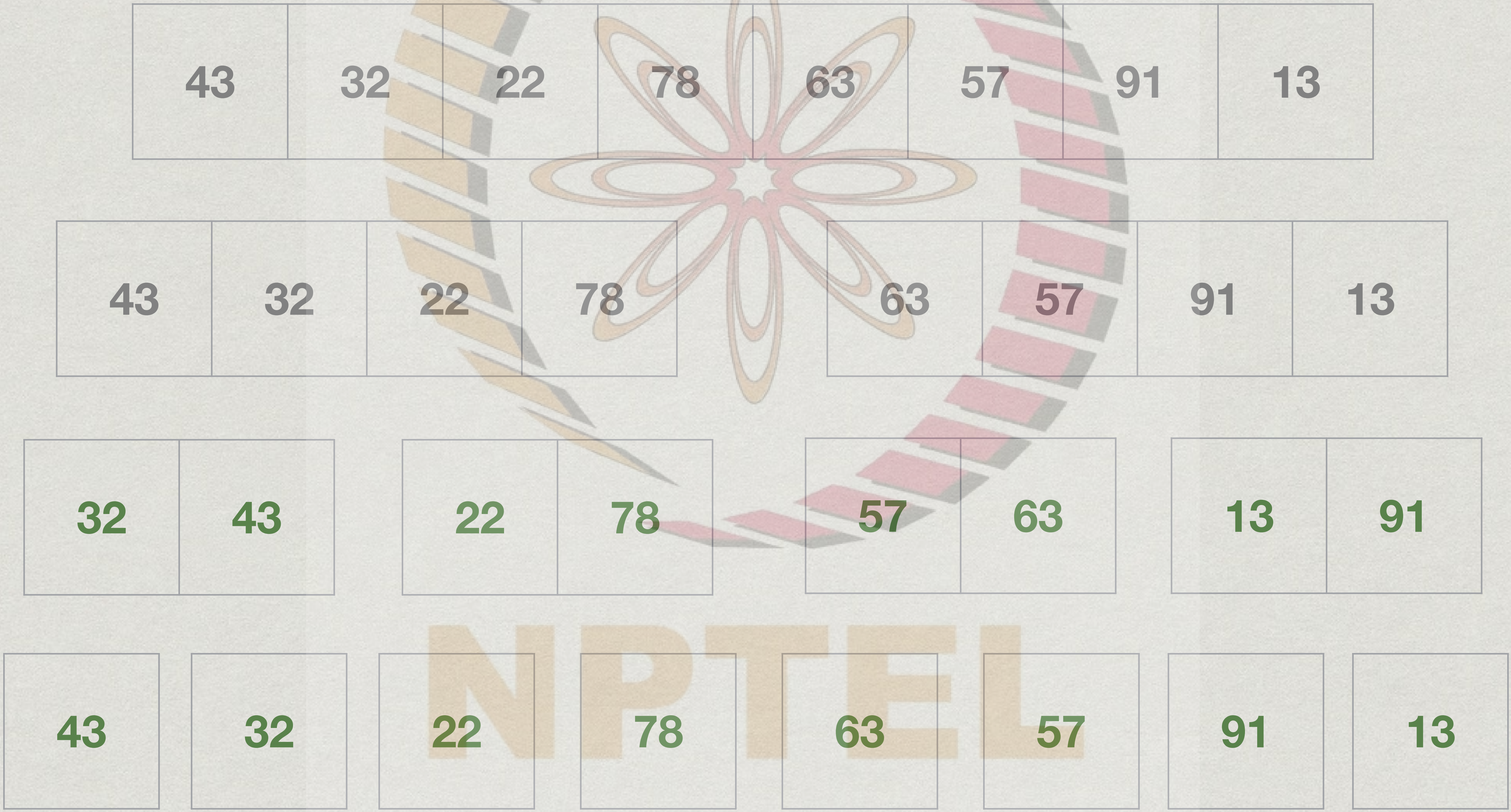
43 32 22 78 63 57 91 13

43 32 22 78 | 63 57 91 13

32 43 | 22 78 | 57 63 | 91 13

43 | 32 | 22 | 78 | 63 | 57 | 91 | 13

Merge Sort



43	32	22	78	63	57	91	13

43	32	22	78

63	57	91	13

32	43

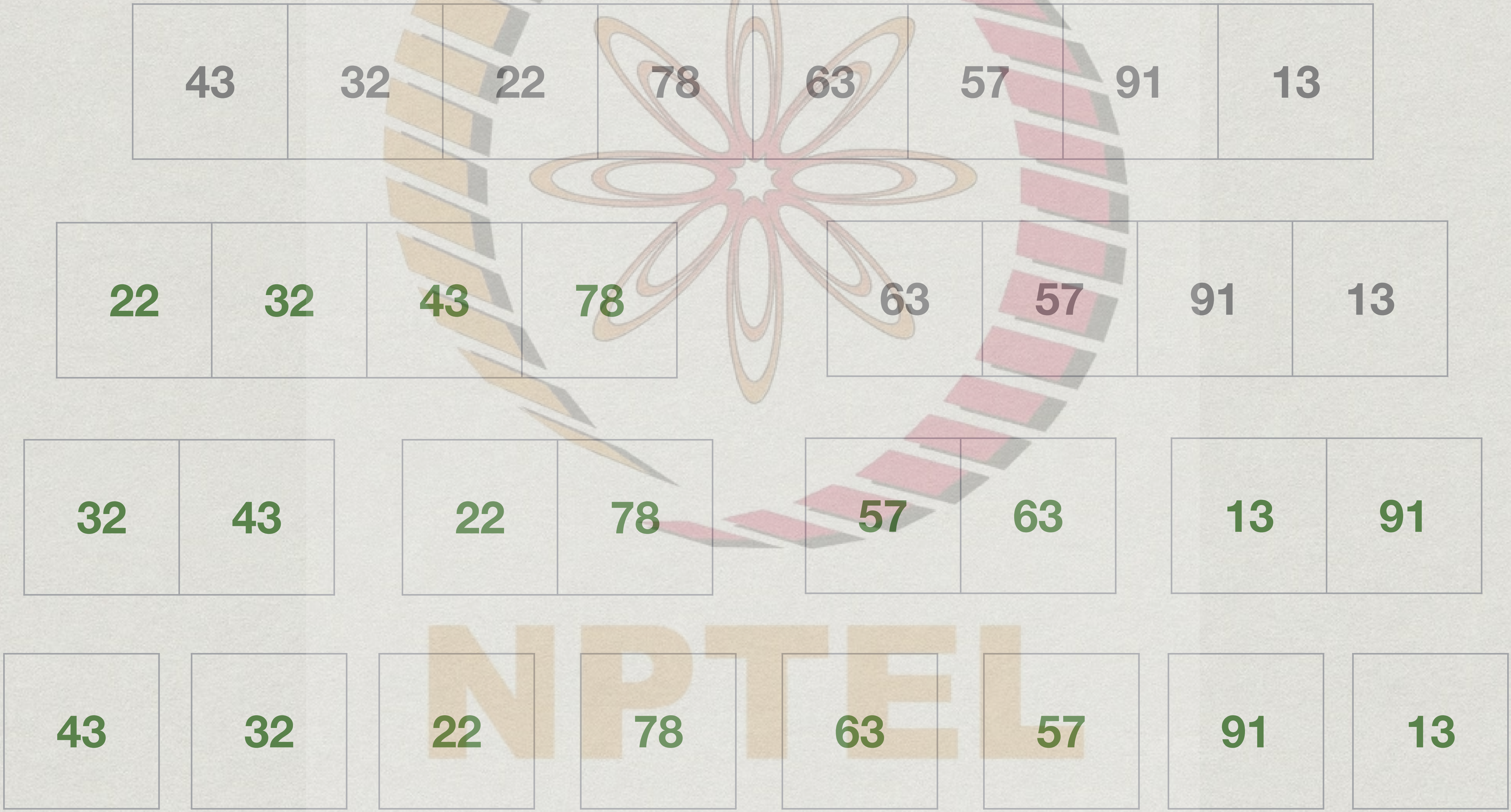
22	78

57	63

13	91

43 32 22 78 63 57 91 13

Merge Sort



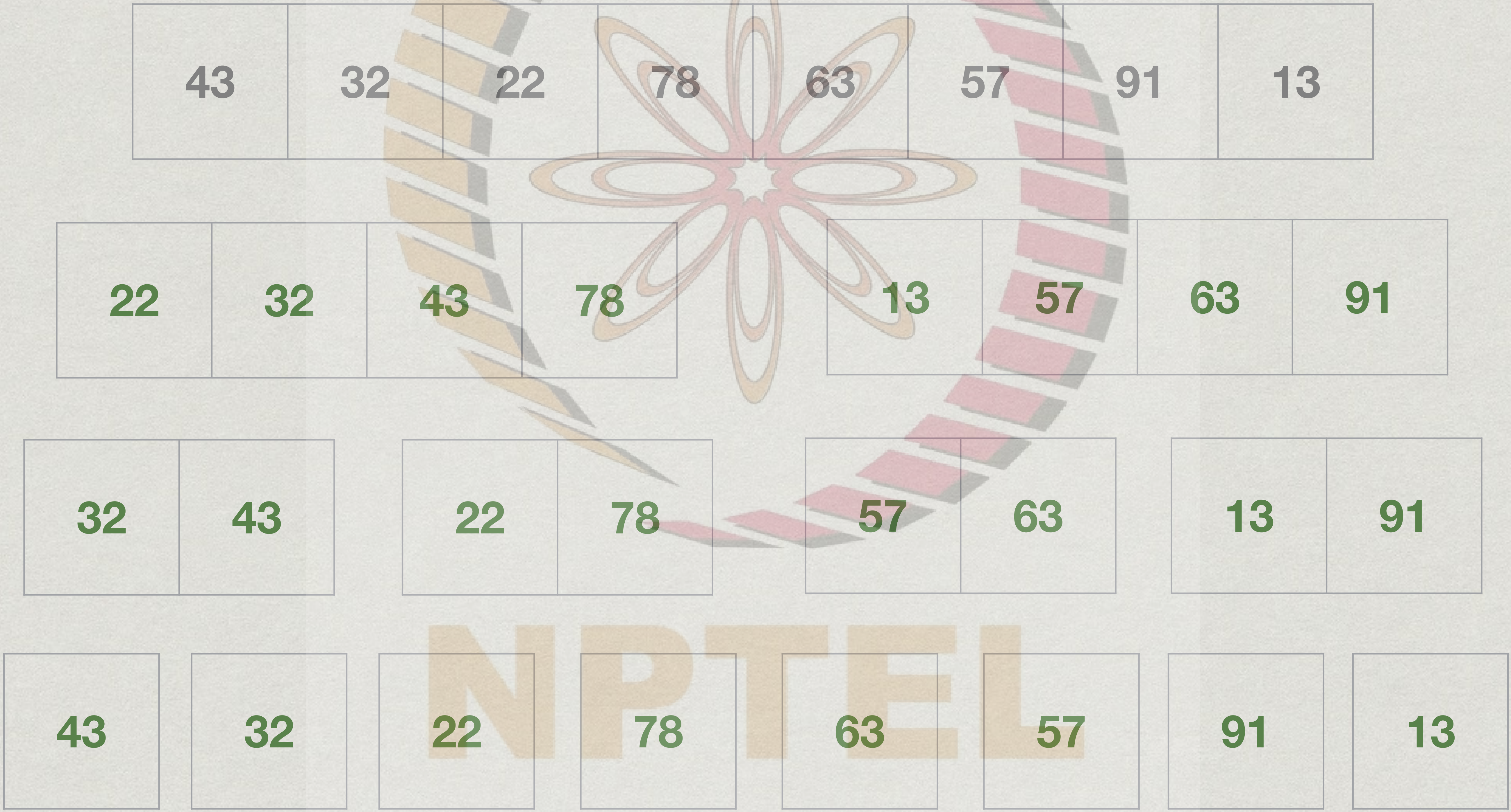
43 32 22 78 63 57 91 13

22 32 43 78 63 57 91 13

32 43 22 78 57 63 13 91

43 32 22 78 63 57 91 13

Merge Sort



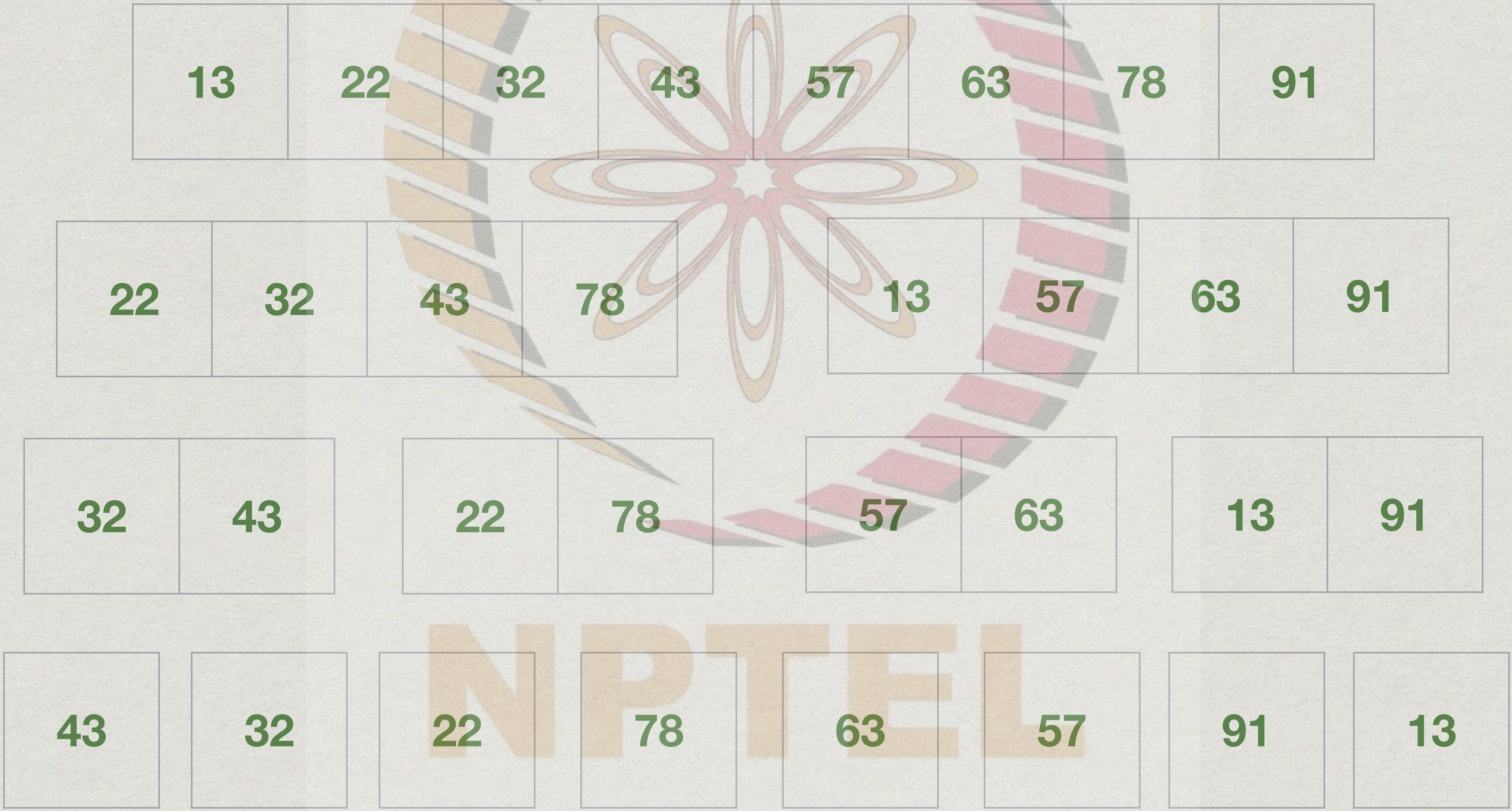
43 32 22 78 63 57 91 13

22 32 43 78 13 57 63 91

32 43 22 78 57 63 13 91

43 32 22 78 63 57 91 13

Merge Sort



13 22 32 43 57 63 78 91

22 32 43 78 13 57 63 91

32 43 22 78 57 63 13 91

43 32 22 78 63 57 91 13

Divide and conquer

- Break up problem into disjoint parts
- Solve each part separately
- Combine the solutions efficiently

Merging sorted lists

Combine two sorted lists A and B into C

- If A is empty, copy B into C
- If B is empty, copy A into C
- Otherwise, compare first element of A and B and move the smaller of the two into C
- Repeat until all elements in A and B have been moved

Merging

```
function Merge(A,m,B,n,C)
  // Merge A[0..m-1], B[0..n-1] into C[0..m+n-1]

  i = 0; j = 0; k = 0;
  // Current positions in A,B,C respectively

  while (k < m+n)
  // Case 0: One of the two lists is empty
     if (i==m) {j++; k++;}
      if (j==n) {i++; k++;}
  // Case 1: Move head of A into C
     if (A[i] <= B[j]) { C[k] = B[j]; j++; k++;}
  // Case 2: Move head of B into C
     if (A[i] > B[j]) {C[k] = B[j]; j++; k++;}
```

Merge Sort

To sort A[0..n-1] into B[0..n-1]

- If n is 1, nothing to be done
- Otherwise
  - Sort A[0..n/2-1] into L (left)
  - Sort A[n/2..n-1] into R (right)
  - Merge L and R into B

Merge Sort

```
function MergeSort(A,left,right,B)
   // Sort the segment A[left..right-1] into B

   if (right - left == 1) // Base case
      B[0] = A[left]

   if (right - left > 1)  // Recursive call

      mid = (left+right)/2

      MergeSort(A,left,mid,L)
      MergeSort(A,mid,right,R)

      Merge(L,mid-left,R,right-mid,B)
```