NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 4, Lecture 1

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 5000

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

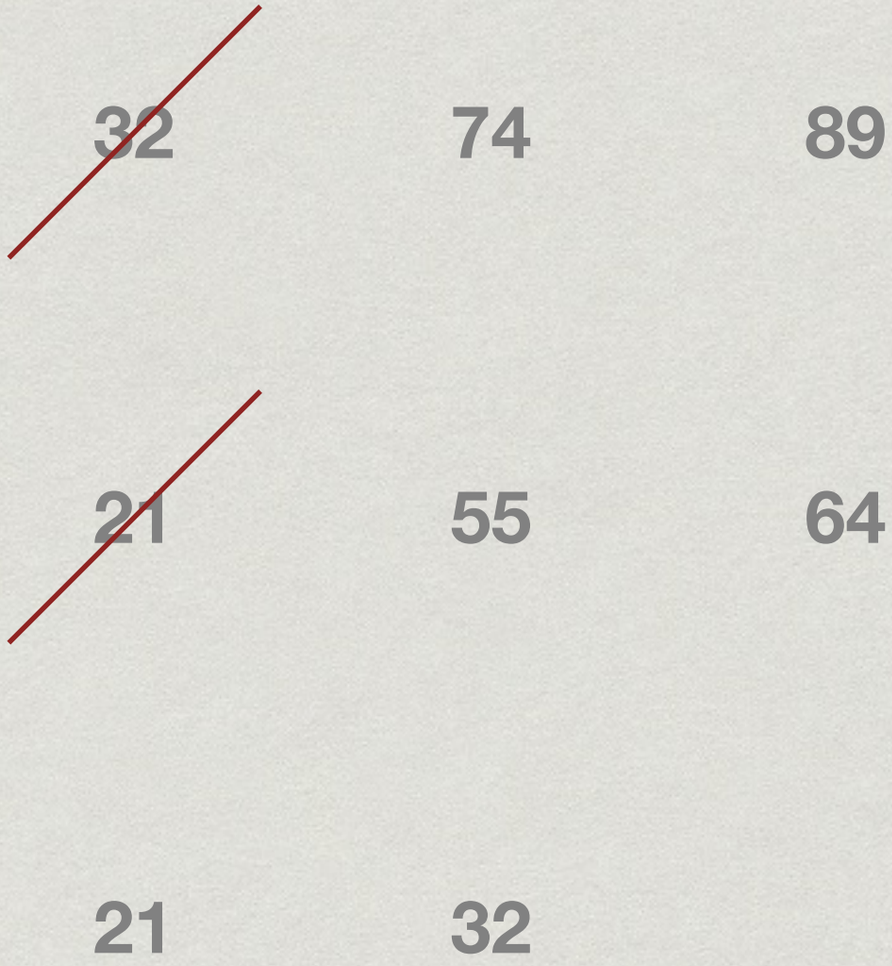
Merging two sorted lists

32 74 89

21 55 64

21

Merging two sorted lists



Merging two sorted lists

Merging two sorted lists

Merging two sorted lists

Merging two sorted lists

21 32 55 64 74 89

Merge Sort

- Sort `A[0:n//2]`
- Sort `A[n//2:n]`
- Merge sorted halves into `B[0:n]`
- 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

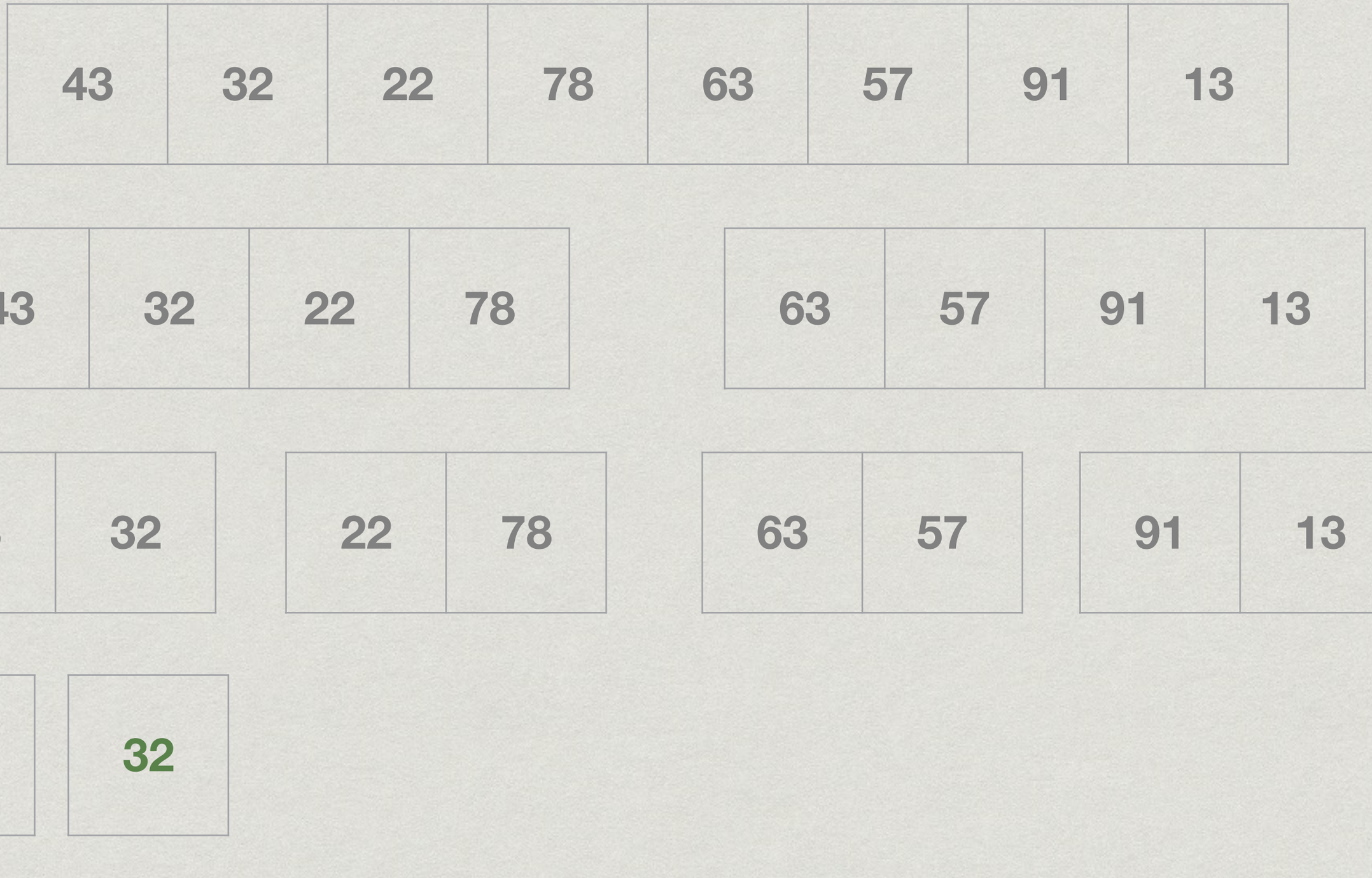
43	32

22	78

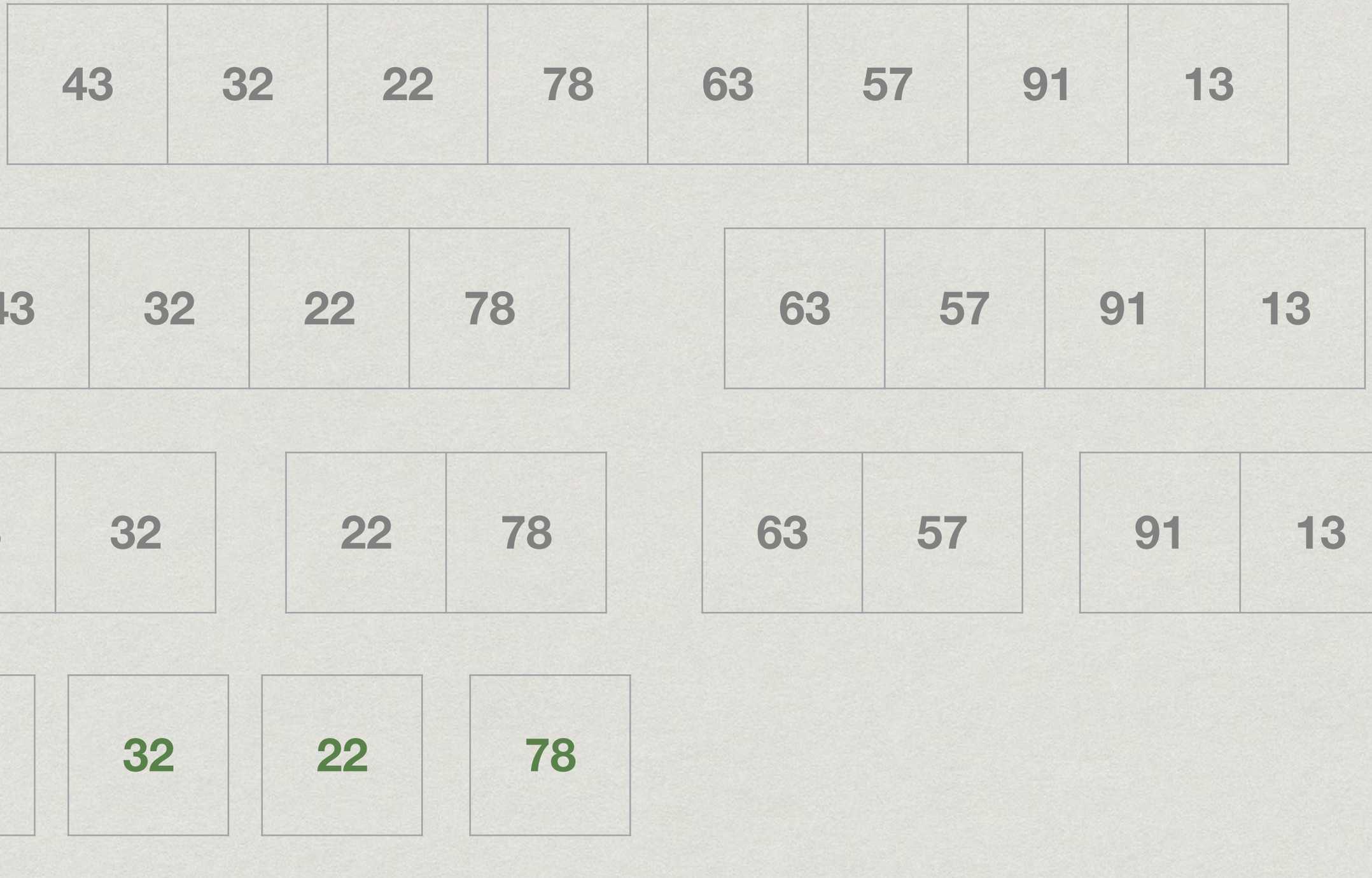
63	57

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

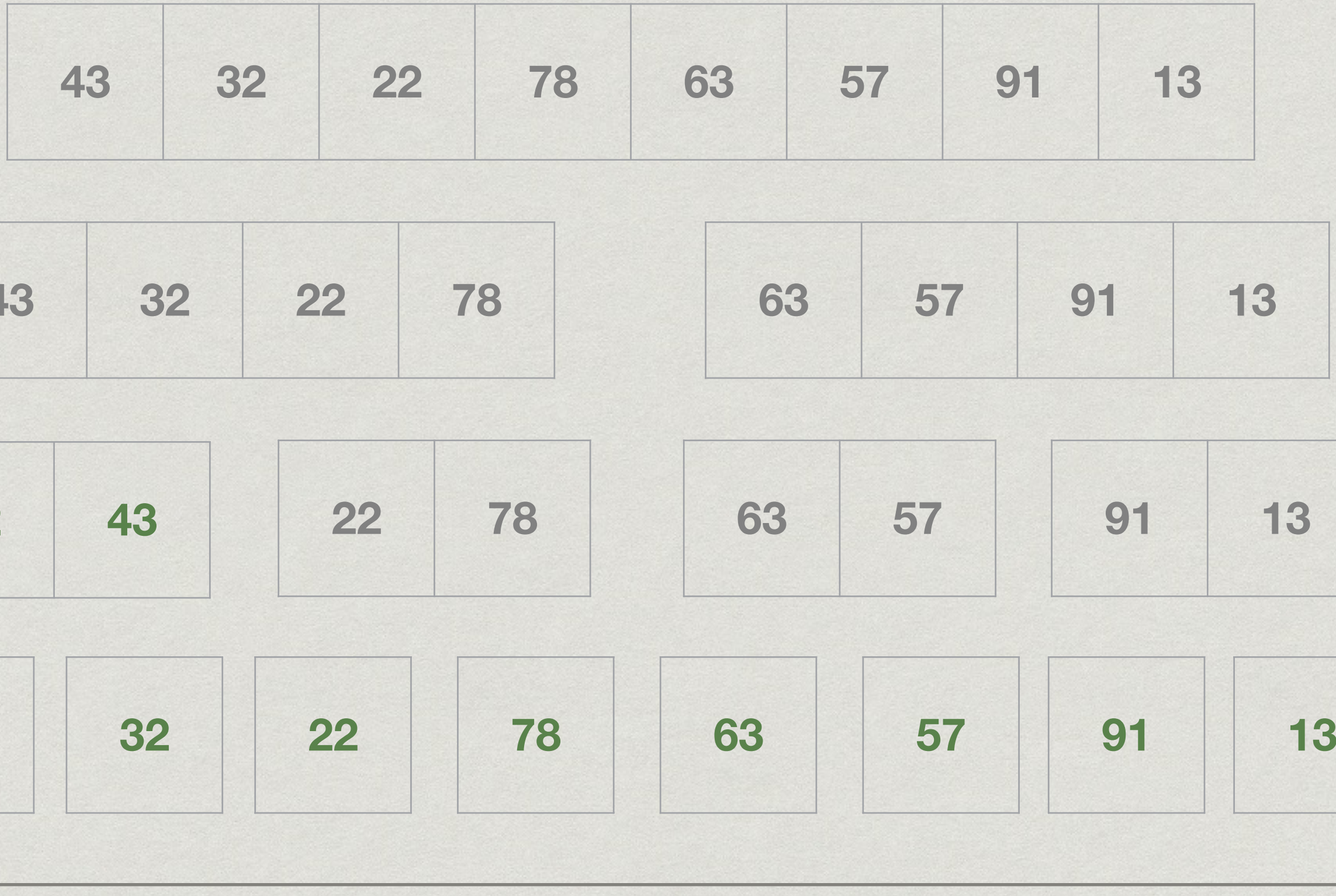
22	78

63	57

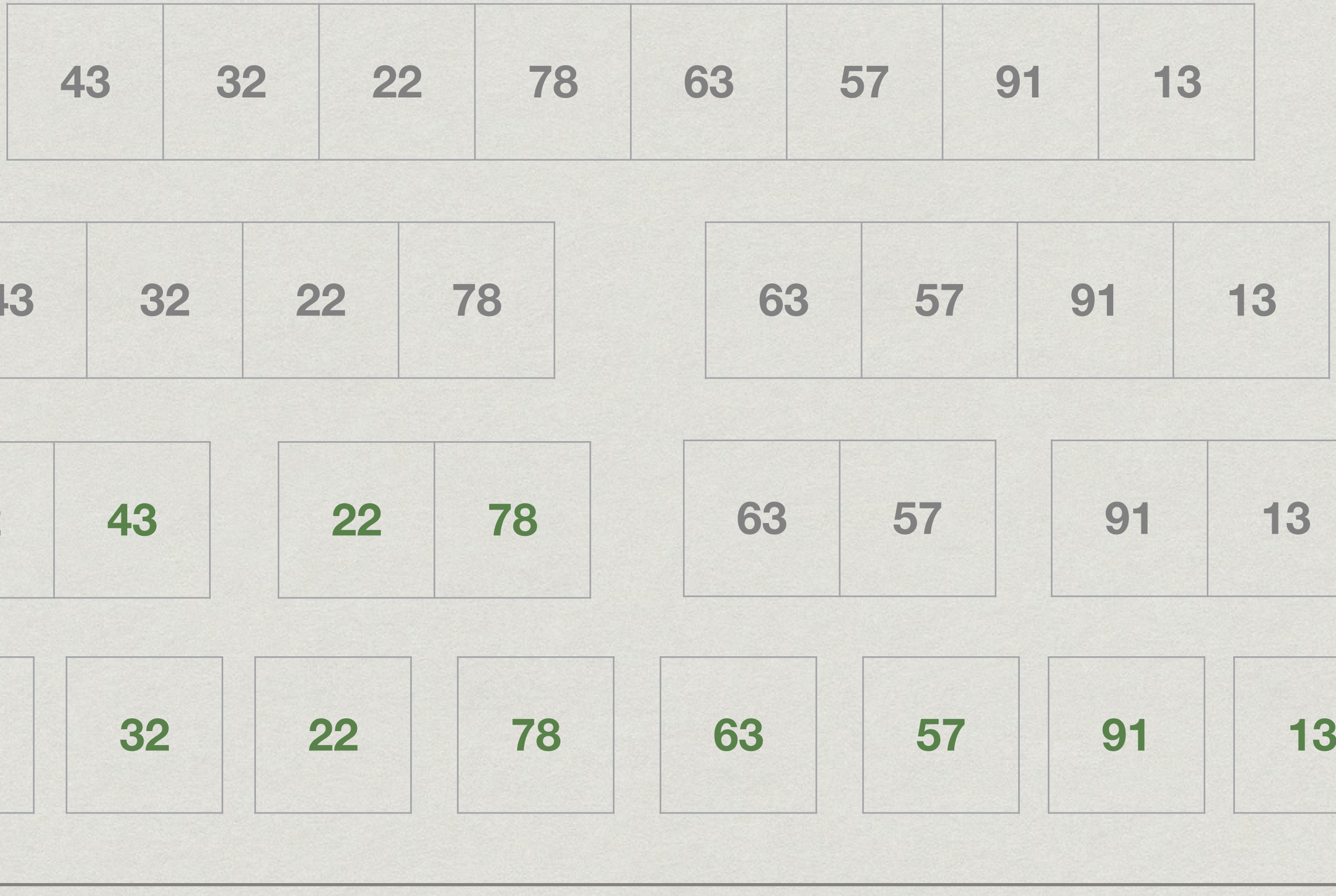
91	13

43	32	22	78	63	57	91	13

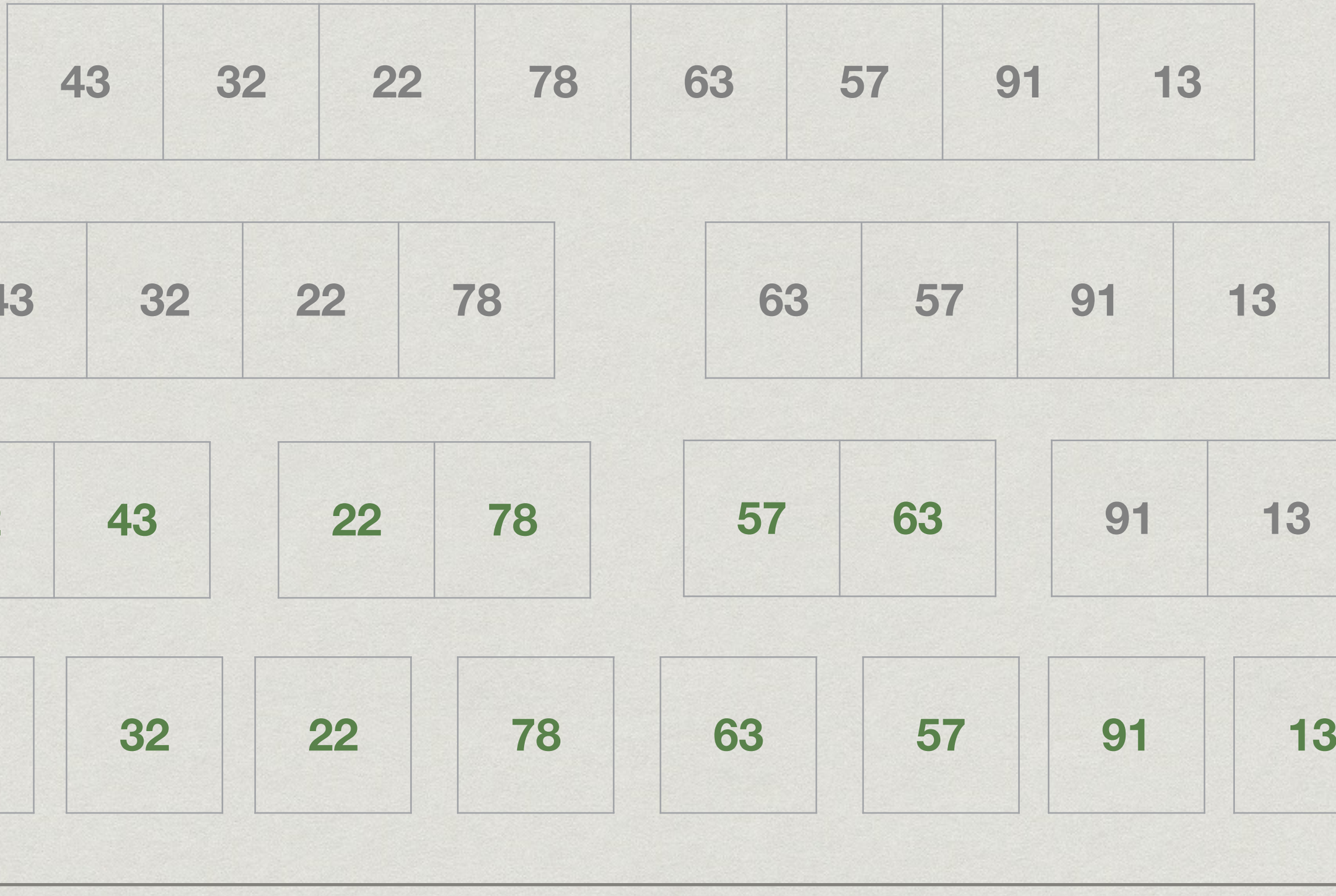
Merge Sort


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

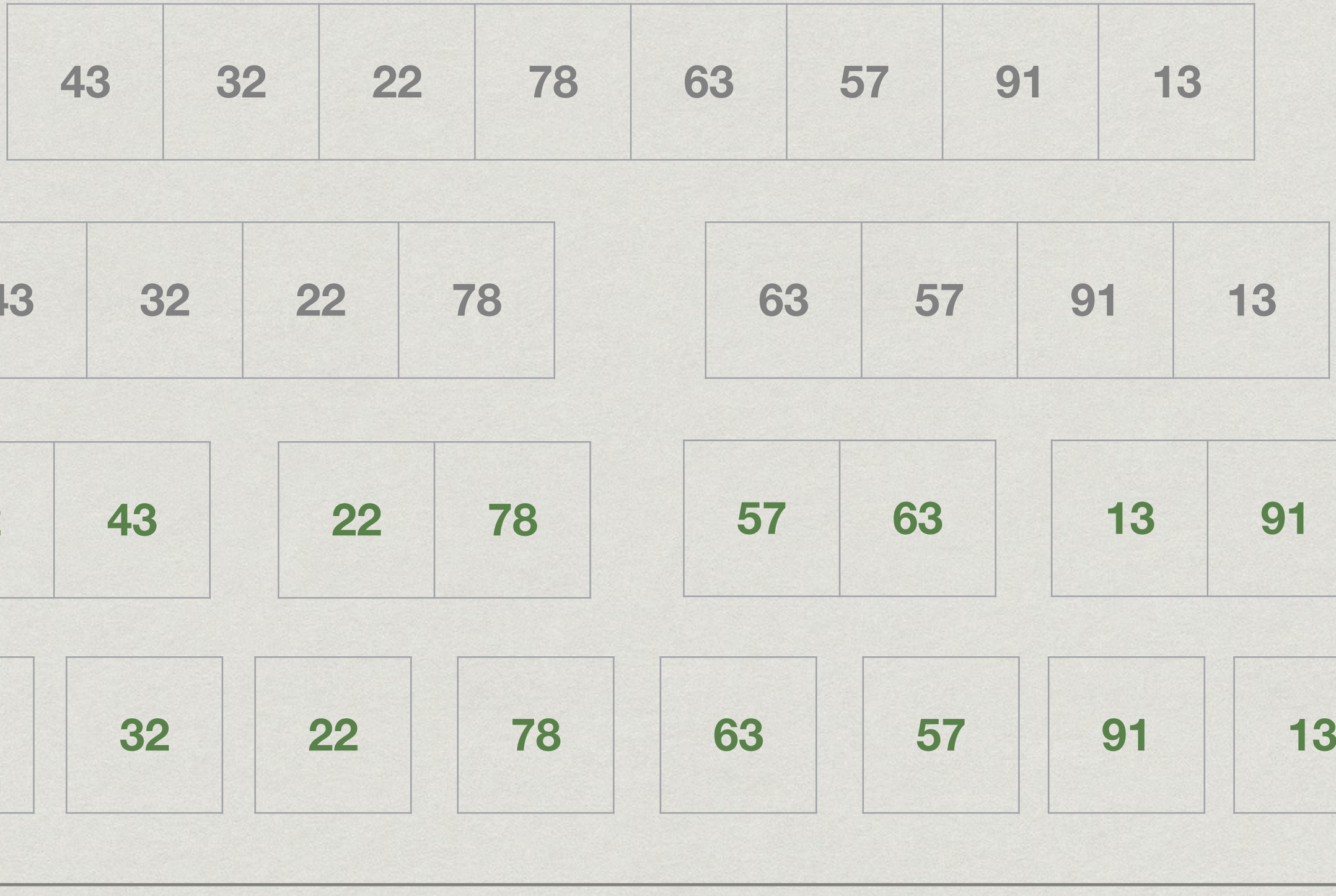
Merge Sort


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

```
def merge(A,B): # Merge A[0:m],B[0:n]
  (C,m,n) = ([],len(A),len(B))
  (i,j) = (0,0) # Current positions in A,B
  while i+j < m+n: # i+j is number of elements merged so far
    if i == m:  # Case 1: A is empty
      C.append(B[j])
      j = j+1
    elif j == n: # Case 2: B is empty
      C.append(A[i])
      i = i+1
    elif A[i] <= B[j]: # Case 3: Head of A is smaller
      C.append(A[i])
      i = i+1
    elif A[i] > B[j]: # Case 4: Head of B is smaller
      C.append(B[j])
      j = j+1
  return(C)
```

Merging, wrong

```
def mergewrong(A,B): # Merge A[0:m],B[0:n]
  (C,m,n) = ([],len(A),len(B))
  (i,j) = (0,0) # Current positions in A,B
  while i+j < m+n:
  # i+j is number of elements merged so far
    # Combine Case 1, Case 4
    if i == m or A[i] > B[j]:
      C.append(B[j])
      j = j+1
    # Combine Case 2, Case 3:
    elif j == n or A[i] <= B[j}:
      C.append(A[i])
      i = i+1
  return(C)
```

Merge Sort

To sort `A[0:n]` into `B[0:n]`

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

Merge Sort

```
def mergesort(A,left,right):
    # Sort the slice A[left:right]

    if right - left <= 1: # Base case
        return(A[left:right])

    if right - left > 1:  # Recursive call

        mid = (left+right)//2

        L = mergesort(A,left,mid)
        R = mergesort(A,mid,right)

        return(merge(L,R))
```