Merge Sort

Madhavan Mukund

`https://www.cmi.ac.in/~madhavan`

Programming, Data Structures and Algorithms using Python

Week 2

Beating the $O(n^2)$ barrier

- Both selection sort and insertion sort take time $O(n^2)$
- This is infeasible for $n > 10000$

Beating the $O(n^2)$ barrier

- Both selection sort and insertion sort take time $O(n^2)$
- This is infeasible for $n > 10000$
- How can we bring the complexity below $O(n^2)$?

Beating the $O(n^2)$ barrier

- Both selection sort and insertion sort take time $O(n^2)$
- This is infeasible for $n > 10000$
- How can we bring the complexity below $O(n^2)$?

Strategy 3

- Divide the list into two halves

Beating the $O(n^2)$ barrier

- Both selection sort and insertion sort take time $O(n^2)$
- This is infeasible for $n > 10000$
- How can we bring the complexity below $O(n^2)$?

Strategy 3

- Divide the list into two halves
- Separately sort the left and right half

Beating the $O(n^2)$ barrier

- Both selection sort and insertion sort take time $O(n^2)$
- This is infeasible for $n > 10000$
- How can we bring the complexity below $O(n^2)$?

Strategy 3

- Divide the list into two halves
- Separately sort the left and right half
- Combine the two sorted halves to get a fully sorted list

Combining two sorted lists

- Combine two sorted lists `A` and `B` into a single sorted list `C`

Combining two sorted lists

- Combine two sorted lists `A` and `B` into a single sorted list `C`
    - Compare first elements of `A` and `B`

Combining two sorted lists

- Combine two sorted lists `A` and `B` into a single sorted list `C`
  - Compare first elements of `A` and `B`
  - Move the smaller of the two to `C`

Combining two sorted lists

- Combine two sorted lists A and B into a single sorted list C
  - Compare first elements of A and B
  - Move the smaller of the two to C
  - Repeat till you exhaust A and B

Combining two sorted lists

- Combine two sorted lists `A` and `B` into a single sorted list `C`
    - Compare first elements of `A` and `B`
    - Move the smaller of the two to `C`
    - Repeat till you exhaust `A` and `B`

32 74 89

21 55 64

Combining two sorted lists

- Combine two sorted lists A and B into a single sorted list C
  - Compare first elements of A and B
  - Move the smaller of the two to C
  - Repeat till you exhaust A and B

32 74 89

~~21~~ 55 64

21

Combining two sorted lists

- Combine two sorted lists A and B into a single sorted list C
  - Compare first elements of A and B
  - Move the smaller of the two to C
  - Repeat till you exhaust A and B

~~32~~ 74 89

~~21~~ 55 64

21 32

Combining two sorted lists

- Combine two sorted lists `A` and `B` into a single sorted list `C`
  - Compare first elements of `A` and `B`
  - Move the smaller of the two to `C`
  - Repeat till you exhaust `A` and `B`

~~32~~ 74 89

~~21~~ ~~55~~ 64

21 32 55

Combining two sorted lists

- Combine two sorted lists A and B into a single sorted list C
  - Compare first elements of A and B
  - Move the smaller of the two to C
  - Repeat till you exhaust A and B

~~32~~ 74 89

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

21 32 55 64

Combining two sorted lists

- Combine two sorted lists A and B into a single sorted list C
  - Compare first elements of A and B
  - Move the smaller of the two to C
  - Repeat till you exhaust A and B

~~32~~ ~~74~~ 89

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

21 32 55 64 74

Combining two sorted lists

- Combine two sorted lists `A` and `B` into a single sorted list `C`
  - Compare first elements of `A` and `B`
  - Move the smaller of the two to `C`
  - Repeat till you exhaust `A` and `B`

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

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

21 32 55 64 74 89

Combining two sorted lists

- Combine two sorted lists A and B into a single sorted list C
  - Compare first elements of A and B
  - Move the smaller of the two to C
  - Repeat till you exhaust A and B
- Merging A and B

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

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

21 32 55 64 74 89

Merge sort

- Let `n` be the length of L

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
    - Recursively, same strategy!

Merge sort

- Let n be the length of L
- Sort A[:n//2]
- Sort A[n//2:]
- Merge the sorted halves into B
- How do we sort A[:n//2] and A[n//2:]?
    - Recursively, same strategy!

43	32	22	78	63	57	91	13

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!

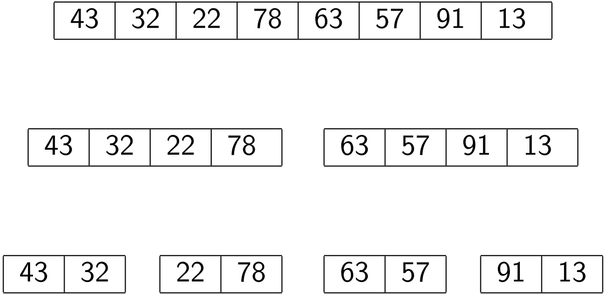
43	32	22	78	63	57	91	13

43	32	22	78

63	57	91	13

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



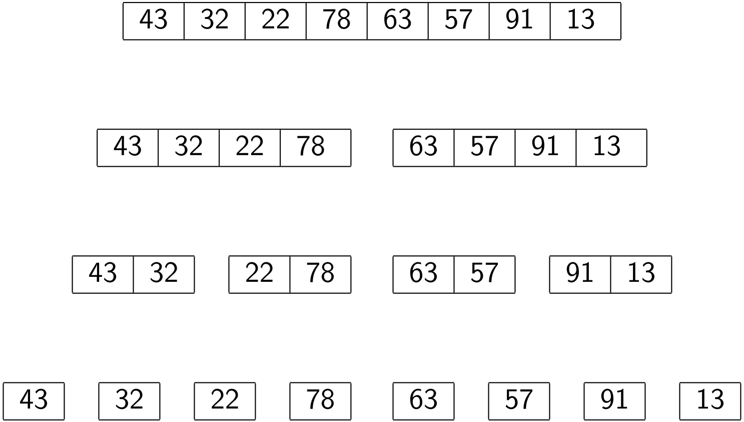
| 43 | 32 | 22 | 78 | 63 | 57 | 91 | 13 |

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

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

Merge sort

- Let n be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
    - Recursively, same strategy!



43 32 22 78 63 57 91 13

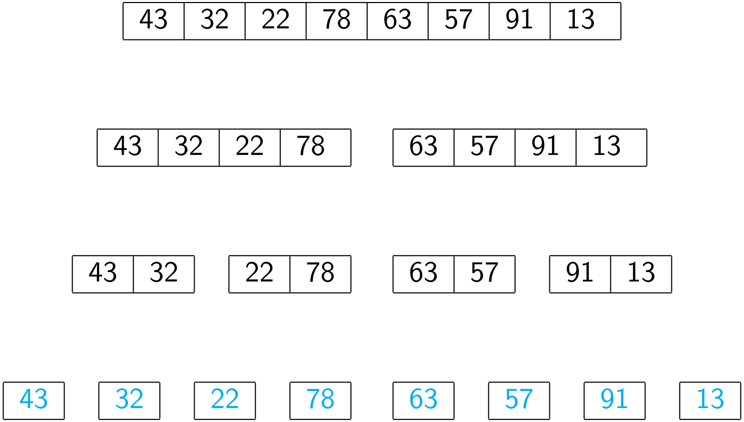
43 32 22 78 | 63 57 91 13

43 32 | 22 78 | 63 57 | 91 13

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

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



43 32 22 78 63 57 91 13

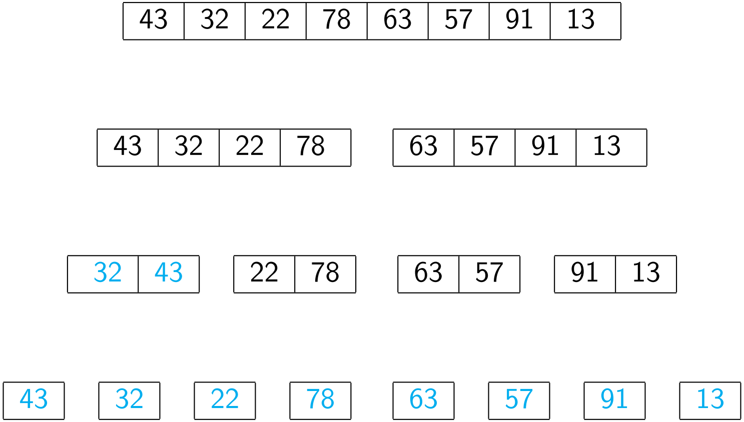
43 32 22 78 | 63 57 91 13

43 32 | 22 78 | 63 57 | 91 13

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

Merge sort

- Let `n` be the length of *L*
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



43 32 22 78 63 57 91 13

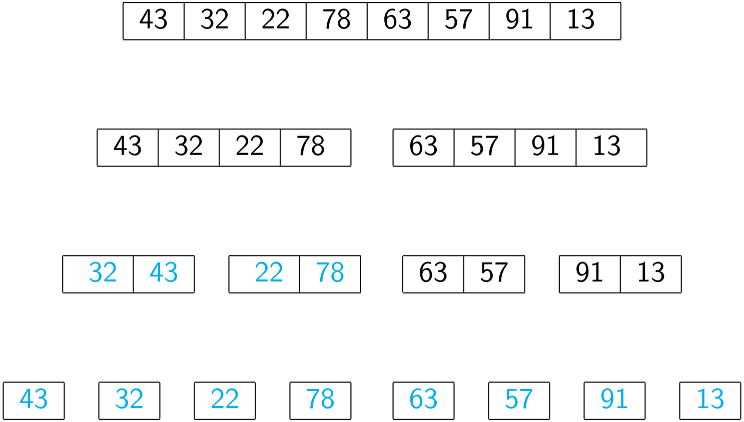
43 32 22 78 | 63 57 91 13

32 43 | 22 78 | 63 57 | 91 13

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

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



43 32 22 78 63 57 91 13

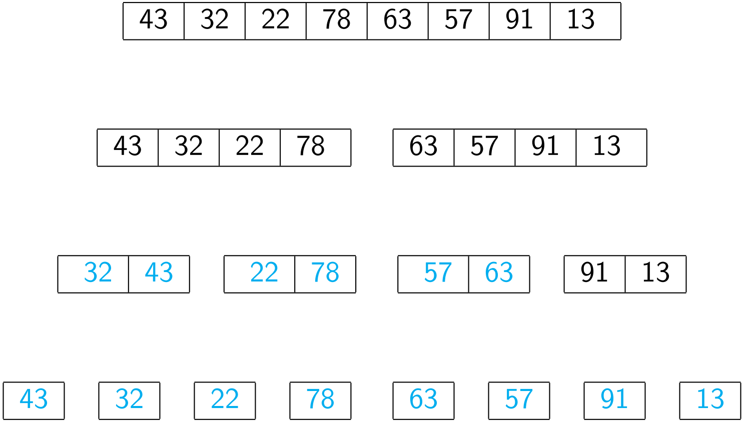
43 32 22 78 | 63 57 91 13

32 43 | 22 78 | 63 57 | 91 13

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

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



43 32 22 78 63 57 91 13

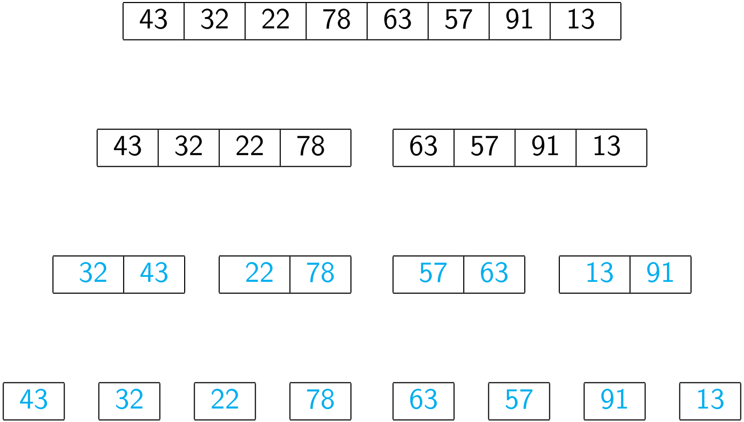
43 32 22 78   63 57 91 13

32 43   22 78   57 63   91 13

43   32   22   78   63   57   91   13

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



43 32 22 78 63 57 91 13

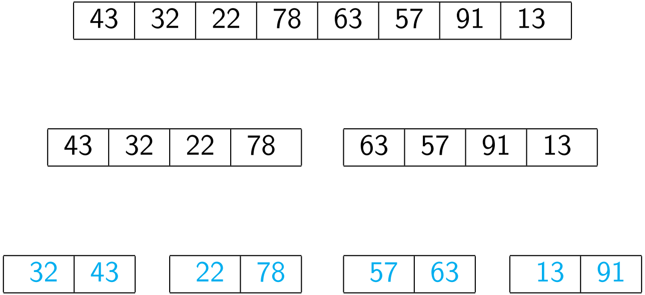
43 32 22 78 | 63 57 91 13

32 43 | 22 78 | 57 63 | 13 91

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

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



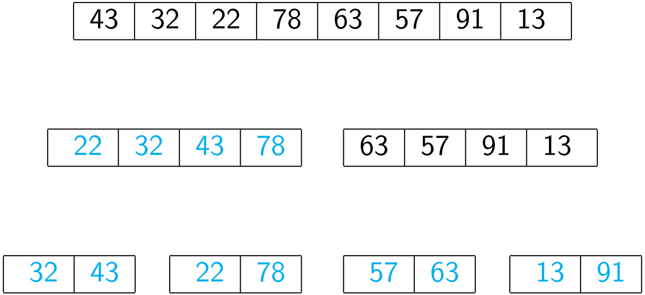
43 32 22 78 63 57 91 13

43 32 22 78   63 57 91 13

32 43   22 78   57 63   13 91

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



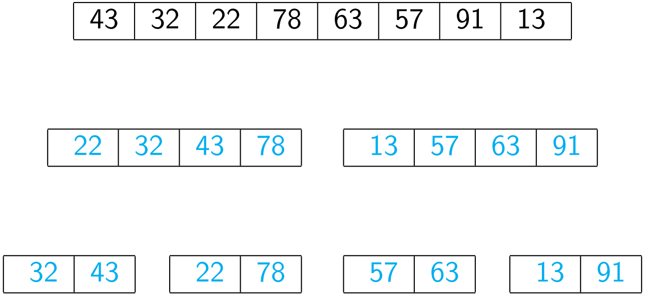
43 32 22 78 63 57 91 13

22 32 43 78 | 63 57 91 13

32 43 | 22 78 | 57 63 | 13 91

Merge sort

- Let n be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!



43	32	22	78	63	57	91	13

22	32	43	78

13	57	63	91

32	43

22	78

57	63

13	91

Merge sort

- Let n be the length of L
- Sort A[:n//2]
- Sort A[n//2:]
- Merge the sorted halves into B
- How do we sort A[:n//2] and A[n//2:]?
  - Recursively, same strategy!

43	32	22	78	63	57	91	13

22	32	43	78

13	57	63	91

Merge sort

- Let n be the length of L
- Sort A[:n//2]
- Sort A[n//2:]
- Merge the sorted halves into B
- How do we sort A[:n//2] and A[n//2:]?
    - Recursively, same strategy!

13	22	32	43	57	63	78	91

22	32	43	78

13	57	63	91

Merge sort

- Let n be the length of L
- Sort A[:n//2]
- Sort A[n//2:]
- Merge the sorted halves into B
- How do we sort A[:n//2] and A[n//2:]?
    - Recursively, same strategy!

13	22	32	43	57	63	78	91

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
    - Recursively, same strategy!

Merge sort

- Let `n` be the length of L
- Sort `A[:n//2]`
- Sort `A[n//2:]`
- Merge the sorted halves into `B`
- How do we sort `A[:n//2]` and `A[n//2:]`?
  - Recursively, same strategy!

Divide and Conquer

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

Merging sorted lists

- Combine two sorted lists `A` and `B` into `C`

Merging sorted lists

- Combine two sorted lists `A` and `B` into `C`
    - If `A` is empty, copy `B` into `C`

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`

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 elements of `A` and `B`
    - Move the smaller of the two to `C`

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 elements of `A` and `B`
    - Move the smaller of the two to `C`
  - Repeat till all elements of `A` and `B` have been moved

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 elements of A and B
    - Move the smaller of the two to C
  - Repeat till all elements of A and B have been moved

```
def merge(A,B):
  (m,n) = (len(A),len(B))
  (C,i,j,k) = ([],0,0,0)
  while k < m+n:
    if i == m:
      C.extend(B[j:])
      k = k + (n-j)
    elif j == n:
      C.extend(A[i:])
      k = k + (n-i)
    elif A[i] < B[j]:
      C.append(A[i])
      (i,k) = (i+1,k+1)
    else:
      C.append(B[j])
      (j,k) = (j+1,k+1)
  return(C)
```

Merge sort

- To sort `A` into `B`, both of length n

Merge sort

- To sort `A` into `B`, both of length n
- If $n \leq 1$, nothing to be done

Merge sort

- To sort `A` into `B`, both of length n
- If $n \leq 1$, nothing to be done
- Otherwise

Merge sort

- To sort `A` into `B`, both of length n
- If $n \leq 1$, nothing to be done
- Otherwise
  - Sort `A[:n//2]` into `L`

Merge sort

- To sort `A` into `B`, both of length n
- If $n \leq 1$, nothing to be done
- Otherwise
  - Sort `A[:n//2]` into `L`
  - Sort `A[n//2:]` into `R`

Merge sort

- To sort `A` into `B`, both of length n
- If $n \leq 1$, nothing to be done
- Otherwise
  - Sort `A[:n//2]` into `L`
  - Sort `A[n//2:]` into `R`
  - Merge `L` and `R` into `B`

Merge sort

- To sort A into B, both of length n
- If $n \leq 1$, nothing to be done
- Otherwise
  - Sort A[:n//2] into L
  - Sort A[n//2:] into R
  - Merge L and R into B

```
def mergesort(A):
  n = len(A)

  if n <= 1:
     return(A)

  L = mergesort(A[:n//2])
  R = mergesort(A[n//2:])

  B = merge(L,R)

  return(B)
```

Summary

- Merge sort using divide and conquer to sort a list

Summary

- Merge sort using divide and conquer to sort a list
- Divide the list into two halves

Summary

- Merge sort using divide and conquer to sort a list
- Divide the list into two halves
- Sort each half

Summary

- Merge sort using divide and conquer to sort a list
- Divide the list into two halves
- Sort each half
- Merge the sorted halves

Summary

- Merge sort using divide and conquer to sort a list
- Divide the list into two halves
- Sort each half
- Merge the sorted halves
- Next, we have to check that the complexity is less than $O(n^2)$