Merge Sort

Madhavan Mukund

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

Programming, Data Structures and Algorithms using Python
Week 2

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

Madhavan Mukund Merge Sort PDSA using Python Week 2

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

Madhavan Mukund Merge Sort PDSA using Python Week 2

- 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

- 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

- 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

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

Madhavan Mukund Merge Sort PDSA using Python Week 2

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

Madhavan Mukund Merge Sort PDSA using Python Week 2

- 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

- 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

- 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

- 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

- 2 74 89
- 21 55 64

21

- 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

- Combine two sorted lists A and B into a single sorted list C
- 32 74 89

■ Compare first elements of A and B

21 55 64

Move the smaller of the two to CRepeat till you exhaust A and B

21 32 55

- 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

- 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

- 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

- 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

■ Let n be the length of L

Madhavan Mukund Merge Sort PDSA using Python Week 2

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

Madhavan Mukund Merge Sort PDSA using Python Week 2

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

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

- 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:]?

- 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!

- 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!

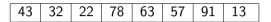
43	32	22	78	63	57	91	13

- 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



- 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!







- 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

- 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

- 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

- 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//2:1]
 - Recursively, same strategy!

43 32 22 78 63 57 91 13

43 32 22 78 63 57 91 13

78 63 57 91 13

- 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

- 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

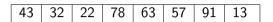
43 32 22 78 63 57 91 13

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

43 32 22 78 63 57 91 13

PDSA using Python Week 2

- 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!







- 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!



22 32 43 78 63 57 91 13

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

- 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!



22	32	43	78	13	57	63	91

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

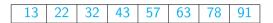
- Let n be the length of L
- Sort A[:n//2]
- Sort A[n//2:]
- Merge the sorted halves into B
- How do we sort A[:n//2] and A[n//2:]?
 - Recursively, same strategy!



22 | 32 | 43 | 78

13 | 57 | 63 | 91

- 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!



22	32	43	78

- 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
13		52	75	31	0.5	10	91

- 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!

- 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

■ Combine two sorted lists A and B into C

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

- 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

- 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

- 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

- 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[i:])
      k = k + (n-j)
    elif i == n:
      C.extend(A[i:])
      k = k + (m-i)
    elif A[i] < B[j]:</pre>
      C.append(A[i])
      (i,k) = (i+1,k+1)
    else:
      C.append(B[j])
      (j,k) = (j+1,k+1)
  return(C)
```

■ To sort A into B, both of length n

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

Madhavan Mukund Merge Sort PDSA using

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

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

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

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

- To sort A into B, both of length n
- If $n \le 1$, 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 \le 1:
     return(A)
 L = mergesort(A[:n//2])
 R = mergesort(A[n//2:])
 B = merge(L,R)
  return(B)
```

Merge sort using divide and conquer to sort a list

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

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

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

- 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 $O(n^2)$

8/8