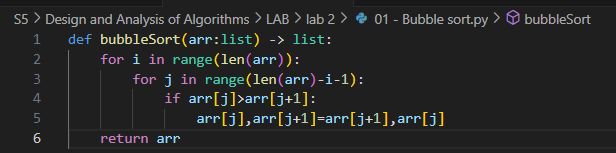
Lab Sheet 2

Sorting and Recursion

Implement the following sorting algorithms and answer the associated questions.

1. Bubble sort
   1. What is the time complexity of a simple bubble sort algorithm?



**Time Complexity : O(n­2)**

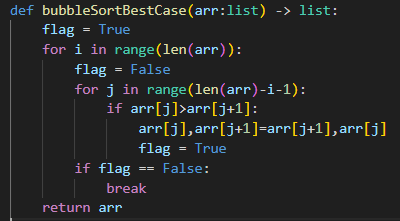
Is there any difference between the best case and the worst case?

Yes, best case is when the array is already sorted.

* 1. How can we change the best-case complexity to Ω(n)?

By using a flag to check if the array is the already sorted or not.

Modify your algorithm accordingly.



What is the worst-case complexity of the improved algorithm?

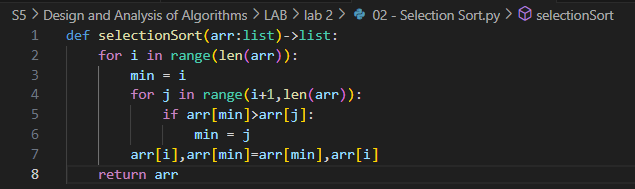
O(n2)

* 1. Give examples for best-case and worst-case inputs.

**Best case : [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]**

**Worst case: [ 9, 8, 7, 6, 5, 4, 3, 2, 1 ]**

1. Selection sort
   1. What is the time complexity of a simple selection sort algorithm?



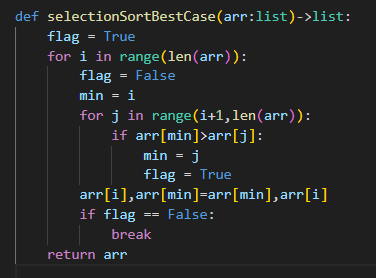
Is there any difference between the best case and the worst case?

Yes, best case O(n) and worst case O(n2)

* 1. How can we change the best-case complexity to Ω(n)?

By using a flag to check if the array is the already sorted or not.

Modify your algorithm accordingly.



What is the worst-case complexity of the improved algorithm?

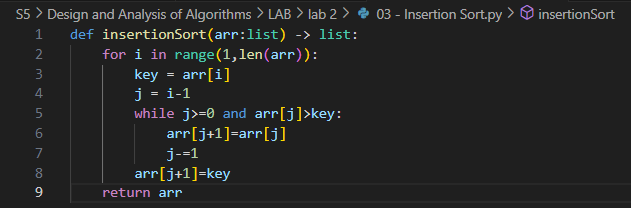
Worst case is O(n2)

* 1. Give examples for best-case and worst-case inputs.

**Best case : [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]**

**Worst case: [ 9, 8, 7, 6, 1, 2, 3, 4, 5]**

1. Insertion sort
   1. What is the time complexity of a simple selection sort algorithm?



**Time Complexity : O(n­2)**

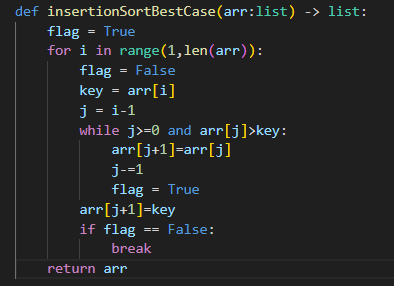
Is there any difference between the best case and the worst case?

Yes, best case is when the array is already sorted.

* 1. How can we change the best-case complexity to Ω(n)?

By using a flag to check if the array is the already sorted or not.

Modify your algorithm accordingly.



What is the worst-case complexity of the improved algorithm?

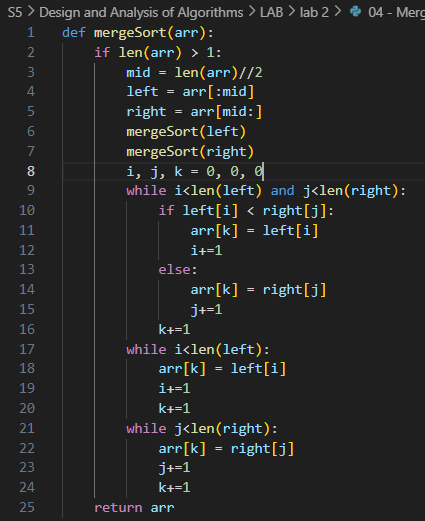
O(n2)

* 1. Give examples for best-case and worst-case inputs.

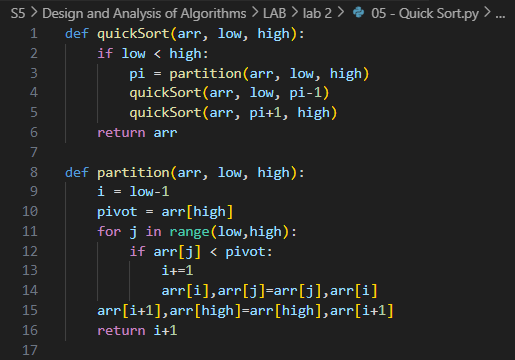
**Best case : [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]**

**Worst case: [ 9, 8, 7, 6, 5, 4, 3, 2, 1 ]**

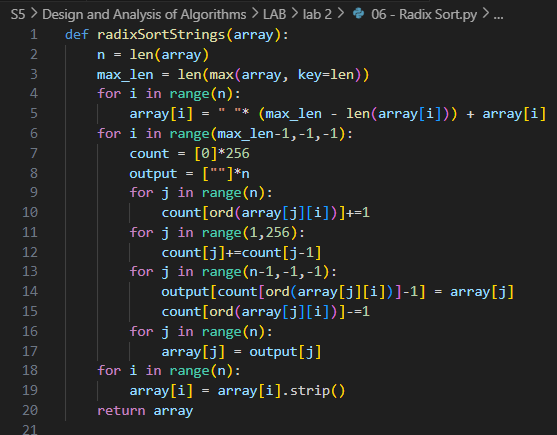
1. Merge sort



1. Quick sort

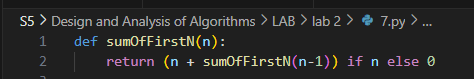


1. Sort a set of strings using Radix sort



Write Recursive algorithms for the following problems. Implement your algorithm, write the recurrence relation, solve it, and find the asymptotic time complexity

1. Print the sum of the first N natural numbers.



T(n)= T( n – 1 ) + 1

Gen => T(n) = T( n – k ) + k

n-k = 0

n=k

∴ T(n) = T( n – n ) + n

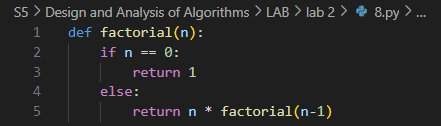
T(n) = T(0) + n

T(n) = 1 + n

T(n) = O(n)

Time Complexity : O(n)

1. Print the product of the first N natural numbers.



T(n)= T( n – 1 ) + 1

Gen => T(n) = T( n – k ) + k

n-k = 0

n=k

∴ T(n) = T( n – n ) + n

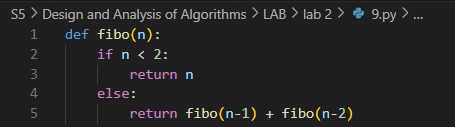
T(n) = T(0) + n

T(n) = 1 + n

T(n) = O(n)

Time Complexity : O(n)

1. Print the Nth Fibonacci number.



T(n)= T( n – 1 ) + T( n – 2 ) + 1

Gen => T(n) = T( n – k ) + T( n – l ) + ( k + l )

n-k = 0 n-l = 0

n=k n=l

∴ T(n) = T( n – n ) + T( n – n ) + n

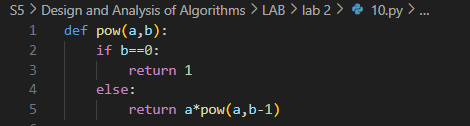
T(n) = T(0) + n

T(n) = 1 + n

T(n) = O(n)

Time Complexity : O(n)

1. Calculate xy.



T(n)= T( n – 1 ) + 1

Gen => T(n) = T( n – k ) + k

n-k = 0

n=k

∴ T(n) = T( n – n ) + n

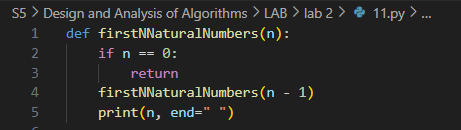
T(n) = T(0) + n

T(n) = 1 + n

T(n) = O(n)

Time Complexity : O(n)

1. Print the first N natural numbers.



T(n)= T( n – 1 ) + 1

Gen => T(n) = T( n – k ) + k

n-k = 0

n=k

∴ T(n) = T( n – n ) + n

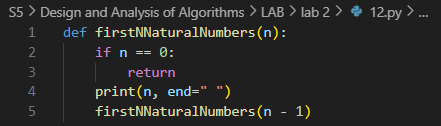
T(n) = T(0) + n

T(n) = 1 + n

T(n) = O(n)

Time Complexity : O(n)

1. Print the first N natural numbers in reverse order.



T(n)= T( n – 1 ) + 1

Gen => T(n) = T( n – k ) + k

n-k = 0

n=k

∴ T(n) = T( n – n ) + n

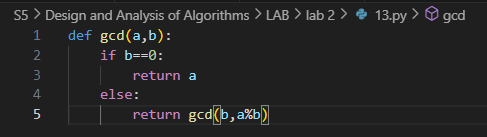
T(n) = T(0) + n

T(n) = 1 + n

T(n) = O(n)

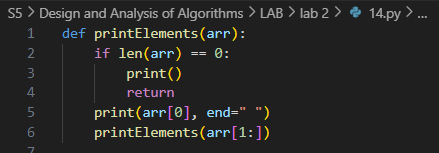
Time Complexity : O(n)

1. Find the GCD(HCF) of two numbers.



O(log(min(a,b)))

1. Print the elements of an array.



T(n)= T( n – 1 ) + 1

Gen => T(n) = T( n – k ) + k

n-k = 0

n=k

∴ T(n) = T( n – n ) + n

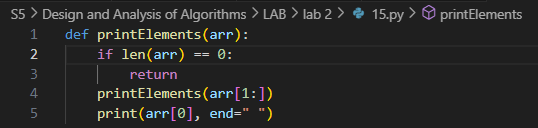
T(n) = T(0) + n

T(n) = 1 + n

T(n) = O(n)

Time Complexity : O(n)

1. Print the elements of an array in reverse order.



T(n)= T( n – 1 ) + 1

Gen => T(n) = T( n – k ) + k

n-k = 0

n=k

∴ T(n) = T( n – n ) + n

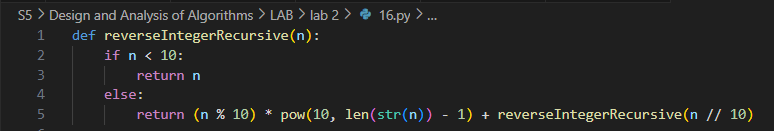
T(n) = T(0) + n

T(n) = 1 + n

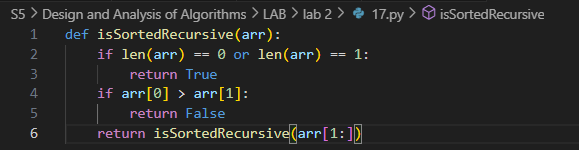
T(n) = O(n)

Time Complexity : O(n)

1. Reverse a given number.



1. Check if an array is sorted or not.



T(n)= T( n – 1 ) + 1

Gen => T(n) = T( n – k ) + k

n-k = 0

n=k

∴ T(n) = T( n – n ) + n

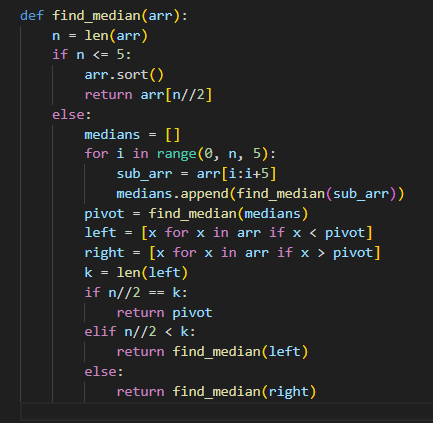
T(n) = T(0) + n

T(n) = 1 + n

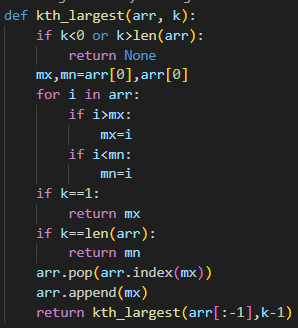
T(n) = O(n)

Time Complexity : O(n)

1. Write a recursive algorithm to find the median of median in O(n) time.



1. Write a recursive algorithm to find the kth largest element.



O(n2)