**Algorithm Mini project**

Index

|  |  |
| --- | --- |
| Sr. No. | Name |
| 1 | Quick sort implementation |
| 2 | Maximum subarray |

1. Python code for implementation of Quick Sort algorithm.

Ans:-

Code:-

def partition(arr, low, high):

i = (low-1) # index of smaller element

pivot = arr[high] # pivot

for j in range(low, high):

# If current element is smaller than or

# equal to pivot

if arr[j] <= pivot:

# increment index of smaller element

i = i+1

arr[i], arr[j] = arr[j], arr[i]

arr[i+1], arr[high] = arr[high], arr[i+1]

return (i+1)

# The main function that implements QuickSort

# arr[] --> Array to be sorted,

# low --> Starting index,

# high --> Ending index

# Function to do Quick sort

def quickSort(arr, low, high):

if len(arr) == 1:

return arr

if low < high:

# pi is partitioning index, arr[p] is now

# at right place

pi = partition(arr, low, high)

# Separately sort elements before

# partition and after partition

quickSort(arr, low, pi-1)

quickSort(arr, pi+1, high)

# Driver code to test above

arr = [10, 7, 8, 9, 1, 5]

n = len(arr)

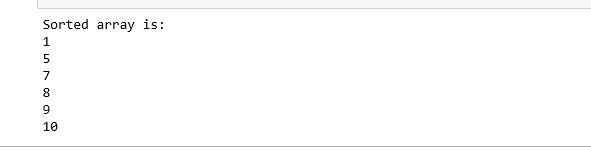
quickSort(arr, 0, n-1)

print("Sorted array is:")

for i in range(n):

print("%d" % arr[i])

o/p:-



2)Python programme for implementation of maximum subarray problem.

Ans:-

Code:-

from math import inf

def max\_crossing\_sublist(input\_list : list[int], low : int, mid : int, high : int) -> tuple[int]:

left\_sum = -1 \* inf

right\_sum = -1 \* inf

sum = 0

max\_left = -1 \* inf

max\_right = -1 \* inf

for i in range(mid, low - 1, -1):

sum += input\_list[i]

if sum > left\_sum:

max\_left = i

left\_sum = sum

sum = 0

for j in range(mid + 1, high + 1):

sum += input\_list[j]

if sum > right\_sum:

max\_right = j

right\_sum = sum

return max\_left, max\_right, left\_sum + right\_sum

def max\_subarray\_problem(input\_list : list[int], low : int, high : int) -> tuple[int]:

if low == high:

return low, high, input\_list[low]

else:

mid = int((low + high) / 2)

left\_low, left\_high, left\_sum =max\_subarray\_problem(input\_list, low, mid)

right\_low, right\_high, right\_sum =max\_subarray\_problem(input\_list, mid + 1, high)

cross\_low, cross\_high, cross\_sum =max\_crossing\_sublist(input\_list, low, mid, high)

if left\_sum >= right\_sum and left\_sum >= cross\_sum: return left\_low, left\_high, left\_sum

elif right\_sum >= left\_sum and right\_sum >= cross\_sum: return right\_low, right\_high, right\_sum

else:

return cross\_low, cross\_high, cross\_sum

if \_\_name\_\_== "\_\_main\_\_":

from random import randrange

input\_list = [randrange(-100, 100) for i in range(20)]

print("Input array is >", input\_list)

low\_index, high\_index, max\_sum = max\_subarray\_problem(input\_list, 0, len(input\_list) - 1)

print("Maximum sum is: ", max\_sum, "with ", low\_index, "and", high\_index, 'as the lower and upper bounds.')

o/p:-

