**Labsheet 6**

**Dijkstra’s**

class Graph():

    def \_\_init\_\_(self, vertices):

        self.V = vertices

        self.graph = [[0 for column in range(vertices)]

                      for row in range(vertices)]

    def printSolution(self, dist):

        print("Vertex \t Distance from Source")

        for node in range(self.V):

            print(node, "\t\t", dist[node])

    def minDistance(self, dist, sptSet):

        min = 1e7

        for v in range(self.V):

            if dist[v] < min and sptSet[v] == False:

                min = dist[v]

                min\_index = v

        return min\_index

    def dijkstra(self, src):

        dist = [1e7] \* self.V

        dist[src] = 0

        sptSet = [False] \* self.V

        for cout in range(self.V):

            u = self.minDistance(dist, sptSet)

            sptSet[u] = True

            for v in range(self.V):

                if (self.graph[u][v] > 0 and

                   sptSet[v] == False and

                   dist[v] > dist[u] + self.graph[u][v]):

                    dist[v] = dist[u] + self.graph[u][v]

        self.printSolution(dist)

g = Graph(9)

g.graph = [[0, 4, 0, 0, 0, 0, 0, 8, 0],

           [4, 0, 8, 0, 0, 0, 0, 11, 0],

           [0, 8, 0, 7, 0, 4, 0, 0, 2],

           [0, 0, 7, 0, 9, 14, 0, 0, 0],

           [0, 0, 0, 9, 0, 10, 0, 0, 0],

           [0, 0, 4, 14, 10, 0, 2, 0, 0],

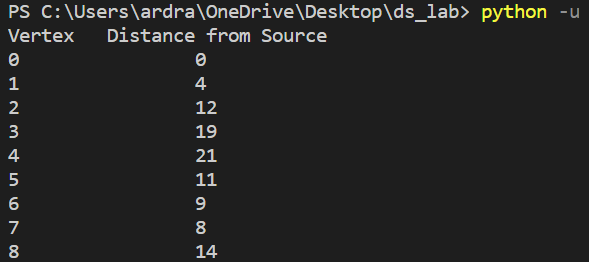
           [0, 0, 0, 0, 0, 2, 0, 1, 6],

           [8, 11, 0, 0, 0, 0, 1, 0, 7],

           [0, 0, 2, 0, 0, 0, 6, 7, 0]

           ]

g.dijkstra(0)



**Max Heap Sort**

# Max Heapify

def max\_heapify(Arr, n, i):

    largest = i

    left = 2\*i + 1

    right = 2\*i + 2

    if left < n and Arr[largest] < Arr[left]:

        largest = left

    if right < n and Arr[largest] < Arr[right]:

        largest = right

    if largest!=i:

        Arr[i], Arr[largest] = Arr[largest], Arr[i]

        max\_heapify(Arr, n, largest)

Arr = [2,66,30,5,9,10]

n = len(Arr)

def heapSort(Arr):

    for i in range(n-1, 0, -1):

        # Swap

        Arr[i], Arr[0] = Arr[0], Arr[i]

        max\_heapify(Arr, i, 0)

for i in range(n//2 - 1, -1, -1):

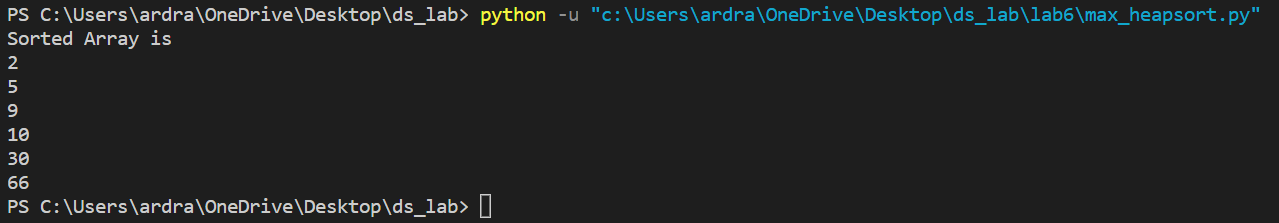
    max\_heapify(Arr, n, i)

heapSort(Arr)

print("Sorted Array is")

for i in range(n):

   print(Arr[i]),



**Min Heap Sort**

# Min Heapify

def Min\_heapify(Arr, n, i):

    smallest = i

    left = 2\*i + 1

    right = 2\*i + 2

    if left < n and Arr[smallest] > Arr[left]:

        smallest = left

    if right < n and Arr[smallest] > Arr[right

        smallest = right

    if smallest!=i:

        Arr[i], Arr[smallest] = Arr[smallest], Arr[i]

        Min\_heapify(Arr, n, smallest)

Arr = [2,66,30,5,9,10]

n = len(Arr)

def heapSort(Arr):

    for i in range(n-1, 0, -1):

        # Swap

        Arr[i], Arr[0] = Arr[0], Arr[i]

        Min\_heapify(Arr, i, 0)

for i in range(n//2 - 1, -1, -1):

    Min\_heapify(Arr, n, i)

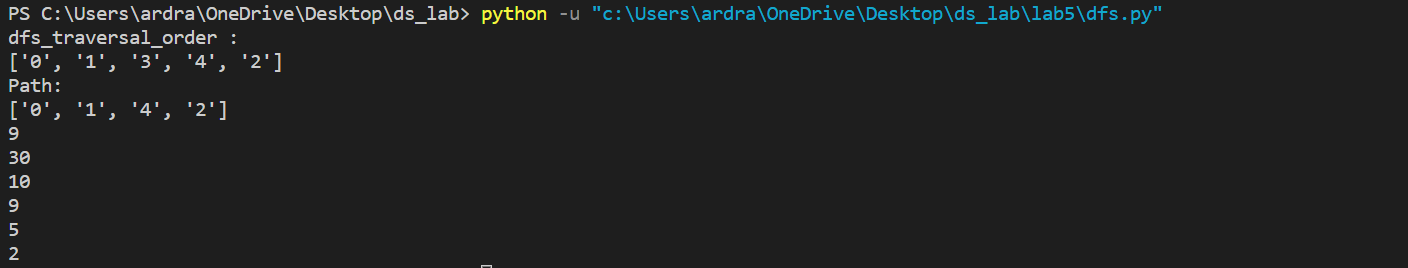
heapSort(Arr)

print("Sorted Array is")

#print("Min heap is")

for i in range(n):

   print(Arr[i]),

****