

Name : Soham Landge
Sub : Analysis of Algorithm & Reasoning Computing
Academic Year : 2022-23

Roll no. 545
Paper : I
Sem : 1

Q.1) Write a Program for Heap Sort Algorithm

```
Python program for  
implementation of heapSort  
  
# To heapify subtree  
rooted at index i. # n  
is size of heap def  
heapify(arr, n, i):  
  
    largest = i # Initialize largest as root  
  
    l = 2 * i + 1 # left = 2*i + 1  
  
    r = 2 * i + 2 # right = 2*i + 2  
  
    # See if left child of root exists and is  
    # greater than root if l < n  
    and arr[i] < arr[l]:  
  
        largest = l  
  
    # See if right child of root exists and is  
    # greater than root if r < n and  
    arr[largest] < arr[r]:  
  
        largest = r  
    # Change root, if needed  
  
    if largest != i:
```

```

        arr[i],arr[largest] = arr[largest],arr[i] # swap
        # Heapify the root. heapify(arr, n, largest)

# The main function to sort an
array of given size def

heapSort(arr):

    n =

    len(arr)

    # Build a maxheap.

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

        heapify(arr, n, i)

    # One by one extract elements

    for i in range(n-1, 0, -1):

        arr[i], arr[0] =

        arr[0], arr[i] #

        swap heapify(arr,

        i,

        0)
# Driver code to test above

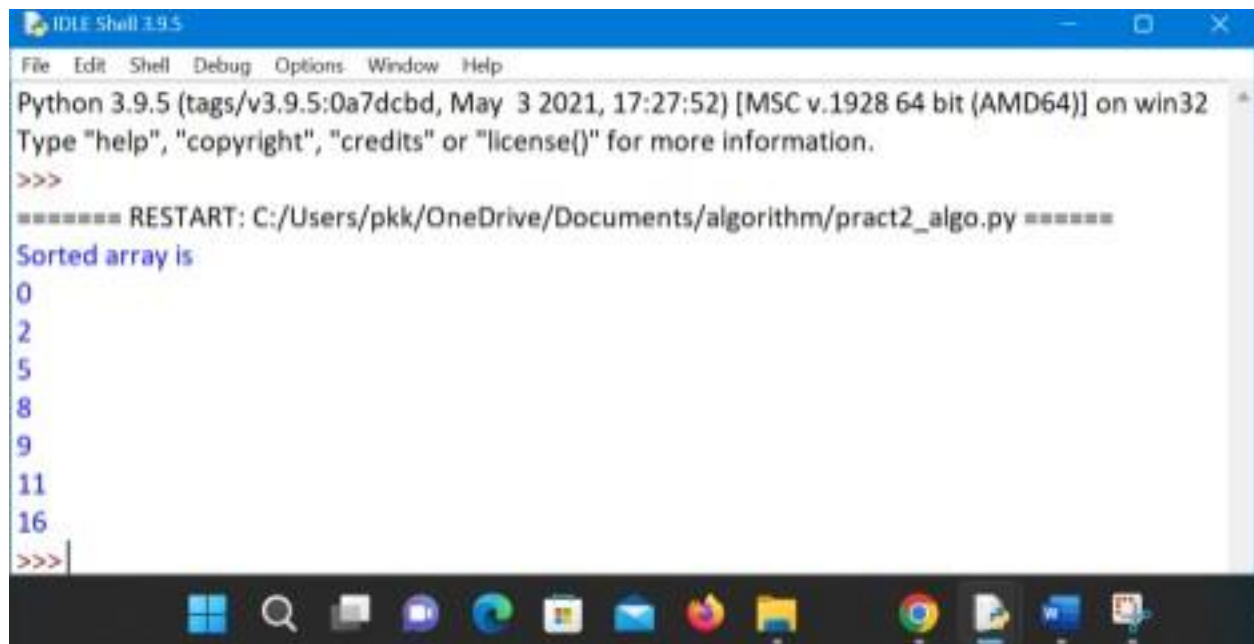
arr = [2,8,16,11,9,5,0]

heapSort(arr) n = len(arr)

print ("Sorted array
is") for i in range(n):
print ("%d" %arr[i]),

```

●Output



```
IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/pkk/OneDrive/Documents/algorithm/pract2_algo.py =====
Sorted array is
0
2
5
8
9
11
16
>>>
```

Q2) Write a Program to perform Radix Sort

Algorithm

Python program for implementation of Radix Sort

Python program for implementation of Radix Sort # A

function to do counting sort of arr[] according to

the digit represented by exp.

def countingSort(arr, exp1):

 n = len(arr)

 output = [0] * (n)

 count = [0] * (10)

 for i in range(0,

 n):

 index = (arr[i]/exp1)

 count[int((index)%10)] += 1

```

        # Change count[i] so that count[i] now contains
        actual

        # position of this digit in output
array for i in range(1,10): count[i] +=

count[i-1]

# Build the output array

i = n-1 while i>=0:

    index = (arr[i]/exp1)
    output[ count[ int((index)%10) ] - 1] = arr[i]

    count[int((index)%10)] -= 1

    i -= 1

# Copying the output array to arr[],

i = 0

for i in range(0,len(arr)):
    arr[i] = output[i]

def radixSort(arr): max1 = max(arr) exp
= 1 while max1/exp > 0:
countingSort(arr,exp) exp *= 10 arr = [
170, 45, 75, 90, 802, 24, 2, 66]
radixSort(arr)

```

```

for i in range(len(arr)):

    print(arr[i],end=" ")

```

Q3) Write a Program for Randomized Selection Algorithm

```

from random import randrange

def partition(x, pivot_index = 0):
    i = 0
    if pivot_index != 0:
        x[0],x[pivot_index] =
        x[pivot_index],x[0]
    for j in
    range(len(x)-1):
        if x[j+1] < x[0]: x[j+1],x[i+1]
        = x[i+1],x[j+1]
    i += 1
    x[0],x[i] =
    x[i],x[0]
    return x,i

```

```

def RSelect(x,k):

    if len(x) == 1:

        return x[0]

    else:

        xpart = partition(x,randrange(len(x)))

        x = xpart[0] # partitioned array j =

        xpart[1] # pivot index if j == k: return

        x[j]

        elif j > k: return

            RSelect(x[:j],k)

        else: k = k - j - 1 return

            RSelect(x[(j+1):], k)

x = [8,4,2,6,1] for i in

range(len(x)): print

(RSelect(x,i))

```

Output :-



```
IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcbbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/pkk/OneDrive/Documents/algorithm/pract1_algo.py =====
1
2
4
6
8
>>>
```

**Q4) Write a Program to Perform Bucket
Sort Algorithm**

```
# Python3 program to sort an array
# using bucket sort
```

```
def insertionSort(b):
    for i in range(1, len(b)):
        up = b[i]
```

```

        j = i - 1 while j >=0 and
        b[j] > up: b[j + 1] =
        b[j]

        j -= 1
        b[j + 1] = up

    return b def

bucketSort(x): arr = []

slot_num = 10 # 10 means 10
slots, each # slot's size is
0.1 for i in
range(slot_num):

    arr.append([]) #

    Put array elements in
    different buckets for j
    in x:

        index_b = int(slot_num * j)

        arr[index_b].append(j)

    # Sort individual buckets

    for i in range(slot_num):

        arr[i] =

        insertionSort(arr[i]) #

    concatenate the result k = 0 for i

    in range(slot_num):

        for j in range(len(arr[i])):

```



```

        x[k] = arr[i][j]

    k += 1
    return x # Driver

```

Code

```

x = [0.453, 0.573, 0.8656, 0.9064, 0.5678,
0.2568]
print("Sorted Array is")
print(bucketSort(x))

```

Output :-

```

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/pkk/OneDrive/Documents/algorithm/pract4.py =====
Sorted Array is
[0.2568, 0.453, 0.5678, 0.573, 0.8656, 0.9064]
>>>

```

Q5) Write a Program to Perform Folyd-Warshall algorithm.

```

# Python Program for Floyd Warshall Algorithm

# Number of vertices in the graph

V = 4

# Define infinity as the large enough value.
This value will be # used for vertices not
connected to each other

INF = 99999
# Solves all pair shortest path via Floyd Warshall

Algorithm def floydWarshall(graph):
    dist = map(lambda
    i : map(lambda j : j , i) , graph)
    for k in range(V):

```

```

# pick all vertices as source one by one

    for i in range(V):
# Pick all vertices as destination for the

# above picked source for j in
    range(V):

# If vertex k is on the shortest path from

# i to j, then update the value of dist[i][j]

        dist[i][j] = min(dist[i][j]

            ,dist[i][k]+ dist[k][j])

        printSolution(dist) # A utility function

to print the solution def printSolution(dist): print

"Following matrix shows the shortest distances\

between every pair of vertices"

    for i in range(V):

        for j in range(V):

            if(dist[i][j] == INF):
                print "%7s" %("INF"),

            else:

                print "%7d\t"

                %(dist[i][j]), if j == V-1:

                print ""

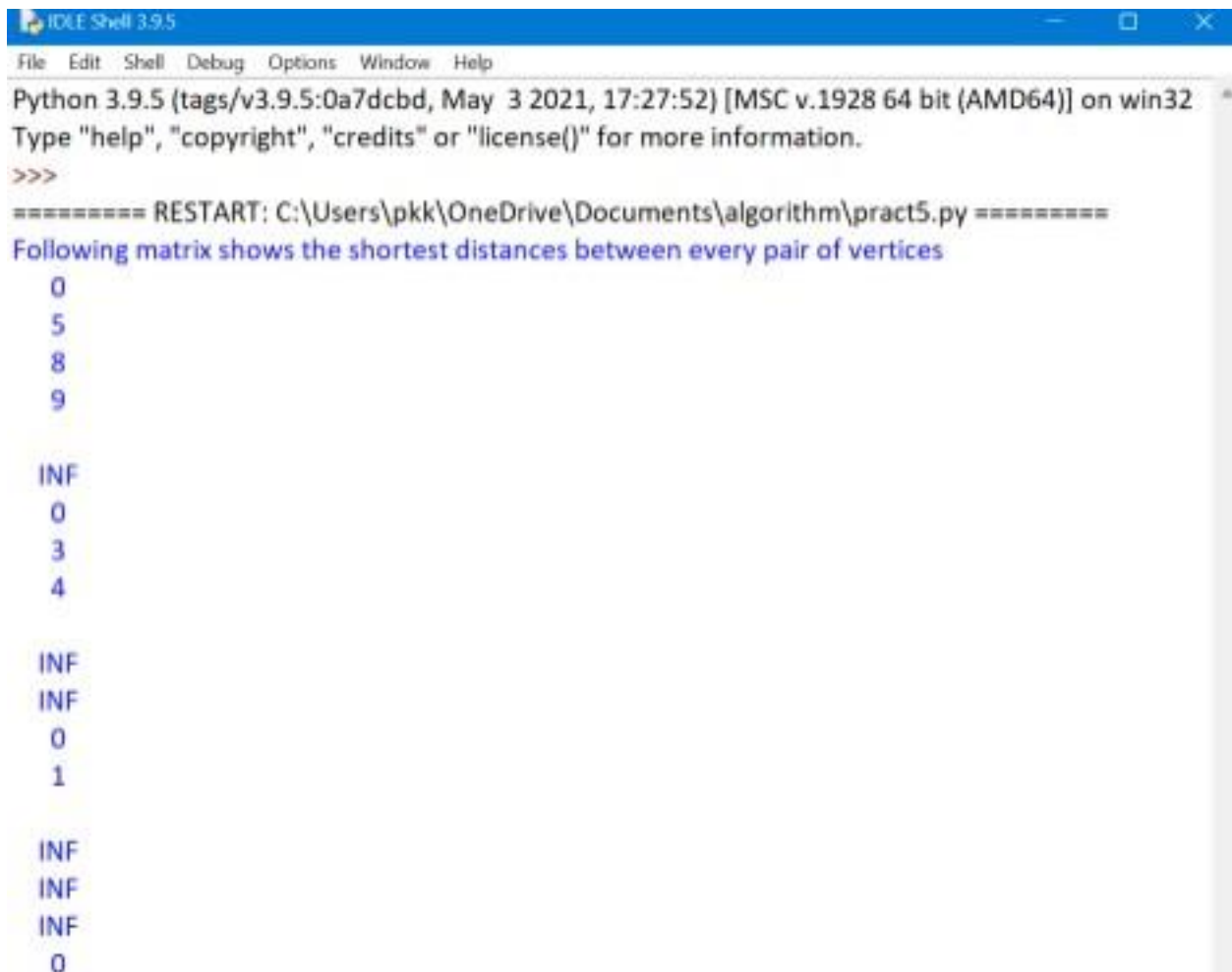
graph = [[0,5,INF,10],

        [INF,0,3,INF],

```

```
        [INF, INF, 0, 1],  
        [INF, INF, INF, 0]]  
floydWarshall(graph);
```

Output: -



```
IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcbbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\pkk\OneDrive\Documents\algorithm\pract5.py =====
Following matrix shows the shortest distances between every pair of vertices
0
5
8
9

INF
0
3
4

INF
INF
0
1

INF
INF
INF
0
```

Q6) Write a Program for Counting

Sort Algorithm in Python

```
Python program for counting sort

# The main function that sort the
given string arr[] in #
alphabetical order def
countSort(arr):

# The output character array that
will have sorted arr output = [0
```

```

        for i in range(256)]

# Create a count array to store
count of inidividul # characters
and initialize count array as 0
count = [0 for i in range(256)]

# For storing the
resulting answer since the
# string is immutable ans =

        [" " for _ in arr]

# Store count of each character

        for i in arr:

                count[ord(i)] += 1 # Change
count[i] so that count[i] now
contains actual # position of this
character in output array for i
in range(256):
        count[i] += count[i-1]

# Build the output character array

        for i in range(len(arr)):

                output[count[ord(arr[i])]-1] = arr[i]
                count[ord(arr[i])] -= 1

# Copy the output array to arr, so that arr now
# contains sorted characters

        for i in range(len(arr)):

```

```

        ans[i] = output[i]
    return

ans

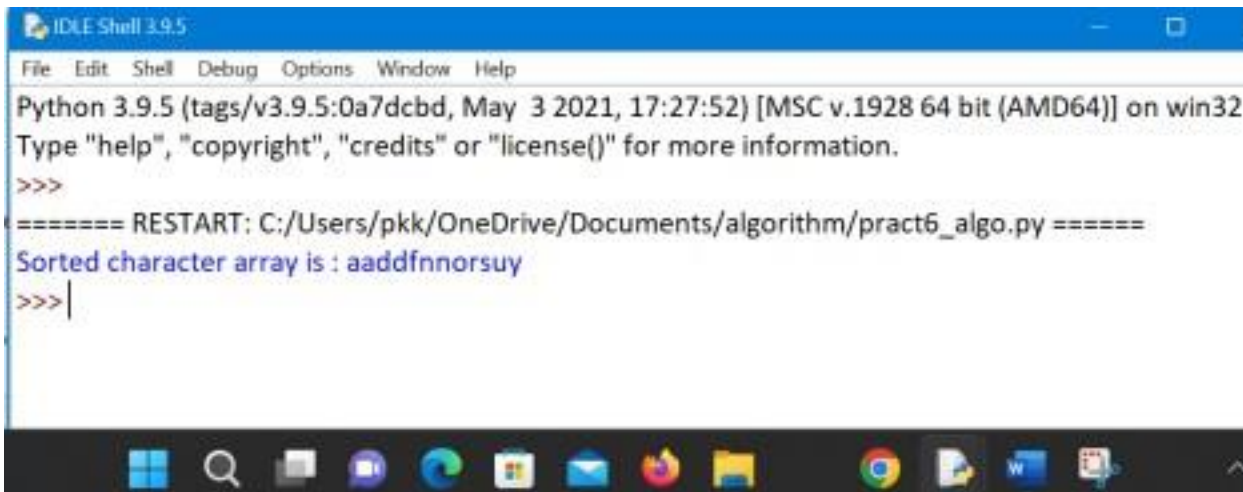
# Driver program to test above function
arr =

"Sandfoundary"
ans = countSort(arr)
print "Sorted

character array is %s" %("".join(ans))

```

Output :-



```

Python 3.9.5 (tags/v3.9.5:0a7dcbbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/pkk/OneDrive/Documents/algorithm/pract6_algo.py =====
Sorted character array is : aaddfnnorsuy
>>>

```

Q7) Write a Program for found a subset with
given sum

```

# A recursive solution for subset sum

# problem

# Returns true if there is a subset

# of set[] with sun equal to given sum

def isSubsetSum(set,n, sum) :

```

```

    # Base Cases
    if (sum == 0) :

```

```

        return True if (n
== 0 and sum != 0) :

        return False
    # If last element is greater
    than # sum, then ignore it if
    (set[n - 1] > sum) :

        return isSubsetSum(set, n - 1, sum);

        # else, check if sum can be obtained
        # by any of the following
        # (a) including the last element
        # (b) excluding the last element return
isSubsetSum(set, n-1, sum) or isSubsetSum(set, n-1, sum-set[n-
1])

    # Driver program to test above
function set = [3, 34, 4, 12, 5, 2] sum = 9 n
= len(set) if (isSubsetSum(set, n, sum) ==
True) :

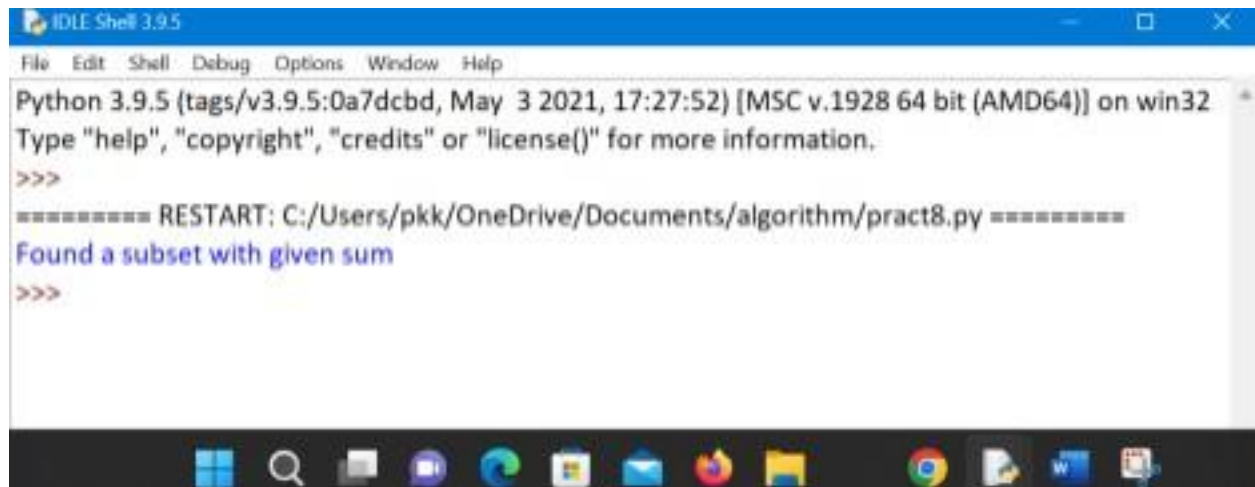
    print("Found a subset with given sum")

    else :

        print("No subset with given sum")

```

Output :-



```
IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/pkk/OneDrive/Documents/algorithm/pract8.py =====
Found a subset with given sum
>>>
```

Q8) Write a program for Set Covering Problem

```
def set_cover(universe, subsets):

    """Find a family of subsets that covers the universal set"""
    elements = set(e for s in subsets for e in s)

    # Check the subsets cover the universe

    if elements != universe:
        return None

    covered = set()

    cover = []

    # Greedily add the subsets with the
    most uncovered points while covered !=
    elements:

        subset = max(subsets, key=lambda s:
                    len(s - covered))

        cover.append(subset)
```



```

        covered |= subset
    return

cover

def main():

    universe = set(range(11, 21))

    subsets = [set([11,12,

        13, 18, 19,20]),

        set([11, 12, 13, 14,

            15]), set([14,15,

                17]), set([15, 16, 17]),

                set([16, 17, 18, 19,20])]

    cover = set_cover(universe, subsets)

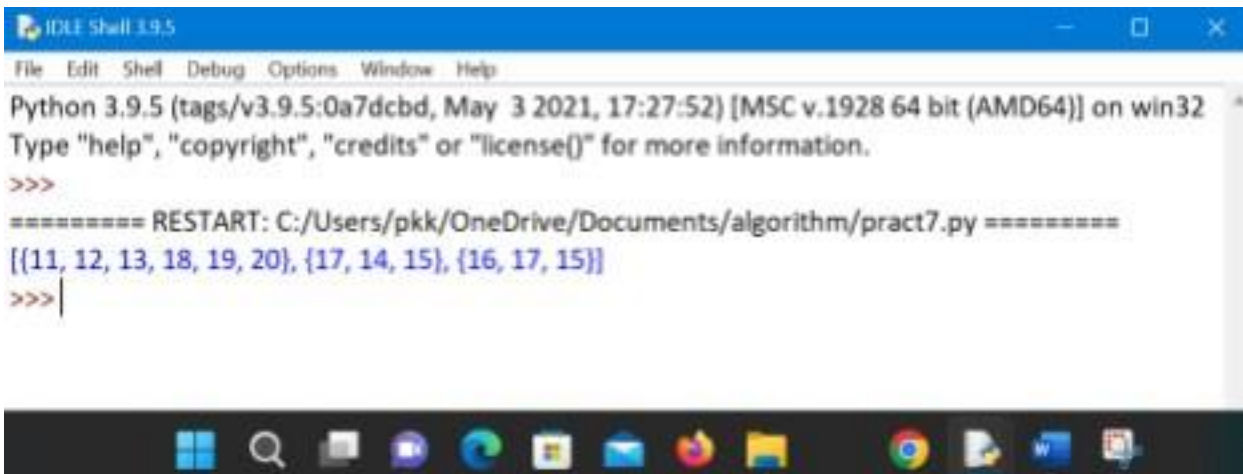
    print(cover)

if __name__ == '__main__':

    main()

```

Output :-



```

IDEE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcdbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/pkk/OneDrive/Documents/algorithm/pract7.py =====
{(11, 12, 13, 18, 19, 20), {17, 14, 15}, {16, 17, 15}}
>>>

```