LAB 5

1. Merge Two Sorted Lists

PROGRAM:

def m(a, b):

r = []

i, j = 0, 0

while i < len(a) and j < len(b):

if a[i] < b[j]:

r.append(a[i])

i += 1

else:

r.append(b[j])

j += 1

while i < len(a):

r.append(a[i])

i += 1

while j < len(b):

r.append(b[j])

j += 1

return r

a=[1,2,4]

b=[1,3,4]

print(m(a,b))

OUTPUT:



1. Merge k Sorted Lists

PROGRAM:

def m(l):

def mt(a, b):

r = []

i, j = 0, 0

while i < len(a) and j < len(b):

if a[i] < b[j]:

r.append(a[i])

i += 1

else:

r.append(b[j])

j += 1

while i < len(a):

r.append(a[i])

i += 1

while j < len(b):

r.append(b[j])

j += 1

return r

while len(l) > 1:

ml = []

for i in range(0, len(l), 2):

l1 = l[i]

l2 = l[i + 1] if (i + 1) < len(l) else []

ml.append(mt(l1, l2))

l = ml

return l[0]

l=[[1,3,2,5],[5,26,9],[8,0,7]]

print(sorted(m(l)))

OUTPUT:

1. Remove Duplicates from Sorted Array

PROGRAM:

def dupli(a):

if not a:

return 0

b=[]

for i in a:

if i not in b:

b.append(i)

return b

a=[1,1,2,2,3,3]

print(dupli(a))

OUTPUT:

1. Search in Rotated Sorted Array

PROGRAM:

def x(a, t):

l, h = 0, len(a) - 1

while l <= h:

m = (l + h) // 2

if a[m] == t:

return m

if a[l] <= a[m]:

if a[l] <= t < a[m]:

h = m - 1

else:

l = m + 1

else:

if a[m] < t <= a[h]:

l = m + 1

else:

h = m - 1

return -1

a=[4,5,6,7,0,1,2]

t=0

print(x(a,t))

OUTPUT:

1. Find First and Last Position of Element in Sorted Array

PROGRAM:

def f(a,k):

n=len(a)

c=[]

for i in range(n):

if a[i]==k:

c.append(a[i])

else :

continue

if i not in a:

return [-1,-1]

elif len(c)==1:

return [c[0],-1]

a=[1,2,3,4,5,6,7,8]

k=3

print(f(a,k))

OUTPUT:

1. Sort Colors

PROGRAM:

def col(ar):

if len(ar)>1:

mid = len(ar)//2

l=col(ar[:mid])

r=col(ar[mid:])

i=j=k=0

a=[0]\*len(ar)

while i<len(l) and j<len(r):

if l[i]<r[j]:

a[k]=l[i]

i+=1

else:

a[k]=r[j]

j+=1

k+=1

while i<len(l):

a[k]=l[i]

i+=1

k+=1

while j<len(r):

a[k]=r[j]

j+=1

k+=1

return a

else:

return ar

ar=[0,2,1,2,0,2,0,1]

print(col(ar))

OUTPUT:

1. Remove Duplicates from Sorted List

PROGRAM:

class L:

def \_\_init\_\_(s, v=0, n=None):

s.v = v

s.n = n

def r(h):

c = h

while c and c.n:

if c.v == c.n.v:

c.n = c.n.n

else:

c = c.n

return h

def p(node):

while node:

print(node.v, end=" ")

node = node.n

h2 = L(1, L(1, L(2, L(3, L(3)))))

x =r(h2)

p(x)

OUTPUT:

1. Merge Sorted Array

PROGRAM:

def m(a1, m, a2, n):

i, j, k = m-1, n-1, m+n-1

while i >= 0 and j >= 0:

if a1[i] > a2[j]:

a1[k] = a1[i]

i -= 1

else:

a1[k] = a2[j]

j -= 1

k -= 1

while j >= 0:

a1[k] = a2[j]

j -= 1

k -= 1

a1 = [1,2,3,0,0,0]

m1 = 3

a2 = [2,5,6]

n1 = 3

m(a1, m1, a2, n1)

print(a1)

OUTPUT:

1. Convert Sorted Array to Binary Search Tree

PROGRAM:

class T:

def \_\_init\_\_(self, v=0, l=None, r=None):

self.v = v

self.l = l

self.r = r

def c(a):

if not a:

return None

m = len(a) // 2

r = T(a[m])

r.l = c(a[:m])

r.r = c(a[m+1:])

return r

def print\_tree(node):

if node:

print(node.v, end=" ")

print\_tree(node.l)

print\_tree(node.r)

a1 = [-10, -3, 0, 5, 9]

t1 = c(a1)

print\_tree(t1)

print()

OUTPUT:

1. Insertion Sort List

PROGRAM:

class L:

def \_\_init\_\_(self, v=0, n=None):

self.v = v

self.n = n

def i(h):

if not h or not h.n:

return h

d = L(0)

c = h

while c:

t = c.n

p = d

while p.n and p.n.v < c.v:

p = p.n

c.n = p.n

p.n = c

c = t

return d.n

def print\_list(node):

while node:

print(node.v, end=" ")

node = node.n

print()

h1 = L(4, L(2, L(1, L(3))))

s1 = i(h1)

print\_list(s1)

OUTPUT:

1. Sort Characters By Frequency

PROGRAM:

from collections import Counter

def s(f):

c = Counter(f)

return ''.join([k \* v for k, v in c.most\_common()])

s1 = "tree"

o1 = s(s1)

print(o1)

OUTPUT:

1. Max Chunks To Make Sorted

PROGRAM:

def m(a):

mx, c = 0, 0

for i, n in enumerate(a):

mx = max(mx, n)

if mx == i:

c += 1

return c

a1 = [4, 3, 2, 1, 0]

o1 = m(a1)

print(o1)

OUTPUT:

1. Intersection of Three Sorted Arrays

PROGRAM:

def i(a1, a2, a3):

s1, s2, s3 = set(a1), set(a2), set(a3)

return sorted(s1 & s2 & s3)

a1\_1 = [1, 2, 3, 4, 5]

a2\_1 = [1, 2, 5, 7, 9]

a3\_1 = [1, 3, 4, 5, 8]

o1 = i(a1\_1, a2\_1, a3\_1)

print(o1)

OUTPUT:

1. Sort the Matrix Diagonally

PROGRAM:

from collections import defaultdict

import heapq

def s(m):

d = defaultdict(list)

for i in range(len(m)):

for j in range(len(m[0])):

heapq.heappush(d[i-j], m[i][j])

for i in range(len(m)):

for j in range(len(m[0])):

m[i][j] = heapq.heappop(d[i-j])

return m

m1 = [[3, 3, 1, 1],[2, 2, 1, 2],[1, 1, 1, 2]]

o1 =s(m1)

print(o1)

OUTPUT: