LAB 16

#1 Sort

def s(a):

if len(a)<=1:

return a

n=len(a)

for i in range(n):

for j in range(n):

if a[j]>=a[i]:

x=a[i]

a[i]=a[j]

a[j]=x

return a

n=int(input("Enter range: "))

a=[]

for i in range(n):

y=int(input("Enter value: "))

a.append(y)

#2 Selection sort

def sel(a,n):

for i in range(n-1):

min=i

for j in range(i+1,n):

if a[j]<a[min]:

min=j

if min!=i:

temp=a[min]

a[min]=a[i]

a[i]=temp

return a

a=[]

n=int(input("Enter range: "))

for i in range(n):

a.append(int(input("Enter value: ")))

print(sel(a,n))

#3 Bubble sort

def bub(a,n):

for i in range(n):

for j in range(n):

if a[j]>a[i]:

x=a[i]

a[i]=a[j]

a[j]=x

return a

a=[]

n=int(input("Enter range: "))

for i in range(n):

a.append(int(input("Enter value: ")))

print(bub(a,n))

#4 Insertion sort

def ins(a,n):

for i in range(1,n):

key=a[i]

j=i-1

while(j>=0 and a[j]>key):

a[j+1]=a[j]

j=j-1

a[j+1]=key

return a

a=[]

n=int(input("Enter range: "))

for i in range(n):

a.append(int(input("Enter value: ")))

print(ins(a,n))

#5

def k(a,n):

t=0

for i in range(1,100):

if i not in a:

t+=1

if t==n:

return i

a=[1,3,5,7,9,]

n=2

print(k(a,n))

#5

def f(a, k):

def co(a, mid):

c = 0

for num in a:

if num <= mid:

c += 1

else:

break

return mid - c + 1

left, right = 0, a[-1]

while left < right:

mid = (left + right) // 2

if co(a, mid) < k:

left = mid + 1

else:

right = mid

return left + k

a = [2, 3, 4, 7, 11]

k = 5

print(f(a, k))

#6 peak element

def p(a,n):

m=0

x=0

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

if a[i]>a[i+1] and a[i]>a[i-1]:

if m<a[i]:

m=a[i]

x=i

return a[i-1]

a=[]

n=int(input("Enter range: "))

for i in range(n):

a.append(int(input("Enter value: ")))

print(p(a,n))

#7 string matching

def s(t,p):

n=len(t)

m=len(p)

for i in range(n-m+1):

j=0

while j<m:

if t[i+j]==p[j]:

j=j+1

if j==m:

return i

else:

break

return None

t=input("Enter Haystack: ")

p=input("Enter needle: ")

print(s(t, p))

#8 string matching in list

def s(t):

n = len(t)

x = []

for i in range(n):

for j in range(n):

if i != j:

m = len(t[i])

o = len(t[j])

if m < o:

for k in range(o - m + 1):

l = 0

while l < m and t[j][k + l] == t[i][l]:

l += 1

if l == m:

x.append(t[i])

break

return x if x else None

t = []

n = int(input("Enter the number of words: "))

for i in range(n):

t.append(input("Enter word: "))

print(s(t))

#9 closest pair

from math import sqrt

def cl(a,n):

min=float('inf')

for i in range(n-1):

for j in range(i+1,n):

x=sqrt((a[j][0]-a[i][0])\*2 + (a[j][1]-a[i][1])\*2)

if min>=x:

min=x

k=i

p=j

else:

continue

return (min,[a[k],a[p]])

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

n=len(a)

print(cl(a,n))

#10 convex hull

def c(o, a, b):

return (a[0] - o[0]) \* (b[1] - o[1]) - (a[1] - o[1]) \* (b[0] - o[0])

def r(o, a, b):

return c(o, a, b) < 0

def bf(pts):

n = len(pts)

if n < 3:

return pts

h = []

for i in range(n):

for j in range(i + 1, n):

rs = True

ls = True

for k in range(n):

if k == i or k == j:

continue

if not r(pts[i], pts[j], pts[k]):

rs = False

if r(pts[i], pts[j], pts[k]):

ls = False

if rs or ls:

h.append(pts[i])

h.append(pts[j])

uh = []

for p in h:

if p not in uh:

uh.append(p)

my = uh[0][1]

mp = uh[0]

for p in uh:

if p[1] < my or (p[1] == my and p[0] < mp[0]):

my = p[1]

mp = p

def pa(p0, p1):

return (p1[1] - p0[1]) \* (p1[0] - p0[0])

sp = sorted(uh, key=lambda p: pa(mp, p))

return sp

pts = [(10, 0), (11, 5), (5, 3), (9, 3.5), (15, 3), (12.5, 7), (6, 6.5), (7.5, 4.5)]

ch = bf(pts)

print(ch)

#11 convex hull

def c(o, a, b):

return (a[0] - o[0]) \* (b[1] - o[1]) - (a[1] - o[1]) \* (b[0] - o[0])

def r(o, a, b):

return c(o, a, b) < 0

def bf(pts):

n = len(pts)

if n < 3:

return pts

h = []

for i in range(n):

for j in range(i + 1, n):

rs = True

ls = True

for k in range(n):

if k == i or k == j:

continue

if not r(pts[i], pts[j], pts[k]):

rs = False

if r(pts[i], pts[j], pts[k]):

ls = False

if rs or ls:

h.append(pts[i])

h.append(pts[j])

uh = []

for p in h:

if p not in uh:

uh.append(p)

my = uh[0][1]

mp = uh[0]

for p in uh:

if p[1] < my or (p[1] == my and p[0] < mp[0]):

my = p[1]

mp = p

def pa(p0, p1):

return (p1[1] - p0[1]) \* (p1[0] - p0[0])

sp = sorted(uh, key=lambda p: pa(mp, p))

return sp

pts = [(1, 1), (4, 6), (8, 1), (0, 0), (3, 3)]

ch = bf(pts)

print(ch)

#12 TSP

import math

def d(c1, c2):

return math.sqrt((c1[0] - c2[0]) \*\* 2 + (c1[1] - c2[1]) \*\* 2)

def permute(lst):

if len(lst) == 0:

return []

if len(lst) == 1:

return [lst]

l = []

for i in range(len(lst)):

m = lst[i]

rem\_lst = lst[:i] + lst[i+1:]

for p in permute(rem\_lst):

l.append([m] + p)

return l

def t(c):

s = c[0]

o = c[1:]

m = float('inf')

b = []

for p in permute(o):

cp = [s] + p + [s]

cd = 0

for i in range(len(cp) - 1):

cd += d(cp[i], cp[i + 1])

if cd < m:

m = cd

b = cp

return m, b

tc1 = [(1, 2), (4, 5), (7, 1), (3, 6)]

md1, bp1 = t(tc1)

print(f"Shortest Distance: {md1}")

print(f"Shortest Path: {bp1}")

#13

import math

def tc(a, cm):

t = 0

for i in range(len(a)):

t += cm[i][a[i]]

return t

def p(lst):

if len(lst) == 0:

return [[]]

perms = []

for i in range(len(lst)):

m = lst[i]

rem\_lst = lst[:i] + lst[i+1:]

for perm in p(rem\_lst):

perms.append([m] + perm)

return perms

def ap(cm):

n = len(cm)

t = list(range(n))

mc = math.inf

ba = []

all\_perms = p(t)

for perm in all\_perms:

current\_cost = tc(perm, cm)

if current\_cost < mc:

mc = current\_cost

ba = perm

return mc, ba

cost\_matrix\_1 = [[3, 10, 7],[8, 5, 12],[4, 6, 9]]

min\_cost\_1, best\_assignment\_1 = ap(cost\_matrix\_1)

print(f"Optimal Assignment: {[(w + 1, t + 1) for w, t in enumerate(best\_assignment\_1)]}")

print(f"Total Cost: {min\_cost\_1}")

#14 Kanpsack

w=[2,3,1]

cost=[4,5,3]

n=len(w)

c=[]

weight=4

a=-1

b=-1

for i in range(n):

if w[i]<=weight:

if w[i]>a or (w[i]==a and cost[i]>b):

a=w[i]

b=cost[i]

c.append(i)

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

x=w[i]

y=cost[i]

c=[]

c.append(i)

for j in range(i+1,n):

x+=w[j]

y+=cost[j]

c.append(j)

if x<=weight:

if x>a or (x==a and y>b):

a=x

b=y

print("Combination: ",c,"Total weight: ",a,"Maximum profit: ",b)"""