LAB 13

N-Queens

def p(b, q):

n = len(b)

bwq = [["." for \_ in range(n)] for \_ in range(n)]

for i in range(len(q)):

r, c = q[i]

bwq[r][c] = f"q{i + 1}"

for r in bwq:

print(" ".join(r))

print("\n")

def s(b, r, c, n):

for i in range(r):

if b[i][c]:

return False

i, j = r, c

while i >= 0 and j >= 0:

if b[i][j]:

return False

i -= 1

j -= 1

i, j = r, c

while i >= 0 and j < n:

if b[i][j]:

return False

i -= 1

j += 1

return True

def u(b, r, n, q):

if r >= n:

p(b, q)

return

for c in range(n):

if s(b, r, c, n):

b[r][c] = 1

q.append((r, c))

u(b, r + 1, n, q)

b[r][c] = 0

q.pop()

def n\_queens(n):

b = [[0 for \_ in range(n)] for \_ in range(n)]

q = []

u(b, 0, n, q)

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

n\_queens(n)

#Sum of subsets

def ss(a, t):

r=[]

p=[]

def b(i,c):

if c>t:

return

if c==t:

r.append(p[:])

return

for j in range(i,len(a)):

p.append(a[j])

b(j+1, c+a[j])

p.pop()

b(0, 0)

return r

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

a=[]

for i in range(n):

a.append(int(input("Enter the values: ")))

t=int(input("enter target sum: "))

subsets = ss(a,t)

print("Subsets with sum",t,"are: ",subsets)

#LONGEST PALLIDROME SUBSEQUENCE

def lps(s):

n = len(s)

m = [[0 for \_ in range(n)] for \_ in range(n)]

for i in range(n):

m[i][i] = 1

for cl in range(2, n + 1):

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

j = i + cl - 1

if s[i] == s[j] and cl == 2:

m[i][j] = 2

elif s[i] == s[j]:

m[i][j] = m[i + 1][j - 1] + 2

else:

m[i][j] = max(m[i][j - 1], m[i + 1][j])

def pmt(m):

for row in m:

print(row)

print()

print("Final table:")

pmt(m)

seq = [""] \* m[0][n - 1]

start = 0

end = m[0][n - 1] - 1

i, j = 0, n - 1

while i <= j:

if s[i] == s[j]:

seq[start] = s[i]

seq[end] = s[i]

start += 1

end -= 1

i += 1

j -= 1

elif m[i][j - 1] > m[i + 1][j]:

j -= 1

else:

i += 1

return ''.join(seq)

s = "teeth"

result\_length = lps(s)

result\_subsequence = lps(s)

print("The length of the longest palindromic subsequence is:", len(result\_subsequence))

print("The longest palindromic subsequence is:", result\_subsequence)

#graph colouring

def s(v, c, i, g):

for j in range(len(g)):

if g[v][j] == 1 and c[j] == i:

return False

return True

def u(g,m,c,v,col):

if v == len(g):

print(' '.join(col[c[k]] for k in range(len(c))))

return

for i in range(m):

if s(v, c, i, g):

c[v]=i

u(g,m,c,v + 1,col)

c[v]=-1

g = [[0, 1, 0, 1],[1, 0, 1, 0],[0, 1, 0, 1],[1, 0, 1, 0]]

m = 3

col = ['R', 'G', 'B']

c = [-1] \* len(g)

u(g,m,c,0,col)