#1

def d(s, n, t):

m = [[0] \* (t + 1) for \_ in range(n + 1)]

m[0][0] = 1

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

for j in range(1, t + 1):

for k in range(1, min(s, j) + 1):

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

return m[n][t]

def p(s, n, t):

r = d(s, n, t)

print(f"Number of ways to reach sum {t}: {r}")

p(6, 2, 7)

#2

def a(n, a1, a2, t1, t2, e1, e2, x1, x2):

f1, f2 = [0] \* n, [0] \* n

f1[0], f2[0] = e1 + a1[0], e2 + a2[0]

for i in range(1, n):

f1[i] = min(f1[i-1] + a1[i], f2[i-1] + t2[i-1] + a1[i])

f2[i] = min(f2[i-1] + a2[i], f1[i-1] + t1[i-1] + a2[i])

return min(f1[-1] + x1, f2[-1] + x2)

Example usage:

n = 4

a1 = [4, 5, 3, 2]

a2 = [2, 10, 1, 4]

t1 = [7, 4, 5]

t2 = [9, 2, 8]

e1, e2 = 10, 12

x1, x2 = 18, 7

print(a(n, a1, a2, t1, t2, e1, e2, x1, x2))

#3

def t(s, l1, l2, l3, tr):

n = len(l1)

f1, f2, f3 = [0]\*n, [0]\*n, [0]\*n

f1[0], f2[0], f3[0] = l1[0], l2[0], l3[0]

for i in range(1, n):

f1[i] = min(f1[i-1], f2[i-1]+tr[1][0], f3[i-1]+tr[2][0]) + l1[i]

f2[i] = min(f2[i-1], f1[i-1]+tr[0][1], f3[i-1]+tr[2][1]) + l2[i]

f3[i] = min(f3[i-1], f1[i-1]+tr[0][2], f2[i-1]+tr[1][2]) + l3[i]

return min(f1[-1], f2[-1], f3[-1])

Example usage:

s = 3

l1 = [5, 9, 3]

l2 = [6, 8, 4]

l3 = [7, 6, 5]

tr = [[0, 2, 3], [2, 0, 4], [3, 4, 0]]

print(t(s, l1, l2, l3, tr))

#4

def f(m):

n = len(m)

for k in range(n):

for i in range(n):

for j in range(n):

m[i][j] = min(m[i][j], m[i][k] + m[k][j])

return sum(min(row) for row in m)

Example usage:

m = [

[0, 10, 15, 20],

[10, 0, 35, 25],

[15, 35, 0, 30],

[20, 25, 30, 0]

]

print(f(m))

#5

import itertools

def t(d):

n = len(d)

c = list(range(n))

return min(sum(d[c[i-1]][c[i]] for i in range(n)) for c in itertools.permutations(c))

Example usage:

d = [

[0, 10, 15, 20, 25],

[10, 0, 35, 25, 30],

[15, 35, 0, 30, 20],

[20, 25, 30, 0, 15],

[25, 30, 20, 15, 0]

]

print(t(d))

#6

def l(s):

n = len(s)

dp = [[False]\*n for \_ in range(n)]

m, r = 1, s[0]

for i in range(n):

dp[i][i] = True

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

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

j = i+l-1

if l == 2:

dp[i][j] = s[i] == s[j]

else:

dp[i][j] = dp[i+1][j-1] and s[i] == s[j]

if dp[i][j] and l > m:

m, r = l, s[i:j+1]

return r

print(l("babad"))

print(l("cbbd"))

#7

def l(s):

m, start, c = 0, 0, {}

for i, ch in enumerate(s):

if ch in c and c[ch] >= start:

start = c[ch] + 1

else:

m = max(m, i - start + 1)

c[ch] = i

return m

print(l("abcabcbb"))

print(l("bbbbb"))

print(l("pwwkew"))

#8

def w(s, d):

n = len(s)

dp = [False] \* (n + 1)

dp[0] = True

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

for j in range(i):

if dp[j] and s[j:i] in d:

dp[i] = True

break

return dp[n]

print(w("leetcode", ["leet", "code"]))

print(w("applepenapple", ["apple", "pen"]))

print(w("catsandog", ["cats", "dog", "sand", "and", "cat"]))

#9

def w(s, d):

n = len(s)

dp = [False] \* (n + 1)

dp[0] = True

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

for j in range(i):

if dp[j] and s[j:i] in d:

dp[i] = True

break

return "Yes" if dp[n] else "No"

d = {"i", "like", "sam", "sung", "samsung", "mobile", "ice",

"cream", "icecream", "man", "go", "mango"}

print(w("ilike", d))

print(w("ilikesamsung", d))

#10

def j(w, m):

r, l, c = [], [], 0

for i in w:

if c + len(l) + len(i) > m:

for j in range(m - c):

l[j % (len(l) - 1 or 1)] += ' '

r.append(''.join(l))

l, c = [], 0

l.append(i)

c += len(i)

return r + [' '.join(l).ljust(m)]

w = ["This", "is", "an", "example", "of", "text", "justification."]

print(j(w, 16))

#11

class W:

def \_\_init\_\_(self, w):

self.t = {}

for i, word in enumerate(w):

for j in range(len(word) + 1):

for k in range(len(word) + 1):

self.t[(word[:j], word[k:])] = i

def f(self, p, s):

return self.t.get((p, s), -1)

w = W(["apple"])

print(w.f("a", "e"))

#12

def f(g):

n = len(g)

for k in range(n):

for i in range(n):

for j in range(n):

g[i][j] = min(g[i][j], g[i][k] + g[k][j])

return g

g = [

[0, 3, 8, float('inf')],

[float('inf'), 0, 4, 1],

[2, float('inf'), 0, float('inf')],

[float('inf'), 6, -5, 0]

]

print(f(g))

#13

def f(g):

n = len(g)

for k in range(n):

for i in range(n):

for j in range(n):

g[i][j] = min(g[i][j], g[i][k] + g[k][j])

return g

def p(g, s, e):

return g[s][e]

g = [

[0, 1, 5, float('inf'), 8, float('inf')],

[1, 0, 2, 1, float('inf'), float('inf')],

[5, 2, 0, float('inf'), 3, float('inf')],

[float('inf'), 1, float('inf'), 0, 1, 6],

[8, float('inf'), 3, 1, 0, 2],

[float('inf'), float('inf'), float('inf'), 6, 2, 0]

]

print("Before failure:", p(f(g), 0, 5))

g[1][3] = g[3][1] = float('inf')

print("After failure:", p(f(g), 0, 5))

#14

def f(g):

n = len(g)

for k in range(n):

for i in range(n):

for j in range(n):

g[i][j] = min(g[i][j], g[i][k] + g[k][j])

return g

g = [

[0, 2, float('inf'), 6, float('inf')],

[2, 0, 3, float('inf'), float('inf')],

[float('inf'), 3, 0, 1, 7],

[6, float('inf'), 1, 0, float('inf')],

[float('inf'), float('inf'), 7, float('inf'), 0]

]

print(f(g))

#15

def o(k, f):

n = len(k)

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

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

for i in range(n):

c[i][i] = f[i]

r[i][i] = i

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

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

j = i + l - 1

c[i][j] = float('inf')

s = sum(f[i:j+1])

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

t = (c[i][k-1] if k > i else 0) + (c[k+1][j] if k < j else 0) + s

if t < c[i][j]:

c[i][j] = t

r[i][j] = k

return c[0][n-1], c, r

k = [10, 12, 16, 21]

f = [4, 2, 6, 3]

cost, cost\_table, root\_table = o(k, f)

print("Cost:", cost)

print("Cost Table:", cost\_table)

print("Root Table:", root\_table)