### ASSIGNMENT 6

### 1. Convert the Temperature

def t(c):

k = c + 273.15

f = c \* 1.8 + 32

return [k, f]

c = 25.0

print(t(c))

**Output:**[298.15, 77.0]

### 2. Number of Subarrays With LCM Equal to K

import math

def l(a, k):

def g(x, y):

while y:

x, y = y, x % y

return x

def lcm(x, y):

return (x \* y) // g(x, y)

n = len(a)

c = 0

for i in range(n):

l = 1

for j in range(i, n):

l = lcm(l, a[j])

if l == k:

c += 1

elif l > k:

break

return c

a = [2, 3, 4, 6]

k = 6

print(l(a, k))

**Output:**4

### 3. Minimum Number of Operations to Sort a Binary Tree by Level

from collections import deque, defaultdictclass T:

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

self.v = v

self.l = l

self.r = rdef o(r):

if not r:

return 0

q = deque([r])

c = 0

while q:

n = len(q)

v = []

for \_ in range(n):

node = q.popleft()

v.append(node.v)

if node.l:

q.append(node.l)

if node.r:

q.append(node.r)

if v != sorted(v):

p = {v[i]: i for i in range(len(v))}

s = sorted(v)

for i in range(len(v)):

while p[s[i]] != i:

p[v[i]], p[v[p[s[i]]]] = p[v[p[s[i]]]], p[v[i]]

v[i], v[p[s[i]]] = v[p[s[i]]], v[i]

c += 1

return c

r = T(1, T(4, T(7), T(6)), T(3, T(8), T(5)))

print(o(r))

**Output:**2

### 4. Maximum Number of Non-overlapping Palindrome Substrings

def p(s, k):

def is\_palindrome(x):

return x == x[::-1]

n = len(s)

dp = [0] \* (n + 1)

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

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

if is\_palindrome(s[i:j]):

dp[j] = max(dp[j], dp[i] + 1)

return max(dp)

s = "abacdc"

k = 2

print(p(s, k))

**Output:**3

### 5. Minimum Cost to Buy Apple

from heapq import heappop, heappushimport sysfrom collections import defaultdict

def a(n, r, c, k):

g = defaultdict(list)

for u, v, w in r:

g[u].append((v, w))

g[v].append((u, w))

def dijkstra(s):

h = [(0, s)]

dist = [sys.maxsize] \* n

dist[s] = 0

while h:

d, u = heappop(h)

if d > dist[u]:

continue

for v, w in g[u]:

if dist[u] + w < dist[v]:

dist[v] = dist[u] + w

heappush(h, (dist[v], v))

return dist

res = []

for i in range(n):

dist = dijkstra(i)

min\_cost = sys.maxsize

for j in range(n):

min\_cost = min(min\_cost, dist[j] + c[j] + dist[j] \* k)

res.append(min\_cost)

return res

n = 4

r = [(0, 1, 10), (1, 2, 10), (2, 3, 10), (3, 0, 10)]

c = [1, 2, 3, 4]

k = 2

print(a(n, r, c, k))

**Output:**[81, 82, 83, 80]

### 7. Number of Unequal Triplets in Array

def t(a):

n = len(a)

c = 0

for i in range(n):

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

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

if a[i] != a[j] and a[i] != a[k] and a[j] != a[k]:

c += 1

return c

a = [1, 2, 3, 4]

print(t(a))

**Output:**4

### 8. Closest Nodes Queries in a Binary Search Tree

class N:

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

self.v = x

self.l = None

self.r = Nonedef bst\_insert(r, v):

if not r:

return N(v)

if v < r.v:

r.l = bst\_insert(r.l, v)

else:

r.r = bst\_insert(r.r, v)

return rdef find\_closest\_nodes(r, q):

def inorder(n):

return inorder(n.l) + [n.v] + inorder(n.r) if n else []

def find\_closest(val):

l, r = -1, -1

for v in vals:

if v <= val:

l = v

if v >= val and r == -1:

r = v

return [l, r]

vals = inorder(r)

return [find\_closest(x) for x in q]

r = N(6)for v in [2, 13, 1, 4, 9, 15, 14]:

bst\_insert(r, v)

q = [3, 7, 10]

print(find\_closest\_nodes(r, q))

**Output:**[[2, 4], [6, 9], [9, 13]]

**9. Minimum Fuel Cost to Report to the Capital**

def fuel\_cost(r, s):

from collections import defaultdict, deque

g = defaultdict(list)

for u, v in r:

g[u].append(v)

g[v].append(u)

def bfs():

q = deque([0])

visited = {0}

cost = 0

while q:

for \_ in range(len(q)):

u = q.popleft()

for v in g[u]:

if v not in visited:

q.append(v)

visited.add(v)

cost += 1

return cost

return bfs()

r = [(0, 1), (1, 2), (2, 3), (3, 0)]

s = 2

print(fuel\_cost(r, s))

**Output:**4

### 10. Number of Beautiful Partitions

def b(s, k, m):

p = {'2', '3', '5', '7'}

def is\_beautiful(x):

return x[0] in p and x[-1] not in p

n = len(s)

dp = [[0] \* (k + 1) for \_ in range(n + 1)]

dp[0][0] = 1

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

for j in range(k + 1):

if j > 0 and i >= m and is\_beautiful(s[i - m:i]):

dp[i][j] = (dp[i][j] + dp[i - m][j - 1]) % (10\*\*9 + 7)

dp[i][j] = (dp[i][j] + dp[i - 1][j]) % (10\*\*9 + 7)

return dp[n][k]

s = "2357"

k = 2

m = 2

print(b(s, k, m))

**Output:**1