#1

def f(p):

p.sort()

n = len(p) // 3

r = 0

for i in range(n):

r += p[-(2 \* (i + 1))]

return r

p = [2, 4, 1, 2, 7, 8]

print(f(p))

#2

def g(c, t):

c.sort()

m, a = 1, 0

i = 0

while m <= t:

if i < len(c) and c[i] <= m:

m += c[i]

i += 1

else:

m += m

a += 1

return a

c = [1, 2, 4, 8, 10]

t = 19

print(g(c, t))

#3

def h(j, k):

def can\_finish(m):

w = [0] \* k

return dfs(0, m, w)

def dfs(i, m, w):

if i == len(j):

return True

for l in range(k):

if w[l] + j[i] <= m:

w[l] += j[i]

if dfs(i + 1, m, w):

return True

w[l] -= j[i]

if w[l] == 0:

break

return False

l, r = max(j), sum(j)

while l < r:

m = (l + r) // 2

if can\_finish(m):

r = m

else:

l = m + 1

return l

j = [3, 2, 3]

k = 3

print(h(j, k))

#4

def i(s, e, p):

n = len(s)

t = sorted(zip(s, e, p), key=lambda x: x[1])

dp = [0] \* n

dp[0] = t[0][2]

for i in range(1, n):

c = t[i][2]

k = -1

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

if t[j][1] <= t[i][0]:

k = j

break

if k != -1:

dp[i] = max(dp[i - 1], dp[k] + c)

else:

dp[i] = max(dp[i - 1], c)

return dp[n - 1]

s = [1, 2, 3, 3]

e = [3, 4, 5, 6]

p = [50, 10, 40, 70]

print(i(s, e, p))

#5

def j(n, g, src):

d = [float('inf')] \* n

d[src] = 0

q = {src}

while q:

u = min(q, key=lambda x: d[x])

q.remove(u)

for v in range(n):

if g[u][v] != float('inf') and d[u] + g[u][v] < d[v]:

d[v] = d[u] + g[u][v]

q.add(v)

return d

n = 5

g = [[0, 10, 3, float('inf'), float('inf')],

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

[float('inf'), 4, 0, 8, 2],

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

[float('inf'), float('inf'), float('inf'), 9, 0]]

src = 0

print(j(n, g, src))

#6

def k(n, e, src, tgt):

d = {i: float('inf') for i in range(n)}

d[src] = 0

q = {src}

while q:

u = min(q, key=lambda x: d[x])

q.remove(u)

for v, w in [(v, w) for u, v, w in e if u == u]:

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

d[v] = d[u] + w

q.add(v)

return d[tgt]

n = 6

e = [(0, 1, 7), (0, 2, 9), (0, 5, 14),

(1, 2, 10), (1, 3, 15),

(2, 3, 11), (2, 5, 2),

(3, 4, 6), (4, 5, 9)]

src = 0

tgt = 4

print(k(n, e, src, tgt))

#7

def l(c, f):

q = [(f[i], [(c[i], '')]) for i in range(len(c))]

while len(q) > 1:

q.sort()

a = q.pop(0)

b = q.pop(0)

merged = [(char, '0' + code) for char, code in a[1]] + [(char, '1' + code) for char, code in b[1]]

q.append((a[0] + b[0], merged))

result = q[0][1]

return sorted(result, key=lambda x: len(x[1]))

c = ['a', 'b', 'c', 'd']

f = [5, 9, 12, 13]

print(l(c, f))

#8

class Node:

def \_\_init\_\_(self, char, freq):

self.char = char

self.freq = freq

self.left = None

self.right = None

def decode\_huffman(root, encoded\_string):

decoded\_string = ""

current = root

for bit in encoded\_string:

if bit == '0':

current = current.left

else:

current = current.right

if current.left is None and current.right is None:

decoded\_string += current.char

current = root

return decoded\_string

root = Node(None, None)

root.left = Node('a', 5)

root.right = Node(None, None)

root.right.left = Node('b', 9)

root.right.right = Node(None, None)

root.right.right.left = Node('c', 12)

root.right.right.right = Node('d', 13)

encoded\_string = '1101100111110'

decoded\_message = decode\_huffman(root, encoded\_string)

print(decoded\_message)

#9

def n(w, mc):

w.sort(reverse=True)

c = 0

for i in w:

if c + i <= mc:

c += i

return c

w = [10, 20, 30, 40, 50]

mc = 60

print(n(w, mc))

#10

def o(w, mc):

w.sort(reverse=True)

c, cnt = 0, 1

for i in w:

if c + i <= mc:

c += i

else:

cnt += 1

c = i

return cnt

w = [5, 10, 15, 20, 25, 30, 35]

mc = 50

print(o(w, mc))

#11

def p(n, m, e):

e.sort(key=lambda x: x[2])

p = list(range(n))

def find(u):

if p[u] != u:

p[u] = find(p[u])

return p[u]

def union(u, v):

p[find(u)] = find(v)

mst, tw = [], 0

for u, v, w in e:

if find(u) != find(v):

union(u, v)

mst.append((u, v, w))

tw += w

return mst, tw

n = 4

m = 5

e = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]

print(p(n, m, e))

#12

def q(n, m, e, gmst):

def find\_mst():

e.sort(key=lambda x: x[2])

p = list(range(n))

def find(u):

if p[u] != u:

p[u] = find(p[u])

return p[u]

def union(u, v):

p[find(u)] = find(v)

mst = []

for u, v, w in e:

if find(u) != find(v):

union(u, v)

mst.append((u, v, w))

return mst

mst1 = find\_mst()

mst2 = find\_mst()

return gmst == mst1 and gmst != mst2, mst2

n = 4

m = 5

e = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]

gmst = [(2, 3, 4), (0, 3, 5), (0, 1, 10)]

print(q(n, m, e, gmst))