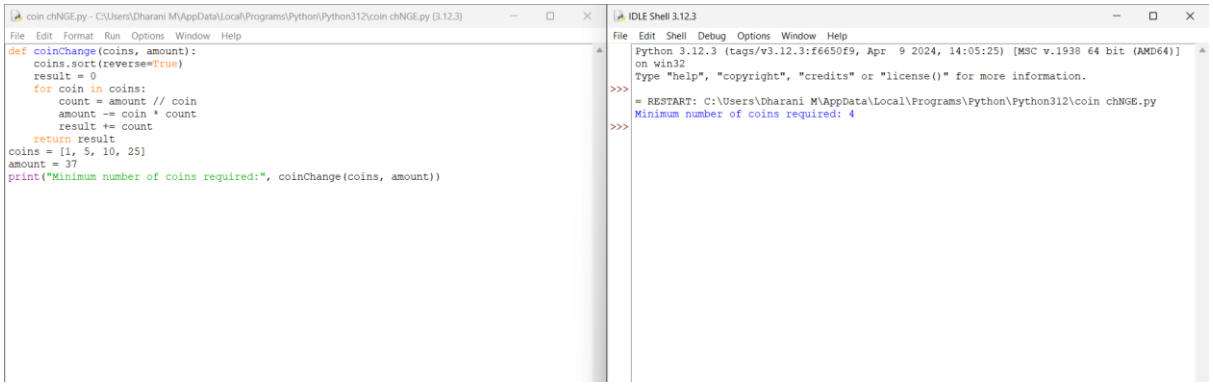
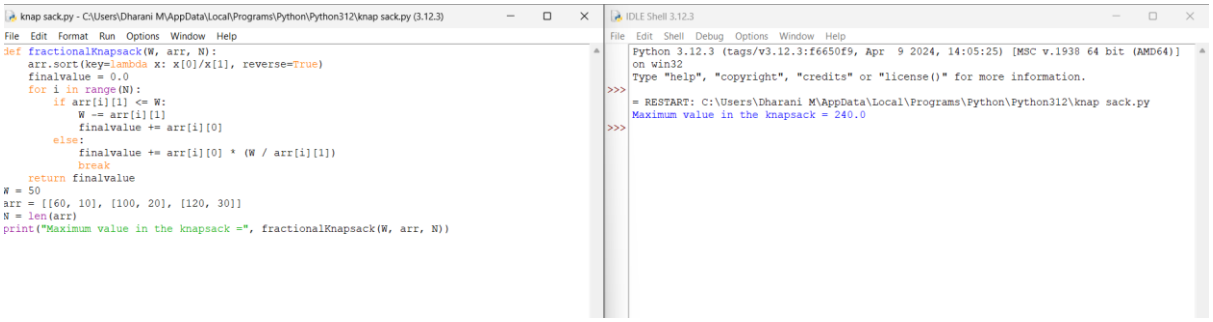
1. Coin Change Problem



```
def coinChange(coins, amount):
    coins.sort(reverse=True)
    result = 0
    for coin in coins:
        count = amount // coin
        amount -= coin * count
        result += count
    return result
coins = [1, 5, 10, 25]
amount = 37
print("Minimum number of coins required:", coinChange(coins, amount))
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\coin chNGE.py
Minimum number of coins required: 4
>>>
```

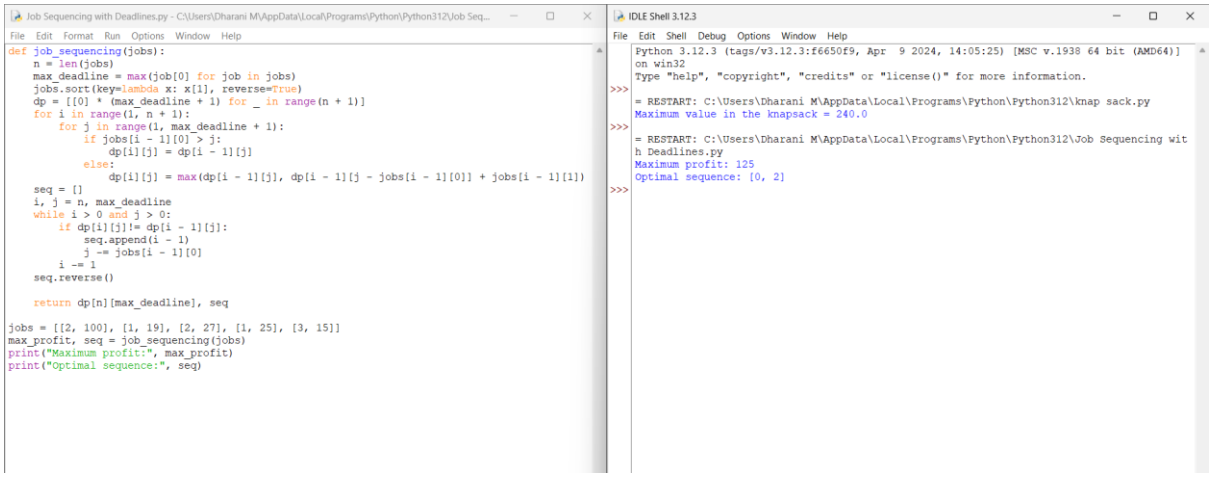
2. Knapsack Problem



```
def fractionalKnapsack(W, arr, N):
    arr.sort(key=lambda x: x[0]/x[1], reverse=True)
    finalvalue = 0.0
    for i in range(N):
        if arr[i][1] <= W:
            W -= arr[i][1]
            finalvalue += arr[i][0]
        else:
            finalvalue += arr[i][0] * (W / arr[i][1])
            break
    return finalvalue
W = 50
arr = [[60, 10], [100, 20], [120, 30]]
N = len(arr)
print("Maximum value in the knapsack =", fractionalKnapsack(W, arr, N))
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\knap sack.py
Maximum value in the knapsack = 240.0
>>>
```

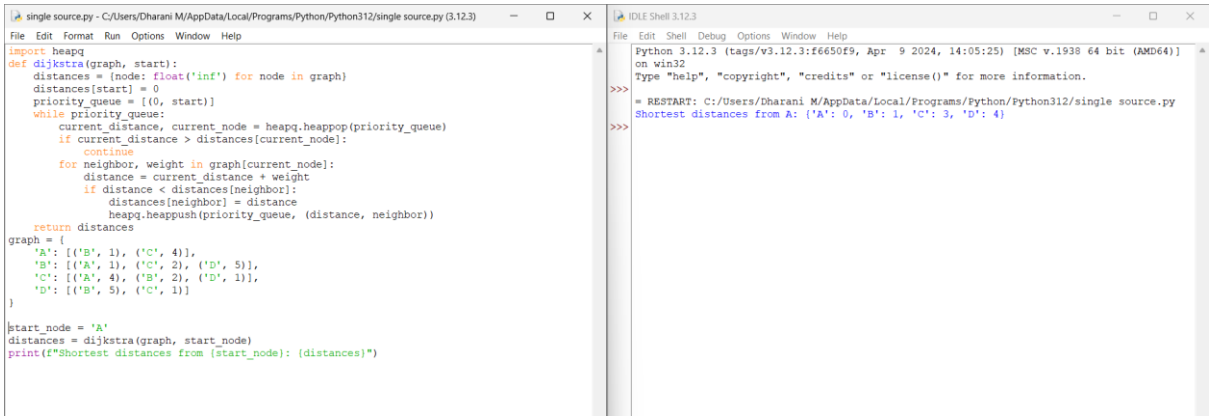
3. Job Sequencing with Deadlines



```
def job_sequencing(jobs):
    n = len(jobs)
    max_deadline = max(job[0] for job in jobs)
    jobs.sort(key=lambda x: x[1], reverse=True)
    dp = [[0] * (max_deadline + 1) for _ in range(n + 1)]
    for i in range(1, n + 1):
        for j in range(1, max_deadline + 1):
            if jobs[i - 1][0] > j:
                dp[i][j] = dp[i - 1][j]
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - jobs[i - 1][0]] + jobs[i - 1][1])
    seq = []
    i, j = n, max_deadline
    while i > 0 and j > 0:
        if dp[i][j]!= dp[i - 1][j]:
            seq.append(i - 1)
            j -= jobs[i - 1][0]
        i -= 1
    seq.reverse()

    return dp[n][max_deadline], seq

jobs = [[2, 100], [1, 19], [2, 27], [1, 25], [3, 15]]
max_profit, seq = job_sequencing(jobs)
print("Maximum profit:", max_profit)
print("Optimal sequence:", seq)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\knap sack.py
Maximum value in the knapsack = 240.0
>>>
= RESTART: C:\Users\Dharani M\AppData\Local\Programs\Python\Python312\Job Sequencing wit
h Deadlines.py
Maximum profit: 125
Optimal sequence: [0, 2]
>>>
```

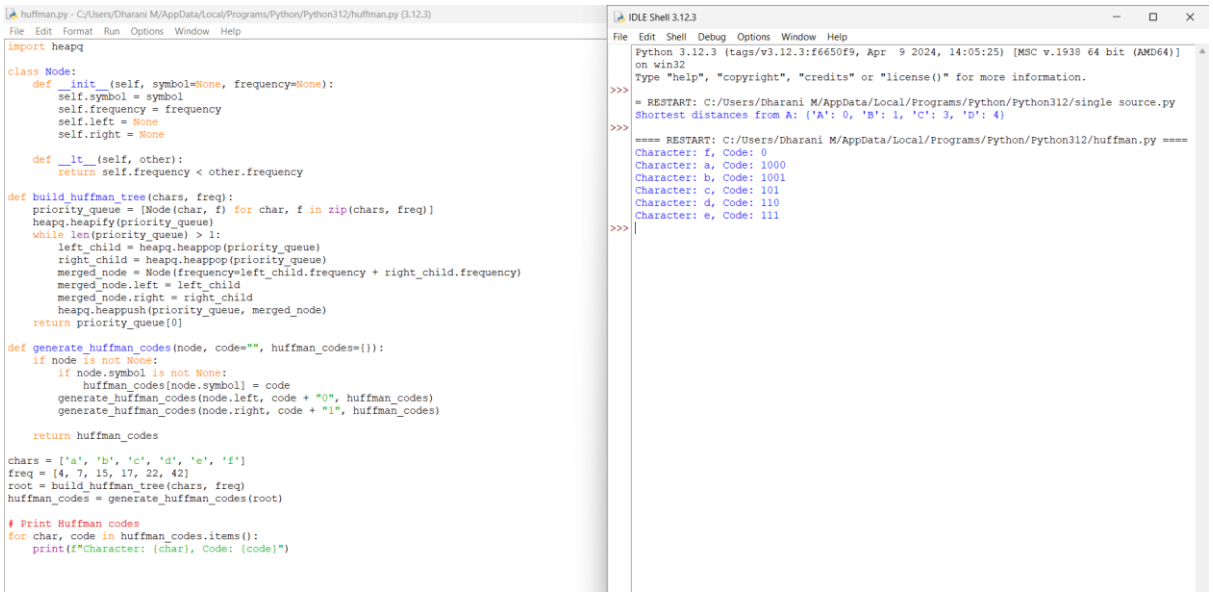
4. Single Source Shortest Paths: Dijkstra's Algorithm



```
import heapq
def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    priority_queue = [(0, start)]
    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)
        if current_distance > distances[current_node]:
            continue
        for neighbor, weight in graph[current_node]:
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(priority_queue, (distance, neighbor))
    return distances
graph = {
    'A': [('B', 1), ('C', 4)],
    'B': [('A', 1), ('C', 2), ('D', 5)],
    'C': [('A', 4), ('B', 2), ('D', 1)],
    'D': [('B', 5), ('C', 1)]
}

start_node = 'A'
distances = dijkstra(graph, start_node)
print(f"Shortest distances from {start_node}: {distances}")
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/single source.py
Shortest distances from A: {'A': 0, 'B': 1, 'C': 3, 'D': 4}
>>>
```

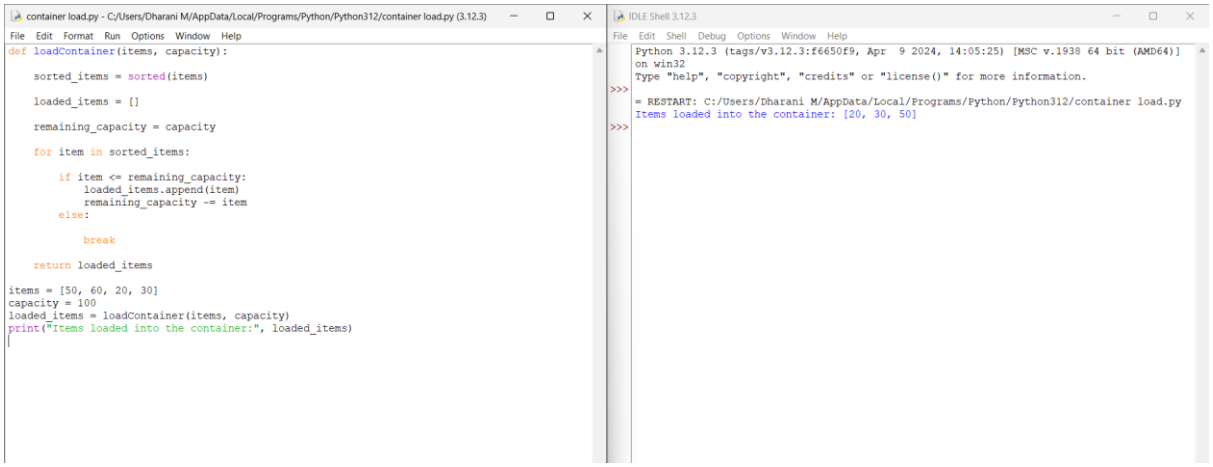
5. Optimal Tree Problem: Huffman Trees and Codes



```
import heapq

class Node:
    def __init__(self, symbol=None, frequency=None):
        self.symbol = symbol
        self.frequency = frequency
        self.left = None
        self.right = None

    def __lt__(self, other):
        return self.frequency < other.frequency

def build_huffman_tree(chars, freq):
    priority_queue = [Node(char, f) for char, f in zip(chars, freq)]
    heapq.heapify(priority_queue)
    while len(priority_queue) > 1:
        left_child = heapq.heappop(priority_queue)
        right_child = heapq.heappop(priority_queue)
        merged_node = Node(frequency=left_child.frequency + right_child.frequency)
        merged_node.left = left_child
        merged_node.right = right_child
        heapq.heappush(priority_queue, merged_node)
    return priority_queue[0]

def generate_huffman_codes(node, code="", huffman_codes={}):
    if node is not None:
        if node.symbol is not None:
            huffman_codes[node.symbol] = code
        generate_huffman_codes(node.left, code + "0", huffman_codes)
        generate_huffman_codes(node.right, code + "1", huffman_codes)

    return huffman_codes

chars = ['a', 'b', 'c', 'd', 'e', 'f']
freq = [4, 7, 15, 17, 22, 42]
root = build_huffman_tree(chars, freq)
huffman_codes = generate_huffman_codes(root)

# Print Huffman codes
for char, code in huffman_codes.items():
    print(f"Character: {char}, Code: {code}")
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/single source.py
Shortest distances from A: {'A': 0, 'B': 1, 'C': 3, 'D': 4}
>>>
==== RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/huffman.py ====
Character: f, Code: 0
Character: a, Code: 1000
Character: b, Code: 1001
Character: c, Code: 101
Character: d, Code: 110
Character: e, Code: 111
>>>
```

6. Container Loading



```
def loadContainer(items, capacity):

    sorted_items = sorted(items)

    loaded_items = []

    remaining_capacity = capacity

    for item in sorted_items:

        if item <= remaining_capacity:
            loaded_items.append(item)
            remaining_capacity -= item
        else:

            break

    return loaded_items

items = [50, 60, 20, 30]
capacity = 100
loaded_items = loadContainer(items, capacity)
print("Items loaded into the container:", loaded_items)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/container load.py
Items loaded into the container: [20, 30, 50]
>>>
```

Minimum Spanning Tree

7. Kruskal's Algorithms

krushkals.py - C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/krushkals.py (3.12.3)

File Edit Format Run Options Window Help

```
class DisjointSet:
    def __init__(self, n):
        self.parent = list(range(n))
    def find(self, u):
        if self.parent[u]!= u:
            self.parent[u] = self.find(self.parent[u])
        return self.parent[u]
    def union(self, u, v):
        root_u = self.find(u)
        root_v = self.find(v)
        if root_u!= root_v:
            self.parent[root_v] = root_u
def kruskal(n, edges):
    edges.sort(key=lambda edge: edge[2])
    ds = DisjointSet(n)
    mst = []
    for u, v, weight in edges:
        if ds.find(u)!= ds.find(v):
            ds.union(u, v)
            mst.append((u, v, weight))
    return mst
edges=[(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]
n = 4
mst = kruskal(n, edges)
print("Edges in MST:", mst)
```

IDLE Shell 3.12.3

File Edit Shell Debug Options Window Help

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/krushkals.py
Edges in MST: [(2, 3, 4), (0, 3, 5), (0, 1, 10)]
>>>
```

8. Prims Algorithm

prims algorithm.py - C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/prims algorithm.py (3.1...

File Edit Format Run Options Window Help

```
def min_key(vertices, key, mst_set):
    min_val = float('inf')
    min_index = -1
    for v in range(vertices):
        if key[v] < min_val and not mst_set[v]:
            min_val = key[v]
            min_index = v
    return min_index

def prim_mst(graph):
    vertices = len(graph)
    parent = [-1] * vertices
    key = [float('inf')] * vertices
    key[0] = 0
    mst_set = [False] * vertices
    for _ in range(vertices):
        u = min_key(vertices, key, mst_set)
        mst_set[u] = True
        print(f"Added edge: {parent[u]} - {u} Key: {key[u]}")
        for v in range(vertices):
            if graph[u][v] > 0 and not mst_set[v] and key[v] > graph[u][v]:
                parent[v] = u
                key[v] = graph[u][v]

# Example graph
graph = [[0, 2, 0, 6, 0],
         [2, 0, 3, 8, 5],
         [0, 3, 0, 0, 7],
         [6, 8, 0, 0, 9],
         [0, 5, 7, 9, 0]]

prim_mst(graph)
```

IDLE Shell 3.12.3

File Edit Shell Debug Options Window Help

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/prims algorithm.py
Added edge: -1 - 0 Key: 0
Added edge: 0 - 1 Key: 2
Added edge: 1 - 2 Key: 3
Added edge: 1 - 4 Key: 5
Added edge: 0 - 3 Key: 6
>>>
```

9. Boruvka's Algorithm

bourukas algorithm.py - C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/bourukas algorithm....

File Edit Format Run Options Window Help

```
def boruvka(graph):
    parent = {}
    cheapest = {}
    for i in range(len(graph)):
        parent[i] = i
        cheapest[i] = -1
    num_trees = len(graph)
    while num_trees > 1:
        for i in range(len(graph)):
            cheapest[i] = -1
        for i in range(len(graph)):
            for j in range(len(graph[i])):
                if parent[i] != parent[j]:
                    if cheapest[parent[i]] == -1 or graph[i][cheapest[parent[i]]] > graph[i]
                        cheapest[parent[i]] = j
        for i in range(len(graph)):
            if cheapest[i] != -1:
                if parent[i] != parent[cheapest[i]]:
                    print(f"Edge {i} - {cheapest[i]}")
                    num_trees -= 1
                    parent[parent[i]] = parent[cheapest[i]]
    return parent
graph = [
    [0, 2, 0, 6, 0],
    [2, 0, 3, 8, 5],
    [0, 3, 0, 0, 7],
    [6, 8, 0, 0, 9],
    [0, 5, 7, 9, 0]
]
boruvka(graph)
```

IDLE Shell 3.12.3

File Edit Shell Debug Options Window Help

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/bourukas algorithm
.py
Edge 0 - 2
Edge 1 - 0
Edge 3 - 2
Edge 4 - 0
>>>
```