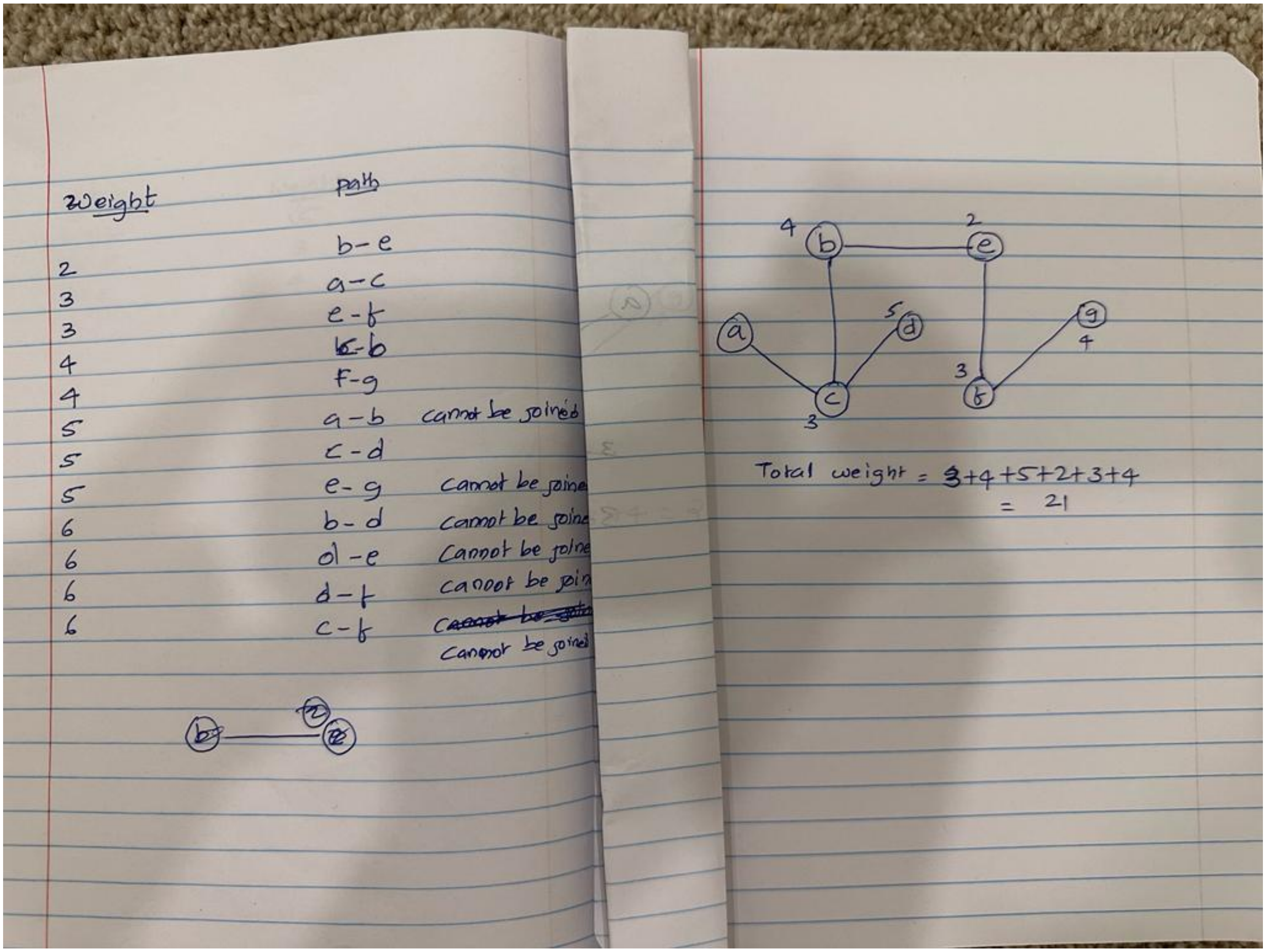
Hw5q2

Kruskal approach



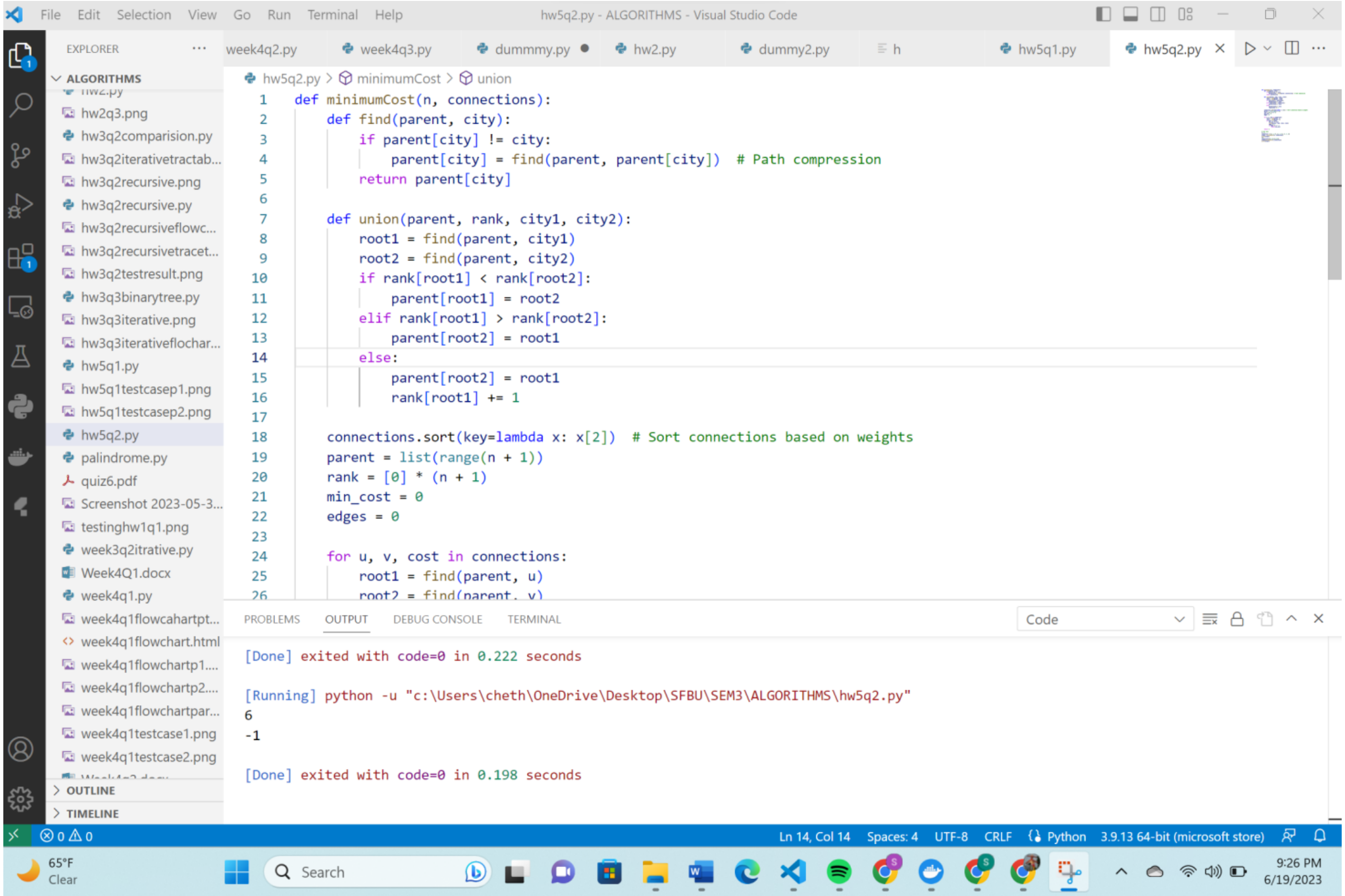
weight	path	
2	b-e	
3	a-c	
3	e-f	
4	~~b~~-b	
4	f-g	
5	a-b	cannot be joined
5	c-d	
5	e-g	cannot be joine
6	b-d	cannot be joine
6	d-e	cannot be joine
6	d-f	cannot be join
6	c-f	~~cannot be join~~ cannot be joined

Total weight = 3+4+5+2+3+4
= 21

Chatgpt python code

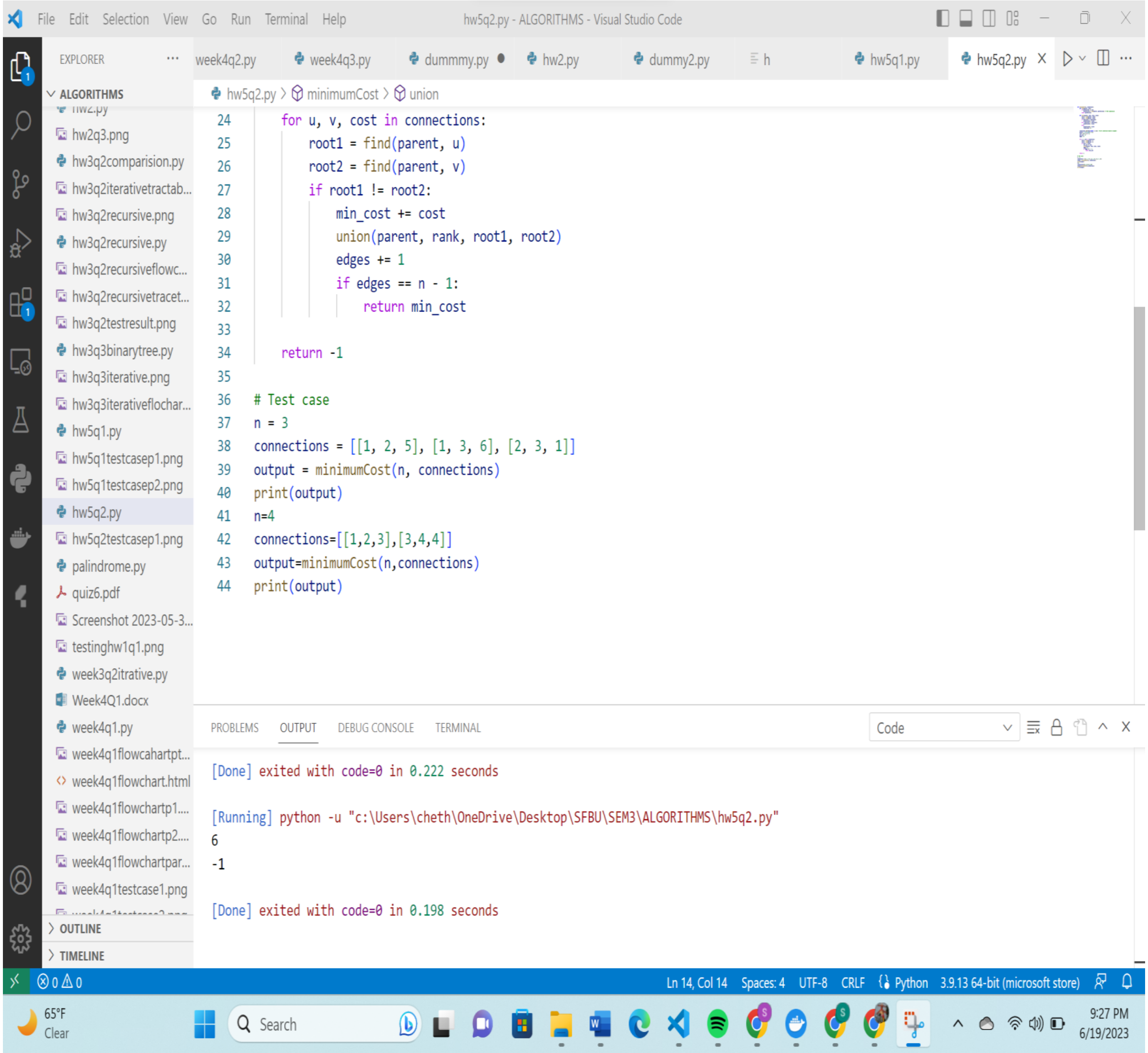
```python
def minimumCost(n, connections):
    def find(parent, city):
        if parent[city] != city:
            parent[city] = find(parent, parent[city])  # Path compression
        return parent[city]

    def union(parent, rank, city1, city2):
        root1 = find(parent, city1)
        root2 = find(parent, city2)
        if rank[root1] < rank[root2]:
            parent[root1] = root2
        elif rank[root1] > rank[root2]:
            parent[root2] = root1
        else:
            parent[root2] = root1
            rank[root1] += 1

    connections.sort(key=lambda x: x[2])  # Sort connections based on weights
    parent = list(range(n + 1))
    rank = [0] * (n + 1)
    min_cost = 0
    edges = 0

    for u, v, cost in connections:
        root1 = find(parent, u)
        root2 = find(parent, v)
        if root1 != root2:
            min_cost += cost
            union(parent, rank, root1, root2)
            edges += 1
            if edges == n - 1:
                return min_cost

    return -1

# Test case
n = 3
connections = [[1, 2, 5], [1, 3, 6], [2, 3, 1]]
output = minimumCost(n, connections)
print(output)
```

testcases



```python
def minimumCost(n, connections):
    def find(parent, city):
        if parent[city] != city:
            parent[city] = find(parent, parent[city])  # Path compression
        return parent[city]

    def union(parent, rank, city1, city2):
        root1 = find(parent, city1)
        root2 = find(parent, city2)
        if rank[root1] < rank[root2]:
            parent[root1] = root2
        elif rank[root1] > rank[root2]:
            parent[root2] = root1
        else:
            parent[root2] = root1
            rank[root1] += 1

    connections.sort(key=lambda x: x[2])  # Sort connections based on weights
    parent = list(range(n + 1))
    rank = [0] * (n + 1)
    min_cost = 0
    edges = 0

    for u, v, cost in connections:
        root1 = find(parent, u)
        root2 = find(parent, v)
```

```
[Done] exited with code=0 in 0.222 seconds

[Running] python -u "c:\Users\cheth\OneDrive\Desktop\SFBU\SEM3\ALGORITHMS\hw5q2.py"
6
-1

[Done] exited with code=0 in 0.198 seconds
```



```python
    for u, v, cost in connections:
        root1 = find(parent, u)
        root2 = find(parent, v)
        if root1 != root2:
            min_cost += cost
            union(parent, rank, root1, root2)
            edges += 1
            if edges == n - 1:
                return min_cost

    return -1

# Test case
n = 3
connections = [[1, 2, 5], [1, 3, 6], [2, 3, 1]]
output = minimumCost(n, connections)
print(output)
n=4
connections=[[1,2,3],[3,4,4]]
output=minimumCost(n,connections)
print(output)
```

```
[Done] exited with code=0 in 0.222 seconds

[Running] python -u "c:\Users\cheth\OneDrive\Desktop\SFBU\SEM3\ALGORITHMS\hw5q2.py"
6
-1

[Done] exited with code=0 in 0.198 seconds
```