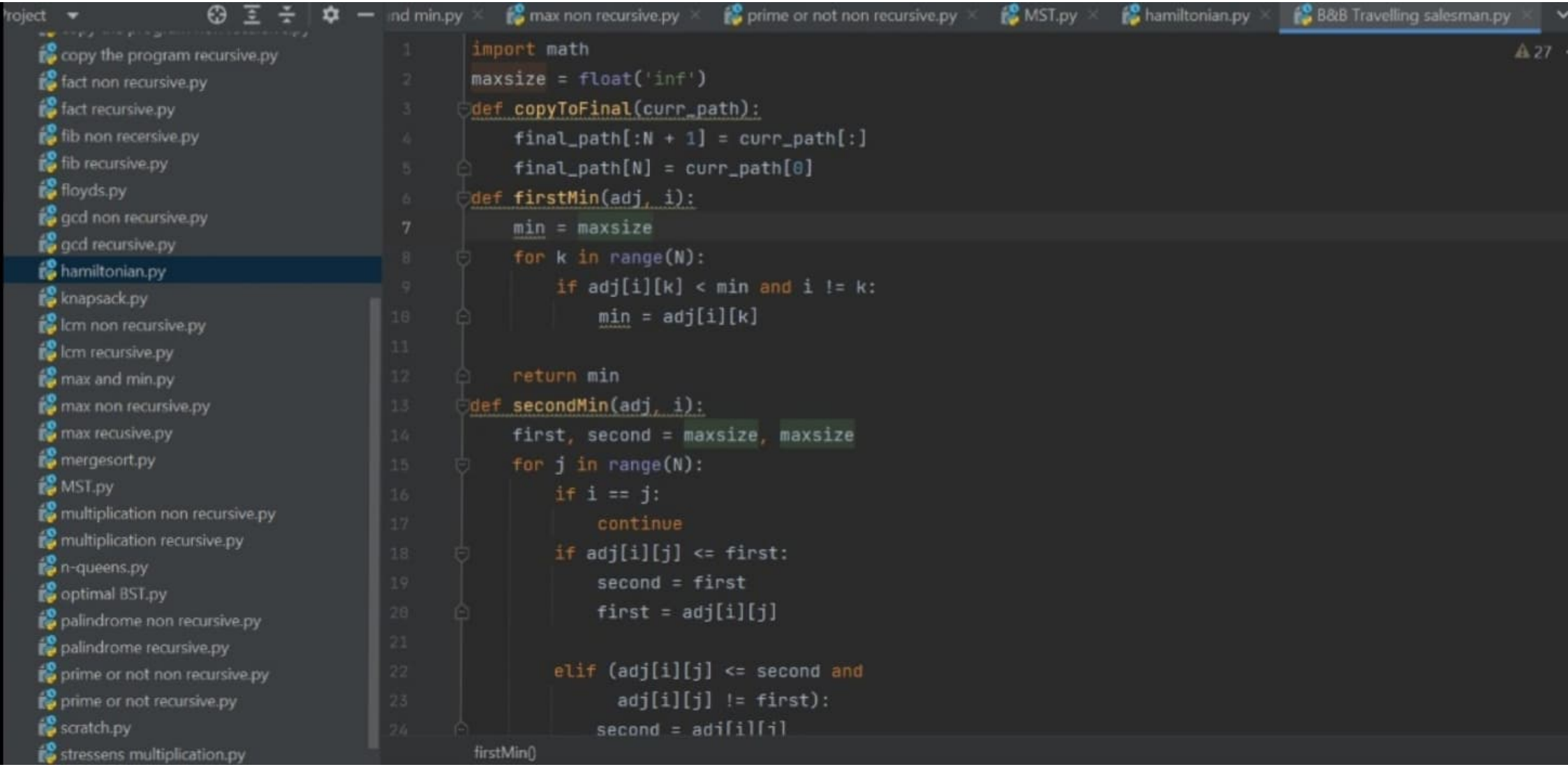
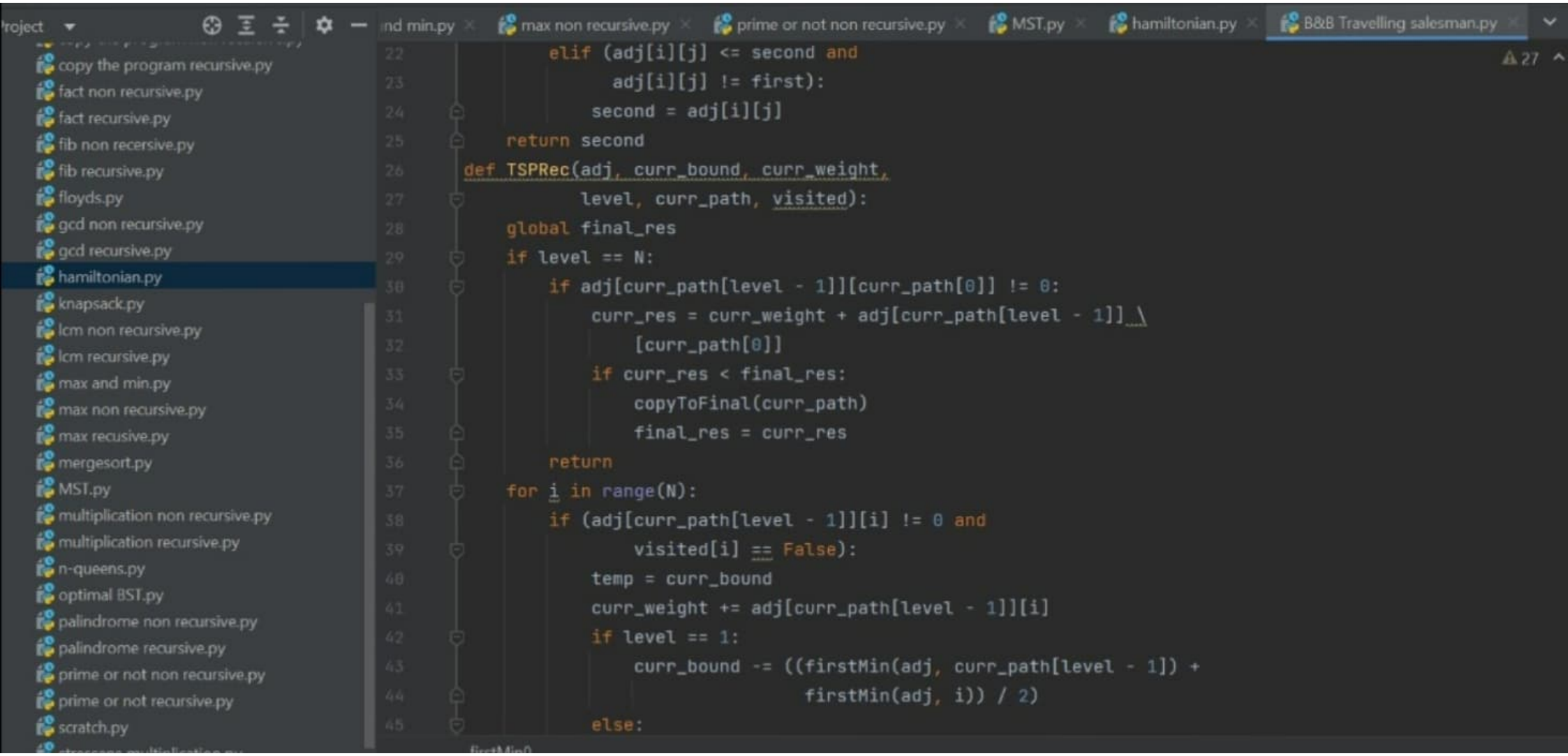
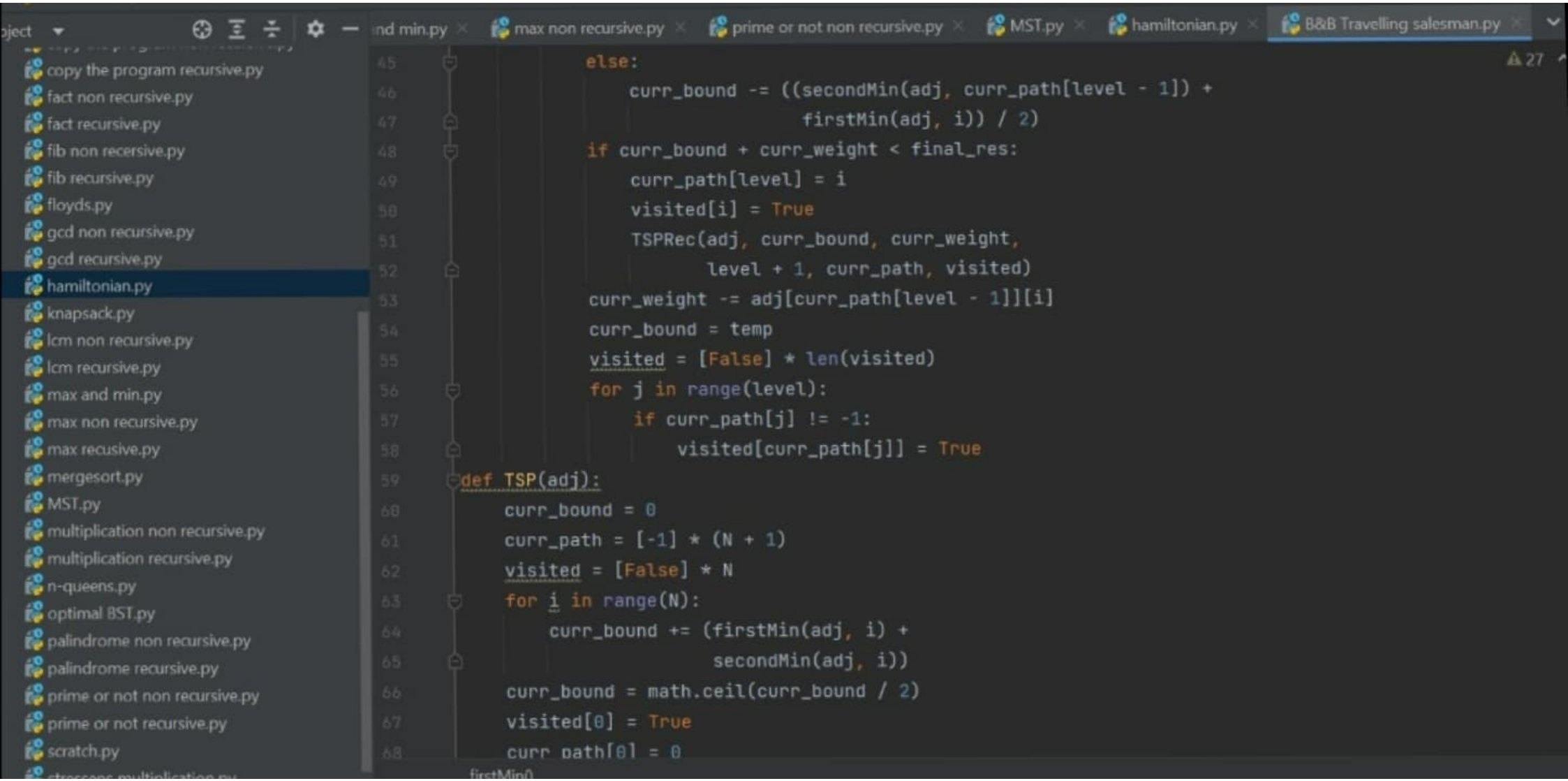
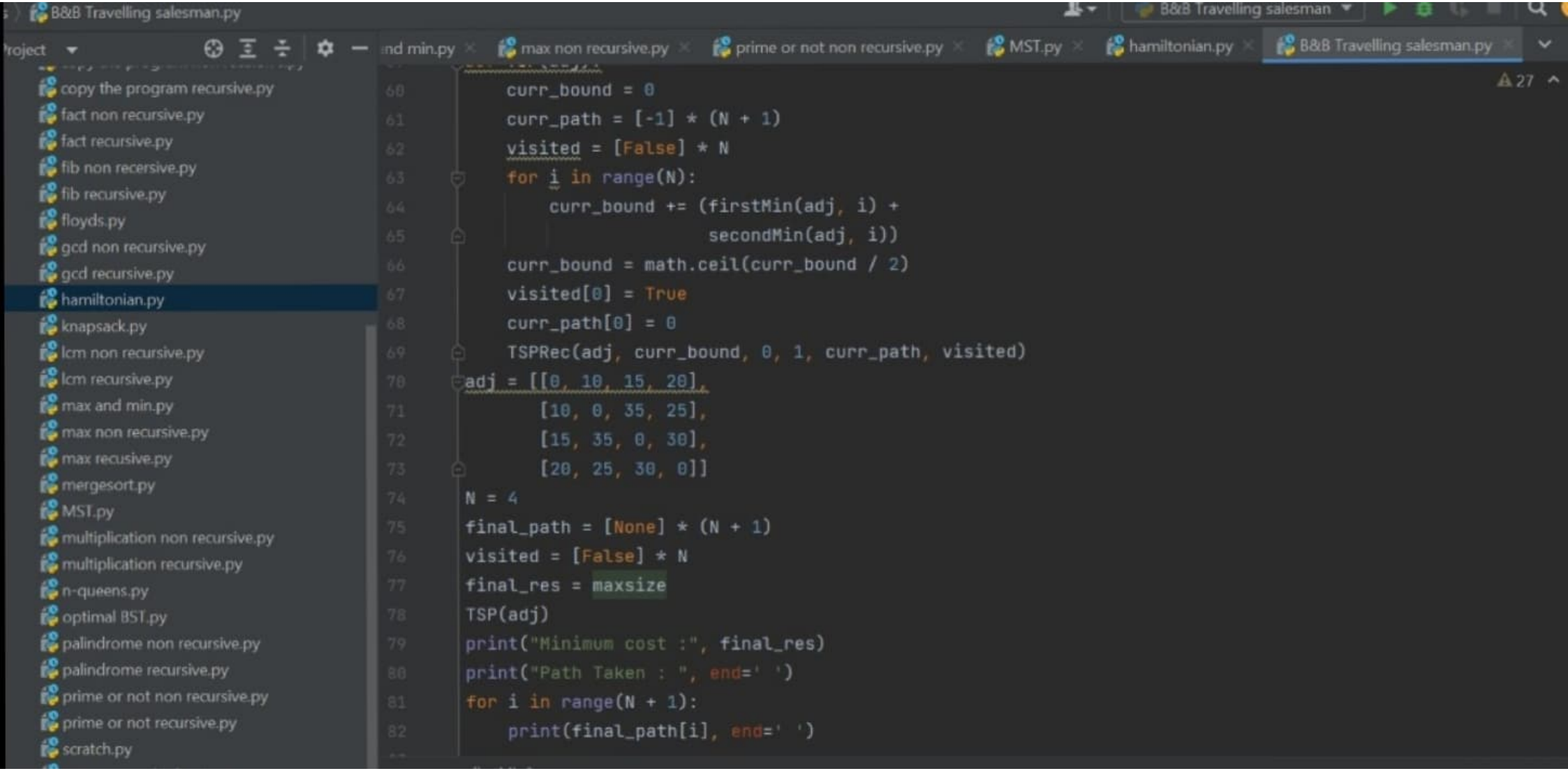
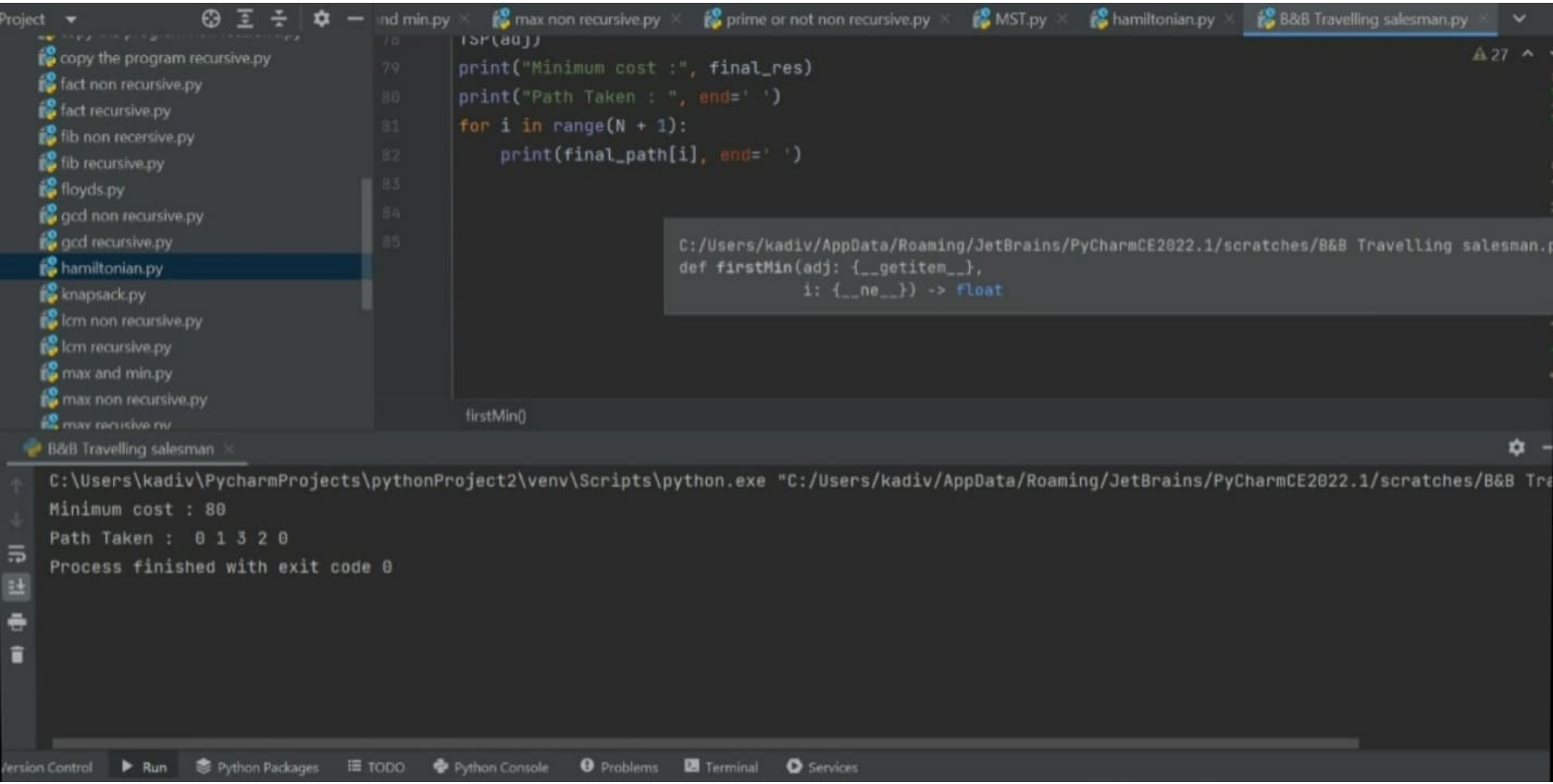

Project
copy the program recursive.py
fact non recursive.py
fact recursive.py
fib non recersive.py
fib recursive.py
floyds.py
gcd non recursive.py
gcd recursive.py
hamiltonian.py
knapsack.py
lcm non recursive.py
lcm recursive.py
max and min.py
max non recursive.py
max recusive.py
mergesort.py
MST.py
multiplication non recursive.py
multiplication recursive.py
n-queens.py
optimal BST.py
palindrome non recursive.py
palindrome recursive.py
prime or not non recursive.py
prime or not recursive.py
scratch.py
stressens multiplication.py
nd min.py
max non recursive.py
prime or not non recursive.py
MST.py
hamiltonian.py
B&B Travelling salesman.py
27
import math
maxsize = float('inf')
def copyToFinal(curr_path):
final_path[:N + 1] = curr_path[:]
final_path[N] = curr_path[0]
def firstMin(adj, i):
min = maxsize
for k in range(N):
if adj[i][k] < min and i != k:
min = adj[i][k]
return min
def secondMin(adj, i):
first, second = maxsize, maxsize
for j in range(N):
if i == j:
continue
if adj[i][j] <= first:
second = first
first = adj[i][j]
elif (adj[i][j] <= second and
adj[i][j] != first):
firstMin()


copy the program recursive.py
fact non recursive.py
fact recursive.py
fib non recersive.py
fib recursive.py
floyds.py
gcd non recursive.py
gcd recursive.py
hamiltonian.py
knapsack.py
lcm non recursive.py
lcm recursive.py
max and min.py
max non recursive.py
max recusive.py
mergesort.py
MST.py
multiplication non recursive.py
multiplication recursive.py
n-queens.py
optimal BST.py
palindrome non recursive.py
palindrome recursive.py
prime or not non recursive.py
prime or not recursive.py
scratch.py
nd min.py
max non recursive.py
prime or not non recursive.py
MST.py
hamiltonian.py
B&B Travelling salesman.py
27
elif (adj[i][j] <= second and
adj[i][j] != first):
second = adj[i][j]
return second
def TSPRec(adj, curr_bound, curr_weight,
level, curr_path, visited):
global final_res
if level == N:
if adj[curr_path[level - 1]][curr_path[0]] != 0:
curr_res = curr_weight + adj[curr_path[level - 1]]\
[curr_path[0]]
if curr_res < final_res:
copyToFinal(curr_path)
final_res = curr_res
return
for i in range(N):
if (adj[curr_path[level - 1]][i] != 0 and
visited[i] == False):
temp = curr_bound
curr_weight += adj[curr_path[level - 1]][i]
if level == 1:
curr_bound -= ((firstMin(adj, curr_path[level - 1]) +
firstMin(adj, i)) / 2)
else:


copy the program recursive.py
fact non recursive.py
fact recursive.py
fib non recersive.py
fib recursive.py
floyds.py
gcd non recursive.py
gcd recursive.py
hamiltonian.py
knapsack.py
lcm non recursive.py
lcm recursive.py
max and min.py
max non recursive.py
max recusive.py
mergesort.py
MST.py
multiplication non recursive.py
multiplication recursive.py
n-queens.py
optimal BST.py
palindrome non recursive.py
palindrome recursive.py
prime or not non recursive.py
prime or not recursive.py
scratch.py
nd min.py
max non recursive.py
prime or not non recursive.py
MST.py
hamiltonian.py
B&B Travelling salesman.py
else:
curr_bound -= ((secondMin(adj, curr_path[level - 1]) +
firstMin(adj, i)) / 2)
if curr_bound + curr_weight < final_res:
curr_path[level] = i
visited[i] = True
TSPRec(adj, curr_bound, curr_weight,
level + 1, curr_path, visited)
curr_weight -= adj[curr_path[level - 1]][i]
curr_bound = temp
visited = [False] * len(visited)
for j in range(level):
if curr_path[j] != -1:
visited[curr_path[j]] = True
def TSP(adj):
curr_bound = 0
curr_path = [-1] * (N + 1)
visited = [False] * N
for i in range(N):
curr_bound += (firstMin(adj, i) +
secondMin(adj, i))
curr_bound = math.ceil(curr_bound / 2)
visited[0] = True
curr_path[0] = 0
firstMin()


B&B Travelling salesman.py
B&B Travelling salesman
max non recursive.py
prime or not non recursive.py
MST.py
hamiltonian.py
B&B Travelling salesman.py
copy the program recursive.py
fact non recursive.py
fact recursive.py
fib non recersive.py
fib recursive.py
floyds.py
gcd non recursive.py
gcd recursive.py
hamiltonian.py
knapsack.py
lcm non recursive.py
lcm recursive.py
max and min.py
max non recursive.py
max recusive.py
mergesort.py
MST.py
multiplication non recursive.py
multiplication recursive.py
n-queens.py
optimal BST.py
palindrome non recursive.py
palindrome recursive.py
prime or not non recursive.py
prime or not recursive.py
scratch.py
curr_bound = 0
curr_path = [-1] * (N + 1)
visited = [False] * N
for i in range(N):
curr_bound += (firstMin(adj, i) +
secondMin(adj, i))
curr_bound = math.ceil(curr_bound / 2)
visited[0] = True
curr_path[0] = 0
TSPRec(adj, curr_bound, 0, 1, curr_path, visited)
adj = [[0, 10, 15, 20],
[10, 0, 35, 25],
[15, 35, 0, 30],
[20, 25, 30, 0]]
N = 4
final_path = [None] * (N + 1)
visited = [False] * N
final_res = maxsize
TSP(adj)
print("Minimum cost :", final_res)
print("Path Taken : ", end=' ')
for i in range(N + 1):
print(final_path[i], end=' ')


Project
copy the program recursive.py
fact non recursive.py
fact recursive.py
fib non recersive.py
fib recursive.py
floyds.py
gcd non recursive.py
gcd recursive.py
hamiltonian.py
knapsack.py
lcm non recursive.py
lcm recursive.py
max and min.py
max non recursive.py
nd min.py
max non recursive.py
prime or not non recursive.py
MST.py
hamiltonian.py
B&B Travelling salesman.py
print("Minimum cost :", final_res)
print("Path Taken : ", end=' ')
for i in range(N + 1):
print(final_path[i], end=' ')
C:/Users/kadiv/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratches/B&B Travelling salesman.
def firstMin(adj: {__getitem__},
i: {__ne__}) -> float
firstMin()
B&B Travelling salesman
C:\Users\kadiv\PycharmProjects\pythonProject2\venv\Scripts\python.exe "C:/Users/kadiv/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratches/B&B Tra
Minimum cost : 80
Path Taken : 0 1 3 2 0
Process finished with exit code 0
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services