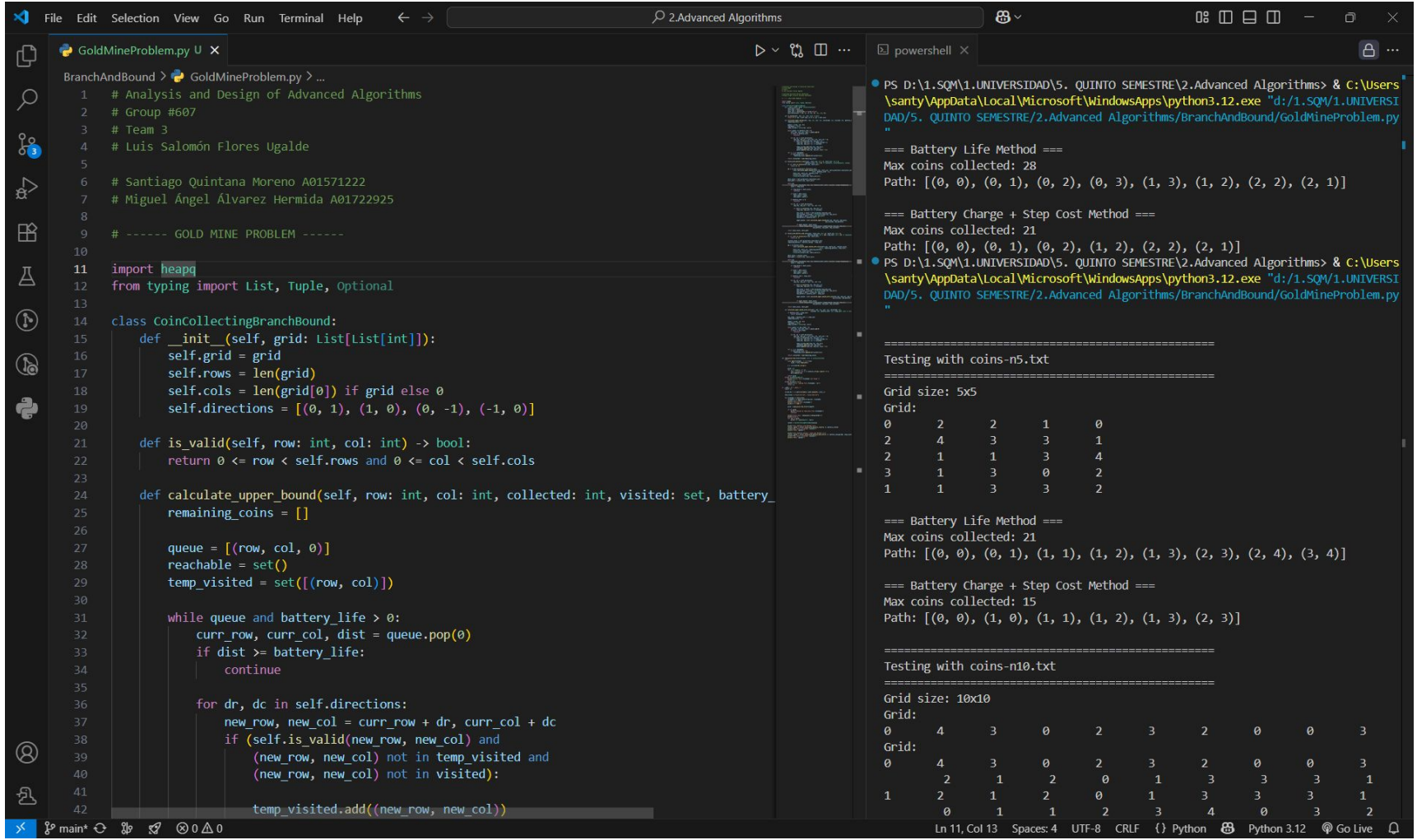
Tecnológico de Monterrey

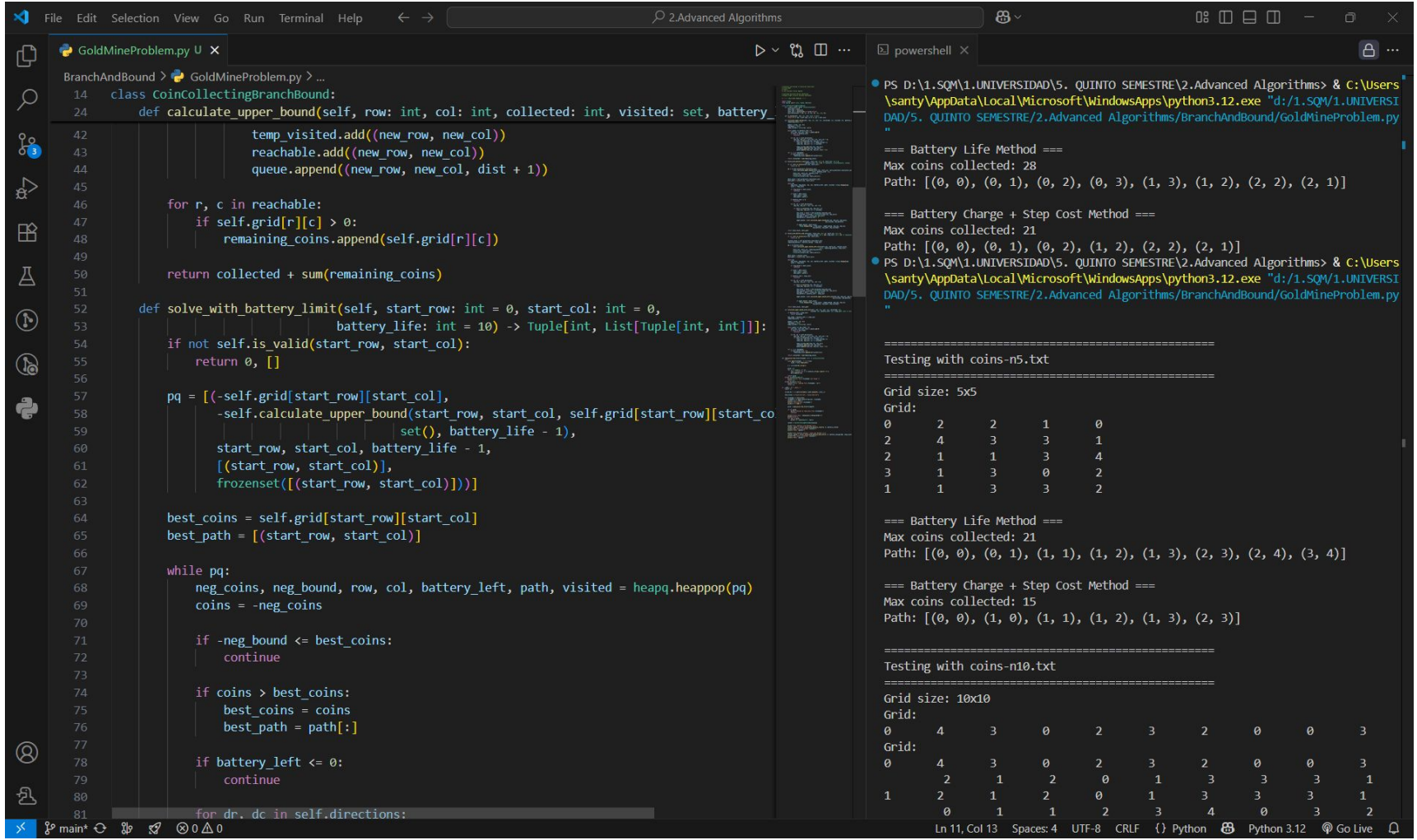
Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

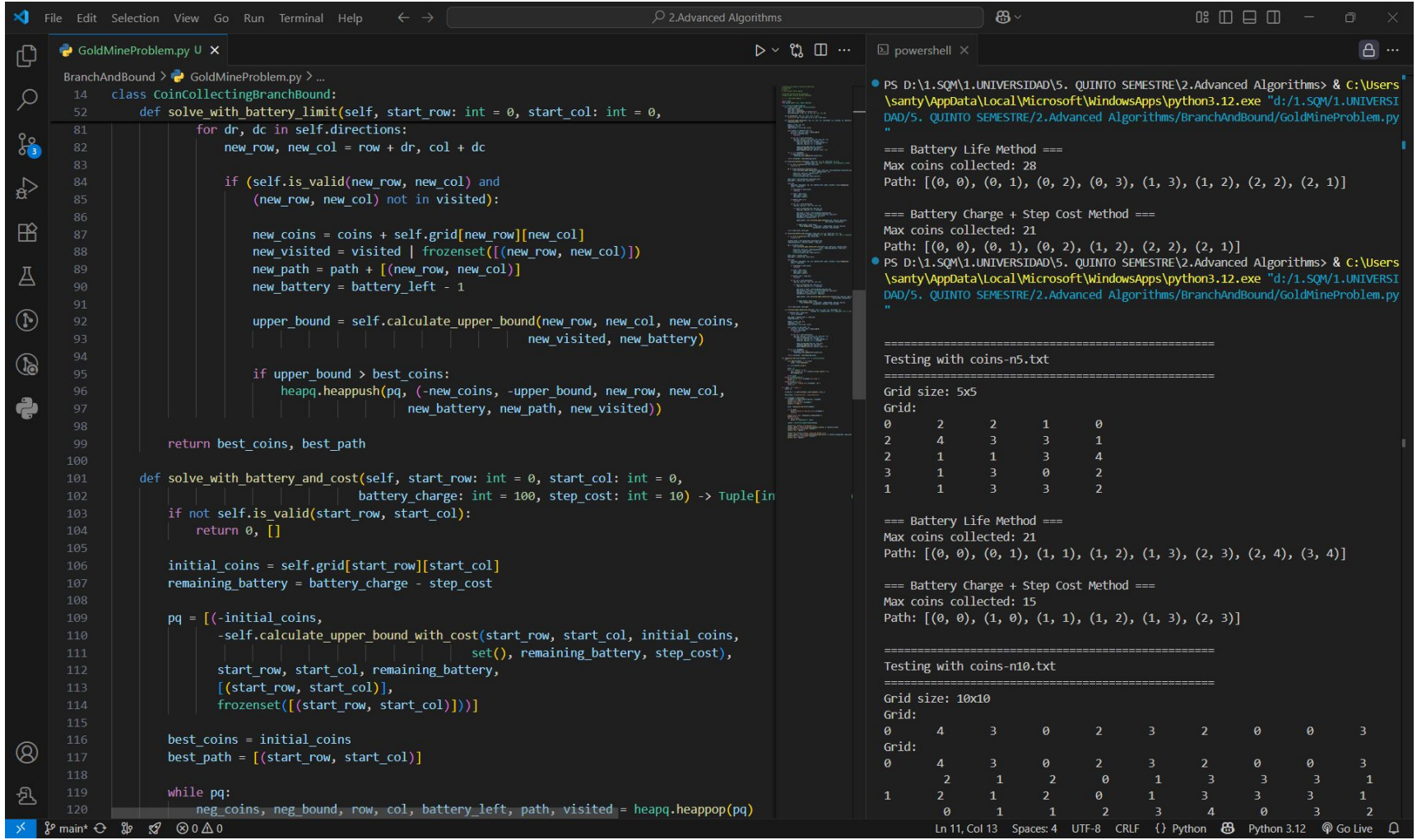
Homework 6:
Gold Mine Problem

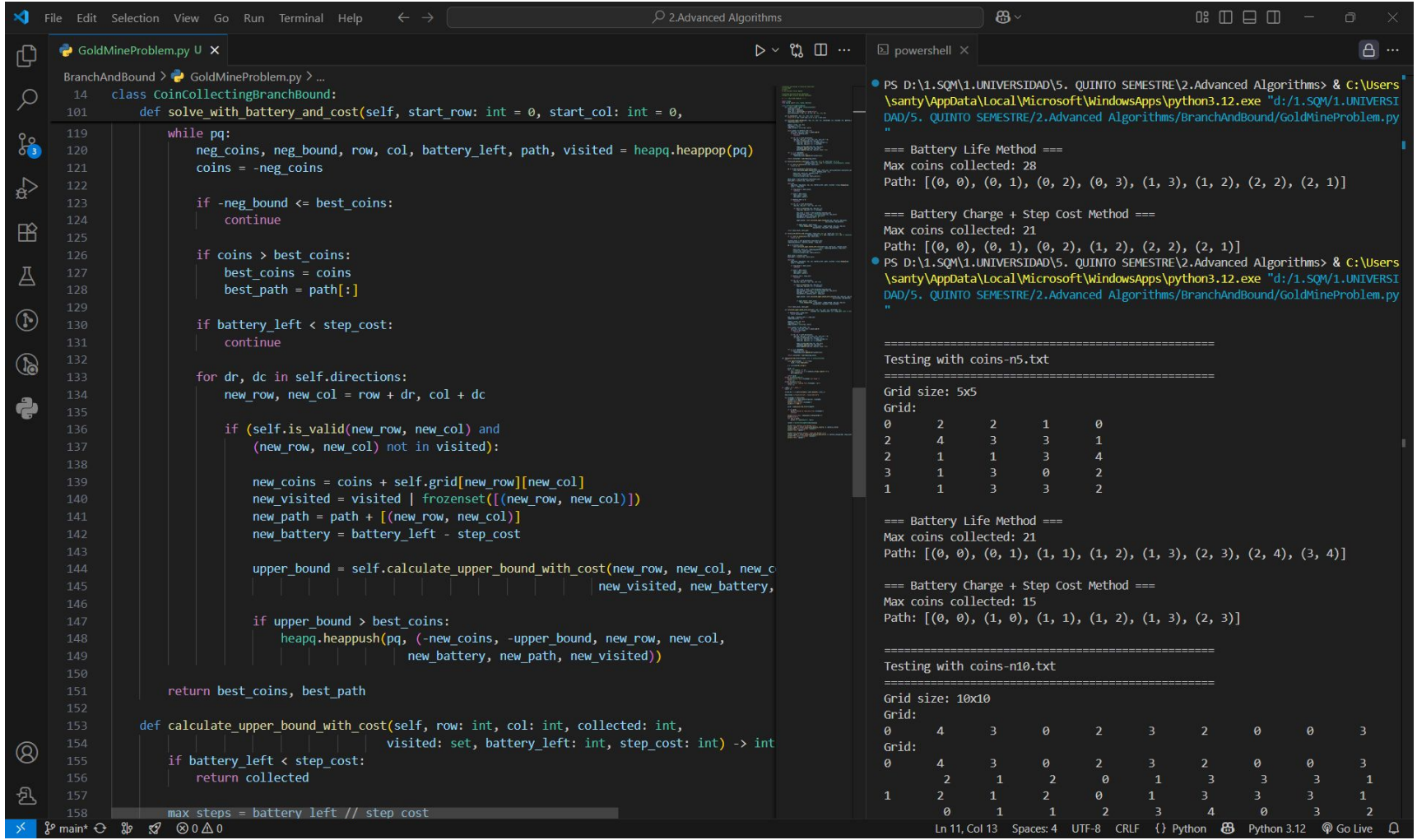
Group: 607
Team #3
Dr. Katie Brodhead

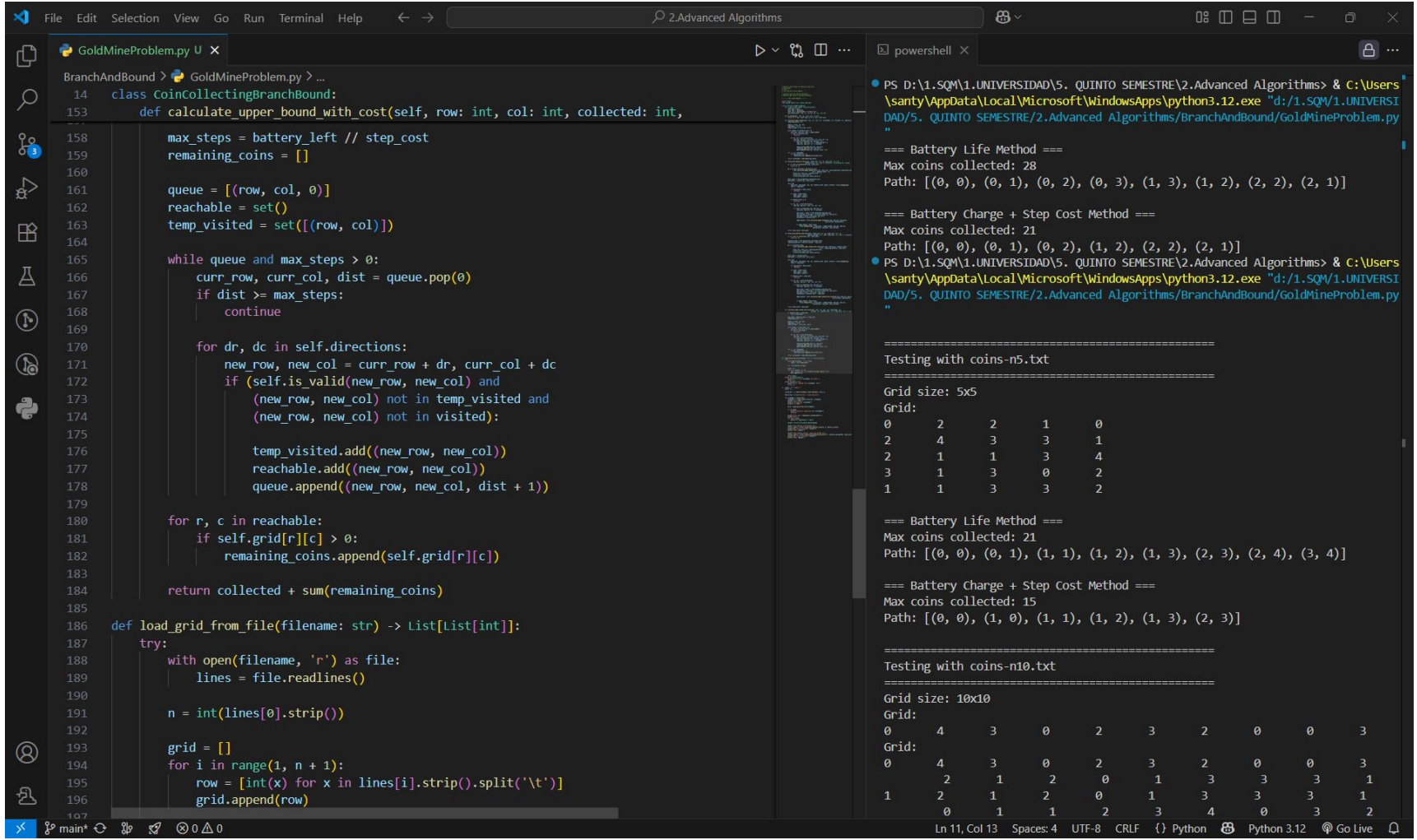
Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925

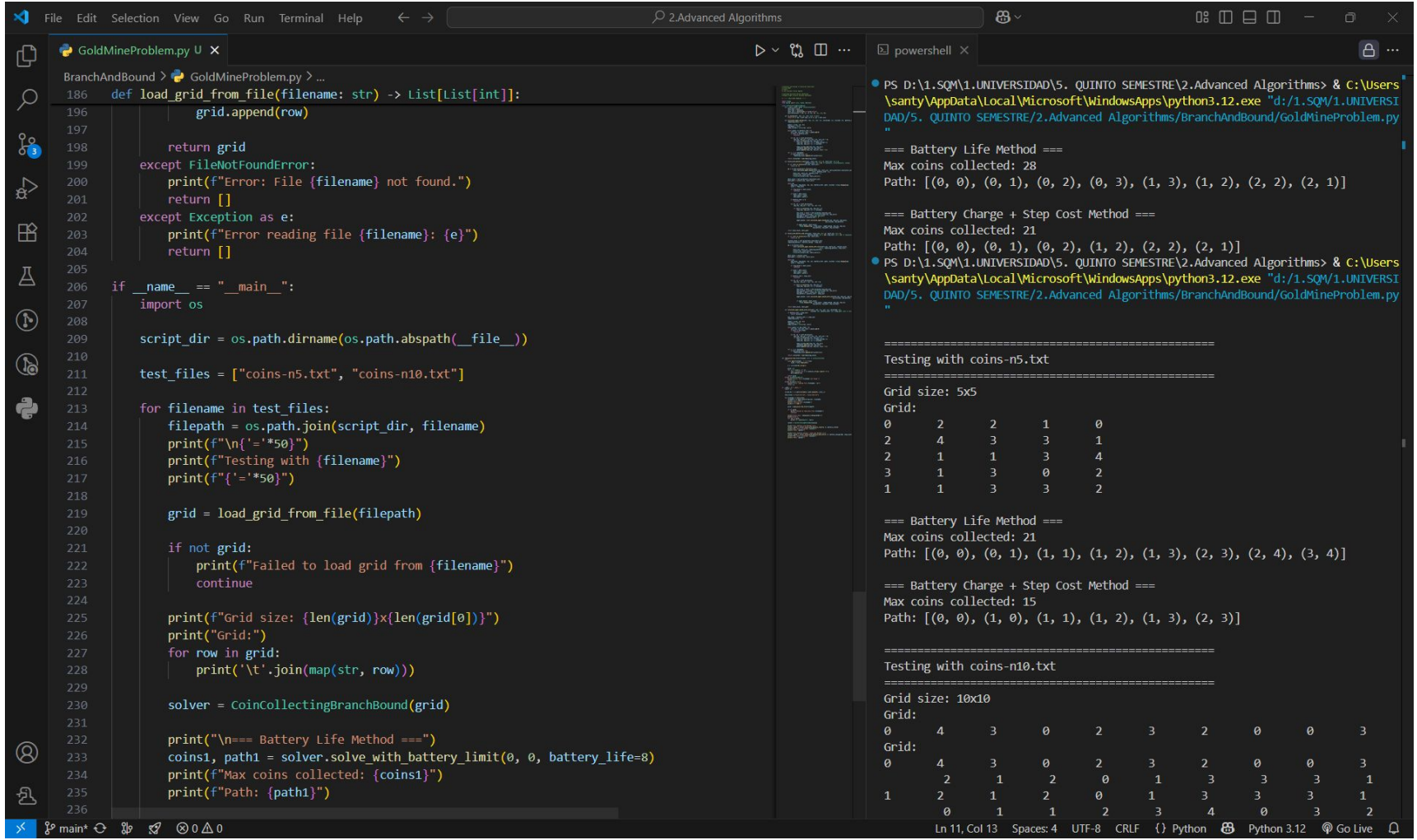

File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GoldMineProblem.py U
BranchAndBound > GoldMineProblem.py > ...
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 3
# Luis Salomón Flores Ugalde
# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925
# ------ GOLD MINE PROBLEM ------
import heapq
from typing import List, Tuple, Optional
class CoinCollectingBranchBound:
def __init__(self, grid: List[List[int]]):
self.grid = grid
self.rows = len(grid)
self.cols = len(grid[0]) if grid else 0
self.directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]
def is_valid(self, row: int, col: int) -> bool:
return 0 <= row < self.rows and 0 <= col < self.cols
def calculate_upper_bound(self, row: int, col: int, collected: int, visited: set, battery_
remaining_coins = []
queue = [(row, col, 0)]
reachable = set()
temp_visited = set([(row, col)])
while queue and battery_life > 0:
curr_row, curr_col, dist = queue.pop(0)
if dist >= battery_life:
continue
for dr, dc in self.directions:
new_row, new_col = curr_row + dr, curr_col + dc
if (self.is_valid(new_row, new_col) and
(new_row, new_col) not in temp_visited and
(new_row, new_col) not in visited):
temp_visited.add((new_row, new_col))
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
=== Battery Life Method ===
Max coins collected: 28
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (1, 2), (2, 2), (2, 1)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1)]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
==========
Testing with coins-n5.txt
==========
Grid size: 5x5
Grid:
0 2 2 1 0
2 4 3 3 1
2 1 1 3 4
3 1 3 0 2
1 1 3 3 2
=== Battery Life Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 15
Path: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3)]
==========
Testing with coins-n10.txt
==========
Grid size: 10x10
Grid:
0 4 3 0 2 3 2 0 0 3
Grid:
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 2
main*
0 0
Ln 11, Col 13
Spaces: 4
UTF-8
CRLF
Python
Python 3.12
Go Live

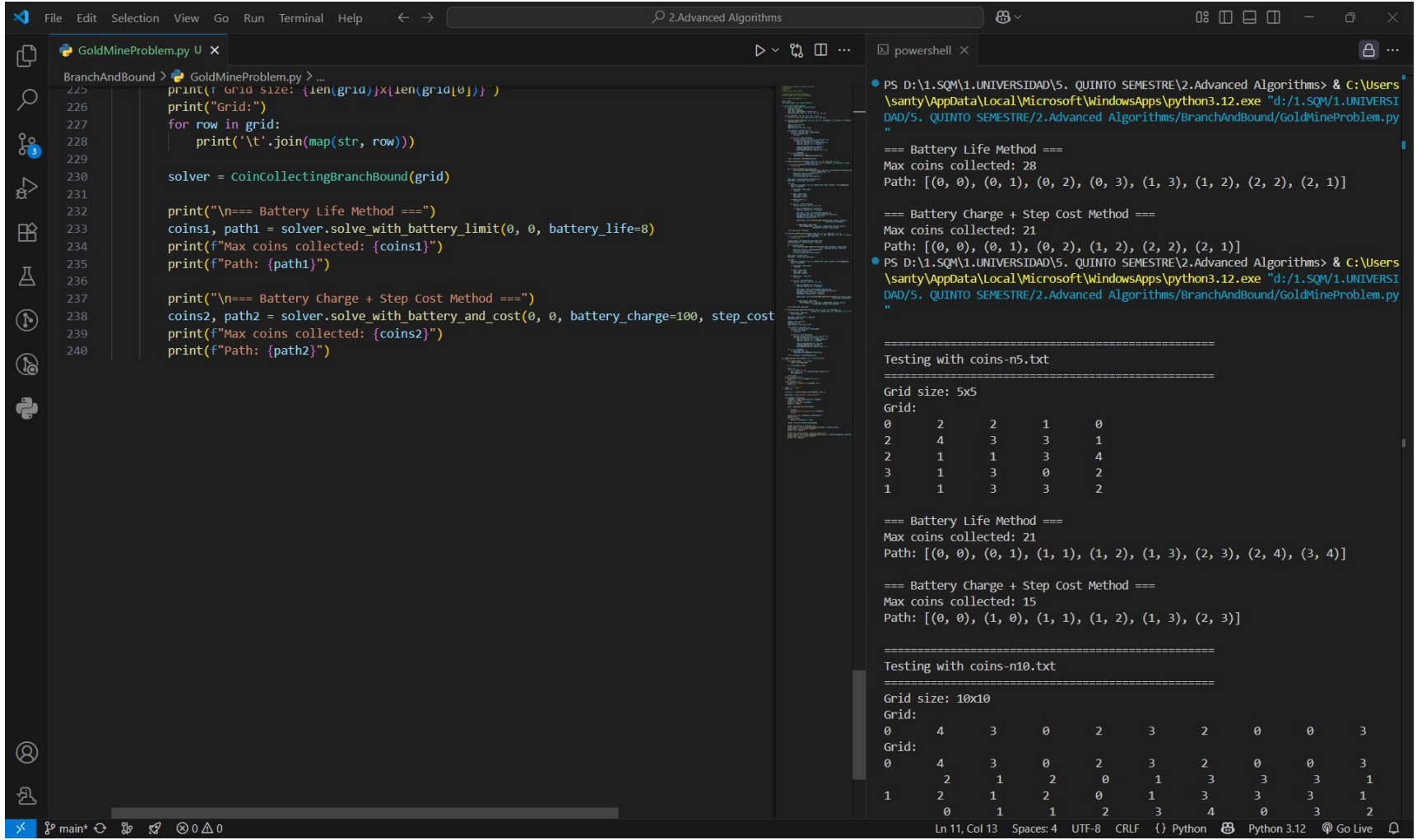

File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GoldMineProblem.py U
BranchAndBound > GoldMineProblem.py > ...
class CoinCollectingBranchBound:
def calculate_upper_bound(self, row: int, col: int, collected: int, visited: set, battery_
temp_visited.add((new_row, new_col))
reachable.add((new_row, new_col))
queue.append((new_row, new_col, dist + 1))
for r, c in reachable:
if self.grid[r][c] > 0:
remaining_coins.append(self.grid[r][c])
return collected + sum(remaining_coins)
def solve_with_battery_limit(self, start_row: int = 0, start_col: int = 0,
battery_life: int = 10) -> Tuple[int, List[Tuple[int, int]]]:
if not self.is_valid(start_row, start_col):
return 0, []
pq = [(-self.grid[start_row][start_col],
-self.calculate_upper_bound(start_row, start_col, self.grid[start_row][start_co
set(), battery_life - 1),
start_row, start_col, battery_life - 1,
[(start_row, start_col)],
frozenset([(start_row, start_col)]))]
best_coins = self.grid[start_row][start_col]
best_path = [(start_row, start_col)]
while pq:
neg_coins, neg_bound, row, col, battery_left, path, visited = heapq.heappop(pq)
coins = -neg_coins
if -neg_bound <= best_coins:
continue
if coins > best_coins:
best_coins = coins
best_path = path[:]
if battery_left <= 0:
continue
for dr, dc in self.directions:
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
=== Battery Life Method ===
Max coins collected: 28
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (1, 2), (2, 2), (2, 1)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1)]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
==========
Testing with coins-n5.txt
==========
Grid size: 5x5
Grid:
0 2 2 1 0
2 4 3 3 1
2 1 1 3 4
3 1 3 0 2
1 1 3 3 2
=== Battery Life Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 15
Path: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3)]
==========
Testing with coins-n10.txt
==========
Grid size: 10x10
Grid:
0 4 3 0 2 3 2 0 0 3
Grid:
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 2
main*
Ln 11, Col 13 Spaces: 4 UTF-8 CRLF Python Python 3.12 Go Live

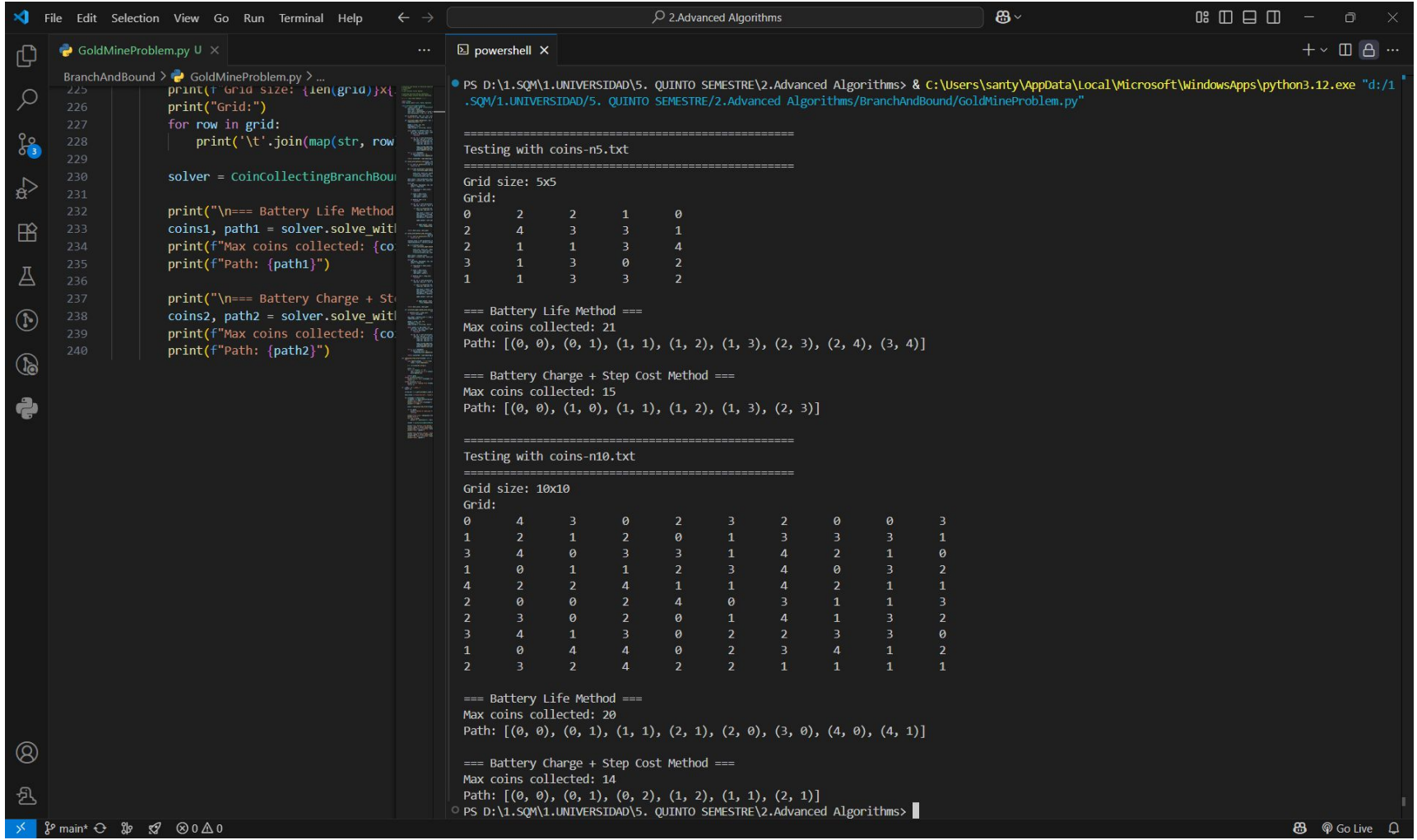

File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GoldMineProblem.py U
BranchAndBound > GoldMineProblem.py > ...
class CoinCollectingBranchBound:
def solve_with_battery_limit(self, start_row: int = 0, start_col: int = 0,
for dr, dc in self.directions:
new_row, new_col = row + dr, col + dc
if (self.is_valid(new_row, new_col) and
(new_row, new_col) not in visited):
new_coins = coins + self.grid[new_row][new_col]
new_visited = visited | frozenset([(new_row, new_col)])
new_path = path + [(new_row, new_col)]
new_battery = battery_left - 1
upper_bound = self.calculate_upper_bound(new_row, new_col, new_coins,
new_visited, new_battery)
if upper_bound > best_coins:
heapq.heappush(pq, (-new_coins, -upper_bound, new_row, new_col,
new_battery, new_path, new_visited))
return best_coins, best_path
def solve_with_battery_and_cost(self, start_row: int = 0, start_col: int = 0,
battery_charge: int = 100, step_cost: int = 10) -> Tuple[in
if not self.is_valid(start_row, start_col):
return 0, []
initial_coins = self.grid[start_row][start_col]
remaining_battery = battery_charge - step_cost
pq = [(-initial_coins,
-self.calculate_upper_bound_with_cost(start_row, start_col, initial_coins,
set(), remaining_battery, step_cost),
start_row, start_col, remaining_battery,
[(start_row, start_col)],
frozenset([(start_row, start_col)]))]
best_coins = initial_coins
best_path = [(start_row, start_col)]
while pq:
neg_coins, neg_bound, row, col, battery_left, path, visited = heapq.heappop(pq)
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
=== Battery Life Method ===
Max coins collected: 28
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (1, 2), (2, 2), (2, 1)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1)]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
==========
Testing with coins-n5.txt
==========
Grid size: 5x5
Grid:
0 2 2 1 0
2 4 3 3 1
2 1 1 3 4
3 1 3 0 2
1 1 3 3 2
=== Battery Life Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 15
Path: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3)]
==========
Testing with coins-n10.txt
==========
Grid size: 10x10
Grid:
0 4 3 0 2 3 2 0 0 3
Grid:
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 2
main*
Ln 11, Col 13 Spaces: 4 UTF-8 CRLF Python Python 3.12 Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GoldMineProblem.py U
BranchAndBound > GoldMineProblem.py > ...
class CoinCollectingBranchBound:
def solve_with_battery_and_cost(self, start_row: int = 0, start_col: int = 0,
while pq:
neg_coins, neg_bound, row, col, battery_left, path, visited = heapq.heappop(pq)
coins = -neg_coins
if -neg_bound <= best_coins:
continue
if coins > best_coins:
best_coins = coins
best_path = path[:]
if battery_left < step_cost:
continue
for dr, dc in self.directions:
new_row, new_col = row + dr, col + dc
if (self.is_valid(new_row, new_col) and
(new_row, new_col) not in visited):
new_coins = coins + self.grid[new_row][new_col]
new_visited = visited | frozenset([(new_row, new_col)])
new_path = path + [(new_row, new_col)]
new_battery = battery_left - step_cost
upper_bound = self.calculate_upper_bound_with_cost(new_row, new_col, new_c
new_visited, new_battery,
if upper_bound > best_coins:
heapq.heappush(pq, (-new_coins, -upper_bound, new_row, new_col,
new_battery, new_path, new_visited))
return best_coins, best_path
def calculate_upper_bound_with_cost(self, row: int, col: int, collected: int,
visited: set, battery_left: int, step_cost: int) -> int
if battery_left < step_cost:
return collected
max_steps = battery_left // step_cost
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
=== Battery Life Method ===
Max coins collected: 28
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (1, 2), (2, 2), (2, 1)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1)]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
==========
Testing with coins-n5.txt
==========
Grid size: 5x5
Grid:
0 2 2 1 0
2 4 3 3 1
2 1 1 3 4
3 1 3 0 2
1 1 3 3 2
=== Battery Life Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 15
Path: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3)]
==========
Testing with coins-n10.txt
==========
Grid size: 10x10
Grid:
0 4 3 0 2 3 2 0 0 3
Grid:
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 2
main*
Ln 11, Col 13 Spaces: 4 UTF-8 CRLF Python Python 3.12 Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GoldMineProblem.py U
BranchAndBound > GoldMineProblem.py > ...
class CoinCollectingBranchBound:
def calculate_upper_bound_with_cost(self, row: int, col: int, collected: int,
max_steps = battery_left // step_cost
remaining_coins = []
queue = [(row, col, 0)]
reachable = set()
temp_visited = set([(row, col)])
while queue and max_steps > 0:
curr_row, curr_col, dist = queue.pop(0)
if dist >= max_steps:
continue
for dr, dc in self.directions:
new_row, new_col = curr_row + dr, curr_col + dc
if (self.is_valid(new_row, new_col) and
(new_row, new_col) not in temp_visited and
(new_row, new_col) not in visited):
temp_visited.add((new_row, new_col))
reachable.add((new_row, new_col))
queue.append((new_row, new_col, dist + 1))
for r, c in reachable:
if self.grid[r][c] > 0:
remaining_coins.append(self.grid[r][c])
return collected + sum(remaining_coins)
def load_grid_from_file(filename: str) -> List[List[int]]:
try:
with open(filename, 'r') as file:
lines = file.readlines()
n = int(lines[0].strip())
grid = []
for i in range(1, n + 1):
row = [int(x) for x in lines[i].strip().split('\t')]
grid.append(row)
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
=== Battery Life Method ===
Max coins collected: 28
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (1, 2), (2, 2), (2, 1)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1)]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
==========
Testing with coins-n5.txt
==========
Grid size: 5x5
Grid:
0 2 2 1 0
2 4 3 3 1
2 1 1 3 4
3 1 3 0 2
1 1 3 3 2
=== Battery Life Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 15
Path: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3)]
==========
Testing with coins-n10.txt
==========
Grid size: 10x10
Grid:
0 4 3 0 2 3 2 0 0 3
Grid:
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 2
main*
0 0
Ln 11, Col 13
Spaces: 4
UTF-8
CRLF
Python
Python 3.12
Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GoldMineProblem.py U
BranchAndBound > GoldMineProblem.py > ...
def load_grid_from_file(filename: str) -> List[List[int]]:
grid.append(row)
return grid
except FileNotFoundError:
print(f"Error: File {filename} not found.")
return []
except Exception as e:
print(f"Error reading file {filename}: {e}")
return []
if __name__ == "__main__":
import os
script_dir = os.path.dirname(os.path.abspath(__file__))
test_files = ["coins-n5.txt", "coins-n10.txt"]
for filename in test_files:
filepath = os.path.join(script_dir, filename)
print(f"\n{'='*50}")
print(f"Testing with {filename}")
print(f"{'='*50}")
grid = load_grid_from_file(filepath)
if not grid:
print(f"Failed to load grid from {filename}")
continue
print(f"Grid size: {len(grid)}x{len(grid[0])}")
print("Grid:")
for row in grid:
print('\t'.join(map(str, row)))
solver = CoinCollectingBranchBound(grid)
print("\n=== Battery Life Method ===")
coins1, path1 = solver.solve_with_battery_limit(0, 0, battery_life=8)
print(f"Max coins collected: {coins1}")
print(f"Path: {path1}")
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
=== Battery Life Method ===
Max coins collected: 28
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (1, 2), (2, 2), (2, 1)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1)]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
==
Testing with coins-n5.txt
==
Grid size: 5x5
Grid:
0 2 2 1 0
2 4 3 3 1
2 1 1 3 4
3 1 3 0 2
1 1 3 3 2
=== Battery Life Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 15
Path: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3)]
==
Testing with coins-n10.txt
==
Grid size: 10x10
Grid:
0 4 3 0 2 3 2 0 0 3
Grid:
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 2
main*
0 0
Ln 11, Col 13 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GoldMineProblem.py U
BranchAndBound > GoldMineProblem.py > ...
print(f"Grid size: {len(grid)}x{len(grid[0])}")
print("Grid:")
for row in grid:
print('\t'.join(map(str, row)))
solver = CoinCollectingBranchBound(grid)
print("\n=== Battery Life Method ===")
coins1, path1 = solver.solve_with_battery_limit(0, 0, battery_life=8)
print(f"Max coins collected: {coins1}")
print(f"Path: {path1}")
print("\n=== Battery Charge + Step Cost Method ===")
coins2, path2 = solver.solve_with_battery_and_cost(0, 0, battery_charge=100, step_cost
print(f"Max coins collected: {coins2}")
print(f"Path: {path2}")
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
=== Battery Life Method ===
Max coins collected: 28
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (1, 2), (2, 2), (2, 1)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1)]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
==========
Testing with coins-n5.txt
==========
Grid size: 5x5
Grid:
0 2 2 1 0
2 4 3 3 1
2 1 1 3 4
3 1 3 0 2
1 1 3 3 2
=== Battery Life Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 15
Path: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3)]
==========
Testing with coins-n10.txt
==========
Grid size: 10x10
Grid:
0 4 3 0 2 3 2 0 0 3
Grid:
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 2
main*
Ln 11, Col 13 Spaces: 4 UTF-8 CRLF Python Python 3.12 Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GoldMineProblem.py U
BranchAndBound > GoldMineProblem.py > ...
print(f"Grid Size: {len(grid)}x{
print("Grid:")
for row in grid:
print('\t'.join(map(str, row
solver = CoinCollectingBranchBou
print("\n=== Battery Life Method
coins1, path1 = solver.solve_wit
print(f"Max coins collected: {co
print(f"Path: {path1}")
print("\n=== Battery Charge + St
coins2, path2 = solver.solve_wit
print(f"Max coins collected: {co
print(f"Path: {path2}")
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/BranchAndBound/GoldMineProblem.py"
==========
Testing with coins-n5.txt
==========
Grid size: 5x5
Grid:
0 2 2 1 0
2 4 3 3 1
2 1 1 3 4
3 1 3 0 2
1 1 3 3 2
=== Battery Life Method ===
Max coins collected: 21
Path: [(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 15
Path: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3)]
==========
Testing with coins-n10.txt
==========
Grid size: 10x10
Grid:
0 4 3 0 2 3 2 0 0 3
1 2 1 2 0 1 3 3 3 1
3 4 0 3 3 1 4 2 1 0
1 0 1 1 2 3 4 0 3 2
4 2 2 4 1 1 4 2 1 1
2 0 0 2 4 0 3 1 1 3
2 3 0 2 0 1 4 1 3 2
3 4 1 3 0 2 2 3 3 0
1 0 4 4 0 2 3 4 1 2
2 3 2 4 2 2 1 1 1 1
=== Battery Life Method ===
Max coins collected: 20
Path: [(0, 0), (0, 1), (1, 1), (2, 1), (2, 0), (3, 0), (4, 0), (4, 1)]
=== Battery Charge + Step Cost Method ===
Max coins collected: 14
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (1, 1), (2, 1)]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
main*
Go Live

https://colab.research.google.com/drive/1QatvvuMiPeTUmElMeaIWLtKsXV0tab1K?usp=sharing