



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

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

GoldMineProblem.py U X

BranchAndBound > GoldMineProblem.py > ...
1 # Analysis and Design of Advanced Algorithms
2 # Group #607
3 # Team 3
4 # Luis Salomón Flores Ugalde
5
6 # Santiago Quintana Moreno A01571222
7 # Miguel Ángel Álvarez Hermida A01722925
8
9 # ----- GOLD MINE PROBLEM -----
10
11 import heapq
12 from typing import List, Tuple, Optional
13
14 class CoinCollectingBranchBound:
15 def __init__(self, grid: List[List[int]]):
16 self.grid = grid
17 self.rows = len(grid)
18 self.cols = len(grid[0]) if grid else 0
19 self.directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]
20
21 def is_valid(self, row: int, col: int) -> bool:
22 return 0 <= row < self.rows and 0 <= col < self.cols
23
24 def calculate_upper_bound(self, row: int, col: int, collected: int, visited: set, battery_remaining_coins = []):
25
26 queue = [(row, col, 0)]
27 reachable = set()
28 temp_visited = set([(row, col)])
29
30
31 while queue and battery_life > 0:
32 curr_row, curr_col, dist = queue.pop(0)
33 if dist >= battery_life:
34 continue
35
36 for dr, dc in self.directions:
37 new_row, new_col = curr_row + dr, curr_col + dc
38 if (self.is_valid(new_row, new_col) and
39 (new_row, new_col) not in temp_visited and
40 (new_row, new_col) not in visited):
41
42 temp_visited.add((new_row, new_col))

powershell X

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 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 3 2

Ln 11, Col 13 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

GoldMineProblem.py U X

BranchAndBound > GoldMineProblem.py > ...

```
14 class CoinCollectingBranchBound:
24     def calculate_upper_bound(self, row: int, col: int, collected: int, visited: set, battery_left: int):
42         temp_visited.add((new_row, new_col))
43         reachable.add((new_row, new_col))
44         queue.append((new_row, new_col, dist + 1))
46     for r, c in reachable:
47         if self.grid[r][c] > 0:
48             remaining_coins.append(self.grid[r][c])
49
50     return collected + sum(remaining_coins)
52
53 def solve_with_battery_limit(self, start_row: int = 0, start_col: int = 0,
54                             battery_life: int = 10) -> Tuple[int, List[Tuple[int, int]]]:
55     if not self.is_valid(start_row, start_col):
56         return 0, []
58
59     pq = [(-self.grid[start_row][start_col],
60           -self.calculate_upper_bound(start_row, start_col, self.grid[start_row][start_col],
61                                       set(), battery_life - 1),
62           start_row, start_col, battery_life - 1,
63           [(start_row, start_col)],
64           frozenset([(start_row, start_col)]))]
66
67     best_coins = self.grid[start_row][start_col]
68     best_path = [(start_row, start_col)]
70
71     while pq:
72         neg_coins, neg_bound, row, col, battery_left, path, visited = heapq.heappop(pq)
73         coins = -neg_coins
75
76         if -neg_bound <= best_coins:
77             continue
79
80         if coins > best_coins:
81             best_coins = coins
82             best_path = path[:]
84
85         if battery_left <= 0:
86             continue
88
89         for dr, dc in self.directions:
```

powershell X

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
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 3 2

Ln 11, Col 13 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

GoldMineProblem.py U X

BranchAndBound > GoldMineProblem.py > ...

```
14 class CoinCollectingBranchBound:
15     def solve_with_battery_limit(self, start_row: int = 0, start_col: int = 0,
81         for dr, dc in self.directions:
82             new_row, new_col = row + dr, col + dc
83
84             if (self.is_valid(new_row, new_col) and
85                 (new_row, new_col) not in visited):
86
87                 new_coins = coins + self.grid[new_row][new_col]
88                 new_visited = visited | frozenset([(new_row, new_col)])
89                 new_path = path + [(new_row, new_col)]
90                 new_battery = battery_left - 1
91
92                 upper_bound = self.calculate_upper_bound(new_row, new_col, new_coins,
93                                                         new_visited, new_battery)
94
95                 if upper_bound > best_coins:
96                     heapq.heappush(pq, (-new_coins, -upper_bound, new_row, new_col,
97                                     new_battery, new_path, new_visited))
98
99     return best_coins, best_path
100
101 def solve_with_battery_and_cost(self, start_row: int = 0, start_col: int = 0,
102     battery_charge: int = 100, step_cost: int = 10) -> Tuple[int,
103     if not self.is_valid(start_row, start_col):
104         return 0, []
105
106     initial_coins = self.grid[start_row][start_col]
107     remaining_battery = battery_charge - step_cost
108
109     pq = [(-initial_coins,
110           -self.calculate_upper_bound_with_cost(start_row, start_col, initial_coins,
111                                                  set(), remaining_battery, step_cost),
112           start_row, start_col, remaining_battery,
113           [(start_row, start_col)],
114           frozenset([(start_row, start_col)]))]
115
116     best_coins = initial_coins
117     best_path = [(start_row, start_col)]
118
119     while pq:
120         neg_coins, neg_bound, row, col, battery_left, path, visited = heapq.heappop(pq)
```

powershell X

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
0	4	3	0	2	3	2	0	0	3
2	1	2	0	1	3	3	3	3	1
1	2	1	2	0	1	3	3	3	1
0	1	1	2	3	4	0	3	2	

Ln 11, Col 13 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

GoldMineProblem.py U ×

BranchAndBound > GoldMineProblem.py > ...

```
14 class CoinCollectingBranchBound:
101 def solve_with_battery_and_cost(self, start_row: int = 0, start_col: int = 0,
119 while pq:
120     neg_coins, neg_bound, row, col, battery_left, path, visited = heapq.heappop(pq)
121     coins = -neg_coins
122
123     if -neg_bound <= best_coins:
124         continue
125
126     if coins > best_coins:
127         best_coins = coins
128         best_path = path[:]
129
130     if battery_left < step_cost:
131         continue
132
133     for dr, dc in self.directions:
134         new_row, new_col = row + dr, col + dc
135
136         if (self.is_valid(new_row, new_col) and
137             (new_row, new_col) not in visited):
138
139             new_coins = coins + self.grid[new_row][new_col]
140             new_visited = visited | frozenset([(new_row, new_col)])
141             new_path = path + [(new_row, new_col)]
142             new_battery = battery_left - step_cost
143
144             upper_bound = self.calculate_upper_bound_with_cost(new_row, new_col, new_coins,
145                                                         new_visited, new_battery,
146                                                         best_coins, best_path)
147
148             if upper_bound > best_coins:
149                 heapq.heappush(pq, (-new_coins, -upper_bound, new_row, new_col,
150                                     new_battery, new_path, new_visited))
151
152     return best_coins, best_path
153
154 def calculate_upper_bound_with_cost(self, row: int, col: int, collected: int,
155                                     visited: set, battery_left: int, step_cost: int) -> int:
156     if battery_left < step_cost:
157         return collected
158
159     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
0 4 3 0 2 3 2 0 0 3
2 1 2 0 1 3 3 3 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 3 2

Ln 11, Col 13 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

GoldMineProblem.py U X

BranchAndBound > GoldMineProblem.py > ...

```
14 class CoinCollectingBranchBound:
153 def calculate_upper_bound_with_cost(self, row: int, col: int, collected: int,
158 max_steps = battery_left // step_cost
159 remaining_coins = []
160
161 queue = [(row, col, 0)]
162 reachable = set()
163 temp_visited = set([(row, col)])
164
165 while queue and max_steps > 0:
166 curr_row, curr_col, dist = queue.pop(0)
167 if dist >= max_steps:
168 continue
169
170 for dr, dc in self.directions:
171 new_row, new_col = curr_row + dr, curr_col + dc
172 if (self.is_valid(new_row, new_col) and
173 (new_row, new_col) not in temp_visited and
174 (new_row, new_col) not in visited):
175
176 temp_visited.add((new_row, new_col))
177 reachable.add((new_row, new_col))
178 queue.append((new_row, new_col, dist + 1))
179
180 for r, c in reachable:
181 if self.grid[r][c] > 0:
182 remaining_coins.append(self.grid[r][c])
183
184 return collected + sum(remaining_coins)
185
186 def load_grid_from_file(filename: str) -> List[List[int]]:
187 try:
188 with open(filename, 'r') as file:
189 lines = file.readlines()
190
191 n = int(lines[0].strip())
192
193 grid = []
194 for i in range(1, n + 1):
195 row = [int(x) for x in lines[i].strip().split('t')]
196 grid.append(row)
197
```

powershell X

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 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 3 2

Ln 11, Col 13 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

GoldMineProblem.py U X

BranchAndBound > GoldMineProblem.py > ...
186 def load_grid_from_file(filename: str) -> List[List[int]]:
196 | grid.append(row)
197
198 | return grid
199 except FileNotFoundError:
200 | print(f"Error: File {filename} not found.")
201 | return []
202 except Exception as e:
203 | print(f"Error reading file {filename}: {e}")
204 | return []
205
206 if __name__ == "__main__":
207 | import os
208
209 | script_dir = os.path.dirname(os.path.abspath(__file__))
210
211 | test_files = ["coins-n5.txt", "coins-n10.txt"]
212
213 | for filename in test_files:
214 | | filepath = os.path.join(script_dir, filename)
215 | | print(f"\n{' '*50}")
216 | | print(f"Testing with {filename}")
217 | | print(f"{' '*50}")
218
219 | | grid = load_grid_from_file(filepath)
220
221 | | if not grid:
222 | | | print(f"Failed to load grid from {filename}")
223 | | | continue
224
225 | | print(f"Grid size: {len(grid)}x{len(grid[0])}")
226 | | print("Grid:")
227 | | for row in grid:
228 | | | print('\t'.join(map(str, row)))
229
230 | | solver = CoinCollectingBranchBound(grid)
231
232 | | print("\n=== Battery Life Method ===")
233 | | coins1, path1 = solver.solve_with_battery_limit(0, 0, battery_life=8)
234 | | print(f"Max coins collected: {coins1}")
235 | | print(f"Path: {path1}")
236

powershell X

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 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 3 2

Ln 11, Col 13 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

GoldMineProblem.py U ×

BranchAndBound > GoldMineProblem.py > ...
225 print(f'Grid size: {len(grid)}x{len(grid[0])}')
226 print("Grid:")
227 for row in grid:
228 print('\t'.join(map(str, row)))
229
230 solver = CoinCollectingBranchBound(grid)
231
232 print("\n=== Battery Life Method ===")
233 coins1, path1 = solver.solve_with_battery_limit(0, 0, battery_life=8)
234 print(f"Max coins collected: {coins1}")
235 print(f"Path: {path1}")
236
237 print("\n=== Battery Charge + Step Cost Method ===")
238 coins2, path2 = solver.solve_with_battery_and_cost(0, 0, battery_charge=100, step_cost
239 print(f"Max coins collected: {coins2}")
240 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.UNIVERSI
DAD/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.UNIVERSI
DAD/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 3 1
1 2 1 2 0 1 3 3 3 1
0 1 1 2 3 4 0 3 3 2

Ln 11, Col 13 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

FileEditSelectionViewGoRunTerminalHelp

2.Advanced Algorithms

GoldMineProblem.py U X

BranchAndBound > GoldMineProblem.py > ...
425 print(f'Grid size: {len(grid)}x{len(grid)}')
426 print("Grid:")
427 for row in grid:
428 print('\t'.join(map(str, row)))
429
430 solver = CoinCollectingBranchAndBound()
431
432 print("\n=== Battery Life Method
433 coins1, path1 = solver.solve_with_battery_life()
434 print(f"Max coins collected: {coins1}")
435 print(f"Path: {path1}")
436
437 print("\n=== Battery Charge + Step Cost Method
438 coins2, path2 = solver.solve_with_battery_charge_and_step_cost()
439 print(f"Max coins collected: {coins2}")
440 print(f"Path: {path2}")

powershell X
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*

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