Design Analysis and Algorithem, Lab Assignment 5

Bhuvana Kanakam - SE21UCSE035

October 6, 2023

Question

Assignment:

- Devise an algorithm to maximize the number of thieves caught by deploying a limited number of policemen while respecting certain constraints.

Problem Description: You are given a grid (2D) representing a city. The grid consists of 'P' (policemen) and 'T' (thieves) characters. Policemen can catch thieves in adjacent cells either horizontally or vertically up to k unit distance, but not diagonally. The objective is to deploy a limited number of policemen to catch as many thieves as possible without exceeding the available policeman count. Each policeman can catch only one thief. A policeman cannot catch a thief who is more than k units away from the policeman.

Instructions:

- Write a Python/JAVA/C/C++ program to solve the Policemen vs. Thieves problem using a greedy algorithm.
- Your program should aim to maximize the number of thieves caught while adhering to the following rules:
- You are given a grid (2D list) representing the city, where 'P' represents a policeman and 'T' represents a thief.
- You have a limited number of policemen available to deploy (policemen_count).
- Policemen can catch thieves in adjacent cells (horizontally or vertically but not diagonally).
- Implement a function that returns the maximum number of thieves that can be caught given the grid with random placement of thieves and policemen.
- Test your program with various city grid configurations and different numbers of available policemen.

Answer

Below is the Python code :

```
import matplotlib.pyplot as plt
import numpy as np

def maxThievesCaught(grid, policemen_count, k):
    m, n = len(grid), len(grid[0])
    policemen = []
    thieves = []

    for i in range(m):
        for j in range(n):
            if grid[i][j] == 'P':
                policemen.append((i, j))
            elif grid[i][j] == 'T':
                thieves.append((i, j))

    caught_thieves = 0
    caught_policemen = set()

    for _ in range(policemen_count):
        min_distance = float('inf')
        chosen_policeman = None

        for i, thief in enumerate(thieves):
            if i not in caught_policemen:
                for policeman in policemen:
                    distance = abs(thief[0] - policeman[0]) + abs(thief[1] - policeman[1
                    if distance <= k and distance < min_distance:
                        min_distance = distance
                        chosen_policeman = i

        if chosen_policeman is not None:
            caught_thieves += 1
            caught_policemen.add(chosen_policeman)

    return caught_thieves

def run_experiments(grid, max_policemen, k):
    x_values = []
    y_values = []

    for policemen_count in range(1, max_policemen + 1):
        caught_thieves = maxThievesCaught(grid, policemen_count, k)
        x_values.append(policemen_count)
        y_values.append(caught_thieves)

    return x_values, y_values

grid = [
    ['P', 'T', 'P', 'P'],
    ['T', 'P', 'T', 'T'],
    ['P', 'T', 'P', 'P']
]

max_policemen = 5
k = 1
x_values, y_values = run_experiments(grid, max_policemen, k)

plt.figure(figsize=(8, 6))
plt.plot(x_values, y_values, marker='o', linestyle='-')
plt.title('Policemen␣vs.␣Thieves␣Caught')
plt.xlabel('Number␣of␣Policemen')
plt.ylabel('Number␣of␣Thieves␣Caught')
plt.grid(True)
plt.show()
```

Explanation

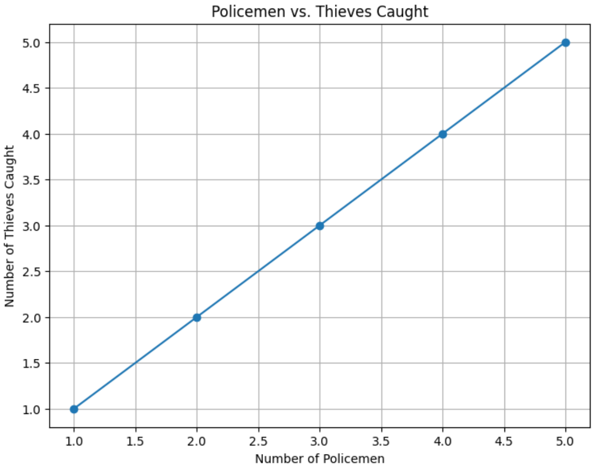

Figure 1: Enter Caption