RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	892
Course Title	Distributed and Cloud Computing
Semester/Year	W2024

Instructor	Dr. Muhammad Jaseemuddin

Lab No.	1

Lab Title	Concurrency vs. Parallelism

Submission Date	February 9th, 2024
Due Date	February 9th, 2024

Student Name	Student ID	Signature*
Abdulrehman Khan	500968727	A.K.

**By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work.*

Introduction

In this lab, we explored the concepts of concurrency and parallelism using Python's threading and multiprocessing modules. The objective was to write programs to process data from Land Mine Detector Robots and disarm mines efficiently.

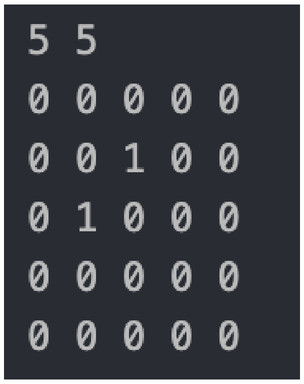
Part 1: (Drawing the path of the Rovers) Data Structures and Functions:

RoverController Class: Manages the state and movement of each rover.

- **fetch_rover_moves:** Retrieves the commands for a rover from the given public API.
- **move_forward:** Moves the rover forward while handling edge cases and mine detection.
- **turn_left** and **turn_right:** Rotate the rover's direction.
- **check_mine:** Checks if the rover is on a mine.
- **initiate_rover_traversal:** Initiates the traversal of a rover and writes its path to a file.

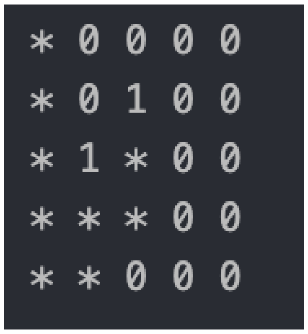
Main Script: Reads the map and initiates rover traversal sequentially and in parallel.

Map.txt:



```
5 5
0 0 0 0 0
0 0 1 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 0 0
```

Sample path_1.txt:



```
* 0 0 0 0
* 0 1 0 0
* 1 * 0 0
* * * 0 0
* * 0 0 0
```

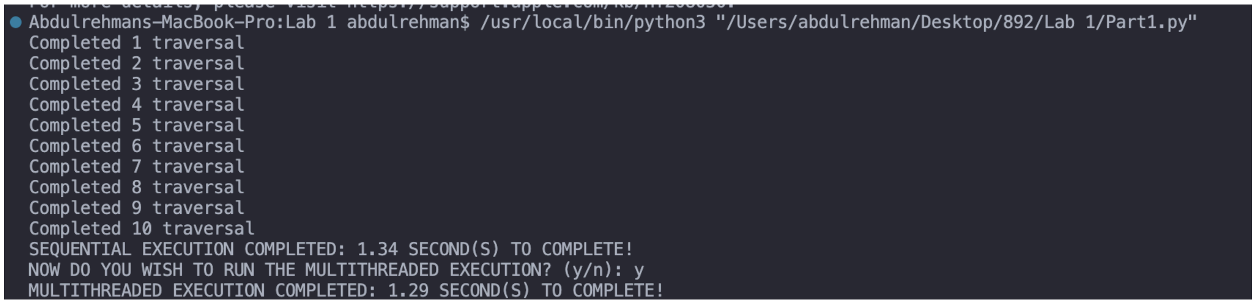
Initial State: The rover starts at position (0, 0) facing south, indicated by the direction 'S'.

First Set of Commands:
"MMMMRMLRRRLRMLMRMLMMMLMMRMMLDMLMMMMMLRLLRDMMMLDMLRDMRLMRMRMRRMLRLLRMDRMRDLMDLM"

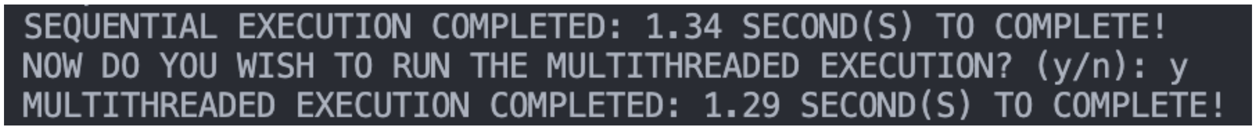
M: The rover moves forward.
R: The rover turns right.
L: The rover turns left.
D: The rover digs.

The rover successfully completed its traversal through the given land space, marked with potential mines, based on a sequence of commands obtained from the API. Despite encountering obstacles, the rover effectively navigates the terrain, avoiding mines and marking its path with '*'. The provided path_1.txt accurately reflects the rover's movement and interactions with the environment, showcasing its ability to follow commands.

Part 1 Terminal Output:



```
For more details, please visit https://support.apple.com/kb/HT208050.
Abdulrehmans-MacBook-Pro:Lab 1 abdulrehman$ /usr/local/bin/python3 "/Users/abdulrehman/Desktop/892/Lab 1/Part1.py"
  Completed 1 traversal
  Completed 2 traversal
  Completed 3 traversal
  Completed 4 traversal
  Completed 5 traversal
  Completed 6 traversal
  Completed 7 traversal
  Completed 8 traversal
  Completed 9 traversal
  Completed 10 traversal
  SEQUENTIAL EXECUTION COMPLETED: 1.34 SECOND(S) TO COMPLETE!
  NOW DO YOU WISH TO RUN THE MULTITHREADED EXECUTION? (y/n): y
  MULTITHREADED EXECUTION COMPLETED: 1.29 SECOND(S) TO COMPLETE!
```



```
SEQUENTIAL EXECUTION COMPLETED: 1.34 SECOND(S) TO COMPLETE!
NOW DO YOU WISH TO RUN THE MULTITHREADED EXECUTION? (y/n): y
MULTITHREADED EXECUTION COMPLETED: 1.29 SECOND(S) TO COMPLETE!
```

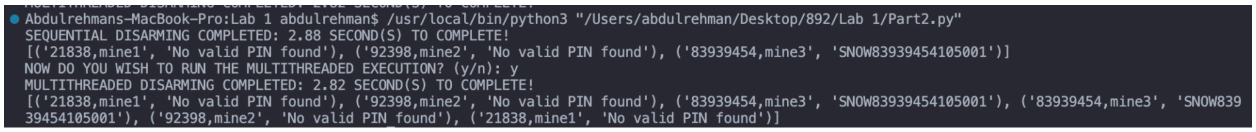
The sequential execution completed in approximately 1.34 seconds.This timing represents the total time taken to process the commands for all 10 rovers sequentially.

The multithreaded execution completed faster, taking 1.29 seconds. This indicates that utilizing Python's Threading module for parallel execution led to an improvement in processing time compared to the sequential approach.

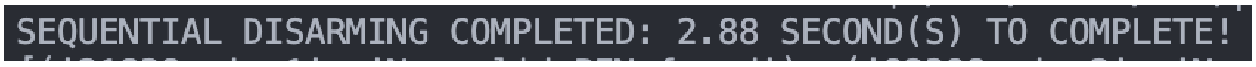
Part 2: (Digging mines) Data Structures and Functions

- **find_valid_pin Function**: Searches for a valid PIN for a given mine based on its serial number, a provided prefix (SNOW), and if the hashed pin starts with “00000”.
- **sequential_disarm_mines Function**: Disarms mines sequentially, reading the provided mines.txt file.
- **threaded_disarm_mines Function**: Disarms mines using multithreading, reading the aforementioned mines.txt file as well.

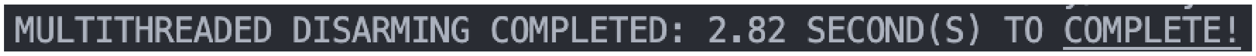
Part 2 Terminal Output:



```
Abdulrehmans-MacBook-Pro:Lab 1 abdulrehman$ /usr/local/bin/python3 "/Users/abdulrehman/Desktop/892/Lab 1/Part2.py"
SEQUENTIAL DISARMING COMPLETED: 2.88 SECOND(S) TO COMPLETE!
[('21838,mine1', 'No valid PIN found'), ('92398,mine2', 'No valid PIN found'), ('83939454,mine3', 'SNOW83939454105001')]
NOW DO YOU WISH TO RUN THE MULTITHREADED EXECUTION? (y/n): y
MULTITHREADED DISARMING COMPLETED: 2.82 SECOND(S) TO COMPLETE!
[('21838,mine1', 'No valid PIN found'), ('92398,mine2', 'No valid PIN found'), ('83939454,mine3', 'SNOW83939454105001'), ('83939454,mine3', 'SNOW839
39454105001'), ('92398,mine2', 'No valid PIN found'), ('21838,mine1', 'No valid PIN found')]
```



```
SEQUENTIAL DISARMING COMPLETED: 2.88 SECOND(S) TO COMPLETE!
```



```
MULTITHREADED DISARMING COMPLETED: 2.82 SECOND(S) TO COMPLETE!
```

Sequential Disarming: The execution time was 2.88 second(s). Firstly, for "mine1" and "mine2", no valid PINs were found. Finally, for "mine3", a valid PIN ('SNOW83939454105001') was discovered and disarmed.

Multithreaded Disarming: The execution time was 2.82 second(s), completed faster. Similar to the sequential approach, the multithreaded execution found no valid PINs for "mine1" and "mine2". Valid PINs were discovered for "mine3" were disarmed using a valid PIN ('SNOW83939454105001') found in the multithreaded approach.

Conclusion

Through this lab, we gained insights into the differences between concurrency and parallelism and their applications in real-world scenarios. We observed significant performance improvements when utilizing parallel processing, especially when dealing with large datasets or computationally intensive tasks. Understanding these concepts and employing the appropriate techniques can lead to more efficient and scalable software solutions.