0x01. Lockboxes

Algorithm Python

In a nutshell…

- **Auto QA review:** 13.0/13 mandatory
- **Altogether: 100.0%**
  - Mandatory: 100.0%
  - Optional: no optional tasks

Must Know

For this project, you will need a solid understanding of several key concepts in order to develop a solution that can efficiently determine if all boxes can be opened. Here's a list of concepts and resources that will be instrumental in tackling this project:

Concepts Needed:

1. **Lists and List Manipulation**:
   - Understanding how to work with lists, including accessing elements, iterating over lists, and modifying lists dynamically.
   - Python Lists (Python Official Documentation) (/rltoken/TtGNy9p1p1d0O5G1rdY1Aw)
2. **Graph Theory Basics**:
   - Although not explicitly required, knowledge of graph theory (especially concepts related to traversal algorithms like Depth-First Search or Breadth-First Search) can be very helpful in solving this problem, as the boxes and keys can be thought of as nodes and edges in a graph.
   - Graph Theory (Khan Academy) (/rltoken/eVcYI8g-6nF0Na46xnRdhw)
3. **Algorithmic Complexity**:
   - Understanding the time and space complexity of your solution is important, as it can help in writing more efficient algorithms.
   - Big O Notation (GeeksforGeeks) (/rltoken/01qym1qAJUkLrb47PvqnKg)
4. **Recursion**:

- Some solutions might require a recursive approach to traverse through the boxes and keys.
- Recursion in Python (Real Python) (/rltoken/zpEuvv0l9EHohlx-HwiAAA)

5. **Queue and Stack**:
   - Knowing how to use queues and stacks is crucial if implementing a breadth-first search (BFS) or depth-first search (DFS) algorithm to traverse through the keys and boxes.
   - Python Queue and Stack (GeeksforGeeks) (/rltoken/CQLm4RJrdwyo2DAcNCtwIA)
6. **Set Operations**:
   - Understanding how to use sets for keeping track of visited boxes and available keys can optimize the search process.
   - Python Sets (Python Official Documentation) (/rltoken/zkmtaPqAbKyxx41kRw7ulA)

By reviewing these concepts and utilizing these resources, you will be well-equipped to develop an efficient solution for this project, applying both your algorithmic thinking and Python programming skills.

Additional Resources

- Mock Technical Interview (/rltoken/TJoFJhWeEGoIlqMpwBn7Pg)

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using `python3` (version 3.4.3)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/python3`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should be documented
- Your code should use the `PEP 8` style (version 1.7.x)
- All your files must be executable

Tasks

0. Lockboxes

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

You have `n` number of locked boxes in front of you. Each box is numbered sequentially from `0` to `n - 1` and each box may contain keys to the other boxes.

Write a method that determines if all the boxes can be opened.

- Prototype: `def canUnlockAll(boxes)`
- `boxes` is a list of lists
- A key with the same number as a box opens that box
- You can assume all keys will be positive integers
  - There can be keys that do not have boxes

- The first box `boxes[0]` is unlocked
- Return `True` if all boxes can be opened, else return `False`

```
carrie@ubuntu: /0x01-lockboxes$ cat main_0.py
#!/usr/bin/python3
canUnlockAll = __import__('0-lockboxes').canUnlockAll

boxes = [[1], [2], [3], [4], []]
print(canUnlockAll(boxes))

boxes = [[1, 4, 2, 3], [0, 4, 1], [2], [0, 4, 1], [3], [4, 1], [5, 6]]
print(canUnlockAll(boxes))

carrie@ubuntu:~/0x01-lockboxes$
```

```
carrie@ubuntu:~/0x01-lockboxes$ ./main_0.py
True
True
False
carrie@ubuntu:~/0x01-lockboxes$
```

Repo:

- GitHub repository: `alx-interview`
- Directory: `0x01-lockboxes`
- File: `0-lockboxes.py`

Check submission | View results

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
Reviews, I can make corrections to review(/)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Copyright © 2024 ALX, All rights reserved.
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)