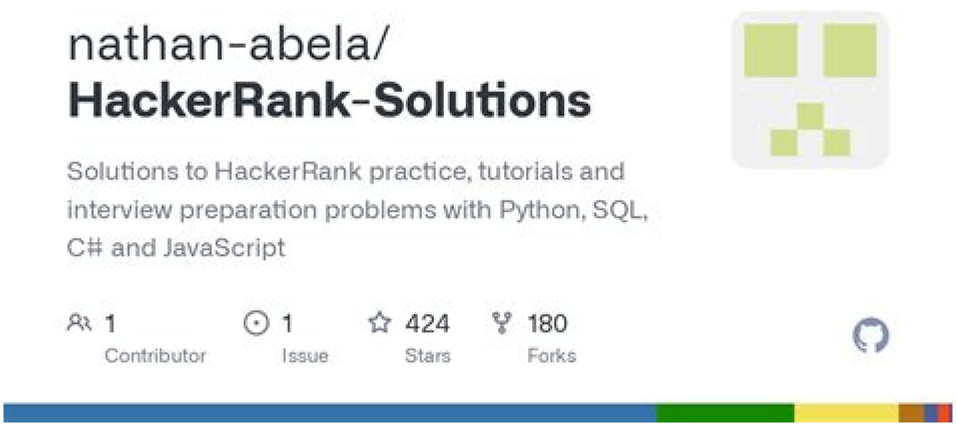
Busy Intersection Hackerrank Solution Github


nathan-abela/
HackerRank-Solutions
Solutions to HackerRank practice, tutorials and interview preparation problems with Python, SQL, C# and JavaScript
1 Contributor
1 Issue
424 Stars
180 Forks

Busy Intersection HackerRank Solution GitHub is a topic that resonates with many programming enthusiasts and competitive coders. HackerRank is a popular platform for coding challenges and contests, where programmers can test their skills and improve their coding proficiency. One such challenge is the "Busy Intersection" problem, which tests one's ability to apply algorithmic thinking and data structure knowledge to solve real-world scenarios. This article delves into the details of the Busy Intersection problem on HackerRank, the strategies for solving it, and how to find solutions on GitHub.

Understanding the Busy Intersection Problem

The "Busy Intersection" problem typically involves simulating a traffic scenario where multiple vehicles approach an intersection. The objective is often to determine how many vehicles can efficiently cross the intersection without causing congestion, or to calculate the time taken for all vehicles to clear the intersection.

Problem Statement

While the specifics may vary, a common version of the problem might involve:

- A set number of vehicles approaching an intersection.
- Each vehicle has a defined entry time and a duration for which it will occupy the intersection.
- The goal is to determine the maximum number of vehicles that can pass through the intersection during a specified time frame without overlapping.

Input and Output Format

Typically, the input format for the Busy Intersection problem might look like this:

- The first line contains an integer `n`, representing the number of vehicles.
- The next `n` lines each contain two integers: `entry_time` and `duration` for each vehicle.

The output would usually be a single integer indicating the maximum number of vehicles that can pass through the intersection.

Approaching the Solution

To solve the Busy Intersection problem efficiently, you can adopt a systematic approach. Here are the general steps:

1. Parse the Input: Read the number of vehicles and their respective entry times and durations.
2. Generate Event Points: For each vehicle, create two events: one for the entry and one for the exit (entry_time and entry_time + duration).
3. Sort Events: Sort these events based on their time, ensuring that entry events are processed before exit events if they occur at the same time.
4. Simulate the Intersection: Use a counter to track the number of vehicles in the intersection at any given time and maximize the count while ensuring no two vehicles overlap.

Algorithm: Greedy Approach

A greedy algorithm can efficiently solve this problem. Here's a brief overview of the approach:

- Initialization: Start with an empty intersection and a count of vehicles.
- Iterate through Events: Loop through the sorted event list:
- If you encounter an entry event and the intersection is empty, allow the vehicle to enter and increment the count.
- If you encounter an exit event, remove the vehicle from the intersection.
- Result: The count at the end of the iteration will give the maximum number of vehicles that can cross.

Implementation Example

Here is a simple Python implementation of the Busy Intersection solution:

```python
def busy_intersection(vehicles):
events = []

Generate events for entry and exit
for entry_time, duration in vehicles:
events.append((entry_time, 'enter'))
events.append((entry_time + duration, 'exit'))

Sort events: first by time, then by type of event
events.sort(key=lambda x: (x[0], x[1] == 'exit'))

max_vehicles = 0
current_vehicles = 0

Simulate the intersection
for time, event in events:
if event == 'enter':
current_vehicles += 1
max_vehicles = max(max_vehicles, current_vehicles)
else:
current_vehicles -= 1

return max_vehicles

Example usage
vehicles = [(1, 2), (2, 1), (3, 3)]
print(busy_intersection(vehicles)) Output: 2
```

Finding Solutions on GitHub

GitHub is a treasure trove of coding solutions, collaborative projects, and repositories dedicated to various coding challenges, including those from HackerRank. To find solutions for the Busy Intersection problem, you can follow these steps:

1. Search GitHub: Use the search bar to look for “Busy Intersection HackerRank solution”. You can also add programming languages to narrow down results (e.g., "Busy Intersection HackerRank solution Python").

2. Explore Repositories: Look through various repositories that might contain solutions for multiple HackerRank problems. Developers often group similar challenges together.

3. Check for Readme Files: Many GitHub repositories include a Readme file that outlines how to use the code, the problem statement, and sometimes even the test cases.

4. Look for Forks and Stars: Check the number of forks and stars on repositories to gauge their popularity and reliability.

5. Contribute: If you come up with an efficient solution or improvements on existing solutions, consider contributing back to the community by creating a pull request.

Best Practices for Coding Challenges

When tackling coding challenges like the Busy Intersection problem, it is essential to adopt best practices:

- Read the Problem Statement Carefully: Ensure you understand the requirements and constraints before jumping into coding.
- Plan Your Solution: Spend a few minutes planning your approach and writing pseudocode if necessary.
- Test with Edge Cases: Consider edge cases and test your solution against them.
- Optimize: Once you have a working solution, think about ways to optimize it for better performance.
- Comment Your Code: Write comments to explain the logic, especially if your solution is complex.

Conclusion

The Busy Intersection problem is an excellent example of how algorithmic thinking can be applied to real-world scenarios. By understanding the problem statement, breaking it down into manageable steps, and employing a greedy algorithm, one can efficiently solve this challenge. Moreover, platforms like GitHub serve as valuable resources for finding solutions, sharing knowledge, and collaborating with others in the coding community. Embracing these practices not only enhances your coding skills but also prepares you for more complex challenges in the future.

Frequently Asked Questions

What is the 'Busy Intersection' problem on HackerRank?

The 'Busy Intersection' problem on HackerRank involves finding the number of intersections that are busy based on car movements in a grid-like city layout, requiring efficient algorithmic solutions to handle potentially large input sizes.

Where can I find solutions to the 'Busy Intersection' problem on GitHub?

You can find various implementations and solutions to the 'Busy Intersection' problem by searching for repositories on GitHub with keywords like 'Busy Intersection HackerRank solution' or by exploring popular coding repositories.

What programming languages are commonly used for the 'Busy Intersection' solutions on GitHub?

Common programming languages used for the 'Busy Intersection' solutions include Python, Java, C++, and JavaScript, with many solutions demonstrating different approaches to the problem.

Are there any specific algorithmic techniques used in 'Busy Intersection' solutions?

Yes, solutions often utilize algorithmic techniques such as coordinate compression, sweep line algorithms, and data structures like segment trees or binary indexed trees to efficiently count busy intersections.

How can I improve my solution for the 'Busy Intersection' problem?

To improve your solution, focus on optimizing time complexity by using efficient data structures, reducing redundant calculations, and ensuring you understand the underlying mathematical principles of intersection counting.

What are common pitfalls when solving the 'Busy Intersection' problem?

Common pitfalls include misunderstanding the input format, miscalculating the intersections based on vehicle paths, and not optimizing for large datasets which can lead to time limit exceeded errors.

Can I collaborate with others on GitHub for solving 'Busy Intersection'?

Yes, GitHub allows for collaboration through features like forking repositories, creating pull requests, and discussing issues, making it a great platform to work with others on solutions to the 'Busy Intersection' problem.

Busy Intersection Hackerrank Solution Github

steam□□□□□□□□□□,□□□□□□□ - □□□□
Mar 31, 2022 · □□□□□□□□□□□□□□□□□□□ steam□□□□□□□Steam□□□

be busy doing sth□be busy with sth □□□□□□□_□□□□
2.□□□□□be busy doing sth□□□□□□□□□□□□□be busy with sth□□□□□□□□□□□□ -She is always busy with handling the company's daily affairs.

be busy to do sth.□be busy doing□□□ - □□□□
be busy doing□□□□□□□□□□□□□□ be too busy to do sth□□□□□□□ □□□□□□ □□be busy doing The plan for the new book is on ice at the moment.I've been busy with too many other things. □□□□□□□□□□□□□,□□□□□□□□□□□□. He is ...

busy □□□□ - □□□□
Mar 1, 2008 · busy adj. □□□, □□□ [□] (□□)□□□ □□□, □□□□□ □□□□□ (in) □□□, □□□ (□□)□□□, □□□□□□□, □□□□ a busy Sunday □□□□□□□□ a busy street □□□□□ get a busy signal □ (□□□)□□□□□□□□ be busy in another's affair □□□□□□□□ □□□□ busy oneself with □□ ...

be busy in □be busy with□□□□□□□ - □□□□
□□□□□□□□ 1□be busy in□busy□□□□□□□□□□□“□□”□□□□□□□□□□□□□□□□□□□□“□□□”□□□□ busy□“□□□; □□□”□□□□□□□□□□ 2□be busy with□busy□□□□□□□□□ □□□

□□□□□□□□□□□□□□□**BUSY**□□□□□□□_□□□□
□□□□□□□□□□□□□□□BUSY□□□□□□□□ □□□□□BUSY□□□1□□□□□□□□□□□□□□□□□□□□□□□□

busy□□□□□□□□ - □□□□
busy□□□□□□□□□□□□ 1□busy□□□□□□□□□□□□□□□ be busy with sth □□ □□□ be busy doing sth □□□□□ busy oneself with sth □□□□□□□□ 2□busy □□□□□□□□□□□□□□□□□□□□□□□□□□□busy□□□□□□□□□□□□□□□□□□ 3□□busy□□□□□□□□□□□□ ...

“□□□□□□□□□□□□□□□□”□□□□□□□□□□□□
“□□□□□□□□□□□□□□□□”□□□□□□□□□□□□□□□□□!□□□□□□□□□□□□□,□□□□□.□□□□Sorry!The subscriber you dialed can not be connected for the moment, please redial la

Linux□□□□□□□ *Text file busy* □□□□□□□□_□□□□
Mar 8, 2025 · □Linux□□□□□□□”Text file busy”□□□□□□□□□□□□□□□□ □□fuser□□□□□□□□□□□□□ □□□□□□fuser□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Xfbdev□□□□□fuser Xfbdev□ □□□□□□□□□□fuser□□□□□□□□□□□□□□□□□□root□□□□□ ...

be busy doing□□be busy to do□□□□□□? - □□□□
"be busy doing sth" □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□"I am busy doing my homework"□□□□□□□□□ "be busy to do sth" □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

steam□□□□□□□□□□,□□□□□□□ - □□□□
Mar 31, 2022 · □□□□□□□□□□□□□□□□□□□ steam□□□□□□□Steam□□□ ...

be busy doing sth□be busy with sth □□□□□□□_□□□□
2.□□□□□be busy doing sth□□□□□□□□□□□□□be busy with sth□□□□□□□□□□□ -She is always busy with handling the company's daily affairs.

be busy to do sth.□be busy doing□□□ - □□□□
be busy doing□□□□□□□□□□□□□□ be too busy to do sth□□□□□□□□ □□□□□□ □□be busy doing The plan for the new book is on ice at the moment.I've been busy with ...

busy □□□□ - □□□□
Mar 1, 2008 · busy adj. □□□, □□□ [□] (□□)□□□ □□□, □□□□□ □□□□□ (in) □□□, □□□ (□□)□□□, □□□□□□□, □□□□ a busy Sunday □□□□□□□□ a busy ...

be busy in □be busy with□□□□□□□ - □□□□
□□□□□□□□ 1□be busy in□busy□□□□□□□□□□□“□□”□□□□□□□□□□□□□□□□□□□□“□□□”□□□□ busy□“□□□; □□□”□□□□ □□□□□□ 2□be ...

□□□□□□□□□□□□□□□□*BUSY*□□□□□□□_□□□□
□□□□□□□□□□□□□□□□BUSY□□□□□□□ □□□□□BUSY□□□1□□□□ □□□ ...

busy□□□□□□□□ - □□□□
busy□□□□□□□□□□□ 1□busy□□□□□□□□□□□□□□ be busy with sth □□ □□□ be busy doing sth □□□□□ busy oneself with sth □□□□□□□□ 2□busy □□□□ ...

“□□□□□□□□□□□□□□□□”□□□□□□□□□□□
“□□□□□□□□□□□□□□□□”□□□□□□□□□□□□□□□□□!□□□□□□□□□□□□□,□□□□□.□□□□Sorry!The subscriber you dialed can not be connected for the ...

Linux□□□□□□□ Text file busy □□□□□□□□_□□□□
Mar 8, 2025 · □Linux□□□□□□□”Text file busy”□□□□□□□□□□□□□□□□ □□fuser□□□□□□□□□□□□□ □□□□□□fuser□□ □□□□□□□□□□□□□□□□ ...

be busy doing□□be busy to do□□□□□□? - □□□□
"be busy doing sth" □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□"I am busy doing my homework"□□□□□□□ □□□ "be busy to do sth" □□□□□□□□□ ...

Discover the Busy Intersection HackerRank solution on GitHub! Improve your coding skills with our clear explanations and practical examples. Learn more now!

Back to Home