busy intersection hackerrank solution

Busy Intersection Hackerrank Solution: A Step-by-Step Guide

busy intersection hackerrank solution is a popular challenge that has caught the attention of many programmers looking to sharpen their problem-solving skills on the Hackerrank platform. This problem involves simulating traffic flow at an intersection controlled by traffic lights, and determining how many cars safely pass through without accidents. If you've been searching for an efficient and clear approach to tackle this coding puzzle, you're in the right place. Let's dive deep into understanding the problem, breaking down the logic, and exploring an optimized solution.

Understanding the Busy Intersection Problem

Before jumping into the coding part, it's essential to comprehend what the problem is asking. At its core, the busy intersection problem simulates a scenario where cars arrive at a traffic light, and the light cycles between green and red. Our goal is to figure out how many cars can pass through safely without causing a collision.

The problem typically provides:

- A sequence or array indicating the arrival time of each car.
- The duration of the green light and red light phases.
- Rules about how cars move during these phases.

The challenge lies in accurately simulating the passage of cars while respecting the timing constraints of the traffic light, and avoiding any overlapping or unsafe crossings.

Key Concepts in the Busy Intersection Challenge

- **Traffic light cycles:** The traffic light alternates between green and red at fixed intervals.
- **Car arrival times:** Each car arrives at a certain timestamp.
- **Passing logic:** Only cars arriving during the green light can pass immediately, while others must wait.
- **Collision prevention:** Cars cannot pass simultaneously if they would collide.

By understanding these components, you can then design an algorithm that processes these inputs and calculates the number of cars that pass without incident.

Breaking Down the Busy Intersection Hackerrank

Solution

To solve this problem effectively, start by outlining the algorithm in simple terms:

1. **Track the traffic light state** at any given time.
2. **Maintain a queue or list** of cars waiting to pass.
3. **Simulate the passage of time** and move cars through the intersection only when the light is green.
4. **Ensure no two cars pass simultaneously** if that would cause a crash.
5. **Count cars that pass safely** and ignore those that cannot.

Step 1: Modeling the Traffic Light Cycle

The traffic light cycle alternates between green and red phases. For example, if the green light lasts `g` seconds and the red light lasts `r` seconds, the total cycle length is `g + r`. Using modular arithmetic, you can determine the state of the light at any given time `t` by calculating `t % (g + r)`:

- If `t % (g + r) < g`, the light is green.
- Otherwise, the light is red.

This simple calculation helps you decide whether cars arriving at time `t` can pass or must wait.

Step 2: Processing Car Arrivals

You’ll receive an array or list of car arrival times. The main task is to iterate through these times and determine if each car can pass immediately or if it must wait.

Here’s what you need to consider:

- A car arriving during the green light phase can pass if no other car is currently passing.
- If a car arrives during a red light or the intersection is occupied, it must wait until the next green phase.
- Multiple cars arriving at the same time should be handled carefully to avoid collisions.

Step 3: Simulating the Intersection Flow

One common approach is to use a queue to represent cars waiting to cross. You simulate the passing of time, and at each time step:

- Check if the light is green.
- If yes, allow one car from the queue to pass.
- Increment your count of safely passed cars.

- Update the state of the intersection (occupied or free).

This simulation continues until all cars have either passed or are stuck indefinitely.

Implementing the Busy Intersection Hackerrank Solution in Code

Here’s a high-level overview of how you might implement the solution in your favorite programming language:

```python
def busy_intersection(arrivals, green, red):
cycle = green + red
cars_passed = 0
current_time = 0
waiting_queue = []

# Sort the arrival times if not already sorted
arrivals.sort()
i = 0
n = len(arrivals)

while i < n or waiting_queue:
# Determine if light is green at current_time
light_state = current_time % cycle < green

# Add arriving cars at current_time to the queue
while i < n and arrivals[i] <= current_time:
waiting_queue.append(arrivals[i])
i += 1

# If light is green and queue is not empty, pass one car
if light_state and waiting_queue:
waiting_queue.pop(0)
cars_passed += 1

current_time += 1

return cars_passed
```

This example uses a straightforward time simulation, checking each second to see if the light is green and if cars can pass. Although simple, this method is effective for moderate input sizes.

Optimizing the Solution

While the above method works, it may not be efficient for very large datasets due to its step-by-step time simulation. To optimize:

- **Use event-driven simulation:** Jump directly to the next arrival time or the next green light start instead of iterating through every second.
- **Batch process cars arriving during continuous green phases:** If multiple cars arrive during a green phase, calculate how many can pass based on green duration and arrival times, reducing unnecessary iteration.
- **Avoid unnecessary queue operations:** Use pointers or indexes instead of popping from lists to improve performance.

By implementing these optimizations, your solution can handle larger inputs swiftly and pass Hackerrank’s time constraints.

Tips for Tackling Busy Intersection Problems on Hackerrank

If you’re preparing for Hackerrank challenges or technical interviews, here are some handy tips when dealing with busy intersection-like problems:

- **Understand the problem constraints carefully.** Knowing the input size can help you decide whether a simulation or a mathematical approach is better.
- **Think in terms of cycles and modular arithmetic.** Many traffic light problems rely on repeating cycles, so modular operations can simplify logic.
- **Consider edge cases:** What happens if multiple cars arrive exactly at the green light’s start? Or if no cars arrive during a green phase? Handling these cases prevents bugs.
- **Write clean and readable code:** Use meaningful variable names like `green_duration`, `red_duration`, and `arrival_times` to make your code easy to follow.
- **Test your solution with sample inputs:** Before submitting, test with various scenarios including minimum and maximum input sizes, and boundary conditions.

Common Pitfalls to Avoid

- **Ignoring the waiting queue:** Sometimes, programmers forget to account for cars that arrive during a red light and must wait until the next green phase.
- **Overcomplicating the timing logic:** Keep the time calculations straightforward to avoid off-by-one errors.
- **Failing to sort arrivals:** Unsynchronized arrival times can lead to incorrect simulation results.

Exploring Variations and Related Problems

The busy intersection problem is part of a broader category of traffic simulation and scheduling challenges. Similar problems you might encounter include:

- **Traffic light synchronization:** Optimizing multiple intersections to minimize waiting times.
- **Car collision detection:** Predicting accidents based on speed and trajectory.
- **Queue management in real-time systems:** Handling resources shared among multiple processes.

Understanding the busy intersection solution lays the groundwork for tackling these more complex scenarios.

If you have been stuck on the busy intersection Hackerrank solution or want to deepen your understanding of traffic flow simulations, approaching the problem methodically—by understanding cycles, modeling arrival times, and simulating traffic—can make a huge difference. With practice, you’ll not only solve this puzzle but also gain valuable insights into event-driven simulations and optimization techniques useful across many programming challenges.

Frequently Asked Questions

What is the Busy Intersection problem on HackerRank about?

The Busy Intersection problem on HackerRank involves simulating traffic light signals and car movements at an intersection to determine how many cars get stuck due to red lights.

What data structures are commonly used to solve the Busy Intersection problem?

Queues are commonly used to model the cars waiting at each direction of the intersection, as they allow processing cars in the order they arrive.

How do you simulate the traffic lights in the Busy Intersection solution?

Traffic lights are simulated by iterating over a string representing the light sequence, switching the allowed directions accordingly and letting cars pass if their corresponding light is green.

What is the time complexity of the Busy Intersection solution?

The time complexity is generally O(n), where n is the length of the traffic light sequence, since each car and light change is processed once.

Can the Busy Intersection problem be solved using arrays instead of queues?

Yes, arrays or lists can be used, but queues provide a more natural and efficient way to process cars in FIFO order.

How do you handle cars arriving simultaneously from different directions in the Busy Intersection problem?

Cars arriving simultaneously are enqueued to their respective direction queues in the order of arrival, and processed based on the current green light direction.

What programming languages are commonly used to solve the Busy Intersection problem on HackerRank?

Common languages include Python, Java, C++, and JavaScript, as they provide built-in queue data structures and easy string manipulation.

Is it necessary to track the total wait time of cars in the Busy Intersection problem?

Typically, tracking total wait time is not required; the main goal is to count how many cars get stuck or pass through the intersection.

How do you determine when the simulation should stop in the Busy Intersection solution?

The simulation stops after processing all traffic light signals and when no more cars are waiting to pass from any direction.

Are there any edge cases to consider when solving the Busy Intersection problem?

Yes, edge cases include no cars at the intersection, continuous green signals for one direction, or all directions having cars but no green light to pass.

Additional Resources

Mastering the Busy Intersection Hackerrank Solution: A Professional Review

busy intersection hackerrank solution represents a classic challenge frequently encountered by programmers aiming to test their algorithmic skills in traffic simulation and logical problem-solving. This problem, often presented in coding platforms like Hackerrank, requires candidates to analyze traffic light signals and vehicle queues at a busy four-way intersection. The goal is to determine whether a traffic jam occurs based on input sequences describing car arrivals and signal states. Understanding how to approach this problem methodically not only helps in cracking coding interviews but also deepens one’s grasp of real-time system simulations.

Understanding the Busy Intersection Problem

At its core, the busy intersection problem involves simulating traffic flow through an intersection controlled by traffic lights. The input typically consists of a series of 4-character strings representing the state of traffic lights and the presence of cars at each of the four roads converging at the intersection. Each character corresponds to a specific lane and can indicate either a green light with a car present, a green light without a car, or a red light with or without a car.

The main task is to analyze whether any car proceeds through the intersection illegally or if a gridlock occurs, causing a traffic jam. The challenge lies in correctly interpreting the light signals and the car positions, then determining if the rules of traffic flow are violated.

Algorithmic Approach to the Busy Intersection Hackerrank Solution

Developing an efficient busy intersection Hackerrank solution involves breaking down the problem into manageable checks. The solution requires examining the current state of the traffic lights and the presence of cars to identify potential conflicts.

Key Points to Consider:

1. **Traffic Light Representation:** Each character in the input string represents a road direction (north, east, south, west). Characters 'G' indicate a green light, and 'R' indicates red. Accompanying these are indicators of whether a car is waiting.

2. **Car Movement Rules:** Cars can proceed only if their traffic light is green. However, cars present at a red light should not move.

3. **Conflict Detection:** The primary challenge is to detect if a car is moving when it shouldn’t or if the configuration of cars and lights could cause collisions, leading to a traffic

jam.

To illustrate, the solution algorithm involves iterating through each character in the input string, checking if a green light coincides with a car, and then verifying if any conflicting rules are broken by checking neighboring lanes or dependent directions.

Step-by-Step Solution Walkthrough

Step 1: Parsing Input

The solution starts by reading the integer `n`, representing the number of signal states to process. For each state, a 4-character string is read, with each character indicating the state of the lane:

- Uppercase letters like 'G' or 'R' specify light status.
- Lowercase letters or dots may be used to indicate the presence or absence of cars.

Step 2: Identifying Cars at Green Lights

For each lane, check if a car is present and if the light is green. A green light with a car means the vehicle is potentially moving.

Step 3: Checking for Potential Traffic Jams

The critical logic involves determining if a car at a green light can cause a jam by conflicting with other cars. This involves:

- Checking if any car is waiting at a red light that would be crossed by a car moving through a green light.
- Assessing if cars are simultaneously waiting to turn into lanes occupied by other vehicles.
- Evaluating if any lane’s green light with a car leads to a conflict with cars at other lanes based on traffic rules.

The solution concludes by outputting "YES" if a traffic jam or illegal movement is detected, or "NO" otherwise.

Optimization and Edge Cases

When implementing the busy intersection Hackerrank solution, efficiency is crucial, especially when dealing with multiple input states. The algorithm should run in O(n) time complexity, iterating through each input string once.

Edge Cases to Consider:

- All lights are red, but cars are present—should not cause a jam as no cars move.
- Multiple cars at green lights simultaneously—verify if their paths intersect or conflict.
- No cars present—result should be "NO" since no movement occurs.
- Cars present only at red lights—no movement expected; thus, no jam.

Comparison with Other Traffic Simulation Problems

The busy intersection problem differs from broader traffic simulation challenges by focusing on a fixed, small-scale intersection with a limited number of lanes and discrete signal states. Unlike large-scale traffic network simulations, this problem prioritizes logical checks rather than continuous modeling.

In contrast, problems like the “Traffic Light Control System” or “Vehicle Queue Management” involve dynamic updates over time and larger datasets, requiring advanced data structures and algorithms such as priority queues or graph traversal.

Practical Applications and Learning Outcomes

Solving the busy intersection Hackerrank problem enhances one’s ability to:

- Analyze real-world systems through logical abstraction.
- Develop conditional checks and simulate state machines.
- Improve proficiency in string manipulation and iteration.
- Prepare for technical interviews where scenario-based problem solving is essential.

Moreover, the problem offers insight into traffic management concepts and how algorithmic thinking can be applied to everyday urban challenges.

Sample Code Outline

Here is a simplified outline of the solution logic in pseudocode:

```plaintext
for each signal_state in input_states:
jam_detected = false
for each lane in signal_state:
if lane has green light and car present:
if conflicting conditions with other lanes:
jam_detected = true
break
```

```
print "YES" if jam_detected else "NO"
```

This approach highlights the importance of clear and concise checks for each lane’s condition, ensuring accurate detection of jams without unnecessary complexity.

Final Thoughts on Busy Intersection Hackerrank Solution

The busy intersection Hackerrank solution exemplifies a problem where analytical reasoning and attention to detail are paramount. Its relevance extends beyond coding contests into practical areas such as traffic engineering and automated control systems. For developers and interviewees, mastering this problem hones skills that are transferable to complex system design and debugging.

In exploring this challenge, programmers not only sharpen their algorithmic capabilities but also gain appreciation for the intricate dynamics governing seemingly simple real-life scenarios like intersection traffic flow.

Busy Intersection Hackerrank Solution

Related to busy intersection hackerrank solution

BUSY Definition & Meaning - Merriam-Webster busy, industrious, diligent, assiduous, sedulous mean actively engaged or occupied. busy chiefly stresses activity as opposed to idleness or leisure
BUSY | definition in the Cambridge English Dictionary Would you like to come over on the weekend, or are you busy? I'm busy on Tuesday morning but we could meet in the afternoon
BUSY Definition & Meaning | Busy definition: actively and attentively engaged in work or a pastime.. See examples of BUSY used in a sentence
Busy - definition of busy by The Free Dictionary 1. actively and attentively engaged, esp. in work. 2. not at leisure; otherwise engaged: He's busy and can't see you. 3. full of activity: a busy life. 4. (of a telephone line) in use. 5. meddlesome;
busy - Wiktionary, the free dictionary The director cannot see you now: he's busy. Her telephone has been busy all day. He is busy with piano practice. Ramzi is busy getting ready for meetings
BUSY definition in American English | Collins English Dictionary When you are busy, you are working hard or concentrating on a task, so that you are not free to do anything else. What is it? I'm busy. They are busy preparing for a hectic day's activity on

busy | meaning of busy in Longman Dictionary of Contemporary Dennis had a very busy schedule with all of these commitments. busy road For this reason, start by choosing a relatively quiet environment rather than a busy road
busy - Dictionary of English Busy means actively employed, temporarily or habitually: a busy official. Diligent suggests earnest and constant effort or application, and usually connotes fondness for, or enjoyment of, what
Busy Definition & Meaning - YourDictionary Busy definition: Engaged in activity, as work; occupied
busy adjective - Definition, pictures, pronunciation and usage notes Definition of busy adjective in Oxford Advanced American Dictionary. Meaning, pronunciation, picture, example sentences, grammar, usage notes, synonyms and more
BUSY Definition & Meaning - Merriam-Webster busy, industrious, diligent, assiduous, sedulous mean actively engaged or occupied. busy chiefly stresses activity as opposed to idleness or leisure
BUSY | definition in the Cambridge English Dictionary Would you like to come over on the weekend, or are you busy? I'm busy on Tuesday morning but we could meet in the afternoon
BUSY Definition & Meaning | Busy definition: actively and attentively engaged in work or a pastime.. See examples of BUSY used in a sentence
Busy - definition of busy by The Free Dictionary 1. actively and attentively engaged, esp. in work. 2. not at leisure; otherwise engaged: He's busy and can't see you. 3. full of activity: a busy life. 4. (of a telephone line) in use. 5. meddlesome;
busy - Wiktionary, the free dictionary The director cannot see you now: he's busy. Her telephone has been busy all day. He is busy with piano practice. Ramzi is busy getting ready for meetings
BUSY definition in American English | Collins English Dictionary When you are busy, you are working hard or concentrating on a task, so that you are not free to do anything else. What is it? I'm busy. They are busy preparing for a hectic day's activity on
busy | meaning of busy in Longman Dictionary of Contemporary Dennis had a very busy schedule with all of these commitments. busy road For this reason, start by choosing a relatively quiet environment rather than a busy road
busy - Dictionary of English Busy means actively employed, temporarily or habitually: a busy official. Diligent suggests earnest and constant effort or application, and usually connotes fondness for, or enjoyment of, what
Busy Definition & Meaning - YourDictionary Busy definition: Engaged in activity, as work; occupied
busy adjective - Definition, pictures, pronunciation and usage notes Definition of busy adjective in Oxford Advanced American Dictionary. Meaning, pronunciation, picture, example sentences, grammar, usage notes, synonyms and more
chatgpt-chinese-gpt/ChatGPT-sites-guide - GitHub 1 day ago ChatGPT □□□□□□□□□ □□□□□ https://chat.openai.com □□□□□□ □□□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□ □□□□□
Chat GPT □□□□ChatGPT □□□□□□□□□ GPT □□ 1 day ago □□□□□2025/09/20 □□□□ ChatGPT □□□□□□□□□□□ □□□□ GPT-4 □□□□□□□ □□□□□□□□□□□□□□ ChatGPT □□□□□□□□□□□□□□□□□□□
chatgpt-chinese-gpt/ChatGPT-Chinese-version - GitHub 2 days ago ChatGPT □□□□□□□□□□□□□4□ □□□□□. Contribute to chatgpt-chinese-gpt/ChatGPT-Chinese-version development by creating an account on
GitHub - chatgpt-china-gpt/ChatGPT_CN: □10□□□□□ □□□□ ChatGPT □□□□□□□□□□□□ GPT-4□4o□o1□o3 □ DeepSeek R1 □□□□□□□□□ □□□□□□□□□□□□□□□ ChatGPT □□□□□□□□□□□□□□
GitHub - chatgpt-zh/Chinese-ChatGPT-Tutorial: ChatGPT □□□ ChatGPT □□□□□□□□□ □□□□□□□ □□ □□□□□ chat.openai.com □□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□
ChatGPT□□□□□□□□□□□□□GPT-4□GPT4o - GitHub 2 days ago □□□□ ChatGPT □□□□□□□□□□□□ GPT-4 □□ □□□□□ □□□□□□□□□□□□□□ ChatGPT □□□□□□□□□□□□□□□□□□□ ChatGPT□□□□ □ □
10 cách dùng ChatGPT - OpenAI Chat miễn phí tại Việt Nam ChatGPT (OpenAI chat gpt)

đang trở thành một trào lưu tại Việt Nam. Đây là trí tuệ nhân tạo AI sử dụng trên trình duyệt web và chưa có ứng dụng chính thức. Sau đây là
ChatGPT □□□□GPT-5 □□□□□□□□ GPT-4□GPT-5 3 days ago ChatGPT □□□□□□ GPT-4□ □□□□□□□□□□ □ GPT-4 □ GPT-3.5□ ChatGPT □□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□
GitHub - 0xk1h0/ChatGPT_DAN: ChatGPT DAN, Jailbreaks prompt NOTE: As of 20230711, the DAN 12.0 prompt is working properly with Model GPT-3.5 All contributors are constantly investigating clever workarounds that allow us to utilize the full
GitHub - openai/gpt-oss: gpt-oss-120b and gpt-oss-20b are two Try gpt-oss Guides Model card OpenAI blog Download gpt-oss-120b and gpt-oss-20b on Hugging Face Welcome to the gpt-oss series, OpenAI's open-weight models designed for
BUSY Definition & Meaning - Merriam-Webster busy, industrious, diligent, assiduous, sedulous mean actively engaged or occupied. busy chiefly stresses activity as opposed to idleness or leisure
BUSY | definition in the Cambridge English Dictionary Would you like to come over on the weekend, or are you busy? I'm busy on Tuesday morning but we could meet in the afternoon
BUSY Definition & Meaning | Busy definition: actively and attentively engaged in work or a pastime.. See examples of BUSY used in a sentence
Busy - definition of busy by The Free Dictionary 1. actively and attentively engaged, esp. in work. 2. not at leisure; otherwise engaged: He's busy and can't see you. 3. full of activity: a busy life. 4. (of a telephone line) in use. 5. meddlesome;
busy - Wiktionary, the free dictionary The director cannot see you now: he's busy. Her telephone has been busy all day. He is busy with piano practice. Ramzi is busy getting ready for meetings
BUSY definition in American English | Collins English Dictionary When you are busy, you are working hard or concentrating on a task, so that you are not free to do anything else. What is it? I'm busy. They are busy preparing for a hectic day's activity on
busy | meaning of busy in Longman Dictionary of Contemporary Dennis had a very busy schedule with all of these commitments. busy road For this reason, start by choosing a relatively quiet environment rather than a busy road
busy - Dictionary of English Busy means actively employed, temporarily or habitually: a busy official. Diligent suggests earnest and constant effort or application, and usually connotes fondness for, or enjoyment of, what
Busy Definition & Meaning - YourDictionary Busy definition: Engaged in activity, as work; occupied
busy adjective - Definition, pictures, pronunciation and usage notes Definition of busy adjective in Oxford Advanced American Dictionary. Meaning, pronunciation, picture, example sentences, grammar, usage notes, synonyms and more

Related to busy intersection hackerrank solution

City officials say they're working on solutions for this busy intersection in Tradition (WPTV-TV1mon) PORT ST. LUCIE, Fla. — You could make your voice heard Tuesday night on possible improvements at a busy intersection in Tradition. Port St. Lucie leaders held a public meeting to discuss the
City officials say they're working on solutions for this busy intersection in Tradition (WPTV-TV1mon) PORT ST. LUCIE, Fla. — You could make your voice heard Tuesday night on possible improvements at a busy intersection in Tradition. Port St. Lucie leaders held a public meeting to discuss the
City officials say they're working on solutions for this busy intersection in Tradition (Hosted on MSN1mon) Port St. Lucie city leaders held a public meeting to discuss the intersection at Village and Tradition parkway. Judge Orders Release of $95M in Blow to Trump David Boreanaz recalls Betty White
City officials say they're working on solutions for this busy intersection in Tradition (Hosted

on MSN1mon) Port St. Lucie city leaders held a public meeting to discuss the intersection at Village and Tradition parkway. Judge Orders Release of $95M in Blow to Trump David Boreanaz recalls Betty White