Problem 3508: Implement Router

Problem Information

Difficulty: Medium
Acceptance Rate: 39.36%
Paid Only: No
Tags: Array, Hash Table, Binary Search, Design, Queue, Ordered Set

Problem Description

Design a data structure that can efficiently manage data packets in a network router. Each data packet consists of the following attributes:

* `source`: A unique identifier for the machine that generated the packet. * `destination`: A unique identifier for the target machine. * `timestamp`: The time at which the packet arrived at the router.

Implement the `Router` class:

`Router(int memoryLimit)`: Initializes the Router object with a fixed memory limit.

* `memoryLimit` is the **maximum** number of packets the router can store at any given time. * If adding a new packet would exceed this limit, the **oldest** packet must be removed to free up space.

`bool addPacket(int source, int destination, int timestamp)`: Adds a packet with the given attributes to the router.

* A packet is considered a duplicate if another packet with the same `source`, `destination`, and `timestamp` already exists in the router. * Return `true` if the packet is successfully added (i.e., it is not a duplicate); otherwise return `false`.

`int[] forwardPacket()`: Forwards the next packet in FIFO (First In First Out) order.

* Remove the packet from storage. * Return the packet as an array `[source, destination, timestamp]`. * If there are no packets to forward, return an empty array.

`int getCount(int destination, int startTime, int endTime)`:

* Returns the number of packets currently stored in the router (i.e., not yet forwarded) that have the specified destination and have timestamps in the inclusive range `[startTime, endTime]`.

Note that queries for `addPacket` will be made in non-decreasing order of `timestamp`.

Example 1:

Input: ["Router", "addPacket", "addPacket", "addPacket", "addPacket", "addPacket", "forwardPacket", "addPacket", "getCount"] [[3], [1, 4, 90], [2, 5, 90], [1, 4, 90], [3, 5, 95], [4, 5, 105], [], [5, 2, 110], [5, 100, 110]]

Output: [null, true, true, false, true, true, [2, 5, 90], true, 1]

Explanation

Router router = new Router(3); // Initialize Router with memoryLimit of 3. router.addPacket(1, 4, 90); // Packet is added. Return True. router.addPacket(2, 5, 90); // Packet is added. Return True. router.addPacket(1, 4, 90); // This is a duplicate packet. Return False. router.addPacket(3, 5, 95); // Packet is added. Return True router.addPacket(4, 5, 105); // Packet is added, `[1, 4, 90]` is removed as number of packets exceeds memoryLimit. Return True. router.forwardPacket(); // Return `[2, 5, 90]` and remove it from router. router.addPacket(5, 2, 110); // Packet is added. Return True. router.getCount(5, 100, 110); // The only packet with destination 5 and timestamp in the inclusive range `[100, 110]` is `[4, 5, 105]`. Return 1.

Example 2:

Input: ["Router", "addPacket", "forwardPacket", "forwardPacket"] [[2], [7, 4, 90], [], []]

Output: [null, true, [7, 4, 90], []]

Explanation

Router router = new Router(2); // Initialize `Router` with `memoryLimit` of 2. router.addPacket(7, 4, 90); // Return True. router.forwardPacket(); // Return `[7, 4, 90]`. router.forwardPacket(); // There are no packets left, return `[]`.

Constraints:

* `2 <= memoryLimit <= 105` * `1 <= source, destination <= 2 * 105` * `1 <= timestamp <= 109` * `1 <= startTime <= endTime <= 109` * At most `105` calls will be made to `addPacket`, `forwardPacket`, and `getCount` methods altogether. * queries for `addPacket` will be made in non-decreasing order of `timestamp`.

Code Snippets

C++:

```
class Router {
public:
Router(int memoryLimit) {

}

bool addPacket(int source, int destination, int timestamp) {

}

vector<int> forwardPacket() {

}

int getCount(int destination, int startTime, int endTime) {

}
};

/**
* Your Router object will be instantiated and called as such:
* Router* obj = new Router(memoryLimit);
* bool param_1 = obj->addPacket(source,destination,timestamp);
* vector<int> param_2 = obj->forwardPacket();
* int param_3 = obj->getCount(destination,startTime,endTime);
*/
```

Java:

```
class Router {

public Router(int memoryLimit) {

}

public boolean addPacket(int source, int destination, int timestamp) {

}

public int[] forwardPacket() {

}

public int getCount(int destination, int startTime, int endTime) {

}
}

/**
* Your Router object will be instantiated and called as such:
* Router obj = new Router(memoryLimit);
* boolean param_1 = obj.addPacket(source,destination,timestamp);
* int[] param_2 = obj.forwardPacket();
* int param_3 = obj.getCount(destination,startTime,endTime);
*/
```

Python3:

```
class Router:

def __init__(self, memoryLimit: int):


def addPacket(self, source: int, destination: int, timestamp: int) -> bool:


def forwardPacket(self) -> List[int]:


def getCount(self, destination: int, startTime: int, endTime: int) -> int:
```

```
# Your Router object will be instantiated and called as such:
# obj = Router(memoryLimit)
# param_1 = obj.addPacket(source,destination,timestamp)
# param_2 = obj.forwardPacket()
# param_3 = obj.getCount(destination,startTime,endTime)
```