Data Structure Design...

video-23

Leetcode - 3508

Medium

→ codestorywithmiK

→ CSwithMIK

→ codestorywithMIK

WeekendWithMIK

@WeekendWithMIK [redacted]

Welcome to WeekendWithMIK 🎥✨ >

[redacted]

LIFE BEHIND THE SCENES...

News

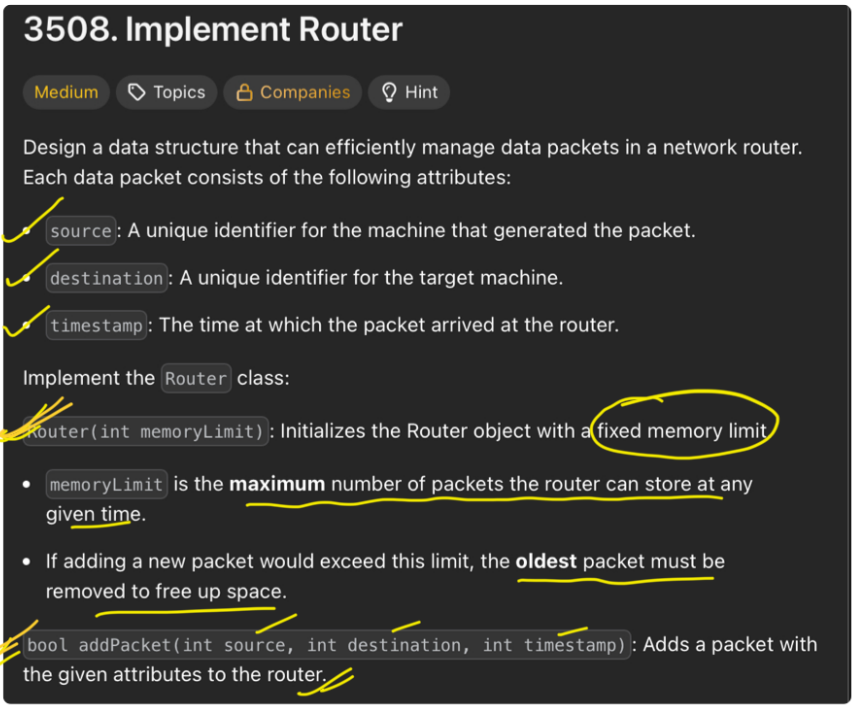
+ Tech

Motivation :-

3 months of hard work & dedication is enough to change your entire life, make you an entiely different person.

Are you willing to give 3 months?
(Oct, Nov, Dec)

MIK...



3508. Implement Router

Medium | Topics | Companies | Hint

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.

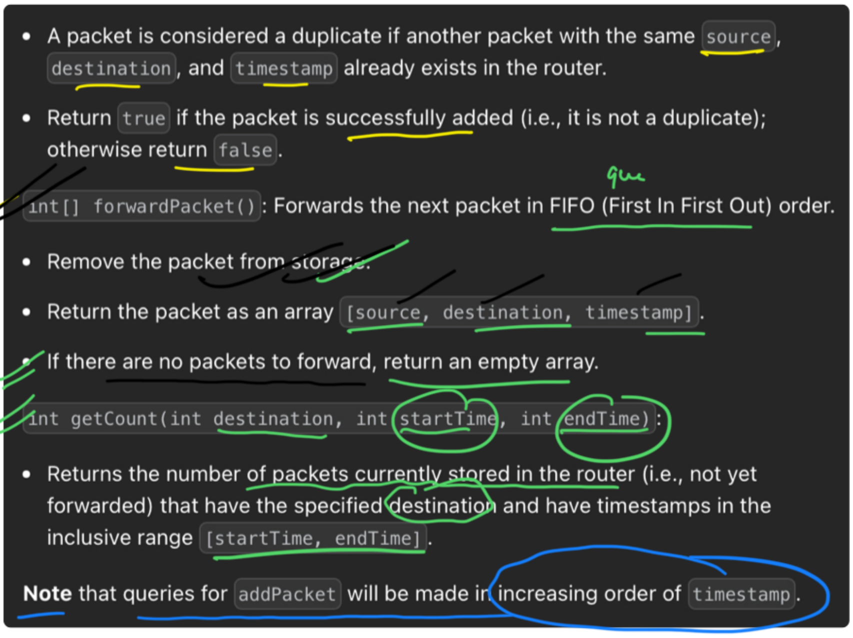
Implement the `Router` class:

`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.

MAX_SIZE=3



- 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`.

que

`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)`:

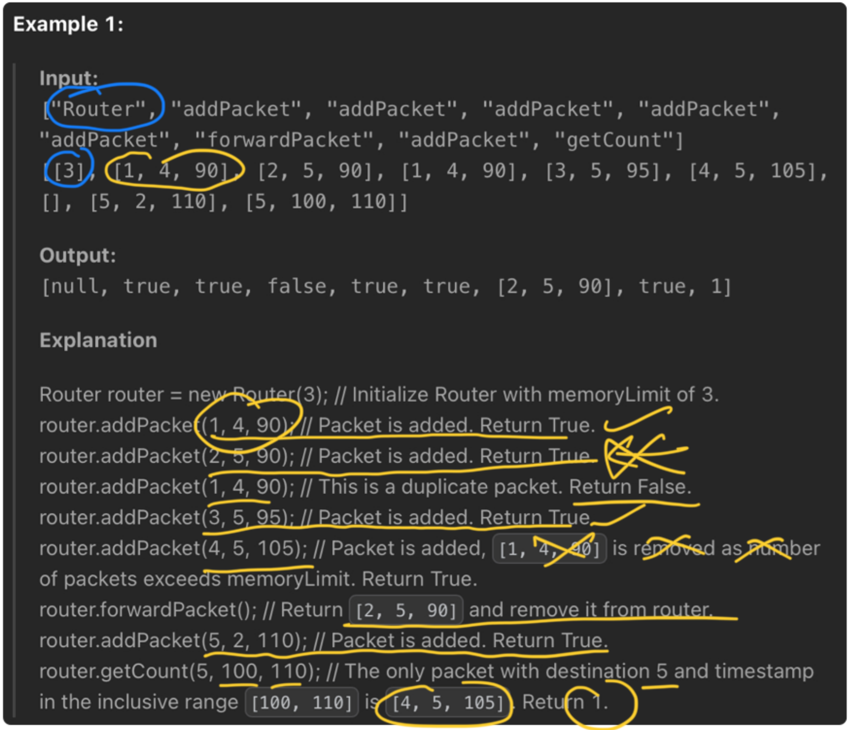
- Returns the number of packets currently stored in the router (i.e., not yet forwarded) that have the specified destination and have timestamps in the inclusive range `[startTime, endTime]`.

Note that queries for `addPacket` will be made in increasing order of `timestamp`.

queue

FIFO

{ }



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]
```

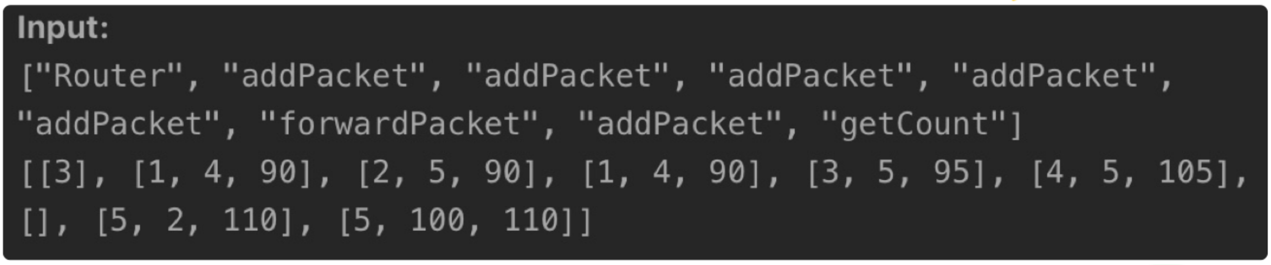
Explanation

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.

MAX-SIZE = 3

Thought Process

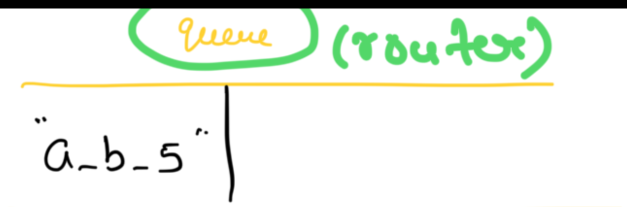
Queue



Input:

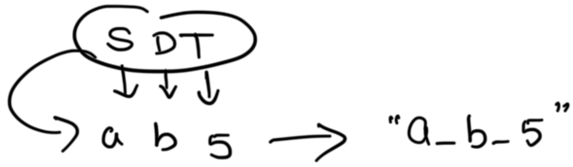
```
["Router", "addPacket", "addPacket", "addPacket", "addPacket",
"addPacket", "forwardPacket", "addPacket", "getCount"]
[[3], [1, 4, 90], [2, 5, 90], [1, 4, 90], [3, 5, 95], [4, 5, 105],
[], [5, 2, 110], [5, 100, 110]]
```

map



"a_b_5" → {a, b, 5}

string → vector



⇒ queue <string> que ;

⇒ packetStore → unor_map <string , vector<int>>;

```
bool addPacket(S, D, T) {

    string Key = to_string(S) + "_" + to-str("D") + "_"+to_st(T)

    if (packetStore.find (key) != pac(S.end())
            return False;

    if (que.size() >= MAX_SIZE) {
            forwardPacket ( ) ;
    }

    packeStore [Key] = {S, D, T};
    que.push (Key);
    destinMap [D]. push_back (T);
    return True;
}
```

```
~~vector~~<int> forwardPacket ( ) {

        if (packetStore.empty()) {

                return { };
        }

        string Key = que.front( );

        que.pop();

        vector<int> packetDetails = packetStore[Key];
               int D              = PacketDetai[1];
        packetStore.erase(Key);

        destMap[D]. erase (destMap[D].begin());

}       return PacketDetails ;
```

{S, D, T}

D → {T1, T2, T3}

```
~~int~~ getCount ( int D , int startTime, int endTime) {
```

queue <String>

packetStore<string, vector>

queu

"3_5_95"	"4_5_105"	"5_2_110"

packetStore

Key	PacketDetails
"3_5_95"	{3, 5, 95}
"4_5_105"	{4, 5, 105}
"5_2_110"	{5, 2, 110}

map
(Destination → {timestamps}

O(1)

5 → {95, 96, 105, 106.. }

sorted.

Binary Search

getCount (5, 95, 106)
D st Et

map
(distination → timestamps)

O(1)

5 → {92, 95, 105, 112}

6 → { . . . . }

7 → { . . }

$\{\overset{0}{92}, \underbrace{\boxed{\overset{\downarrow}{\overset{1}{95}}, \overset{2}{105},}} \overset{\downarrow}{\overset{3}{112}}\}$ sorted.

st. time = $\boxed{95}$ → lower_bound (95) ; $\overset{i}{\boxed{1}}$

end Time = $\boxed{106}$ → upper_bound (106) ; $\underset{j}{\boxed{3}}$

Count = $3 - 1 = 2$ ✓ log(K)

$(j - i)$

=