Problem 3408: Design Task Manager

Problem Information

Difficulty: Medium
Acceptance Rate: 49.32%
Paid Only: No
Tags: Hash Table, Design, Heap (Priority Queue), Ordered Set

Problem Description

There is a task management system that allows users to manage their tasks, each associated with a priority. The system should efficiently handle adding, modifying, executing, and removing tasks.

Implement the `TaskManager` class:

* `TaskManager(vector<vector<int>>& tasks)` initializes the task manager with a list of user-task-priority triples. Each element in the input list is of the form `[userId, taskId, priority]`, which adds a task to the specified user with the given priority.

* `void add(int userId, int taskId, int priority)` adds a task with the specified `taskId` and `priority` to the user with `userId`. It is **guaranteed** that `taskId` does not _exist_ in the system.

* `void edit(int taskId, int newPriority)` updates the priority of the existing `taskId` to `newPriority`. It is **guaranteed** that `taskId` _exists_ in the system.

* `void rmv(int taskId)` removes the task identified by `taskId` from the system. It is **guaranteed** that `taskId` _exists_ in the system.

* `int execTop()` executes the task with the **highest** priority across all users. If there are multiple tasks with the same **highest** priority, execute the one with the highest `taskId`. After executing, the****`taskId`**** is **removed** from the system. Return the `userId` associated with the executed task. If no tasks are available, return -1.

Note that a user may be assigned multiple tasks.

Example 1:

Input: ["TaskManager", "add", "edit", "execTop", "rmv", "add", "execTop"] [[[[1, 101, 10], [2, 102, 20], [3, 103, 15]]], [4, 104, 5], [102, 8], [], [101], [5, 105, 15], []]

Output: [null, null, null, 3, null, null, 5]

Explanation

TaskManager taskManager = new TaskManager([[1, 101, 10], [2, 102, 20], [3, 103, 15]]); // Initializes with three tasks for Users 1, 2, and 3. taskManager.add(4, 104, 5); // Adds task 104 with priority 5 for User 4. taskManager.edit(102, 8); // Updates priority of task 102 to 8. taskManager.execTop(); // return 3. Executes task 103 for User 3. taskManager.rmv(101); // Removes task 101 from the system. taskManager.add(5, 105, 15); // Adds task 105 with priority 15 for User 5. taskManager.execTop(); // return 5. Executes task 105 for User 5.

Constraints:

* `1 <= tasks.length <= 105` * `0 <= userId <= 105` * `0 <= taskId <= 105` * `0 <= priority <= 109` * `0 <= newPriority <= 109` * At most `2 * 105` calls will be made in **total** to `add`, `edit`, `rmv`, and `execTop` methods. * The input is generated such that `taskId` will be valid.

Code Snippets

C++:

```
class TaskManager {
public:
TaskManager(vector<vector<int>>& tasks) {

}

void add(int userId, int taskId, int priority) {

}

void edit(int taskId, int newPriority) {

}
```

```
void rmv(int taskId) {

}

int execTop() {

}
};

/**
* Your TaskManager object will be instantiated and called as such:
* TaskManager* obj = new TaskManager(tasks);
* obj->add(userId,taskId,priority);
* obj->edit(taskId,newPriority);
* obj->rmv(taskId);
* int param_4 = obj->execTop();
*/
```

Java:

```
class TaskManager {

public TaskManager(List<List<Integer>> tasks) {

}

public void add(int userId, int taskId, int priority) {

}

public void edit(int taskId, int newPriority) {

}

public void rmv(int taskId) {

}

public int execTop() {

}
```

```
}

/**
* Your TaskManager object will be instantiated and called as such:
* TaskManager obj = new TaskManager(tasks);
* obj.add(userId,taskId,priority);
* obj.edit(taskId,newPriority);
* obj.rmv(taskId);
* int param_4 = obj.execTop();
*/
```

Python3:

```
class TaskManager:

def __init__(self, tasks: List[List[int]]):


def add(self, userId: int, taskId: int, priority: int) -> None:


def edit(self, taskId: int, newPriority: int) -> None:


def rmv(self, taskId: int) -> None:


def execTop(self) -> int:



# Your TaskManager object will be instantiated and called as such:
# obj = TaskManager(tasks)
# obj.add(userId,taskId,priority)
# obj.edit(taskId,newPriority)
# obj.rmv(taskId)
# param_4 = obj.execTop()
```