2353. Design a Food Rating System

Hint

Medium 527 101

Companies

Design a food rating system that can do the following:

- **Modify** the rating of a food item listed in the system.
- Return the highest-rated food item for a type of cuisine in the system.

Implement the `FoodRatings` class:

- `FoodRatings(String[] foods, String[] cuisines, int[] ratings)` Initializes the system. The food items are described by `foods`, `cuisines` and `ratings`, all of which have a length of `n`.
  - `foods[i]` is the name of the i^{th} food,
  - `cuisines[i]` is the type of cuisine of the i^{th} food, and
  - `ratings[i]` is the initial rating of the i^{th} food.
- `void changeRating(String food, int newRating)` Changes the rating of the food item with the name `food`.
- `String highestRated(String cuisine)` Returns the name of the food item that has the highest rating for the given type of `cuisine`. If there is a tie, return the item with the **lexicographically smaller** name.

Note that a string `x` is lexicographically smaller than string `y` if `x` comes before `y` in dictionary order, that is, either `x` is a prefix of `y`, or if `i` is the first position such that `x[i] != y[i]`, then `x[i]` comes before `y[i]` in alphabetic order.

Example 1:

```
Input
["FoodRatings", "highestRated", "highestRated", "changeRating", "highestRated",
"changeRating", "highestRated"]
[[["kimchi", "miso", "sushi", "moussaka", "ramen", "bulgogi"], ["korean", "japanese",
"japanese", "greek", "japanese", "korean"], [9, 12, 8, 15, 14, 7]], ["korean"], ["japanese"],
["sushi", 16], ["japanese"], ["ramen", 16], ["japanese"]]
Output
[null, "kimchi", "ramen", null, "sushi", null, "ramen"]

Explanation
FoodRatings foodRatings = new FoodRatings(["kimchi", "miso", "sushi", "moussaka", "ramen",
"bulgogi"], ["korean", "japanese", "japanese", "greek", "japanese", "korean"], [9, 12, 8, 15,
14, 7]);
foodRatings.highestRated("korean"); // return "kimchi"
                                    // "kimchi" is the highest rated korean food with a
rating of 9.
foodRatings.highestRated("japanese"); // return "ramen"
                                      // "ramen" is the highest rated japanese food with a
rating of 14.
foodRatings.changeRating("sushi", 16); // "sushi" now has a rating of 16.
foodRatings.highestRated("japanese"); // return "sushi"
                                      // "sushi" is the highest rated japanese food with a
rating of 16.
foodRatings.changeRating("ramen", 16); // "ramen" now has a rating of 16.
foodRatings.highestRated("japanese"); // return "ramen"
                                      // Both "sushi" and "ramen" have a rating of 16.
                                      // However, "ramen" is lexicographically smaller than
"sushi".
```

Constraints:

- $1 <= n <= 2 * 10^4$
- `n == foods.length == cuisines.length == ratings.length`
- `1 <= foods[i].length, cuisines[i].length <= 10`
- `foods[i]`, `cuisines[i]` consist of lowercase English letters.
- $1 <= ratings[i] <= 10^8$
- All the strings in `foods` are **distinct**.
- `food` will be the name of a food item in the system across all calls to `changeRating`.
- `cuisine` will be a type of cuisine of **at least one** food item in the system across all calls to `highestRated`.
- At most $2 * 10^4$ calls **in total** will be made to `changeRating` and `highestRated`.

foodrating sys
↓
highestrated
foodreturn

constructor

food
cuisine
rating
i

To read

Unordered map

```cpp
struct cmp {//custom comparator for set
    bool operator()(pair<int, string> a, pair<int, string> b) const {
        if (a.first > b.first)return 1;
        if (a.first < b.first) return 0;
        return (a.second < b.second);
    }
};

class FoodRatings {
public:
    // cuisine  TO (rating and food)   s
    unordered_map<string, set<pair<int, string>, cmp>> s;
    // food TO (cuisine and rating)   m
    unordered_map<string, pair<string, int>> m;

    FoodRatings(vector<string>& foods, vector<string>& cuisines, vector<int>& ratings) {
        int n = foods.size();
        for(int i = 0; i < n; ++i){
            m[foods[i]] = {cuisines[i], ratings[i]};
            s[cuisines[i]].insert({ratings[i], foods[i]});
        }
    }

    void changeRating(string food, int newRating) {
        string cuisine = m[food].first;
        int oldRating = m[food].second;
        s[cuisine].erase({oldRating, food});

        m[food] = {cuisine, newRating};
        s[cuisine].insert({newRating, food});
    }
```

```
    string highestRated(string cuisine) {
        return (*s[cuisine].begin()).second;;
    }
};
```

cpp

Copy code

```cpp
// Part 1: Definition of cmp struct
struct cmp {
    bool operator()(pair<int, string> a, pair<int, string> b) const {
        if (a.first > b.first) return true;
        if (a.first < b.first) return false;
        return (a.second < b.second);
    }
};

// Part 2: Definition of FoodRatings class
class FoodRatings {
    vector<string> Foods, Cuisines;
    vector<int> Ratings;

public:
    // Part 3: Unordered maps as member variables
    unordered_map<string, set<pair<int, string>, cmp>> s;
    unordered_map<string, pair<string, int>> m;

    // Part 4: Constructor
    FoodRatings(vector<string>& foods, vector<string>& cuisines, vector
        for (int i = 0; i < foods.size(); i++) {
            m[foods[i]] = {cuisines[i], ratings[i]};
            s[cuisines[i]].insert({ratings[i], foods[i]});
        }
    }

    // Part 5: Method to change rating
    void changeRating(string food, int newRating) {
        string cuisine = m[food].first;
        int oldRating = m[food].second;
        s[cuisine].erase({oldRating, food});

        m[food] = {cuisine, newRating};
        s[cuisine].insert({newRating, food});
    }

    // Part 6: Method to find highest rated food
    string highestRated(string cuisine) {
        return (*s[cuisine].begin()).second;
    }
};
```