2353. Design a Food Rating System

Medium | Topics | Companies | Hint

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".
```

```python
class FoodRatings:

    def __init__(self, foods: List[str], cuisines: List[str], ratings: List[int]):


    def changeRating(self, food: str, newRating: int) -> None:


    def highestRated(self, cuisine: str) -> str:



# Your FoodRatings object will be instantiated and called as such:
# obj = FoodRatings(foods, cuisines, ratings)
# obj.changeRating(food,newRating)
# param_2 = obj.highestRated(cuisine)
```

```
def _init_ (self, foods, cuisines, ratings):
        self.foodInfo = {}
        self.cuisineInfo = {}

        for f, c, r in zip(foods, cuisines, ratings):
                foodInfo[f] = (c, r)
                if c not in self.cuisineInfo:
                        self.cuisineInfo[c] = []
                heapq.heappush(self.cuisineInfo[c], (-r, f) )

def changeRating(self, food, newRating):
        c, _ = self.foodInfo[food]
        self.foodInfo[food] = (c, newRating)
        heapq.heappush(self.cuisineInfo[c], (-newRating, food) )
        # will not clean the old rating here, will do it in the highestRating()
```

```
def highestRating(self, cuisine):
        heap = self.cusineInfo[cuisine]

        while heap:
                r, f = heap[0]
                cur_c, cur_r = self.foodInfo[f]

                if cur_r == -r:
                        return f

                heapq.heappop(heap)
```