Mizzza54

Курс по JavaScript. 2021

19 июн 2022, 18:08:03
старт: 27 сен 2021, 14:53:11
начало: 1 сен 2020, 00:00:00

> Мы очень хотим, чтобы код вы написали сами, а не пользовались внешними библиотеками.

Основное задание

Как-то на одной из лекций Аркадий познакомился с девушкой и в скором времени его корабль, под названием «Любовь», наконец-то готов пришвартоваться в бухте, под названием «Семья». Настало время организовать весёлую свадьбу.

Для этого, по мнению Аркадия, должен быть соблюдён ряд условий:

- Во-первых, на свадьбе должны быть не только друзья, но и друзья друзей
- Во-вторых, слишком незнакомые парни смущают Аркадия и он планирует ограничить уровень неизвестности определённым кругом
- В-третьих, чтобы никому не было грустно – он их собирает в пары «парень + девушка»

Аркадий вновь достаёт свою телефонную книгу с записями о друзьях и дополняет её информацией о том, кто и с кем дружит (`friends`), и кто является его лучшими друзьями (`best`) таким образом:

```
const friends = [
    {
        name: 'Sam',
        friends: ['Mat', 'Sharon'],
        gender: 'male',
        best: true
    },
    {
        name: 'Sally',
        friends: ['Brad', 'Emily'],
        gender: 'female',
        best: false
    }
];
```

И чтобы помочь Аркадию выбрать кого пригласить необходимо подготовить для него удобные фильтры и итераторы для работы с телефонной книгой.

```
// Создаем фильтры парней и девушек
const maleFilter = new lib.MaleFilter();
const femaleFilter = new lib.FemaleFilter();

// Создаем итераторы
const femaleIterator = new lib.Iterator(friends, femaleFilter);

// Среди парней приглашаем только луших друзей и друзей лучших друзей
const maleIterator = new lib.LimitedIterator(friends, maleFilter, 2);

const invitedFriends = [];

// Собираем пары «парень + девушка»
while (!maleIterator.done() && !femaleIterator.done()) {
    invitedFriends.push([
        maleIterator.next(),
        femaleIterator.next()
    ]);
}

// Если остались девушки, то приглашаем остальных
while (!femaleIterator.done()) {
    invitedFriends.push(femaleIterator.next());
}
```

Мы выложили пример того как можно работать с вашим кодом под условием задачи, а заготовку для того чтобы реализовать свой код вы можете найти здесь.

Общие условия:

⚠ Код необходимо написать в интерфейсах, зафиксированных в заготовке. То есть использовать функции-конструкторы и методы работы с прототипами, и не использовать «классы».

- Лучшие друзья помечены флагом best
- Для каждого друга указан список его друзей
- Дружба всегда взаимная
- Обход должен происходить, начиная с лучших друзей
- Обход всегда идет в алфавитном порядке имен
- Друзья не должны обходиться дважды
- Первый круг друзей – это лучшие друзья
- Второй круг друзей – это друзья лучших друзей
- Третий круг и остальные строятся аналогичным образом
- Гарантируется, что на входе будут корректные условия
- Неориентированный граф друзей
- Все перечисленные друзья в свойствах friends будут существовать во входном массиве
- Граф друзей может быть несвязным и/или цикличным

Условия для LimitedIterator

- Наследник Iterator
- Имеет ограничение по кругу maxLevel. Если передан maxLevel равен 1, то итератор обойдет только первый круг друзей, если 2 – первый и второй. И так далее.

Условия для Filter

- Создает фильтр, который решает какой друг подходит для итерации
- По умолчанию такой фильтр никого не отсеивает

Условия для MaleFilter

- Наследник Filter
- Позволяет итерироваться по друзьям мужского пола

Условия для FemaleFilter

- Наследник Filter
- Позволяет итерироваться по друзьям женского пола

Пример использования:

```
const assert = require('assert');
const lib = require('./lib'); // Ваш файл с кодом задачи

const friends = [
    {
        name: 'Sam',
        friends: ['Mat', 'Sharon'],
        gender: 'male',
        best: true
```

```
        },
        {
            name: 'Sally',
            friends: ['Brad', 'Emily'],
            gender: 'female',
            best: true
        },
        {
            name: 'Mat',
            friends: ['Sam', 'Sharon'],
            gender: 'male'
        },
        {
            name: 'Sharon',
            friends: ['Sam', 'Itan', 'Mat'],
            gender: 'female'
        },
        {
            name: 'Brad',
            friends: ['Sally', 'Emily', 'Julia'],
            gender: 'male'
        },
        {
            name: 'Emily',
            friends: ['Sally', 'Brad'],
            gender: 'female'
        },
        {
            name: 'Itan',
            friends: ['Sharon', 'Julia'],
            gender: 'male'
        },
        {
            name: 'Julia',
            friends: ['Brad', 'Itan'],
            gender: 'female'
        }
    ];

    function friend(name) {
        let len = friends.length;

        while (len--) {
            if (friends[len].name === name) {
                return friends[len];
            }
        }
    }

    const maleFilter = new lib.MaleFilter();
    const femaleFilter = new lib.FemaleFilter();
    const maleIterator = new lib.LimitedIterator(friends, maleFilter, 2);
    const femaleIterator = new lib.Iterator(friends, femaleFilter);

    const invitedFriends = [];

    while (!maleIterator.done() && !femaleIterator.done()) {
        invitedFriends.push([
            maleIterator.next(),
            femaleIterator.next()
        ]);
    }

    while (!femaleIterator.done()) {
        invitedFriends.push(femaleIterator.next());
    }

    assert.deepStrictEqual(invitedFriends, [
        [friend('Sam'), friend('Sally')],
        [friend('Brad'), friend('Emily')],
        [friend('Mat'), friend('Sharon')],
```

```
        friend('Julia')
    ]);
```

Набрать здесь | Отправить файл

```
'use strict';

const bfs = function (phonebook) {
    const queue = [];
    const layers = [];
    layers[0] = [];
    const visited = new Set();

    const map = new Map();
    phonebook.forEach(f => map.set(f.name, f))


    const bestPhonebook = phonebook.filter(x => x.best);
    bestPhonebook.forEach(x => {
        x.level = 0;
        visited.add(x);
        queue.push(x);
        layers[0].push(x);
    });

    while (queue.length > 0) {
        const node = queue.shift();

        if (!layers[node.level + 1]) {
            layers[node.level + 1] = []
        }

        for (let i = 0; i < node.friends.length; i++) {
            const friend = map.get(node.friends[i]);

            if (!visited.has(friend)) {
                visited.add(friend);
                layers[node.level + 1].push(friend);
                friend.level = node.level + 1;
                queue.push(friend);
            }
        }
    }
```

Отправить

Предыдущая Следующая