

# А. Поиск коллеги

Сегодня мы станем частью большой и очень значительной транснациональной корпорации Yandex Contest 2024. В ней работает много сотрудников и, как часто это бывает, нельзя просто так взять и найти нужный контакт человека из соседней команды, соседней группы, соседнего бизнес-юнита. Иной раз требуется обойти большую цепочку коллег для того, чтобы получить необходимый контакт. Требуется написать программу, которая поможет нашим коллегам находить самый короткий путь от них до желаемого контакта.

- `graph` — объект ключ/значение. Количество ключей  $0 \leq N \leq 50$ , количество значений  $0 \leq N \leq 50$
- `startVertex` — начальная вершина
- `endVertex` — конечная вершина

## Шаблон

```
/**
 * @param {{
 *   graph: Record<string, string[]>,
 *   startVertex: string,
 *   endVertex: string,
 * }}
 * @returns {string[]}
 */
module.exports = function solution({ graph, startVertex, endVertex }) {
  // ваш код
}
```

## Формат ввода

### Пример 1

```
module.exports = {  
  graph: {  
    Александра: ["Борис"],  
    Борис: ["Александра", "Светлана"],  
    Светлана: ["Борис"],  
  },  
  startVertex: "Александра",  
  endVertex: "Светлана",  
};
```

### Пример 2

```
module.exports = {  
  graph: {  
    Артемий: ["Бронислав", "Дементий"],  
    Бронислав: ["Артемий", "Софья", "Дементий"],  
    Софья: ["Бронислав"],  
    Дементий: ["Артемий", "Бронислав"],  
    Фаина: ["Гаврила"],  
    Гаврила: ["Фаина"],  
  },  
  startVertex: "Артемий",  
  endVertex: "Фаина",  
};
```

## Формат вывода

### Пример 1

Самый короткий путь от Александры до Светланы

```
["Александра", "Борис", "Светлана"]
```

### Пример 2

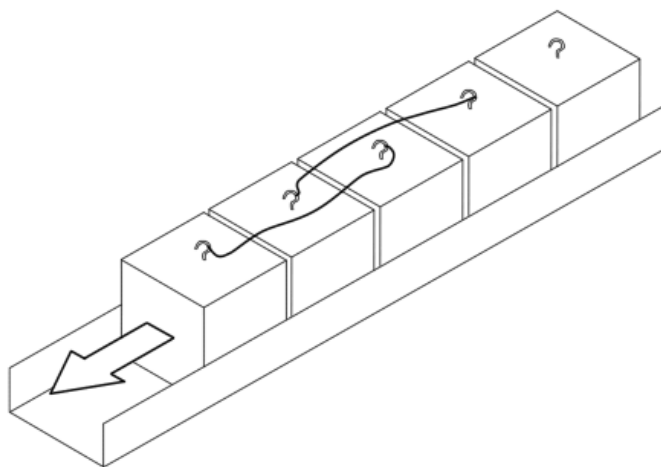
Путь от Артемия до Фаины отсутствует, поэтому выводим пустой массив

```
[]
```

## В. Желоб

Ограничение времени	1 секунда
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

В продолговатом желобе размещены один за другим  $n$  кубических грузиков с крючками на верхней стороне. Некоторые пары крючков соединены между собой прочными нитями.



Максим стоит возле желоба так, что видит все грузики. Он начинает тянуть ближайший грузик вдоль желоба на себя. Определите общее количество грузиков, которые будут сдвинуты.

Максим достаточно силен, чтобы сдвинуть любое количество грузиков, а соединяющие грузики нити никогда не рвутся.

## Формат ввода


В первой строке входных данных содержится два целых числа  $n$  и  $m$ , разделённых одним пробелом - количество грузиков и количество нитей ( $1 \leq n \leq 10^9, 1 \leq m \leq 10^5$ ).

Последующие  $m$  строк содержат по два разделённых пробелом целых числа  $a_i$  и  $b_i$  - номера соединённых  $i$ -той нитью грузиков ( $1 \leq a_i, b_i \leq n$ ). Грузики нумеруются с единицы в порядке удаления от Максима.


## Формат вывода

Выходные данные должны содержать единственное целое число - общее количество грузиков, которые Максим потянет на себя.

### Пример 1


Ввод 

```
5 2
1 3
4 2
```


Вывод 

```
4
```

### Пример 2

Ввод 

```
1000 5
1 100
100 200
200 300
300 400
350 421
```

Вывод 

```
421
```

## Примечания

Рисунок в условии иллюстрирует первый пример. Обратите внимание, что номера соединённых нитью грузиков могут идти как в порядке возрастания, так и в порядке убывания (как в первом примере), а к одному грузику может быть привязано множество разных нитей (во втором примере к некоторым грузикам привязано по две нити).

## С. Капризный король

Ограничение времени	1 секунда
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Шахматный король стоит в левом верхнем углу шахматной доски размером  $N \times M$ . За один шаг он может перейти в соседнюю клетку ниже, либо в клетку правее, либо в клетку правее и ниже (то есть сделать шаг по диагонали). Его цель — добраться таким образом до правого нижнего угла доски.

Да только вот король тот капризен, а потому не наступает трижды подряд на клетки одного цвета. Доска имеет стандартную шахматную покраску: каждая клетка является либо чёрной, либо белой, а цвет каждой пары имеющих общую сторону клеток различен.

Сколько существует различных путей, которыми капризный король смог бы добраться до своей цели? В ответ вывести число таких путей по модулю 1000 (иначе говоря - остаток от деления реального числа путей на 1000).

### Формат ввода

Входные данные содержат два натуральных числа  $N$  и  $M$ , разделённые одним пробелом. Оба числа не превосходят 100.


### Формат вывода

Вывести одно целое число — ответ на вопрос задачи.

### Пример 1


Ввод 

2 2


Вывод 

3

### Пример 2


Ввод 

2 3


Вывод 

5

### Пример 3

Ввод 

3 3

Вывод 

12

Язык Node.js 18.16.0

Набрать здесь

Отправить файл

```
1 // Для чтения входных данных в Node.js необходимо использовать
2 // модуль readline, который работает с потоком ввода-вывода
3 // (stdin/stdout) и позволяет читать строки.
4 const readline = require('readline');
5
6 const rl = readline.createInterface({
7   input: process.stdin,
8   output: process.stdout
9 });
10
11 // Данные во входном потоке могут состоять из нескольких строк.
12 // Чтобы прочитать их, можно использовать метод rl.on(),
13 // который вызывается каждый раз при появлении новой строки
14 // в потоке ввода.
15 // Чтобы вывести результат в поток вывода (stdout),
16 // можно использовать метод console.log().
17 // Пример:
18 // console.log('Результат:', result);
19
20 // Пример решения задачи "Вычислите сумму A+B":
21 // rl.on('line', line => {
22 //   const [a, b] = line.split(' ').map(Number);
23 //   console.log(a + b);
24 //   rl.close();
25 // });
```

## D. Внедрение кэширования

После запуска проекта выяснилось, что у пользователей наблюдаются проблемы с памятью при длительном использовании приложения. Результаты, полученные на предыдущих запросах к бэкенду, остаются в памяти, что приводит к зависанию страницы и прочим неприятным последствиям.

Ваша команда разработала план, и вам поручили исправить эту проблему и реализовав его. План заключается в том, чтобы разработать алгоритм кэширования.

Нужно написать функцию, которая получает на вход размер кэша и возвращает интерфейс для встраивания в проект, который будет управлять кэшированием данных на странице. В процессе обсуждений было решено использовать приоритетное кэширование, где дольше сохраняются в кэше те данные, которые использовались последними.

```
interface EntityDataI<Data extends {id: number}> {  
  setCacheChunk: (value: Data | Data[]): void;  
  changeItem: (newData: Data | Data[]): void;  
  getCacheItemById: (id: number): Data;  
  getData: (): Data[];  
}
```

- `setCacheChunk` — Функция чтобы положить новую порцию данных с сервера в кэш;
- `changeItem` — Функция для пользовательского изменения кэшированных данных;
- `getCacheItemById` — Функция для получения элемента из кэша по `id`;
- `getAppCache` — Функция для получения текущего состояния кэша;

В результате у вас должна получится следующая функция:

```
module.exports = function getCache(maxSize) {  
  // Ваш JavaScript код  
}
```

### Пример использования

```
const cache = getCache(3); // Размер кэша равен трем элементам

cache.setCacheChunk({id: 1}); // Добавляем 1 объект в кэш
cache.setCacheChunk([{id: 2}, {id: 3}]); // Добавляем несколько объектов в кэш
cache.getData() // [{id: 1}, {id: 2}, {id: 3}]

cache.setCacheChunk({id: 4}); // Добавляем 1 объект в заполненный кэш
cache.getData() // [{id: 2}, {id: 3}, {id: 4}]

cache.changeItem({id: 3, log: 'some data'}); // Изменяем объект в кэше
cache.getData() // [{id: 2}, {id: 3, log: 'some data'}, {id: 4}]

cache.setCacheChunk([{id: 5}, {id: 6}]); // Добавляем несколько объектов в заполн
cache.getData() // [{id: 3, log: 'some data'}, {id: 4}, {id: 5}, {id: 6}]

cache.setCacheChunk({id: 3, field: 'some value'}); // Измененный объект перестает
// Из-за того что объект был установлен снова, его приоритет повышен
cache.getData() // [{id: 5}, {id: 6}, {id: 3, field: 'some value'}]

cache.setCacheChunk([{id: 7}, {id: 8}, {id: 9}]); // Добавляем объекты в кэш
cache.getCacheItemById(7); //Читаем объекты из кэша
cache.getCacheItemById(8);
cache.getData() // [{id: 9}, {id: 7}, {id: 8}]
```

### На что следует обратить внимание:

- Размер кэша (без измененных данных) должен быть не больше `maxSize`;
- При вызове функции получения данных `getCacheItemById`, приоритет полученного элемента кэша повышается над остальными;
- Добавленные элементы через `setCacheChunk` так же получают повышенный приоритет над остальными в порядке добавления;
- Измененные пользователем данные должны оставаться в кэше;
- Измененные пользователем данные не учитываются в подсчете размера кэша;
- При получении с сервера новой версии измененных данных (совпадают по `id`) они перестают быть измененными и новая версия данных кэшируется по основным правилам.

Для корректной проверки решения, при отправке выбирайте (make) компилятор.



# Е. Магические ссылки

## Легенда

В одном волшебном королевстве очень злые маги начали подделывать ссылки на ценные знания, находящиеся на сервисах Яндекса. Вашей задачей, как чародея-защитника, является создать заклинание, которое сможет распознавать истинные магические ссылки и блокировать поддельные.

## Условие

Напишите функцию `isValidYandexLink(url)`, которая проверяет, является ли переданная строка параметра `url` ссылкой на любой из сервисов Яндекса.

1. Ссылка считается валидной, если она ведет на один из доменов, принадлежащих Яндексу.
2. Домен может быть как основным (.ru), так и международным (.by, .com и т.д.).
3. Примеры доменов и поддоменов:  
домены: ya.ru, yandex.com, yandex.kz, yandex.by, yandex.az, и т.д.  
поддомены: education.yandex., lyceum.yandex., shad.yandex., и т.д.

## Примеры

```
isValidYandexLink("https://ya.ru") // true
isValidYandexLink("https://education.yandex.ru") // true
isValidYandexLink("http://yandex.ru/cup") // true
isValidYandexLink("https://dataschool.yandex.com") // true
isValidYandexLink("https://education.yandex.ru/uchebnik") // true
isValidYandexLink("https://google.com") // false
isValidYandexLink("http://example.com") // false
isValidYandexLink("hts://y*ndex.ru/somepath") // false
```

- Функция должна возвращать `true`, если переданная ссылка валидна и ведет на один из сервисов Яндекса.
- Функция должна возвращать `false`, если переданная ссылка не ведет на сервис Яндекса или если ссылка невалидна (например, `http://wrong.url`).
- Функция должна корректно обрабатывать различные виды URL (с/без `http(s)`), с различными разделителями и параметрами).

## Формат ввода

```
function isValidYandexLink(url) {
  // Your code here...
}

module.exports = isValidYandexLink;
```