Задания по PHP

1 . Две суммы

Учитывая массив целых чисел `nums` и целое число `target`, верните *индексы двух чисел так, чтобы они составляли в сумме* `target` .

Вы можете предположить, что каждый вход будет иметь ***ровно* одно решение** , и вы не можете использовать один *и тот же* элемент дважды.

Вы можете вернуть ответ в любом порядке.

Пример 1:

```
Ввод: nums = [2,7,11,15], target = 9
 Вывод: [0,1]
 Объяснение: Поскольку nums[0] + nums[1] == 9, мы возвращаем [0, 1].
```

Пример 2:

```
Ввод: числа = [3,2,4], цель = 6
 Вывод: [1,2]
```

Пример 3:

```
Ввод: числа = [3,3], цель = 6
 Вывод: [0,1]
```

Ограничения:

- `2 <= nums.length <= 10`^{4}
- `-10`^{9} `<= nums[i] <= 10`^{9}
- `-10`^{9} `<= target <= 10`^{9}
- **Существует только один правильный ответ.**

Дополнение: можете ли вы придумать алгоритм, сложность которого меньше временной? `O(n`^{2}`)`

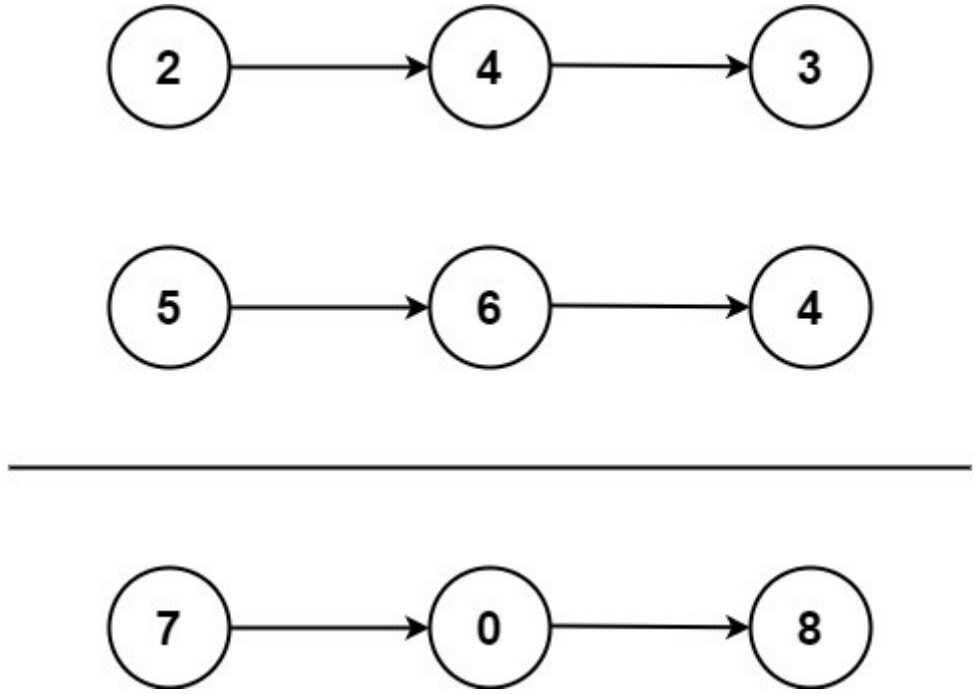
```
class Solution {

    /**
    * @param Integer[] $nums
    * @param Integer $target
    * @return Integer[]
    */
    function twoSum($nums, $target) {

    }
}
```

2 . Добавить два числа

Вам даны два непустых связанных списка, представляющих два неотрицательных целых числа. Цифры хранятся в обратном порядке , и каждый из их узлов содержит одну цифру. Добавьте два числа и верните сумму в виде связанного списка.

Вы можете предположить, что эти два числа не содержат начальных нулей, кроме самого числа 0.

Пример 1:




```
Ввод: l1 = [2,4,3], l2 = [5,6,4]
 Вывод: [7,0,8]
 Объяснение: 342 + 465 = 807.
```

Пример 2:

```
Вход: l1 = [0], l2 = [0]
 Выход: [0]
```

Пример 3:

```
Ввод: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]
 Вывод: [8,9,9,9,0,0,0,1]
```

Ограничения:

- Количество узлов в каждом связанном списке находится в диапазоне `[1, 100]`.
- `0 <= Node.val <= 9`
- Гарантируется, что список представляет собой число, не имеющее лидирующих нулей.

```php
/**
 * Definition for a singly-linked list.
 * class ListNode {
 *     public $val = 0;
 *     public $next = null;
 *     function __construct($val = 0, $next = null) {
 *         $this->val = $val;
 *         $this->next = $next;
 *     }
 * }
 */
class Solution {

    /**
     * @param ListNode $l1
     * @param ListNode $l2
     * @return ListNode
     */
    function addTwoNumbers($l1, $l2) {

    }
}
```

3 . Медиана двух отсортированных массивов

Учитывая два отсортированных массива `nums1` и `nums2` размер `m` и `n` соответственно, вернуть **медиану** двух отсортированных массивов.

Общая сложность времени выполнения должна быть `O(log (m+n))`.

Пример 1:

```
Ввод: nums1 = [1,3], nums2 = [2]
 Вывод: 2,00000
 Объяснение: объединенный массив = [1,2,3] и медиана равна 2.
```

Пример 2:

```
Ввод: nums1 = [1,2], nums2 = [3,4]
 Вывод: 2,50000
 Объяснение: объединенный массив = [1,2,3,4] и медиана (2 + 3) / 2 = 2,5.
```

Ограничения:

- `nums1.length == m`
- `nums2.length == n`
- `0 <= m <= 1000`
- `0 <= n <= 1000`
- `1 <= m + n <= 2000`
- -10^6 `<= nums1[i], nums2[i] <=` 10^6

```
class Solution {

    /**
     * @param Integer[] $nums1
     * @param Integer[] $nums2
     * @return Float
     */
    function findMedianSortedArrays($nums1, $nums2) {

    }
}
```