643 Maximum Average Subarray I

2018年4月21日 17:12

Given an array consisting of n integers, find the contiguous subarray of given length k that has the maximum average value. And you need to output the maximum average value.

Example 1:

Input: [1,12,-5,-6,50,3], k = 4

Output: 12.75

Explanation: Maximum average is (12-5-6+50)/4 = 51/4 = 12.75

Note:

1. 1 <= k <= n <= 30,000.
2. Elements of the given array will be in the range [-10,000, 10,000].

来自 <https://leetcode.com/problems/maximum-average-subarray-i/description/>

给定 n 个整数，找出平均数最大且长度为 k 的连续子数组，并输出该最大平均数。

注意:

1. 1 <= k <= n <= 30,000。
2. 所给数据范围 [-10,000，10,000]。

Solution for Python3:

```python
class Solution1:
    def findMaxAverage(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: float
        """
        accusum = sum(nums[:k])
        res = accusum
        for i in range(k, len(nums)):
           accusum += nums[i] - nums[i - k];
           if accusum > res:
               res = accusum
        return res / k

class Solution2:
    def findMaxAverage(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: float
        """
        sums = [0] + list(itertools.accumulate(nums))
        return max(map(operator.sub, sums[k:], sums)) / k

class Solution2:
    def findMaxAverage(self, nums, k):
        """
        :type nums: List[int]
```

```
        :type k: int
        :rtype: float
        """
        import numpy as np
        sums = np.cumsum([0] + nums)
        return int(max(sums[k:] - sums[:-k])) / k
```

Solution for C++:

```
class Solution {
public:
    double findMaxAverage(vector<int>& nums, int k) {
        int accusum = accumulate(nums.begin(), nums.begin() + k, 0);
        int res = accusum;
        for (int i = k; i < nums.size(); i++) {
            accusum +=nums[i] - nums[i - k];
            if (accusum > res)
                res = accusum;
        }
        return double(res) / k;
    }
};
```