Minimum Moves to Equal Array Elements

Given a **non-empty** integer array of size n, find the minimum number of moves required to make all array elements equal, where a move is incrementing n - 1 elements by 1.

Example:

```
Input:
[1,2,3]

Output:
3

Explanation:
Only three moves are needed (remember each move increments two elements):

[1,2,3]  =>  [2,3,3]  =>  [3,4,3]  =>  [4,4,4]
```

Solution 1

Add `1` to `n - 1` elements is the same as subtracting `1` from one element, w.r.t goal of making the elements in the array equal.
So, best way to do this is make all the elements in the array equal to the `min` element.
`sum(array) - n * minimum`

```java
public class Solution {
    public int minMoves(int[] nums) {
        if (nums.length == 0) return 0;
        int min = nums[0];
        for (int n : nums) min = Math.min(min, n);
        int res = 0;
        for (int n : nums) res += n - min;
        return res;
    }
}
```

written by kdtree original link here

Solution 2

Incrementing all but one is equivalent to decrementing that one. So let's do that instead. How many single-element decrements to make all equal? No point to decrementing below the current minimum, so how many single-element decrements to make all equal to the current minimum? Just take the difference from what's currently there (the sum) to what we want (n times the minimum).

Python:

```
def minMoves(self, nums):
    return sum(nums) - len(nums) * min(nums)
```

Ruby:

```
def min_moves(nums)
  nums.inject(:+) - nums.size * nums.min
end
```

Java (ugh :-):

```
public int minMoves(int[] nums) {
    return IntStream.of(nums).sum() - nums.length * IntStream.of(nums).min().getAsInt();
}
```

C++ (more ugh):

```
int minMoves(vector<int>& nums) {
    return accumulate(begin(nums), end(nums), 0) - nums.size() * *min_element(begin(nums), end(nums));
}
```

written by StefanPochmann original link here

Solution 3

let's define sum as the sum of all the numbers, before any moves; minNum as the min number int the list; n is the length of the list;

After, say m moves, we get all the numbers as x , and we will get the following equation

```
sum + m * (n - 1) = x * n
```

and actually,

```
  x = minNum + m
```

and finally, we will get

```
  sum - minNum * n = m
```

So, it is clear and easy now.

written by wang.senyuan original link here