Sherlock And Array

Welcome to the first of many challenges that reference “Sherlock,” for no good reason at all.

A simple approach would be to consider all possible splits and sum up the subarrays on either side. This produces the right answers, but it will time out on the larger inputs.

If we consider the splits in order, we can realize that the sums will only change by a known amount each step, namely the item moving to or from the split location. With clever ordering of the adjustments of the sums, we can do this with a single pass through the array. This converts an $\mathcal{O}(n^2)$ algorithm to $\mathcal{O}(n)$.

```python
def balancedSums(arr):
    sum1 = 0
    sum2 = sum(arr)

    for item in arr:
        sum2 -= item
        if sum1 == sum2:
            return "YES"
        sum1 += item
    return "NO"
```

Note the use of the short-circuit `return` statement. As soon as we find a split that works, we can return `"YES"`, rather than waiting until the end of the loop.