

1. 给出不同策略返回一系列数中的众数（出现次数不小于序列长度的一半），并分析。

算法一

将序列排序，由于众数出现次数不小于序列长度的一半，所以众数必然在序列的中间位置，因此直接返回中间位置的数即可。时间复杂度为 $O(n\log n)$ ，空间复杂度为 $O(1)$ 。

至于排序的方法，我们在前面的几次作业中提到过很多种，不再赘述。

代码实现


```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int majorityElement(vector<int>& nums) {
    sort(nums.begin(), nums.end());
    return nums[nums.size() / 2];
}

int main() {
    vector<int> nums = { 3, 2, 1, 9, 3 };
    int majority = majorityElement(nums);
    cout << "此序列的众数为" << majority << endl;
    return 0;
}
```

结果展示

 Microsoft Visual Studio 调试控制台

此序列的众数为3

C:\Users\lenovo\Desktop\Project1\Debug\Project1.exe (进程 23348) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .

算法二

Boyer-Moore 投票算法：从头开始遍历序列，假设第一个数为众数，遍历后面的数，遇到相同的数则计数器加1，否则计数器减1，当计数器减到0时，说明前面的数不是众数，重新选取下一个数作为众数。遍历完整个序列后，最终得到的数即为众数。时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$ 。

代码实现


```
#include <iostream>
#include <vector>

using namespace std;

int majorityElement(vector<int>& nums) {
    int count = 0, candidate;
    for (int num : nums) {
        if (count == 0) {
            candidate = num;
        }
        count += (num == candidate) ? 1 : -1;
    }
    return candidate;
}

int main() {
    vector<int> nums = { 7,6,6,4,5,6,1,7,6,6 };
    int majority = majorityElement(nums);
    cout << "该序列的众数为: " << majority << endl;
    return 0;
}
```

结果展示

 Microsoft Visual Studio 调试控制台

该序列的众数为: 6

C:\Users\lenovo\Desktop\Project1\Debug\Project1.exe (进程 12688) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。...

2.给出不同策略将偶数个元素形成的序列分成两个子序列S1,S2，要求s1，s2的和差值最大。

算法一

我们可以直接将给定序列进行从小到大排序，如果限制S1和S2元素个数相同，那么将排好的序列的前 $n/2$ 个数放入S1，后 $n/2$ 个数放入S2，可以实现和差值最大。如果不限制S1，S2元素个数相同，可以将正数放入S1，负数放入S2即可实现和差值最大）。

这种方法比较简单不再赘述。

算法二

创立两个序列S1，S2和SUM1和SUM2。先将原序列的第一个元素放入S1，更新SUM1和SUM2，将第二个元素放入序列时，判断放入S1或S2的情况下，哪种情况的 $\text{abs}(\text{SUM1}-\text{SUM2})$ 更大，选择更大的情况。这样一直循环，直到最后原序列遍历完毕。

（实际上在本题中没有规定S1，S2中元素个数相同，求和差值最大可以直接正数放S1，负数放S2更简单。本方法可以判断和差值最小，只要修改一下判断就可以实现）

代码部分

```

#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int maxDiff(vector<int>& nums) {
    int n = nums.size();
    vector<int> nums1(0);
    vector<int> nums2(0);
    int x = nums[0], y = 0; // 初始化x为第一个元素, y为0
    nums1.push_back(nums[0]);
    int maxDiff = abs(x - y); // 初始化maxDiff为当前的差值
    for (int i = 1; i < n; i++) {
        // 如果将nums[i]放入S1
        int diff1 = abs(x + nums[i] - y);
        // 如果将nums[i]放入S2
        int diff2 = abs(x - (y + nums[i]));
        if (diff1 >= diff2) {
            x += nums[i];
            nums1.push_back(nums[i]);
        }
        else {
            y += nums[i];
            nums2.push_back(nums[i]);
        }
        maxDiff = abs(x-y); // 更新maxDiff
    }
    cout << "序列S1为: " << endl;
    for (int i = 0; i < nums1.size(); i++) {
        cout << nums1[i] << " ";
    }
    cout << endl;
    cout << "序列S2为: " << endl;
    for (int i = 0; i < nums2.size(); i++) {
        cout << nums2[i] << " ";
    }
    cout << endl;
    cout << "最大和差值为: ";
    return maxDiff;
}

int main() {
    int n;
    cout << "请输入序列的长度n: " << endl;
    cin >> n;
    vector<int> nums(n);
    cout << "请输入序列元素" << endl;
    for (int i = 0; i < n; i++) {
        cin >> nums[i];
    }
}

```

```
}  
cout << maxDiff(nums) << endl;  
return 0;  
}
```

结果展示



The screenshot shows the Microsoft Visual Studio Debug Console. The title bar reads "Microsoft Visual Studio 调试控制台". The console output is as follows:

```
请输入序列的长度n:  
4  
请输入序列元素  
1 2 -4 8  
序列S1为:  
1 2 8  
序列S2为:  
-4  
最大和差值为: 15  
  
C:\Users\lenovo\Desktop\Project1\Debug\Project1.exe (进程 21744) 已退出, 代码为 0。  
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。  
按任意键关闭此窗口. . .
```