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。

- 法一：最容易想到的方法是将序列排序，此时相同元素都聚集中一起，逐步遍历寻找只需记录当前出现次数最多元素及其次数，在遍历过程中更新即可。

代码如下：

```
#include<iostream>
#include<algorithm>
using namespace std;
#define MaxSize 25
int main(){
    int arr[MaxSize];
    srand((unsigned)time(NULL));
    cout<<"数组元素为："<<endl;
    for (int i = 0;i<MaxSize;i++){
        arr[i] = rand()%20;
        cout<<arr[i]<<" ";
    }
    cout<<endl;

    //将序列排序
    sort(arr,arr+MaxSize);
    cout<<"经排序后数组为："<<endl;
    for (int i = 0;i<MaxSize;i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;

    //排序后相同数字都聚在一起
    int modalNumber=arr[0];//众数
    int currentNum = 0;
    int mostNum = 0;//最多数的数量
    for (int i = 0; i < MaxSize; i++){
        //特殊情况，i为最后一个元素索引
        currentNum++;
        if (arr[i] != arr[i + 1] || i==MaxSize-1){
            if(currentNum>mostNum){
                mostNum = currentNum;
                modalNumber=arr[i];
            }
            currentNum = 0;//下一个元素不同，归零
        }
    }
    cout << "众数是：" << modalNumber << endl;
    return 0;
}
```

该算法首先需要将序列排序，快速排序的时间复杂度为O(nlogn)，排序后需要遍历整个序列因此时间复杂度为O(n)，总时间复杂度为O(nlogn)。空间复杂度为O(1)，只需要几个常数级的空间来处理。

- 法二：统计每个元素出现次数，可使用哈希表来储存每个数字出现的次数，之后遍历序列更新最大值，得到众数。

```
#include<iostream>
using namespace std;
#include<unordered_map>
#define MaxSize 25

int main(){
    int arr[MaxSize];
    srand((unsigned)time(NULL));
    cout<<"数组元素为：">endl;
    for (int i = 0;i<MaxSize;i++){
        arr[i] = rand()%20;
        cout<<arr[i]<<" ";
    }
    cout<<endl;

    int modalNumber;//众数
    int MaxNum = 0;//最多数的数量

    unordered_map <int,int> map;//定义一个哈希表
    for (int i = 0;i<MaxSize;i++){
        map[arr[i]]++;//将数组中的元素作为哈希表的键，出现的次数作为值
        if(map[arr[i]]>MaxNum){
            MaxNum = map[arr[i]];
            modalNumber = arr[i];
        }
    }
    cout<<"众数是：">modalNumber<<endl;
    return 0;
}
```

该方法只需遍历一次数组，故时间复杂度为O(n)；因要创建哈希表，其空间复杂度为O(n)。

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

最容易想到的方法是将序列排序，排序后将序列前后等分一定可以得到最大差值。

代码如下:

```
#include<iostream>
#include<algorithm>
using namespace std;
#define MaxSize 24
int main(){
    int arr[MaxSize];
    srand((unsigned)time(NULL));
    cout<<"数组元素为：">endl;
    for (int i = 0;i<MaxSize;i++){
        arr[i] = rand()%20;
        cout<<arr[i]<<" ";
    }
    cout<<endl;

    sort(arr,arr+MaxSize);

    cout<<"序列分为两部分：">endl;
```

```
    cout << "s1:";
    for (int i = 0; i < MaxSize / 2; i++){
        cout<<arr[i]<<" ";
    }
    cout << endl;
    cout << "s2:";
    for (int i = MaxSize / 2; i < MaxSize; i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

该方法的时间复杂度为O(nlogn)。