一、题目说明

题目是41. First Missing Positive，求一个未排序队列中缺失的最小正整数。时间复杂度要求是O(n)。难度是Hard，确实难。

二、我的解答

不考虑时间复杂度，首先对队列进行排序，然后从第一个正数开始，如果不是1就返回1，否则继续查找2....找不到就返回，找到就继续。

代码如下：

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<unordered_map>
using namespace std;
class Solution{
    public:
        int firstMissingPositive(vector<int>& nums){
            sort(nums.begin(),nums.end());
            int cur = 1;

            int start = 0;
            while(start<nums.size() && nums[start]<=0){
                start++;
            }

            if(start>= nums.size()){
                return 1;
            }
            if(nums[start] != 1){
                return 1;
            }else{
                for(int t=start;t<nums.size();t++){
                    if(nums[t] == cur){
                        cur++;
                    }
                    if(nums[t]<=cur){
                        continue;
                    }
                }

                if(cur==nums[nums.size()-1]){
                    cur++;
                }
            }

            return cur;
        }
};
int main(){
    Solution s;
    vector<int> r;

    r = {1,2,0};
```

```
    cout<<(3==s.firstMissingPositive(r))<<"\n";

    r = {3,4,-1,1};
    cout<<(2==s.firstMissingPositive(r))<<"\n";

    r = {7,8,9,11,12};
    cout<<(1==s.firstMissingPositive(r))<<"\n";

    r = {0,2,2,1,1};
    cout<<(3==s.firstMissingPositive(r))<<"\n";

    r = {1,2,3};
    cout<<(4==s.firstMissingPositive(r))<<"\n";
    return 0;
}
```

性能如下：

```
Runtime: 4 ms, faster than 65.35% of C++ online submissions for First Missing
Positive.
Memory Usage: 8.6 MB, less than 92.00% of C++ online submissions for First
Missing Positive.
```

三、优化措施

上述实现，排序的时间复杂度一般是O(Nlog(N)),是不满足要求的。对于这个未排序的队列，可以这样处理：从第1个数开始，负数不做处理，如果nums[i] ！ = 1+i，就将nums[i]和 nums[nums[i] - 1]交换,一个循环就可以了。代码如下：

```
class Solution{
    public:
        int firstMissingPositive(vector<int>& nums){
            int len = nums.size();
            for(int i=0;i<len;){
                if(nums[i]<=len && nums[i]>=1 && nums[i]!=nums[nums[i]-1]){
                    int temp = nums[nums[i] - 1];
                    nums[nums[i] - 1] = nums[i];
                    nums[i] = temp;
                }else{
                    i++;
                }
            }

            int i=0;
            while(i<len && i+1==nums[i]){
                i++;
            }
            return i+1;
        }
};
```