一、时间复杂度分析

1.

$f_1()$ 执行 n 次插入，set底层实现是红黑树（一种旋转平衡树），单次操作的时间为 $O(logn)$， 因此总时间复杂度为 $O(nlogn)$ 。
$f_2()$ 遍历set，单次操作复杂度均摊为 $O(1)$，因此总时间复杂度为 $O(n)$ 。
2.
读入总复杂度为 $O(nm)$， 正常排序的复杂度为 $O(nlogn)$，但排序对象是string类型，应乘上m，因此总时间复杂度为 $O(mnlogn)$ 。

二、简答

1. 前者广泛适用于多种数据结构，后者是专门为list定制的排序方法，效率更优。
2. 第一问同上。set的 $find()$ 时间复杂度为 $O(logn)$,map采用的是hash方法，平均复杂度为 $O(1)$ 。
3. 重载小于号或者使用仿函数greater。

三、作业H2

C. 桶装数字

题目大意

给定m个无序数对，排序输出。

解法

使用make_pair()方法构造数对，sort排序即可。

时间复杂度

$O(mlogm+n)$

代码

```
int main (){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<pair<int, int> > a;
    int x, y;

    for (int i = 0; i < m; i++) {
        cin >> x >> y;
        a.push_back({y, x});
    }

    sort(a.begin(), a.end());
    auto it = a.begin();

    for (int i = 1; i <= n;) {
        if (it != a.end() && it->first == i) {
            cout << it->second << " ";
            it++;
        }
        else {
            i++;
            cout << '\n';
        }
    }
    return 0;
}
```

D. 笔记本

题目大意

模拟一个单词本。

解法

根据定义模拟，由于 $op = 3$ 时要求按照字典序输出，使用有序结构set。

时间复杂度

$O(mlogm)$,考虑m次执行 $op = 1$ 操作。

代码

```cpp
int main (){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int m;
    cin >> m;

    set<string> S;
    while (m--) {
        int op;
        string str;
        cin >> op;

        if (op == 1) {
            cin >> str;
            if (S.find(str) != S.end()) {
                cout << "found\n";
            }
            else {
                cout << "write\n";
                S.insert(str);
            }
        }
        else if (op == 2) {
            cin >> str;
            if (S.find(str) != S.end()) {
                cout << "erased\n";
                S.erase(str);
            }
            else {
                cout << "not found\n";
            }
        }
        else {
            for (auto i : S) {
                cout << i << ' ';
            }
            cout << '\n';
        }
    }
    return 0;
}
```