《第七次上机实验》解题报告

1.序列调度

1.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

有一个 N 个数的序列 A：1，2，……，N。有一个后进先出容器 D，容器的容量为 C。如果给出一个由 1 到 N 组成的序列，那么可否由 A 使用容器 D 的插入和删除操作得到。

输入格式:
第 1 行，2 个整数 T 和 C，空格分隔，分别表示询问的组数和容器的容量，1≤T≤10，1≤C≤N。
第 2 到 T+1 行，每行的第 1 个整数 N，表示序列的元素数，1≤N≤10000。接下来 N 个整数，表示询问的序列。

输出格式:
T 行。若第 i 组的序列能得到，第 i 行输出 Yes；否则，第 i 行输出 No,1≤i≤T。

1.2 思路

使用一个栈模拟，记录入栈的最小元素 m；
每次从先 m 开始入栈，一直到当前数字，当栈顶元素和当前元素相同时弹栈，反之则判定不能满足；

1.3 代码

```
#include<iostream>
#include<stack>
inline int read() {
    int x = 0; int w = 1; register char c = getchar();
    for (; c ^ '-' && (c < '0' || c > '9'); c = getchar());
    if (c == '-') w = -1, c = getchar();
```

```
    for (; c >= '0' && c <= '9'; c = getchar()) x = (x << 3) + (x << 1)
+ c - '0';
    return x * w;
}
using namespace std;
const int maxn = 10005;
int n, ssize;

stack<int>s;
bool check() {
    bool ret = 1;
    int N = read();
    while (!s.empty())s.pop();
    int ex = 1, x;
    for (int i = 1; i <= N; i++) {
        x = read();
        if (!ret)continue;
        if (x == ex)s.push(ex++);
        while (x >= ex)s.push(ex++);
        if (s.size() > ssize)  ret = 0;
        if (s.top() == x)s.pop();
        else ret = 0;
    }
    return ret;
}

int main() {
    n = read(), ssize = read();
    for (int i = 0; i < n; i++) {
        if (check())printf("Yes\n");
        else printf("No\n");
    }
    return 0;
}
```

2. 学院通知

2.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

学院有 n 个学生，每个学生可以通知其余若干学生。通知不一定是相互的，即 a 能通知 b、b 不一定能通知 a。
现在，学院要发布一个重要的通知，必须通知学院的所有学生。问最少需要通知多少学生、才可能确保通知到学院里所有学生。

输入格式:
第一行有两个整数 n 和 m， 表示 n 个学生（编号 1~n）和 m 种通知方式，n≤500，m≤100000。
以下 m 行，每行两个整数有 a 和 b，表示 a 能通知 b， 1≤a,b≤n。

输出格式:
一行 最少需要通知的学生人数。

2.2 思路

使用并查集，但要处理多个父节点的情况：因为要求最少通知数,当一个节点已经被合并过时，不必再次合并；
另须特判处理环；

2.3 代码

```
#include<iostream>
using namespace std;
int fa[505];
int Find(int x){
    if (fa[x] <= 0) return x;
    return  fa[x] = Find(fa[x]);
}
bool Union(int x, int y) {
    int yf = Find(y);
    if (x == yf)return false;//判环
    fa[x] = yf;
    return true;
}
int main() {
    int n = 0, m = 0;
    scanf("%d%d", &n, &m);

    int u, v;
    for (int i = 0; i < m; i++) {
```

```
        scanf("%d%d", &u, &v);

        if (0 == fa[v])
            if(Union(v, u))n--;
    }
    printf("%d", n);
    return 0;
}
```

3.前缀查询

3.1 题目描述

分数 100
作者 谷方明
单位 吉林大学
已知有 n 个单词，单词均由小写字母构成。给出一个字符串，请统计以该字符串为前缀的单词的数量。规定：一个单词是其自身的前缀。单词和字符串都不空，长度都不超过 20。

输入格式:
第 1 行包含两个整数 n 和 m，n≤10000,m≤10000，分别表示单词的个数和查询的个数。
接下来的 n 行，每行一个单词。
接下来的 m 行，每行一个字符串。

输出格式:
多行，每行一个整数，对应查询字符串的统计数量。

3.2 思路

要查询多个单词的前缀，使用字典树。
因查询的是前缀数量，所以每次插入时，路径上所有节点次数增 1；

3.3 代码

```cpp
#include<iostream>

using namespace std;
const int maxn = 1e4 + 5;
struct node {
    int end;
    int son[26];
}trie[maxn*20];
int tp=1;
void insert(string s) {
    int cur = 0;
    for (int i = 0; i < s.size(); i++) {
        int c = s[i] - 'a';
        if (trie[cur].son[c] == 0) {
            trie[cur].son[c] = ++tp;
        }
        trie[trie[cur].son[c]].end++;
        cur = trie[cur].son[c];
    }
}
int query(string s) {
    int cur = 0;
    for (int i = 0; i < s.size(); i++) {
        int c = s[i] - 'a';
        if (trie[cur].son[c] == 0)return 0;

        cur = trie[cur].son[c];
    }
    return trie[cur].end;
}

int main() {
    int m, n;
    cin >> m >> n;
    string s;
    for (int i = 1; i <= m; i++) {
        cin >> s;
        insert(s);
    }
    for (int i = 1; i <= n; i++) {
        cin >> s;
        cout<<query(s)<<'\n';
    }
    return 0;
```

}

4. 数据结构设计 I

4.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

小唐正在学习数据结构。他尝试应用数据结构理论处理数据。最近，他接到一个任务，要求维护一个动态数据表，并支持如下操作：
插入操作（I）：从表的一端插入一个整数。
删除操作（D）：从表的另一端删除一个整数。
取反操作（R）：把当前表中的所有整数都变成相反数。
取最大值操作（M）：取当前表中的最大值。 如何高效实现这个动态数据结构呢？

输入格式:
第 1 行，包含 1 个整数 M，代表操作的个数， 2≤M≤1000000。
第 2 到 M+1 行，每行包含 1 个操作。每个操作以一个字符开头，可以是 I、D、R、M。如果是 I 操作，格式如下：I x, x 代表插入的整数，-10000000≤x≤10000000。

输出格式:
若干行，每行 1 个整数，对应 M 操作的返回值。如果 M 和 D 操作时队列为空，忽略对应操作。

4.2 思路

难点为 1.取负值和插入操作交替 2.查询；
考虑使用 queue 保存输入顺序；
设置一个标志 flag，标记累计取负值的结果，flag 只有两种状态；
每次插入时，若 flag 标记负，则将元素取负值后插入，而查询时，若 flag 标记负，则将找到的元素取负值后输出；
查询时，同样依据 flag 判断找最大或最小，因要同时满足查找最大最小，考虑使用 mutiset；

4.3 代码

```
#include<iostream>
#include<set>
#include<queue>

inline int read() {
    int x = 0; int w = 1; register char c = getchar();
    for (; c ^ '-' && (c < '0' || c > '9'); c = getchar());
    if (c == '-') w = -1, c = getchar();
    for (; c >= '0' && c <= '9'; c = getchar()) x = (x << 3) + (x << 1)
+ c - '0';
    return x * w;
}
using namespace std;

const int maxn = 1e6 + 5;
bool isneg= false;
queue<int>Q;
multiset<int>S;
void Insert() {
    int x = read(); x = (isneg) ? -x : x;
    S.insert(x), Q.push(x);
}
void Del(){
    if (Q.empty())return;
    int x = Q.front(); Q.pop();
    S.erase(S.find(x));
}
void Mfind() {
    if (Q.empty())return;
    if (isneg)printf("%d\n",- *S.begin());
    else printf("%d\n",*S.rbegin());
}


int main() {
    int n = read();
    char ch;
    for (int i = 1; i <= n;i++) {
        while ((ch = getchar()) != 'I'&& ch != 'D'&&ch != 'M' && ch !=
'R');
        switch (ch) {
        case 'I':Insert(); break;
        case 'D':Del(); break;
        case 'M':Mfind(); break;
```

```
            case 'R':isneg= !isneg; break;
            }
        }
        return 0;
}
```