哈希

整数哈希

- 有一个空数组，有Q次操作，每次操作要么是给数组中添加一个元素，要么是查询某个元素是否存在

解法

- 暴力的做法：每次查找枚举前面所有出现过的数，复杂度是Q^2的

- Q * logQ的做法：使用map<int,int>

```
    int q, t, num;
    map<int, int> mp;
    scanf("%d", &q);
    while (q--) {
        scanf("%d%d", &t, &num);
        if (t == 1) {
            mp[num] = 1;
        } else {
            if (mp.find(num) != mp.end()) {
                printf("Yes");
            } else {
                printf("No");
            }
        }
    }
```

字符串哈希

题目描述：

聪明的chnlkw收到了很多礼物，但是作为chnlkw的经纪人萝卜，很想知道chnlkw这次共收到了价值多少的礼物。Chnlkw对每一件礼物都作了自动的登记。而经纪人通过网络查找到了它们的价值。现在请你帮他们计算一下礼物的总价值。

输入格式：

第一行正整数n，k 表示共收到了n件礼物，经纪人查到了k件物品的价值；
接下来的n行，包含礼物的名称(名称不包含空格)和这件礼物件数，中间有且仅有一个空格；
接下来的k行，包含了经纪人查到物品名称和价值，可能有重复，以最后一次出现的名称和价值为准，中间有且仅有一个空格；

map<string,int>

- 将字符串映射到一个整数，后面出现相同的会覆盖前面的。

更多哈希

- map<pair<int,int>, int>
- map<vector<int>, int>
- …

追求更高的效率

- 散列表（哈希表）

- 以整数为例，通过一个哈希函数，将大整数投射到小整数上，如果有多个大整数投射到了同一个小整数，就在小整数的地方建立一个链表，保存所有投射到同一个地方的数。

- 查找某个大整数是否存在，只需去投射到的位置的链表上查找即可

- 复杂度取决于哈希函数，哈希函数设计的好，投射后的数就会分布的比较均匀，平均访问的代价就会比较小

整数哈希函数

- 对一个素数取模，可以选取比数据量大一个量级的数，比如有10w个整数需要哈希，模数就选取1000007，在空间允许的范围内，多花点空间换取时间效率是值得的

```
struct hash_table {
    int head[MOD];
    int nxt[N];
    int value[N];
    int pnt[N];
    int E;
    void init () {
        E = 0;
        memset(head, -1, sizeof(head));
    }
    void insert(int number) {
        int h = number % MOD;
        for (int i = head[h]; i != -1; i = nxt[i]) {
            if (pnt[i] == number) {
                return ;
            }
        }
        pnt[E] = number;
        nxt[E] = head[h];
        head[h] = E++;
    }
    bool find(int number) {
        int h = number % MOD;
        for (int i = head[h]; i != -1; i = nxt[i]) {
            if (pnt[i] == number) {
                return true;
            }
        }
        return false;
    }
};
```

字符串哈希函数

- 选取两个素数mod1，mod2，将哈希函数出来的结果对这两个素数取模，得到一个pair，可以基本认为不会产生冲突

应用

- 求两个字符串的最长公共子串

- 首先可以二分答案mid
- 然后O(n)求出两个串中所有长度为mid的子串的哈希值，看两个串有没有公共的哈希值
- 判有没有公共的哈希值可以再用一个二分解决
- 那么主要问题就变成了如何快速求出所有长度为mid的子串的哈希值

- 每一个字符都是一个整数，因此我们可以考虑P进制哈希，P可以设为256.
- 比如abc的哈希值就为
- make_pair（（’a’*p^2 + ‘b’ * P + ‘c’）% MOD1,（’a’*p^2 + ‘b’ * P + ‘c’）% MOD2 ）
- 利用P进制哈希可以在O（1）时间内求下一个长度为mid的子串的哈希值