字符串 Hash

一、Hash 引入

问题：读入 n 个正整数，查询某个数是否在这 n 个数中出现，一共查询 m 次。（ai<=20000）

方法一：排序+二分查找。

方法二：开一个 f[20001]，f[i]表示 i 是否出现，直接 O(1)查询。

如果 ai 的值很大，方法二则无法解决，而类似这样查询某个数是否存在的问题经常遇见，如何高效的动态维护查询呢？我们可以采取 Hash 的办法。所有 Hash 就是将一个数值映射为另一个数值，即将一个值（键值）通过一定的函数运算得到一个值，得到的函数值也称为 hash 值，一般键值是一个比较大的范围，而 hash 值是一个比较小的范围。

常用 hash 方法：

1、直接寻址法

取关键字或关键字的某个线性函数值为散列地址。即 H(key)=key 或 H(key) = a • key + b

2、数字分析法

分析键值规律，选择键值中互不相同部分作为 hash 值

3、平方取中法

4、折叠法

5、除留余数法

其中，除留余数法最为简单，但是也最容易产生冲突。冲突即两个不同的键值映射为一个 Hash 值。解决冲突有以下办法：

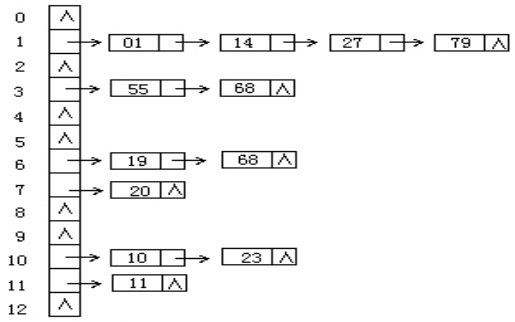
1、再 Hash 法

2、拉链法

3、公共溢出区

4、双 hash

常见的采取拉链法解决冲突。如图



有 1,14,27,79,55,68,19,68,20,10,23,11,采取取余法对 13 取余，1,14,27,79 的 hash 值都是 1，于是在值 1 后拉一条链来记录冲突。查询时，先找到 hash 值，然后在对应的链上查询。

例 1：数对 P3507

给出一串数以及一个数字 C，要求计算出所有 A-B=C 的数对的个数。

不同位置的数字一样的数对算不同的数对。

方法一：暴力求解

直接枚举两两数对，看是否满足要求即可

时间复杂度 O(N^2)

方法二：

问题中 C 是一个固定值，而朴素方法在计算时，需要重复去查找每个数是否能相减得到 C。
可以考虑 X-Y=C
变形：X-C=Y，只需要看所有比 c 大的数，然后看 X-C 有多少个即可。
我们可以设数组 F[i]表示数字 i 有多少个。
不过 i 的范围是 10^9 明显，数组不能开下，所以考虑 hash，这里采取直接取模法，mod 一个比较大的质数即可，然后用拉链法来解决冲突问题

```
/*拉链法 hash*/
#define N 200010
#define mod 1654573
using namespace std;
int first[mod+10],a[N],n,C,ans,cnt;
struct node{
    int v,nxt,sum;
}e[N];
void add(int x,int y){
    for(int i=first[x];i;i=e[i].nxt)
        if(e[i].v==y){
            e[i].sum++;return;
        }
    ++cnt;
    e[cnt].v=y;e[cnt].sum++;
    e[cnt].nxt=first[x];
   ________________
}
int query(int x,int y){
    for(int i=first[x];i;i=e[i].nxt)
        if(e[i].v==y){
            return e[i].sum;
        }
    return 0;
}

int main(){
    scanf("%d%d",&n,&C);
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
        if(a[i]-C<0)continue;
        add((a[i]-C)%mod,a[i]-C);
    }
    int ams=0;
    for(int i=1;i<=n;i++){
        ans+=query(a[i]%mod,a[i]);
    }
    printf("%d",ans);
```

```
    return 0;
}
```

二、字符串 Hash

将引入问题一中的正整数改称字符串，（每个字符串的长度不超过 20），还是输入 n 个字符串，一共 m 次询问。

做法一：我们对这些字符串排序，然后二分查找。

做法二：对每个字符串进行 Hash 操作，字符串 Hash 算法较多，这里介绍比较实用的一种算法， BKDRHash，这个算法的具体做法是，将每个字符串看着是一个大进制数，比如可以看着一个 31 进制，然后将这个进制数转为 10 进制后取模得到一个整数。即：

$$h(S)=\sum_{i=0}^{len(S)-1} S_i K^{len(S)-1-i} \mod P$$

一般我们取 $P=2^{64}$，即直接使用 unsigned long long。

K 一般设为 31,131,1313,13131 之类的。

常见操作参考代码：

```
#define BASE 31
#define ULL unsigned long long
#define N 10010
ULL H[N],p[N];
void init(){
    p[0]=1;
    for(int i=1;i<=N;i++)//求出每一位权值
        p[i]=p[i-1]*BASE;
}
ULL gethash(int l,int r){//计算一个字符串片段
    return H[r]-H[l]*p[r-l+1];
}
ULL geth(char s[]){//求字符串s的hash值
    int len=strlen(s);
    ULL ans=0;
    for(int i=0;i<len;i++)
        ans=ans*BASE+s[i];
    return ans;
}
```

例 2：凯撒密码 P3506

题意：N 个长度为 5 的密文与明文，密文->明文的加密方式是移动 m 位完成，找出对应关系及每对密文&明文的 m。

【分析】

我们发现密文不管怎么变，相邻的字母的差值（摸 26）不变，五个字母有四个差值，用四个差值构成 4 位 26 进制的哈希值，哈希值相同的是一组，找到对应关系即可。

```
const int mod=998244353;
const int maxn=5e5+10;
int n,hsh[maxn],ans=0;
char s[maxn],tmp[10];
int gethash(char s[]){
    int res=0;
    for(int i=0;i<4;i++)
        res=res * 26 + (s[i]-s[i+1]+26)%26;
    return res;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        scanf("%s",tmp);
        s[i]=tmp[0];
        hsh[gethash(tmp)]=i;
    }
    for(int i=1;i<=n;i++){
        scanf("%s",tmp);
        int tt=hsh[gethash(tmp)];
        int m=(tmp[0] - s[tt] + 26)%26;
        ans=(ans+ (i^tt^m))%mod;
    }
    printf("%d\n",ans);
    return 0;
}
```

例 2：stringP3508

定义两个字符串匹配为它们的最小表示法相同 给定一个模式串和 n 个主串，求模式串对每一个主串的模式匹配次数 字符集大小{‘a’ ^’..’z’}

【分析】

建立哈希表，将模式串的所有最小表示法，建立一个哈希表。

依次枚举主串每一位开始的子串是否在哈希表中

```
int main(){
    scanf("%s",ch+1);m=strlen(ch+1);
    for(int i=1;i<=m;++i)hs=hs*base+ch[i];
    for(int i=1;i<m;++i)pw=pw*base;
    for(int i=1;i<=m;++i){
        f[hs]=1;
        hs-=pw*ch[i];
        hs=hs*base+ch[i];
    }
    scanf("%d",&n);
    while(n--)  {
        scanf("%s",ch+1);l=strlen(ch+1);
        if(l<m){puts("0");continue;}
        int ans=0;hs=0;
        for(int i=1;i<m;++i)hs=hs*base+ch[i];
        for(int i=m;i<=l;++i)       {
            hs=hs*base+ch[i];
            ans+=f[hs];
            hs-=pw*ch[i-m+1];
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

三、Map 的使用

功能：将一个值映射为另外一个值

格式：

map<类型 1，类型 2>变量名

如：map<string ,int >f;

使用方法：

1、[]使用，如 f[“abc”]=1，如果已经有键值则修改否则是插入

2、count 函数，查找键值

如：f.count(“abc”) 返回是否存在

例 3：P10034. 「一本通 2.1 例 2」图书管理

题意: 图书管理是一件十分繁杂的工作，在一个图书馆中每天都会有许多新书加入。为了更方便的管理图书（以便于帮助想要借书的客人快速查找他们是否有他们所需要的书），我们需要设计一个图书查找系统。

该系统需要支持 2 种操作：

add(s) 表示新加入一本书名为 s 的图书。

find(s) 表示查询是否存在一本书名为 s 的图书。

【分析】直接将每个字符串通过 Hash 映射为一个值，然后直接查询即可。

参考代码

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 30050
int n;
typedef unsigned long long ull;
map<ull, bool> exist;
ull read(){
    char c = getchar(); ull z = 0;
    while(c != '\n') z = z * 223 + c, c = getchar(); return z;
}
int main(){
    scanf("%d", &n); char op[10];
    for(int i = 1; i <= n; i++){
        scanf("%s", op);
        ull now = read();
        if(op[0] == 'a') exist[now] = true;
        else exist[now] ? puts("yes") : puts("no");
    } return 0;
}
```