哈希表

哈希的应用主要分成两类：一类是哈希表，用于查找；另一类是字符串哈希，用于匹配。

【查找问题】查找数值x在n个数中是否存在？

【分析】该问题的解决思路有2个：

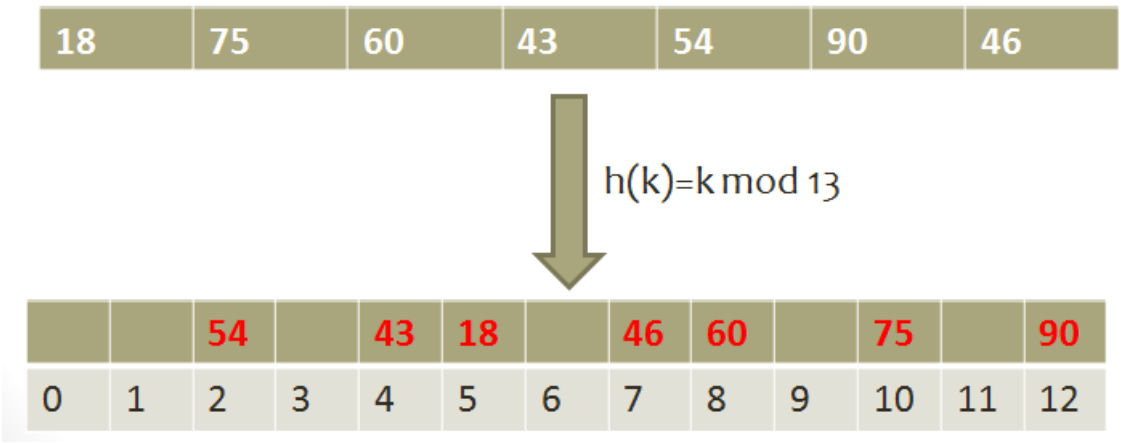
1.顺序查找，这个方式的时间复杂度为O（n）。如果序列是有序的，借助二分查找，能达到O（log n）的效率。但若序列是无序的，又遇到稍大的数据量如10^9，将面临时间超限的问题。

2.桶排。输入x后做出标记，a[x]=x。查找即是对a[x]的值进行判断，如等于x则存在，时间复杂度为O（1）。该处理方式虽然快但涉及到空间问题，若序列中的数值相对较稀疏时，空间的开销会较大。

如面临大数据量的查找时，我们能将这两种方式的有点相结合，综合时空优势，无疑是一个有效的解决途径。而哈希表，便是处于这两种方式的中间点，平衡时空问题，有效进行查找。

哈希表原理

哈希表的基本原理是使用一个下标范围比较大的数组A来存储元素，设计一个函数h，对于要存储的线性表的每个元素node，取一个关键字key，算出一个函数值h（key），把h（key）作为数组下标，用A[h(key)]这个数组单元来存储node。也可以简单理解为，按照关键字为每一个元素“分类”，然后将这个元素存储在相应“类”所对应的地方。 如图所示：

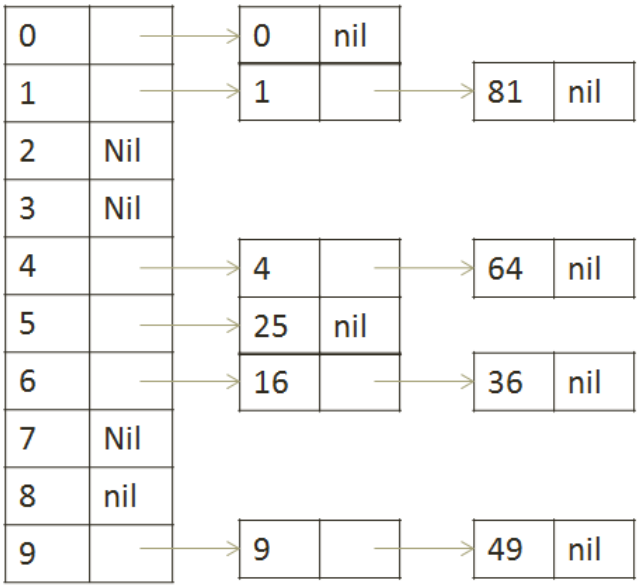


关于哈希函数的构造有三点需要注意：

1.构造哈希函数有两个标准：简单和均匀。
2.哈希函数的关键字一般是整数或者字符串。
3.最常用的哈希函数：除余法，即h(key)=key mod m。m为素数，可以使得较好保证分布均匀。 ·

【思考】在这个过程中，会遇到两个关键字均映射到同一个值的情况。如增加第8个元素，值为30，那么按照原设定的哈希函数将计算出一个函数值4，但A[4]已经存储了原第4个元素的值，不能被覆盖。这种情况该如何处理？

这种情况我们称为“冲突”，是在哈希表构造中出现的一个常见问题，我们可采用拉链法，把哈希值相同的关键字串成链表，即哈希值为x的所有元素存储在以head[x]为表头的单链表中。



也常用动态数组vector来解决，下面以vector的使用为例，学习哈希表的基本操作。

```
#include<iostream>
#include<vector>
using namespace std;
const int mod=10007;
vector<int> h[10007];
int main(){
    int n,m,x;
    cin>>n;
    for(int i=0;i<n;i++){
        cin>>x;
        insert(x);//输入数据，分类构建哈希表
    }
    cin>>m;
    for(int i=0;i<m;i++){
        cin>>x;
        if(find(x))cout<<"yes"<<endl;//构建哈希表查询函数，在小范围内进行查询
        else cout<<"no"<<endl;
    }
}

```

在代码的书写中要注意mod的取值：首先，mod为素数能更好的处理均匀分布的问题。其次，mod的值是时空的平衡点。数值低，则分配类别少，占用空间小，但每个类别存数量增大，查询遍历时间长。相反，占用空间大，查询遍历时间少。

【元素的插入】

```
void insert(int x){
    h[x%mod].push_back(x);
}
```

【元素查找】

```
bool find(int x){
    for(int i=0;i<h[x%mod].size();i++){
        if(h[x%mod][i]==x)return true;
    }
    return false;
}
```

A.RY哥查字典

【题目描述】

RY哥最近新买了一本字典,他十分高兴,因为这上面的单词都十分的和谐,他天天查字典。

【输入】

1个整数N,表示字典里面的单词数量。
接下来N行,每行一个字符串,表示一个单词。
然后第N+2行,一个整数M,表示要查的单词数。
接下来M行,每行一个字符串,表示一个要查的单词。

【输出】

对于每一个要查的单词,如果在字典里面,就输出'Yes',否则输出'No',一行一个。

样例输入	样例输出
2 i you 1 love	No

【分析】数值映射为一个数字，可以通过求膜的方式实现。而本题是查找一个字符串，该如何映射成一个数字？

方法一：将各个字符的ascii码值相加将取模。但该方法存在不能区分不同顺序的字符串的弊端，如abc和acb的值相同。

方法二：将各个字符的ascii码值乘以位权后相加再取模。

相比之下，方法二更为准确。

【代码】

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;
int n,m,ans;
string s;
vector<int> hs[10007];
bool check;
int main(){
    cin>>n;
    while(n--){
        cin>>s;
        int h=0;
```

```
        for(int i=0;i<s.size();i++){
            h+=s[i]*(i+1);//字符哈希函数，将各个字符的ascii码值乘以位权后相加
        }
        hs[h%10007].push_back(h);//哈希值取模后对数据分类存储
    }
    cin>>m;
    while(m--){
        cin>>s;
        int h=0;
        check=false;
        for(int i=0;i<s.size();i++){
            h+=s[i]*(i+1);
        }
        for(int i=0;i<hs[h%10007].size();i++){
            if(h==hs[h%10007][i]){
                cout<<"Yes"<<endl;
                check=true;
                break;
            }
        }
        if(!check) cout<<"No"<<endl;
    }
}
```

【解法二】键值对map也具有类哈希功能，在该数据结构中数组的下标不再局限于数字，字符串可直接作为数组下标，即A[“ss”]=值。但是插入、查找的效率不一定比哈希高，时间复杂度为O（logn）。

【代码】

```
#include<iostream>
#include<map>
#include<string>
using namespace std;
map<string,bool> mp;//map<健，值>  注意定义的写法
int n,m;
string s;
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>s;
        mp[s]=true;
    }
    cin>>m;
    for(int i=1;i<=m;i++){
        cin>>s;
        if(mp[s]==true)cout<<"Yes"<<endl;
        else cout<<"No"<<endl;
    }
}
```

B.数列重复项

【问题描述】

有一个数列a ，a_0=1, a_i+1 = (A*a_i+a_i mod B)mod C要求这个数列第一次出现重复的项的标号。

【输入】

一行三个数，分别表示A,B,C 。

【输出】

输出第一次出现重复项的位置，如果答案超过2*10^6输出-1 。

样例输入	样例输出
2 2 9	4

【提示】

30%的数据 $A, B, C <= 10^5$ ；
100%的数据$A, B, C <= 10^9$ 。

【分析】根据题意计算每一项a_i，在已存在的序列中查找，如存在即可输出结果。本题的数据量达10^9，若每一项都往前逐个遍历，时间复杂度接近O（n^2）。因此需要通过哈希表来维护，如用map来处理会出现时间超限的问题。

【代码】

```
#include<iostream>
#include<vector>
using namespace std;
long long a,b,c,x=1;
vector<int> h[100007];//通过确定合适的mod值，平衡时间和空间消耗
void insert(int x){
    int hx=x%100007;
    h[hx].push_back(x);
}
bool find(int x){
    int hx=x%100007;
    for(int i=0;i<h[hx].size();i++){
        if(h[hx][i]==x) return true;
    }
    return false;
}
int main(){
    cin>>a>>b>>c;
    scanf("%lld%lld%lld",&a,&b,&c);
    h[1].push_back(1);//注意：a_0项必须先插入，若忽略则可能造成查询结果出错
    for(int i=1;i<=2000000;i++){
        x=(x*a+x%b)%c;
        if(find(x)){
            printf("%d",i);
            return 0;
        }else{
            insert(x);
        }
    }
    printf("-1");
}
```

C.收集雪花

【问题描述】

不同的雪花往往有不同的形状。在北方的同学想将雪花收集起来，作为礼物送给在南方的同学们。一共有n个时刻，给出每个时刻下落雪花的形状，用不同的整数表示不同的形状。在收集的过程中，同学们不希望有重复的雪花。你可以从任意a时刻开始，在b时刻停止。a到b时刻中间的雪花也都将被收集。他们希望收集的雪花最多。

【输入】

第一行一个正整数n；第2行到第n+1行表示n个时刻雪花的形状。

【输出】

最多能收集雪花的数量。

样例输入	样例输出
5 1 2 3 2 1	3

【提示】

$n \leq 10^6$, $xi \leq 10^9$

【分析】

本题是要在序列中寻找一个区间，区间内元素不重复，且长度为全局最长。结合线性动规思想，f_i记录以每一项为终点的最长区间的起始位置。因此，每一项元素要通过结构体记录雪花的类别，以及该类别在此之前最后出现的位置。遍历每一项，通过哈希表的构建查询当前项类别的雪花是否存在，无则插入，有则相应修改该类别的末尾位置信息，通过末尾信息逐项更新f_i的值，求出区间长度，全局判断求最大值。

【代码】

```
#include<iostream>
#include<vector>
#include<cstdio>
using namespace std;
struct snow{
    int x,last;
};
vector<snow> v[10007];
int n,a,f[1000001],ans;
int search(int a,int p){
    int i=a%10007;
    int id=0;
    for(int j=0;j<v[i].size();j++){
        if(v[i][j].x==a){
            id=v[i][j].last;
            v[i][j].last=p;
            return id;
        }
    }
    v[i].push_back((snow){a,p});
    return 0;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        scanf("%d",&a);
```

```
        int p=search(a,i);//查询与第i项同类别的雪花在此之前的最末尾
        f[i]=max(f[i-1],p+1);//结合前一项情况决策当前最有策略
        ans=max(ans,i-f[i]+1);
    }
    printf("%d",ans);
}
```

D.马农

【问题描述】

在观看完战马检阅之后，来自大草原的两兄弟决心成为超级“马农”，专门饲养战马。
兄弟两回到草原，将可以养马的区域，分为N*N的单位面积的正方形，并实地进行考察，归纳出了每个单位面积可以养马所获得的收益。接下来就要开始规划他们各自的马场了。
首先，两人的马场都必须是矩形区域。同时，为了方便两人互相照应，也为了防止马匹互相走散，规定两个马场的矩形区域相邻，且只有一个交点。最后，互不认输的两人希望两个马场的收益相当，这样才不会影响他们兄弟的感情。
现在，兄弟两找到你这位设计师，希望你给他们设计马场，问共有多少种设计方案。

【输入】

第一行一个整数N，表示整个草原的大小为N*N。
接下来N行，每行N个整数A(i,j)，表示第i行第j列的单位草地的收成。
（注意：收益可能是负数，养马也不是包赚的，马匹也可能出现生病死亡等意外。）

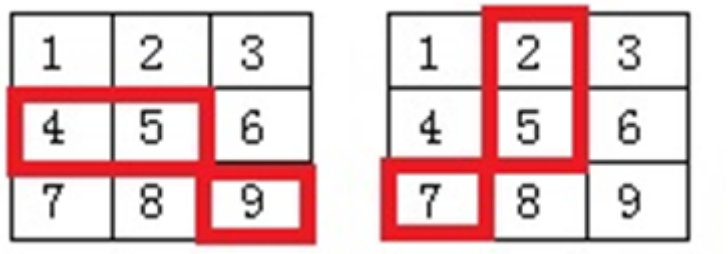
【输出】

输出符合两人要求的草原分配方案数。

样例输入	样例输出
3 1 2 3 4 5 6 7 8 9	2

【提示】

[样例解释]



[数据范围]

40%的数据，$N \leq 10$。
100%的数据，$N \leq 50, -1000 < A(i,j) < 1000$。

【分析】方案成型的两个关键点：区间和相等、只有一个相交点。因此可预先通过计算求出该二维区间的前缀和，通过枚举区间所有可能的相交点（i，j）。第一步先计算以点（i，j）为右下角的矩形区间和s，将s的所有可能数值构建出哈希表。第二步以点（i+1，j+1）为左上角计算所有可行的矩形区间和ss，每一个值均在哈希表中查找，如存在则说明该方案可成型。但在其中注意的有两个问题：1.可能有

多个不同的区域，但区间和的值是相等的，需注意方案数的累加值；2.以（i，j）为相交点，会有左上右下及左下右上两类情况，必须分开枚举。

【代码】

```
#include<iostream>
#include<vector>
using namespace std;
int n,s[52][52],ans;
const int mod=1007;
struct sum{
    int data,k;
};
vector<sum> h[1007];
void clean_h(){
    for(int i=0;i<1007;i++) h[i].clear();
}
int cal(int i,int j,int x,int y){
    return s[i][j]-s[i][y-1]-s[x-1][j]+s[x-1][y-1];
}
void insert(int x){
    int i=(x+250000)%mod;
    int len=h[i].size();
    for(int j=0;j<len;j++){
        if(x==h[i][j].data){
            h[i][j].k++;
            return;
        }
    }
    h[i].push_back((sum){x,1});
}
int finds(int x){
    int i=(x+250000)%mod;
    int len=h[i].size();
    for(int j=0;j<len;j++){
        if(x==h[i][j].data) return h[i][j].k;
    }
    return 0;
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            cin>>s[i][j];
            s[i][j]=s[i][j]+s[i-1][j]+s[i][j-1]-s[i-1][j-1];
        }
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){//枚举交点
            clean_h(); //注意需清空原哈希表
            //左上右下组合
            for(int x=1;x<=i;x++){
                for(int y=1;y<=j;y++){//左上区域
                    int t=cal(i,j,x,y);
                    insert(t);
                }
            }
```

```
            for(int x=i+1;x<=n;x++){//右下区域
                for(int y=j+1;y<=n;y++){
                    int t=cal(x,y,i+1,j+1);
                    ans+=finds(t);
                }
            }
            //右上左下组合
            clean_h(); //注意需清空原哈希表
            for(int x=1;x<=i;x++){
                for(int y=j;y<=n;y++){//右上区域
                    int t=cal(i,y,x,j);
                    insert(t);
                }
            }
            for(int x=i+1;x<=n;x++){//左下区域
                for(int y=1;y<=j-1;y++){
                    int t=cal(x,j-1,i+1,y);
                    ans+=finds(t);
                }
            }
        }
    }
    cout<<ans;
}
```

E.四轮车

【问题描述】

在地图上散落着 n 个车轮，小 J 想用它们造一辆车。要求如下：
1、一辆车需要四个车轮，且四个车轮构成一个正方形
2、车轮不能移动
你需要计算有多少种造车的方案（两个方案不同当且仅当所用车轮不全相同，坐标相同的两个车轮视为不同车轮）。

【输入】

第一行一个整数 n
接下来 n 行，每行两个整数 x y，表示在(x,y)处有一个车轮

【输出】

一行一个整数，表示方案数

样例输入	样例输出
9 0 0 1 0 2 0 0 2 1 2 2 2 0 1 1 1 2 1	6

【提示】

30%的数据保证 $n \leq 30$
100%的数据保证 $1 \leq n \leq 1000; |x|, |y| < 20000$

【分析】

本题的解决思路是枚举出序列中的两个点a与b，计算出以次两点连线为边的正方形另外两个顶点c与d的位置，查询c与d在序列中是否存在，如有则存在方案。本题的难点是c与d位置的计算，考察几何运算能力。在解题过程中还需要注意一下3点：1.确定以a与b连线为边的正方形存在两个方位的情况，需要将所有可能罗列齐全；2.将一个点的横纵坐标两个值映射为数字构建哈希表，需注意坐标点本身的值会存在负数情况，这个在设定哈希函数需要考虑在内；3.对于一个正方形，四条边都会被枚举到，因此最后累计出的方案数需要除以四。

【代码】

```
#include<iostream>
#include<vector>
using namespace std;
struct point{
    int x,y;
}a[1001];
int n,ans;
vector<point> v[10007];
void insert(int x,int y){
    int i=(x+y+40000)%10007;
    v[i].push_back((point){x,y});
}
bool find(int x,int y){
    int i=(x+y+40000)%10007;//确保值为非负数
    for(int j=0;j<v[i].size();j++){
        if(v[i][j].x==x&&v[i][j].y==y) return true;
    }
    return false;
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>a[i].x>>a[i].y;
        insert(a[i].x,a[i].y);
    }
    for(int i=1;i<n;i++){
        for(int j=i+1;j<=n;j++){
            int x1=a[i].x,y1=a[i].y;
            int x2=a[j].x,y2=a[j].y;
            int x3,y3,x4,y4;//注意有两个方位的正方形，分别计算出两种情况的第3、第4个点
            x3=x1+(y2-y1);
            y3=y1-(x2-x1);
            x4=x2+(y2-y1);
            y4=y2-(x2-x1);
            if(find(x3,y3)&&find(x4,y4)) ans++;
            x3=x1-(y2-y1);
            y3=y1+(x2-x1);
            x4=x2-(y2-y1);
            y4=y2+(x2-x1);
            if(find(x3,y3)&&find(x4,y4)) ans++;

```

```
        }
    }
    cout<<ans/4;
}
```

F.八数码问题

【问题描述】

在3×3的棋盘上，摆有八个棋子，每个棋子上标有1至8的某一数字。棋盘中留有一个空格，空格用0来表示。空格周围的棋子可以移到空格中。要求解的问题是：给出一种初始布局（初始状态）和目标布局（为了使题目简单,设目标状态为123804765），找到一种最少步骤的移动方法，实现从初始布局到目标布局的转变。

【输入】

输入初试状态，一行九个数字，空格用0表示

【输出】

只有一行，该行只有一个数字，表示从初始状态到目标状态需要的最少移动次数(测试数据中无特殊无法到达目标状态数据)

样例输入	样例输出
283104765	4

【分析】本题属于最少步骤求解问题，我们可以通过广搜实现。在这个过程中要对已有的过程状态进行剪枝，这就需要借助哈希进行较高效的查询。难点在于字符串的处理，注意细节问题。

【代码】

```
#include<iostream>
#include<queue>
#include<map>
#include<string>
using namespace std;
string s;
int xx[4]={0,0,1,-1};
int yy[4]={1,-1,0,0};
const string sg="123804765";//目标状态
struct num{
    string ss;
    int step;
};
map<string,bool> mp;
queue<num> q;
int a[5][5],k;
bool check(int x,int y){
    if(x<1||x>3||y<1||y>3) return false;
    return true;
}
int main(){
    cin>>s;
    if(s==sg){
        cout<<0;
        return 0;
```

```
    }
    q.push((num){s,0});
    while(!q.empty()){
        num f=q.front();
        q.pop();
        int x,y;
        k=-1;
        for(int i=1;i<4;i++){
            for(int j=1;j<4;j++){
                a[i][j]=f.ss[++k]-'0';//将字符串分解为九宫格
                if(a[i][j]==0){//定位‘0’的位置
                    x=i;
                    y=j;
                }
            }
        }
        for(int i=0;i<4;i++){//可操作步奏
            int nx=x+xx[i];
            int ny=y+yy[i];
            if(check(nx,ny)){
                swap(a[x][y],a[nx][ny]);
                string ss;
                for(int ii=1;ii<4;ii++){
                    for(int jj=1;jj<4;jj++){
                        ss=ss+(char)(a[ii][jj]+'0');//交换位置后将产生新的字符串
                    }
                }
                if(!mp[ss]){//查询
                    if(ss==sg){
                        cout<<f.step+1;
                        return 0;
                    }
                    mp[ss]=true;//注意标记该状态已出现过
                    q.push((num){ss,f.step+1});
                }
            }
            swap(a[x][y],a[nx][ny]);
        }
    }
}
```