

## 目录

HDU 3613 Best Reward.....	2
Codeforces 126B Password(KMP,next 数组) .....	4
Codeforces 182D Common Divisors(KMP,求循环节) .....	5
Codeforces 149E Martian Strings(KMP).....	7
HDU 4333 Revolving Digits(KMP) .....	9
ZOJ 3587 Marlon's String.....	11
Uva 10054: The Necklace 欧拉回路.....	15
__builtin_函数 unordered_map 自定义结构体的写法 .....	19
计算几何.....	20
狗坑！ .....	20
bitset 各个函数.....	22
Stirling 数 .....	23
Bell 数 .....	25
卡特兰数.....	27
LCM 组合数 .....	34
Fwt ( 快速沃尔什变换 ) .....	36
特征方程求数列通项 .....	46
欧拉函数.....	48
pollard_rho 大数分解 Java 版.....	50
等比数列二分求和.....	51
LCA+倍增+带权并查集：POJ 3728 The Merchant .....	52
LCA：POJ 3694 Network .....	58
DFS 序线段树：HDU 5039 Hilarity .....	59
DFS 序线段树：CodeForces 258 E .....	63
DFS 序线段树：CodeForces 343 D.....	67
DFS 序线段树：CodeForces 384 E .....	70
基尔霍夫矩阵.....	73
数位 DP-D. Nass Number .....	73
数位 DP-B. Claire_ 's problem .....	75
威佐夫博弈 .....	78

2015 年 11 月 8 日  
17:31

## HDU 3613 Best Reward

2015 年 7 月 19 日  
20:22

题目链接:

<http://acm.hdu.edu.cn/showproblem.php?pid=3613>

### 题目大意：

有一个串  $S$ , 我们可以将其分成两个串, 如果某个串为回文串, 那么那个串的价值为串中各个字母的价值之和, 否则无价值, 每个字母的价值已知, 问最多能够得到多少价值。

### 做法：

扩展 KMP 的简单运用。如果一个后缀  $S(i...n)$  是回文串, 那么一定存在  $extend[i] = n - i$ 。其中  $n$  为原串的长度,  $extend[i]$  为原串从  $i$  开始的后缀与翻转串的最长公共前缀。发现这个规律的话, 只需要先求出原串所有后缀的价值, 再求出翻转串的所有后缀的价值, 再取能够组成原串的价值之和的最大值就行了。

### 代码：

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
// #include <unordered_map>
#define N 500010
#define id(x, y) ((x)*m+(y))
#define check(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int, int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
int nxt[N], extend[N];
void GetNext(const char* T)
{
    int len = strlen(T), a = 0;
```

```

    nxt[0] = len;
    while(a<len-1 && T[a] == T[a+1]) a++;
    nxt[1] = a;
    a = 1;
    for(int k=2;k<len;k++){
        int p = a + nxt[a] - 1, L = nxt[k - a];
        if(k - 1 + L >= p){
            int j = max(0, p-k+1);
            while(k+j<len && T[k+j] == T[j]) j++;
            nxt[k] = j;a = k;
        }else nxt[k] = L;
    }
}

void GetExtand(const char* S, const char* T)
{
    GetNext(T);
    int slen = strlen(S), tlen = strlen(T), a = 0;
    int Minlen = min(slen, tlen);
    while(a < Minlen && S[a] == T[a]) a++;
    extand[0] = a;
    a = 0;
    for(int k=1;k<slen;k++){
        int p = a + extand[a] - 1, L = nxt[k-a];
        if(k-1+L >= p){
            int j = max(0, p-k+1);
            while(k+j < slen && j < tlen && S[k+j] == T[j]) j++;
            extand[k] = j; a = k;
        }else extand[k] = L;
    }
}

int val[44];
char str[N], T[N];
int preval[N];
int saveans[N];
int main()
{
    Open();
    int T_T;scanf("%d", &T_T);
    while(T_T--){
        for(int i=0;i<26;i++) scanf("%d", &val[i]);
        scanf("%s", str);
        int len = strlen(str);
        preval[0] = val[str[0] - 'a'];
        for(int i=1;i<len;i++) preval[i] = preval[i-1] + val[str[i] - 'a'];
        strcpy(T, str);
        reverse(T, T+len);
        GetExtand(str, T);
        for(int i=1;i<len;i++){
            saveans[i] = (extand[i] == len - i) ? (preval[len - 1] - preval[i-1]) : 0;
            int ans = 0;
            GetExtand(T, str);
            for(int i=1;i<len;i++){
                int tmp = (extand[i] == len - i) ? (preval[extand[i] - 1]) : 0;
                ans = max(ans, tmp + saveans[len - i]);
            }
            cout<<ans<<endl;
        }
        return 0;
    }
}

```

来自 <<http://acm.hdu.edu.cn/viewcode.php?rid=14062600>>

# Codeforces 126B Password(KMP,next 数组)

2015 年 7 月 20 日  
2:50

链接:

<http://www.codeforces.com/problemset/problem/126/B>

题目大意:

给定一个字符串，需要找到一个子串，使得这个子串即是原串的前缀，又是原串的后缀，而且必须出现在原串中（不是前缀也不是后缀）。如果有多个，输出最长的一个。

做法:

这个题能够加深一下对 next 数组的理解，我们首先找到所有的满足既是前缀又是后缀的串，并用 vis 数组标记长度。然后再利用 next 数组找到所有的是原串前缀的中间串，如果出现在 vis 数组中，那么说明这个串满足条件，取最大就行了。其中第一步，“找到所有满足既是前缀又是后缀的串”这里，由于 next[len]记录的只有最长的一个，还可能存在一些小的串满足条件，但是这些小的串一定都在最长的串中。

代码:

```
int pre = nxt[len];  
while(pre > 0) vis[pre] = 1, pre = nxt[pre];
```

代码:

```
#include <iostream>  
#include <cstdio>  
#include <stack>  
#include <cstring>  
#include <queue>  
#include <algorithm>  
#include <cmath>  
//#include <unordered_map>  
#define N 1000010  
#define id(x, y) ((x)*m+(y))  
#define check(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)  
using namespace std;  
typedef pair<int,int> PII;  
const int INF=0x3f3f3f3f;  
void Open()  
{  
    #ifndef ONLINE_JUDGE
```

```

        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
int nxt[N];
void GetNext(const char* T)
{
    nxt[0] = -1; int j=0, k=-1, len = strlen(T);
    while(j < len)
        if(k == -1 || T[j] == T[k]) nxt[++j] = ++k;
        else k = nxt[k];
}
char str[N], T[N];
bool vis[N];
int main()
{
    //Open();
    scanf("%s", str);
    GetNext(str);
    int len = strlen(str);
    int ans = 0;
    int pre = nxt[len];
    while(pre > 0) vis[pre] = 1, pre = nxt[pre];
    for(int i=1; i<len; i++)
        if(vis[nxt[i]]) ans = max(nxt[i], ans);
    if(ans) str[ans] = '\0', cout<<str<<endl;
    else printf("Just a legend\n");
    return 0;
}

```

来自 <<http://codeforces.com/problemset/status?friends=on>>

## Codeforces 182D Common Divisors(KMP,求循环

节)

2015 年 7 月 20 日  
3:00

题意:

一个串的因子为这个串的循环节，比如“abababab”的因子有“ab”，  
“abab”，“abababab”，现在给出两个串，需要输出这两个串的公共因子有多少个。

## 做法:

首先可以利用 **next** 数组求出这个串的最小循环节，该串存在小于原串的因子的话，一定是由若干个相同子串链接而成，于是首先求出一个最小的循环节，再由此求出所有因子（这里可以求出这个串由多少个最小循环节构成，然后所有的因子肯定都是这个数目的约数），再标记所有因子的长度。再求出另一个串的所有因子，统计满足这样的条件（因子的长度小于两个串的最长公共前缀）的公共因子有多少个就行了。

## 代码:

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
#define id(x, y) ((x)*m+(y))
#define check(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int, int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
int nxt[N];
void GetNext(const char* T, int* nxt)
{
    int len = strlen(T), j=0, k=-1; nxt[0] = -1;
    while(j<len) if(k == -1 || T[j] == T[k]) nxt[++j] = ++k; else k = nxt[k];
}
char S[N], T[N];
bool vis[N];
int main()
{
    //Open();
    scanf("%s%s", S, T);
    int slen = strlen(S), tlen = strlen(T);
    int prefix = 0;
    int ans = 0;
    while(prefix < min(slen, tlen) && S[prefix] == T[prefix]) prefix++;
    //cout<<prefix<<endl;
    GetNext(S, nxt);
    int len = slen - nxt[slen];
    int num = (slen%len) ? 1 : slen/len;
```

```

        //cout<<len<<endl;
        if(num > 1) for(int i = 1;i*i<=num;i++) if(num % i == 0) vis[i*len] =
vis[num/i*len] = 1;
        vis[slen] = 1;
        GetNext(T, nxt);
        len = tlen - nxt[tlen];
        num = (tlen%len) ? 1 : tlen/len;
        //cout<<len<<endl;
        if(num > 1) for(int i = 1;i*i<=num;i++) if(num % i == 0) ans+=(i*len <=
prefix?vis[i*len]:0), ans+=((i*i==num || num/i*len >
prefix)?0:vis[num/i*len]);
        if(num == 1) ans += (tlen <= prefix ? vis[tlen] : 0);
        cout<<ans<<endl;
        return 0;
    }

```

来自 <http://codeforces.com/problemset/status?friends=on>

## Codeforces 149E Martian Strings(KMP)

2015 年 7 月 20 日  
3:07

### 题意:

这个题的题意简直醉。。看得我蛋疼。。

他说有种怪物有  $n$  只眼睛，排成一行...睡觉的时候要给每个眼睛带上眼罩，同时每个眼罩里面有一个大写字母，这样的话每天怪物醒来都能看到一个字符串（hhhh...我在吃泡面...），但是一次性打开所有的眼睛太累了，所以只能打开两个不重叠的区间的眼睛  $[a,b][c,d]$  ( $a \leq b < c \leq d$ )

然而这里有  $m$  个美丽的字符串，问这只怪物可能看到的字符串的个数。（Orz 出题人的脑洞 = =）

### 做法:

这题和 zoj 月赛的有个题非常像，不过这里有 100 个串，且不重复。那个题只有一个串且可重复，但是那个题需要计数，所以其实那个题更难一些（其实扩展 KMP 的话难度一样，但是 KMP 的做法，ZOJ 难度比这个大些）。所以这里同样用扩展 KMP 做。

挨个挨个串的判定，对每一个美丽串，首先正序求出  $S$  所有后缀的与  $T$  的最长公共前缀，然后记录每个前缀在  $S$  中出现的最小坐标。再翻转两个字符串，利用 EX-KMP 跑一下，只要存在一个能够容纳这个字符串  $T$  的位置，那么这个串就是可能出现的。

### 代码:

```

#include <iostream>
#include <cstdio>

```

```

#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
#define id(x, y) ((x)*m+(y))
#define check(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
void GetNext(const char* T, int* nxt)
{
    int len = strlen(T), a = 0;
    nxt[0] = len;
    while(a<len-1 && T[a] == T[a+1]) a++;
    nxt[1] = a;
    a = 1;
    for(int k=2;k<len;k++){
        int p = a + nxt[a] - 1, L = nxt[k - a];
        if(k - 1 + L >= p){
            int j = max(0, p-k+1);
            while(k+j<len && T[k+j] == T[j]) j++;
            nxt[k] = j; a = k;
        }else nxt[k] = L;
    }
}
void GetExtend(const char* S, const char* T, int* nxt, int* extend)
{
    GetNext(T, nxt);
    int slen = strlen(S), tlen = strlen(T), a = 0;
    int Minlen = min(slen, tlen);
    while(a < Minlen && S[a] == T[a]) a++;
    extend[0] = a;
    a = 0;
    for(int k=1;k<slen;k++){
        int p = a + extend[a] - 1, L = nxt[k-a];
        if(k-1+L >= p){
            int j = max(0, p-k+1);
            while(k+j < slen && j < tlen && S[k+j] == T[j]) j++;
            extend[k] = j; a = k;
        }else extend[k] = L;
    }
}
int nxt[N], ex1[N], idx[N];

```



```

char S[N] , T[N];
bool vis[N];
int main()
{
    //Open();
    scanf("%s", S);
    int m;scanf("%d", &m);
    int slen = strlen(S);
    int ans = 0;
    while(m--)
    {
        scanf("%s", T);
        int tlen = strlen(T);
        if(tlen == 1) continue;
        memset(idx, 0x3f, sizeof(idx));
        GetExtend(S, T, nxt, exl);
        for(int i=0;i<slen;i++) idx[exl[i]] = min(idx[exl[i]], i);
        for(int i = tlen - 1; i>=0;i--) idx[i] = min(idx[i], idx[i+1]);
        reverse(S, S+slen);
        reverse(T, T+tlen);
        GetExtend(S, T, nxt, exl);
        for(int i=0;i<slen;i++)
            if(idx[tlen - exl[i]] <= (slen - i - tlen)) {ans++;break;}
        reverse(S, S+slen);
    }
    cout<<ans<<endl;
    return 0;
}

```

来自 <<http://codeforces.com/problemset/status?friends=on>>

## HDU 4333 Revolving Digits(KMP)

2015 年 7 月 20 日  
3:18

### 题意:

给出一个数字（十进制表示法长度在  $1e5$ ），可以做这样的操作：将这个数字的最后一位移到最开始，形成一个新的数字。然后需要统计的是所有的操作组成的数字有多少小于原数，多少大于，多少等于。

### 做法:

扩展 KMP。首先用原串接在原串后面（重复一遍），于是所有能够出现的数都为新串中长度为  $len$  的子串，于是跑一边 EX-KMP，得到每个后缀与原串的最长公共前缀，这样的话，只需要比较一位即可判断这个数字与原串的大小关系。需要注意的

是，这里可能出现原串有循环节的情况，需要将最后算出来的个数除以最大循环节个数即可。

代码：

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
// #include <unordered_map>
#define N 200010
#define id(x, y) ((x)*m+(y))
#define check(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
int nxt[N], extend[N];
void getKMPNext(char *T)
{
    nxt[0] = -1;
    int tlen = strlen(T), j=0, k=-1;
    while(j<tlen)
    {
        if(k == -1 || T[j] == T[k])
            nxt[++j] = ++k;
        else k = nxt[k];
    }
}
void GetNext(const char *T) {
    int len=strlen(T), a=0;
    nxt[0]=len;
    while(a<len-1 && T[a]==T[a+1]) a++;
    nxt[1]=a;
    a=1;
    for(int k=2; k<len; k++) {
        int p=a+nxt[a]-1, L=nxt[k-a];
        if((k-1)+L >= p) {
            int j = (p-k+1)>0 ? (p-k+1) : 0;
            while(k+j<len && T[k+j]==T[j]) j++;
            nxt[k]=j;
            a=k;
        }
        else
            nxt[k]=L;
    }
}
void GetExtend(const char *S, const char *T) {
    GetNext(T);
    int slen=strlen(S), tlen=strlen(T), a=0;
    int MinLen = slen < tlen ? slen : tlen;
    while(a<MinLen && S[a]==T[a]) a++;
    extend[0]=a;
    a=0;
    for(int k=1; k<slen; k++) {
        int p=a+extend[a]-1, L=nxt[k-a];
        if((k-1)+L >= p) {
            int j = (p-k+1) > 0 ? (p-k+1) : 0;
```

```

        while(k+j<slen && j<tlen && S[k+j]==T[j]) j++;
        extand[k]=j;
        a=k;
    }
    else
        extand[k]=L;
}
}
char S[N],T[N];
int main()
{
    Open();
    int T_T;scanf("%d", &T_T);int cas=1;
    while(T_T--)
    {
        scanf("%s", S);
        strcpy(T, S);
        strcat(S, T);
        //cout<<S<<endl;
        GetExtand(S, T);
        int slen = strlen(S);
        int tlen = strlen(T);
        int L = 0, E = 0, G = 0;
        for(int i = 0; i < tlen; i++)
        {
            if(extand[i] == tlen)
                E++;
            else {
                S[i + extand[i]] < T[extand[i]] ? L++ : G++;
            }
        }
        getKMPNext(T);
        int kk = (tlen % (tlen - nxt[tlen])) ? 1 : tlen/(tlen -
nxt[tlen]);
        printf("Case %d: %d %d %d\n", cas++, L/kk, E/kk, G/kk);
    }
    return 0;
}

```

来自 <<http://acm.hdu.edu.cn/viewcode.php?rid=14059308>>

## ZOJ 3587 Marlon's String

2015 年 7 月 20 日  
3:24

### 题意:

给出两个字符串 S,T, 问有多少个四元组满足  $S(a, b)+S(c, d) = T$ 。

### 做法:

做法有两种, 一种是 KMP, 一种是 ex-KMP。我想说: 这两种方法的难度根本不在一个等级上好嘛!! (其实都是自己太弱了。。kmp 理解不深。。导致比赛的时候 hh), 如果用 ex-kmp 做的话, 这题异常简单。。和 CF149E 差不多的方法, 统计 T

的每个前缀出现在 S 中的次数，然后翻转，在统计每个前缀在 S 反中的次数...然后就乘乘加就可以了= =。

但是如果用 KMP 的话，统计 T 中的前缀出现在 S 中的次数有些麻烦，因为同样的，这里的 next 数组记录的只是满足条件的最大值，可能有一些小的串也满足，却被省掉了，但不是所有比这个小的串都满足，所以加的时候还需要考虑这个。详见代码。

代码：

```
ex-kmp:
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
#define id(x, y) ((x)*m+(y))
#define check(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<long long, long long> PII;
const long long INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
void GetNext(const char* T, long long* nxt)
{
    long long len = strlen(T), a = 0;
    nxt[0] = len;
    while(a<len-1 && T[a] == T[a+1]) a++;
    nxt[1] = a;
    a = 1;
    for(long long k=2; k<len; k++){
        long long p = a + nxt[a] - 1, L = nxt[k - a];
        if(k - 1 + L >= p){
            long long j = max(0LL, p-k+1);
            while(k+j<len && T[k+j] == T[j]) j++;
            nxt[k] = j; a = k;
        }else nxt[k] = L;
    }
}
void GetExtand(const char* S, const char* T, long long* nxt,
long long* extand)
{
    GetNext(T, nxt);
    long long slen = strlen(S), tlen = strlen(T), a = 0;
    long long Minlen = min(slen, tlen);
    while(a < Minlen && S[a] == T[a]) a++;
    extand[0] = a;
    a = 0;
    for(long long k=1; k<slen; k++){
```

```

        long long p = a + extand[a] - 1, L = nxt[k-a];
        if(k-1+L >= p) {
            long long j = max(0LL, p-k+1);
            while(k+j < slen && j < tlen && S[k+j] == T[j])
j++;
            extand[k] = j; a = k;
        }else extand[k] = L;
    }
}
long long nxt[N];
char S[N], T[N];
long long num[N], pre[N], revpre[N];
int main()
{
    //Open();
    long long T_T; scanf("%lld", &T_T);
    while(T_T--)
    {
        memset(pre, 0, sizeof(pre));
        memset(revpre, 0, sizeof(revpre));
        scanf("%s%s", S, T);
        long long slen = strlen(S), tlen = strlen(T);
        GetExtand(S, T, nxt, num);
        for(long long i=0; i<slen; i++) pre[num[i]]++;
        for(long long i=tlen-1; i>0; i--) pre[i] += pre[i+1];
        reverse(S, S + slen);
        reverse(T, T + tlen);
        GetExtand(S, T, nxt, num);
        for(long long i=0; i<slen; i++) revpre[num[i]]++;
        for(long long i=tlen-1; i>0; i--) revpre[i] +=
revpre[i+1];
        long long ans = 0;
        for(long long i=1; i<tlen; i++) ans +=
pre[i]*revpre[tlen-i];
        cout<<ans<<endl;
    }
    return 0;
}

```

来自 <<http://acm.hust.edu.cn/vjudge/contest/viewSource.action?id=4061946>>

KMP:

```

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
using namespace std;
typedef pair<long long, long long> PII;
const long long INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE

```

```

        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
long long nxt[N];
char s[N];
char t[N];
long long vis1[N];
long long vis2[N];
void GetNextval(char* p, long long nxt[])
{
    long long pLen = strlen(p);
    nxt[0] = -1;
    long long k = -1;
    long long j = 0;
    while (j < pLen )
    {
        if (k == -1 || p[j] == p[k])
        {
            ++j;
            ++k;
            nxt[j] = k;
        }
        else
        {
            k = nxt[k];
        }
    }
}
void KmpSearch(char* s, char* p, long long vis[])
{
    long long i = 0;
    long long j = 0;
    long long sLen = strlen(s);
    long long pLen = strlen(p);
    while (i < sLen )
    {
        if (j == -1 || s[i] == p[j])
        {
            i++;
            j++;
            if(j != -1) vis[j]++;
        }
        else
        {
            {j = nxt[j];} //if(j-1 >= 0) vis[j-1]++;}
            //    j = nxt[j];
        }
    }
    if(j==pLen)
    {
        {j = nxt[j];} //if(j-1 >= 0) vis[j-1]++;}
        //    j = nxt[j];
    }
}
int main()
{
    Open();
}

```

```

int T;scanf("%d",&T);
while(T--)
{
    memset(vis1, 0, sizeof vis1);
    memset(vis2, 0, sizeof vis2);
    scanf("%s", s);
    scanf("%s", t);
    long long slen = strlen(s);
    long long tlen = strlen(t);
    GetNextval(t, nxt);
    KmpSearch(s, t, vis1);
    for(int i = tlen; i>=0;i--)
        vis1[nxt[i]] += vis1[i];
reverse(s, s+slen);
reverse(t, t+tlen);
GetNextval(t, nxt);
KmpSearch(s, t, vis2);
for(int i = tlen; i>=0;i--)
    vis2[nxt[i]] += vis2[i];
long long ans = 0;
for(long long i=1;i<tlen;i++)
{
    ans += vis1[i] * vis2[tlen-i];
}
cout<<ans<<endl;
}
return 0;
}

```

来自 <<http://acm.hust.edu.cn/vjudge/contest/viewSource.action?id=4009527>>

## Uva 10054: The Necklace 欧拉回路

2015 年 3 月 24 日  
19:10

### Problem D: The Necklace

My little sister had a beautiful necklace made of colorful beads. Two successive beads in the necklace shared a common color at their meeting point. The figure below shows a segment of the necklace:

But, alas! One day, the necklace was torn and the beads were all scattered over the floor. My sister did her best to recollect all the beads from the floor, but she is not sure

whether she was able to collect all of them. Now, she has come to me for help. She wants to know whether it is possible to make a necklace using all the beads she has in the same way her original necklace was made and if so in which order the beads must be put.

Please help me write a program to solve the problem.

## Input

The input contains  $T$  test cases. The first line of the input contains the integer  $T$ .

The first line of each test case contains an integer  $N$  (

) giving the number of beads my sister was able to collect. Each of the next  $N$  lines contains two integers describing the colors of a bead. Colors are represented by integers ranging from 1 to 50.

## Output

For each test case in the input first output the test case number as shown in the sample output. Then if you apprehend that some beads may be lost just print the sentence ``some beads may be lost" on a line by itself. Otherwise, print  $N$  lines with a single bead description on each line. Each bead description consists of two integers giving the colors of its two ends. For

, the second integer on line  $i$  must be the same as the first integer on line  $i + 1$ .

Additionally, the second integer on line  $N$  must be equal to the first integer on line 1.

Since there are many solutions, any one of them is acceptable.

Print a blank line between two successive test cases.

## Sample Input

```
2
5
1 2
2 3
3 4
4 5
5 6
5
2 1
2 2
3 4
3 1
2 4
```

## Sample Output



Case #1  
some beads may be lost

Case #2  
2 1  
1 3  
3 4  
4 2  
2 2

来自 <<http://blog.csdn.net/frankiller/article/details/7777989>>

## 题目大意：

有一条项链断了，变成很多颗珠子，一个珠子分为两半有两种颜色，用 1 到 50 来表示 50 种不同的颜色。把这些珠子串起来，两个紧挨着的珠子要满足一个条件就是接触的那部分颜色要相同

例如(1,2)(2,4)，两个珠子的接触部分颜色相同都为 2。当然，因为珠子最后是连成环的，第一个珠子和最后一个珠子也会接触，也要满足这个条件,比如(1,2)(2,4)(4,1)

现给出 n 个珠子，问你能不能连成一条链，能的话输出任意一种连接情况即可，不能的话输出失败

## 解法：

每个珠子有左右两种颜色，可以将颜色看成顶点，而每个珠子看成一条边。抽象成这样的话，我们需要处理的问题就是，在一个无向图中，找出一条欧拉回路。而求欧拉回路，首先得判断该无向图是不是欧拉图，一个无向图是欧拉图必须满足如下两个条件：

- 图是联通图
- 图中每个点的度数必须全是偶数。

由于该无向图存在重边以及自环，所以此处利用邻接矩阵来做。

## 代码：

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
#define N 55
using namespace std;
int du[N];
int par[N];
int eNum[N][N];
int Find(int x)
{
    return par[x]==x?x:(par[x]=Find(par[x]));
}
```

```

}
bool connectable(int u)
{
    for(int i=1;i<N;i++)
        if(du[i] && (Find(i)!=Find(u) || (du[i] & 1)))
            return false;
    return true;
}
int dfs(int u)
{
    for(int i=1;i<N;i++)
    {
        if(eNum[u][i])
        {
            eNum[i][u]--;
            eNum[u][i]--;
            dfs(i);
            printf("%d %d\n",i,u);\\此处必须是逆序输出。
        }
    }
}
int main()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
    #endif // ONLINE_JUDGE
    int t,cas=1;
    scanf("%d",&t);
    while(t--)
    {
        printf("Case #%d\n",cas++);
        memset(du,0,sizeof(du));
        memset(eNum,0,sizeof(eNum));
        for(int i=1;i<N;i++)
            par[i]=i;
        int n;
        int V=0;
        scanf("%d",&n);
        int pre;
        for(int i=0;i<n;i++)
        {
            int u,v;
            scanf("%d%d",&u,&v);
            pre=u;
            du[u]++,du[v]++;
            eNum[u][v]++;
            eNum[v][u]++;
            if(Find(u)!=Find(v))
                par[Find(u)]=Find(v);
        }
        if(!connectable(pre)){
            printf("some beads may be lost\n");
            if(t) printf("\n");
            continue;
        }
        dfs(pre);
        if(t) printf("\n");
    }
    return 0;
}

```

# \_\_builtin\_函数 unordered\_map 自定义结构体的 写法

2015 年 4 月 7 日  
11:43

```
struct Node
{
    string str;
    int idx;
    bool operator==(const Node& o) const
    {
        return str==o.str && idx == o.idx;
    }
};
struct Node_hash
{
    size_t operator()(const Node& o) const
    {
        return hash<string>()(o.str) ^
(hash<int>()(o.idx) >> 1);
    }
};
unordered_map<Node,int,Node_hash> vis;
```

我们需要重载==符号，另外在另一个结构体里面写 hash 函数，格式如上：

- int \_\_builtin\_ffs (unsigned int x)  
返回 x 的最后一位 1 的是从后向前第几位，比如 7368 (1110011001000) 返回 4。
- int \_\_builtin\_clz (unsigned int x)  
返回前导的 0 的个数。
- int \_\_builtin\_ctz (unsigned int x)  
返回后面的 0 的个数，和 \_\_builtin\_clz 相对。
- int \_\_builtin\_popcount (unsigned int x)  
返回二进制表示中 1 的个数。
- int \_\_builtin\_parity (unsigned int x)  
返回 x 的奇偶校验位，也就是 x 的 1 的个数模 2 的结果。

# 计算几何

2015 年 7 月 2 日  
2:46

Atan2 取值范围为  $(-\pi, \pi]$

ATAN2(a, b) 与 ATAN(a/b) 稍有不同，ATAN2(a,b) 的取值范围介于  $-\pi$  到  $\pi$  之间（不包括  $-\pi$ ），

而 ATAN(a/b) 的取值范围介于  $-\pi/2$  到  $\pi/2$  之间（不包括  $\pm\pi/2$ ）。

acos

返回的是一个数值的反余弦弧度值，其范围是  $0 \sim \pi$ 。例如：acos(1) 返回值是 0。

acos(-0.5) 返回的是 2.0944 弧度。

需要注意的是  $\text{acos}(x)$  必须属于  $[-1, 1]$  这个区间，否则会出现计算失误，而计算机中的精度常常会有误差，一定要判断这个

二分一定要用次数限制 100 次左右

来自 <[http://baike.baidu.com/link?url=JYtc3hjmmi567OdAvLSLJ-aR5yKi0QGB\\_tvHHN84NlrCD4YLV\\_9-fzmSu9uQp51aVfQZmLJOarxLg17Q9Cc2jq](http://baike.baidu.com/link?url=JYtc3hjmmi567OdAvLSLJ-aR5yKi0QGB_tvHHN84NlrCD4YLV_9-fzmSu9uQp51aVfQZmLJOarxLg17Q9Cc2jq)>

# 狗坑！

2015 年 7 月 16 日  
15:21

1. 用 %s 利用字符串读取数据时，特别小心读取数字时，只有在一位数的时候才能用 `curchar-'0'` 这种方式得到当前数字
2. 开数组的时候，数组大小尽量比数据范围大，并且超过数据的大小应大于 1，比如  $1 \leq n \leq 10$ ，那么最好开 12 以上

```

    }
    cout<<dfs(s, d, 1)<<" "<<step<<endl;
return 0;

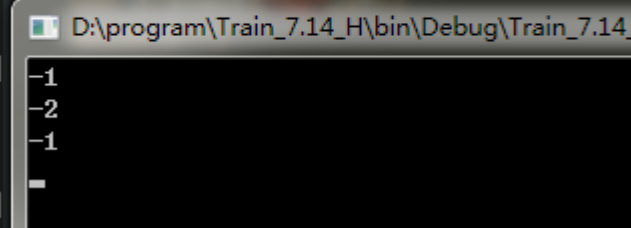
```

3. 如上这种输出，如果 `step` 在 `dfs` 中改变的话，输出的是改变之前的值，需要分开输出。

```

//Open();
cout<<ceil(-1.12123)<<endl;
cout<<floor(-1.123123)<<endl;
cout<<(int)-1.1111<<endl;

```



4. 注意负数的向下取整，向上取整的时候需要注意，取中点尽量使用  $(l+r)>>1$ ，这样就算负数也是向下取整，比如  $(-1 + (-2))/2 = -1$ ,  $(-1 + (-2))>>1 = -2$

5. 

```
a /= __gcd(a, b), b /= __gcd(a, b);
```

上面这种肯定错了啊！！

6. 数学的时候，需要特别小心中间的结果可能会爆 `long long` !!!  
比如 `x%m`，如果为了防止负数换成  $(x\%m+m)\%m$  的话，可能在加 `m` 的那一步就爆了 `long long`.....，不能大意啊！！真是 hhh。  
ZOJ, 3562...果然模板也是爆了 `long long` 所以防止负数模最稳妥的方法果然是：  
`if(x<0) x+=m!!!!!!`
7. `sort` 使用的比较函数中(比如重载小于符号)里面不能用 `<=` 或者 `>=` 符号，不然会 RE (应该是在有值相等的时候 RE)。
8. 使用 `vector` 的时候，在 `for(int i = 0; i < vector.size()-1; i++)` 时，`vector` 的 `size` 为空的时候会出现大问题，因为 `vector.size` 返回的是一个无符号数，会出现死循环。
9. 交题时的 `system("pause")` 必须删掉，注释掉也是不行的，会被 OJ 硬匹配当成危险代码出现 "judge error" 之类的结果
10. 一定要注意好各个数组的大小，特别下标运算的时候。
11. 求最大值的时候，一定要注意初始值的选取，当结果可能为负数的时候，一定要特别小心。

12. 输出%的时候一定要用 `putchar`，如果用 `printf("%")`的话，有的编译器并不会输出%，用 `printf("%%")`好像可以输出。。但是保险还是用 `putchar` 吧
13. 在计算一条路径(u, v)的信息的时候(如求和)，常常会需要使用如下公式  
 $val[u]$ 表示 u 节点到根节点的和，那么  $val[u, v] = val[u] + val[v] - val[lca]*2$ ，此时  $lca$  位置的值有没有加上，需不需要加需要特别小心！！
14. `long long` 做 `mod` 运算特别慢，在特别多乘法 `mod` 运算的时候，避免使用全局 `long long` 替换。

## bitset 各个函数

2015 年 11 月 5 日  
15:11

<code>b.any()</code>	b 中是否存在置为 1 的二进制位？
----------------------	--------------------

<code>b.none()</code>	b 中不存在置为 1 的二进制位吗？
-----------------------	--------------------

<code>b.count()</code>	b 中置为 1 的二进制位的个数
------------------------	------------------

<code>b.size()</code>	b 中二进制位的个数
-----------------------	------------

<code>b[pos]</code>	访问 b 中在 pos 处的二进制位
---------------------	--------------------

<code>b.test(pos)</code>	b 中在 pos 处的二进制位是否为 1？
--------------------------	-----------------------

<code>b.set()</code>	把 b 中所有二进制位都置为 1
----------------------	------------------

<code>b.set(pos)</code>	把 b 中在 pos 处的二进制位置为 1
-------------------------	-----------------------

b.reset()	把 b 中所有二进制位都置为 0
b.reset(pos)	把 b 中在 pos 处的二进制位置为 0
b.flip()	把 b 中所有二进制位逐位取反
b.flip(pos)	把 b 中在 pos 处的二进制位取反
b.to_ulong()	用 b 中同样的二进制位返回一个 unsigned long 值
os << b	把 b 中的位集输出到 os 流

## Stirling 数

2015 年 8 月 11 日  
2:42

第一类 Stirling 数  $s(p, k)$

$s(p, k)$  的一个组合学解释是：将  $p$  个物体排成  $k$  个非空循环排列的方法数。

$s(p, k)$  的递推公式： $s(p, k) = (p-1) * s(p-1, k) + s(p-1, k-1)$  ,  $1 \leq k \leq p-1$   
边界条件： $s(p, 0) = 0$  ,  $p \geq 1$     $s(p, p) = 1$  ,  $p \geq 0$

递推关系的说明：

考虑第  $p$  个物品， $p$  可以单独构成一个非空循环排列，这样前  $p-1$  种物品构成  $k-1$  个非空循环排列，方法数为  $s(p-1, k-1)$ ；

也可以前  $p-1$  种物品构成  $k$  个非空循环排列，而第  $p$  个物品插入第  $i$  个物品的左边，这有  $(p-1) * s(p-1, k)$  种方法。

## 第二类 Stirling 数 $S(p, k)$

$S(p, k)$  的一个组合学解释是：将  $p$  个物体划分成  $k$  个非空的不可辨别的（可以理解为盒子没有编号）集合的方法数。

$k!S(p, k)$  是把  $p$  个人分进  $k$  间有差别（如：被标有房号）的房间（无空房）的方法数。

$S(p, k)$  的递推公式是： $S(p, k) = k \cdot S(p-1, k) + S(p-1, k-1)$  ,  $1 \leq k \leq p-1$

边界条件： $S(p, p) = 1$  ,  $p \geq 0$        $S(p, 0) = 0$  ,  $p \geq 1$

递推关系的说明：

考虑第  $p$  个物品， $p$  可以单独构成一个非空集合，此时前  $p-1$  个物品构成  $k-1$  个非空的不可辨别的集合，方法数为  $S(p-1, k-1)$ ；

也可以前  $p-1$  种物品构成  $k$  个非空的不可辨别的集合，第  $p$  个物品放入任意一个中，这样有  $k \cdot S(p-1, k)$  种方法。

第一类斯特林数和第二类斯特林数有相同的初始条件，但递推关系不同。

### [题目：HDU3625](#)

题意：给  $N$  个元素，让我们求  $K$  个环排列的方法数。

斯特林第一类数的递推公式：

$S(N, 0) = 0$ ;  $S(N, N) = 1$ ;  $S(0, 0) = 0$ ;  $S(N, K) = S(N-1, K-1) + S(N-1, K) \cdot (N-1)$ ;

这个公式的意思是：当前  $N-1$  个数构成  $K-1$  个环的时候，加入第  $N$  个， $N$  只能构成单环！— $S(N-1, K-1)$  如果  $N-1$  个数构

成  $K$  个环的时候，加入第  $N$  个， $N$  可以任意加入， $N-1$  内的一个环里，所以  $(N-1) \cdot S(N-1, K)$  这个题目里，因为不能破坏

第 1 个门：所以  $S(N, K) - S(N-1, K-1)$  才是能算构成  $K$  个环的方法数！就是去掉 1 自己成环的情况。

```
1. #include<stdio.h>
2. #include<string.h>
3. #define N 21
4.
5. __int64 fac[N]={1,1};
6. __int64 stir[N][N];
7.
8. void init()
9. {
10.     int i, j;
11.     for(i=2;i<N;i++)
12.         fac[i]=i*fac[i-1];
```



```

13.     memset(stir, 0, sizeof(stir));
14.     stir[0][0]=0;
15.     stir[1][1]=1;
16.     for(i=2; i<N; i++)
17.         for(j=1; j<=i; j++)
18.             stir[i][j]=stir[i-1][j-1]+(i-1)*stir[i-1][j];
19. }
20. int main()
21. {
22.     init();
23.     int t;
24.     scanf("%d", &t);
25.     while(t--)
26.     {
27.         int n, k, i;
28.         scanf("%d %d", &n, &k);
29.         __int64 cnt=0;
30.         for(i=1; i<=k; i++)
31.             cnt+= stir[n][i] - stir[n-1][i-1]; //注意：去
掉 1 自己成环的
32.         printf("%.4lf\n", 1.0*cnt/fac[n]);
33.     }
34.     return 0;
35. }

```

来自 <<http://blog.csdn.net/acdreamers/article/details/8521134>>

## Bell 数

2015 年 8 月 11 日  
2:43

**Bell 数的定义：**第  $n$  个 **Bell 数** 表示集合  $\{1, 2, 3, \dots, n\}$  的划分方案数，即： $B[0] = 1$ ;

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k$$

每一个 **Bell 数** 都是第二类 **Stirling 数** 的和，即：

$$B_n = \sum_{k=1}^n S(n, k)$$

第二类 Stirling 数的意义是：\$S(n, k)\$ 表示将 \$n\$ 个物体划分成 \$k\$ 个非空的不可辨别的（可以理解为盒子没有编号）集合的方法数。很明显，每一个 Bell 是对应的第二类 Stirling 数之和。

Bell 数的指数生成函数是：

$$\sum_{n=0}^{\infty} \frac{B_n}{n!} x^n = e^{e^x - 1}$$

关于它的推导过程详见：<http://ftiasch.github.io/useless/posts/2013-09-27-generating-function-of-bell-number.html>

Bell 三角形 (构建方法类似于杨辉三角形)

Bell 三角形的构造方法：

第一行第一个元素是 1，即 \$a[1][1] = 1\$

对于 \$n > 1\$，第 \$n\$ 行第一项等于第 \$n-1\$ 行最后一项，即 \$a[n][1] = a[n-1][n-1]\$；

对于 \$m, n > 1\$，第 \$n\$ 行第 \$m\$ 项等于它左边和左上方的两个数之和，即 \$a[n][m] = a[n][m-1] + a[n-1][m-1]\$；

如图：

1								
1	2							
2	3	5						
5	7	10	15					
15	20	27	37	52				
52	67	87	114	151	203			
203	255	322	409	523	674	877		
877	1080	1335	1657	2066	2589	3263	4140	
			⋮		⋮		⋮	

可以看出，每行首项是贝尔数，每行之和是第二类 Stirling 数。

Bell 还有两个重要的同余性质：

$$B_{p+n} \equiv B_n + B_{n+1} \pmod{p}$$

$$B_{p^m+n} \equiv mB_n + B_{n+1} \pmod{p}$$

其中这里的  $p$  是不大于 100 的素数，这样，我们可以通过上面的性质来计算 Bell 数模小于 100 的素数值。

Bell 数模素数  $p$  的周期为：

$$N_p = \frac{p^p - 1}{p - 1}$$

# 卡特兰数

## 1 简介 [编辑](#)

卡特兰数又称卡塔兰数，英文名 Catalan number，是[组合数学](#)中一个常出现在各种计数问题中出现的[数列](#)。由以[比利时](#)的数学家欧仁·查理·卡塔兰 (1814–1894)命名，其前几项为：1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, ...

### 3 原理 [编辑](#)

令  $h(0)=1, h(1)=1$  , catalan 数满足递推式[1] :

$$h(n) = h(0)*h(n-1) + h(1)*h(n-2) + \dots + h(n-1)h(0) \quad (n \geq 2)$$

例如 :  $h(2)=h(0)*h(1)+h(1)*h(0)=1*1+1*1=2$

$$h(3)=h(0)*h(2)+h(1)*h(1)+h(2)*h(0)=1*2+1*1+2*1=5$$

另类递推式[2] :

$$h(n) = h(n-1) * (4^n - 2) / (n+1);$$

递推关系的解为 :

$$h(n) = C(2n, n) / (n+1) \quad (n=0, 1, 2, \dots)$$

递推关系的另类解为 :

$$h(n) = c(2n, n) - c(2n, n-1) \quad (n=0, 1, 2, \dots)$$

### 4 应用 [编辑](#)

实质上都是递推等式的应用

#### 括号化

矩阵连乘 :  $P=a_1 \times a_2 \times a_3 \times \dots \times a_n$  , 依据乘法结合律 , 不改变其顺序 , 只用括号表示成对的乘积 , 试问有几种括号化的方案 ? ( $h(n-1)$ 种)[3]

#### 出栈次序

一个栈(无穷大)的进栈序列为 1, 2, 3, ..., n, 有多少个不同的出栈序列?[4-5]

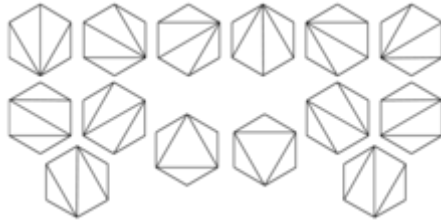
#### 类似问题 买票找零

有  $2n$  个人排成一行进入剧场。入场费 5 元。其中只有  $n$  个人有一张 5 元钞票 , 另外  $n$  人只有 10 元钞票 , 剧院无其它钞票 , 问有多少中方法使得只要有 10 元的人买票 , 售票处就有 5 元的钞票找零 ? (将持 5 元者到达视作将 5 元入栈 , 持 10 元者到达视作使栈中某 5 元出栈)

## 凸多边形三角划分

在一个[凸多边形](#)中，通过若干条互不相交的对角线，把这个多边形划分成了若干个三角形。任务是键盘上输入凸多边形的边数  $n$ ，求不同划分的方案数  $f(n)$ 。

比如当  $n=6$  时， $f(6)=14$ 。[\[6\]](#)



### 分析

如果纯粹从  $f(4)=2, f(5)=5, f(6)=14, \dots, f(n)=n$  慢慢去归纳，恐怕很难找到问题的递推式，我们必须从一般情况出发去找规律。

因为凸多边形的任意一条边必定属于某一个三角形，所以我们以某一条边为基准，以这条边的两个顶点为起点  $P_1$  和终点  $P_n$  ( $P$  即 Point)，将该凸多边形的顶点依序标记为  $P_1, P_2, \dots, P_n$ ，再在该凸多边形中找任意一个不属于这两个点的顶点  $P_k$  ( $2 \leq k \leq n-1$ )，来构成一个三角形，用这个三角形把一个凸多边形划分成两个凸多边形，其中一个凸多边形，是由  $P_1, P_2, \dots, P_k$  构成的凸  $k$  边形 (顶点数即是边数)，另一个凸多边形，是由  $P_k, P_{k+1}, \dots, P_n$  构成的凸  $n-k+1$  边形。

此时，我们若把  $P_k$  视为确定一点，那么根据[乘法原理](#)， $f(n)$  的问题就等价于——凸  $k$  多边形的划分方案数乘以凸  $n-k+1$  多边形的划分方案数，即选择  $P_k$  这个顶点的  $f(n) = f(k) \times f(n-k+1)$ 。而  $k$  可以选 2 到  $n-1$ ，所以再根据加法原理，将  $k$  取不同值的划分方案相加，得到的总方案数为： $f(n) = f(2)f(n-2+1) + f(3)f(n-3+1) + \dots + f(n-1)f(2)$ 。看到此处，再看看卡特兰数的递推式，答案不言而喻，即为  $f(n) = h(n-2)$  ( $n=2, 3, 4, \dots$ )。

最后，令  $f(2)=1, f(3)=1$ 。

此处  $f(2)=1$  和  $f(3)=1$  的具体缘由须参考详尽的“卡特兰数”，也许可从[凸四边形](#)  $f(4) = f(2)f(3) + f(3)f(2) = 2 \times f(2)f(3)$  倒推，四边形的划分方案不用规律推导都可以知道是 2，那么  $2 \times f(2)f(3) = 2$ ，则  $f(2)f(3) = 1$ ，又  $f(2)$  和  $f(3)$  若存在的话一定是整数，则  $f(2)=1, f(3)=1$ 。(因为我没研究过卡特兰数的由来，此处仅作刘转羽的臆测)。

### 类似问题

一位大城市的律师在她住所以北  $n$  个街区和以东  $n$  个街区处工作。每天她走  $2n$  个街区去上班。如果她从不穿越（但可以碰到）从家到办公室的对角线，那么有多少条可能的道路？

在圆上选择  $2n$  个点，将这些点成对连接起来使得所得到的  $n$  条线段不相交的方法数？

## 给定节点组成二叉树

给定  $N$  个节点，能构成多少种不同的二叉树？ [7]

（能构成  $h(N)$  个）

（这个公式的下标是从  $h(0)=1$  开始的）

## 5 扩展 [编辑](#)

对于在  $n$  位的 2 进制中，有  $m$  个 0，其余为 1 的 catalan 数为： $C(n, m) - C(n, m-1)$ 。证明可以参考标准 [catalan](#) 数的证明。 [8]

问题 1 的描述：有  $n$  个 1 和  $m$  个 -1 ( $n > m$ )，共  $n+m$  个数排成一列，满足对所有  $0 \leq k \leq n+m$  的前  $k$  个数的部分和  $S_k > 0$  的排列数。问题等价于在一个格点阵列中，从  $(0, 0)$  点走到  $(n, m)$  点且不经对角线  $x=y$  的方法数 ( $x > y$ )。

考虑情况 I：第一步走到  $(0, 1)$ ，这样从  $(0, 1)$  走到  $(n, m)$  无论如何也要经过  $x=y$  的点，这样的方法数为  $((n+m-1, m-1))$ ;

考虑情况 II：第一步走到  $(1, 0)$ ，又有两种可能：

a. 不经过  $x=y$  的点；（所要求的情况）

b. 经过  $x=y$  的点，我们构造情况 II.b 和情况 I 的一一映射，说明 II.b 和 I 的方法数是一样的。设第一次经过  $x=y$  的点是  $(x_1, y_1)$ ，将  $(0, 0)$  到  $(x_1, y_1)$  的路径沿对角线翻折，于是唯一对应情况 I 的一种路径；对于情况 I 的一条路径，假设其与对角线的第一个焦点是  $(x_2, y_2)$ ，将  $(0, 0)$  和  $(x_2, y_2)$  之间的路径沿对角线翻折，唯一对应情况 II.b 的一条路径。

问题的解就是总的路径数  $((n+m, m))$  - 情况 I 的路径数 - 情况 II.b 的路径数。

$((n+m, m)) - 2*((n+m-1, m-1))$

或： $((n+m-1, m)) - ((n+m-1, m-1))$

问题 2 的描述：有  $n$  个 1 和  $m$  个 -1 ( $n \geq m$ )，共  $n+m$  个数排成一列，满足对所有  $0 \leq k \leq n+m$  的前  $k$  个数的部分和  $S_k \geq 0$  的排列数。（和问题 1 不同之处在于此处部分和可以为 0，这也是更常见的情况）问题等价于在一个格点阵列中，从  $(0, 0)$  点走到  $(n, m)$  点且不穿过对角线  $x=y$  的方法数（可以走到  $x=y$  的点）。

把  $(n, m)$  点变换到  $(n+1, m)$  点，问题变成了问题 1。

方法数为：

$((n+m+1, m)) - 2*((n+m+1-1, m-1))$

或： $((n+m+1-1, m)) - ((n+m+1-1, m-1))$

来自

[http://baike.baidu.com/link?url=mqOG8L8g3HGy5z1KpCTLFhwjvRuwaXiEkCoNq\\_SEd7e25mDtON5t6uaHpDQYSXiHxUMrLHxTxmO4OQtsjLrG](http://baike.baidu.com/link?url=mqOG8L8g3HGy5z1KpCTLFhwjvRuwaXiEkCoNq_SEd7e25mDtON5t6uaHpDQYSXiHxUMrLHxTxmO4OQtsjLrG)

卡特兰数[编辑]

卡特兰数

**卡特兰数**是组合数学中一个常在各种计数问题中出现的数列。以比利时的数学家欧仁·查理·卡特兰（1814–1894）命名。历史上，清代数学家明安图(1692 年 - 1763 年)在其《割圆密率捷法》最早用到“卡特兰数”，远远早于卡特兰[1][2][3]。有中国学者建议将此数命名为“明安图数”或“明安图-卡特兰数” [4]。

卡特兰数的一般项公式为

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$

前 20 项为（OEIS 中的数列 A000108）：1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190

目录

性质[编辑]

$C_n$  的另一个表达形式为

$$C_n = \binom{2n}{n} - \binom{2n}{n+1} \quad \text{for } n \geq 1$$

所以， $C_n$  是一个自然数；这一点在先前的通项公式中并不显而易见。这个表达形式也是 André 对前一公式证明的基础。（见下文的第二个证明。）

[递推关系](#)

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^n C_i C_{n-i} \quad \text{for } n \geq 0.$$

它也满足

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \frac{2(2n+1)}{n+2} C_n,$$

这提供了一个更快速的方法来计算卡特兰数。

卡特兰数的渐近增长为

$$C_n \sim \frac{4^n}{n^{3/2} \sqrt{\pi}}$$

它的含义是当  $n \rightarrow \infty$  时，左式除以右式的商[趋向于](#) 1。（这可以用  $n!$  的[斯特灵公式](#)来证明。）

所有的奇卡特兰数  $C_n$  都满足

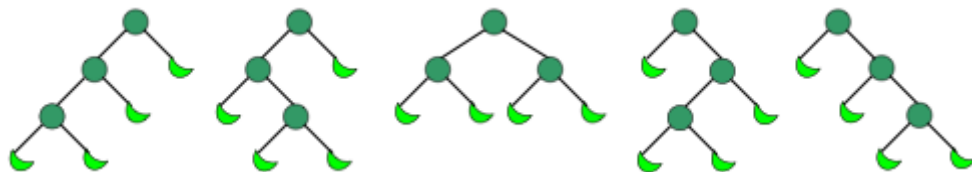
$$n = 2^k - 1$$

。所有其他的卡特兰数都是偶数。

应用[\[编辑\]](#)

[组合数学](#)中有非常多的组合结构可以用卡特兰数来计数。在 Richard P. Stanley 的 Enumerative Combinatorics: Volume 2 一书的习题中包括了 66 个相异的可由卡特兰数表达的组结构。以下用  $n=3$  和  $n=4$  举若干例：

- $C_n$  表示长度  $2n$  的 dyck word 的个数。Dyck word 是一个有  $n$  个 X 和  $n$  个 Y 组成的字符串，且所有的前缀字符串皆满足 X 的个数大于等于 Y 的个数。以下为长度为 6 的 dyck words:  
XXXYYY XYXXYY XYXYXY XXYYXY XXYXYY
- 将上例的 X 换成左括号，Y 换成右括号， $C_n$  表示所有包含  $n$  组括号的合法运算式的个数：  
((( ))) 0(0) 000 (0)0 (00)
- $C_n$  表示有  $n$  个节点组成不同构[二叉树](#)的方案数。下图中， $n$  等于 3，圆形表示节点，月牙形表示什么都没有。
- $C_n$  表示有  $2n+1$  个节点组成不同构满[二叉树](#) ( full binary tree ) 的方案数。下图中， $n$  等于 3，圆形表示内部节点，月牙形表示外部节点。本质同上。



证明：

令 1 表示进栈，0 表示出栈，则可转化为求一个  $2n$  位、含  $n$  个 1、 $n$  个 0 的二进制数，满足从左往右扫描到任意一位时，经过的 0 数不多于 1 数。显然含  $n$  个 1、 $n$  个 0 的  $2n$  位二进制数共有

$$\binom{2n}{n}$$

个，下面考虑不满足要求的数目。



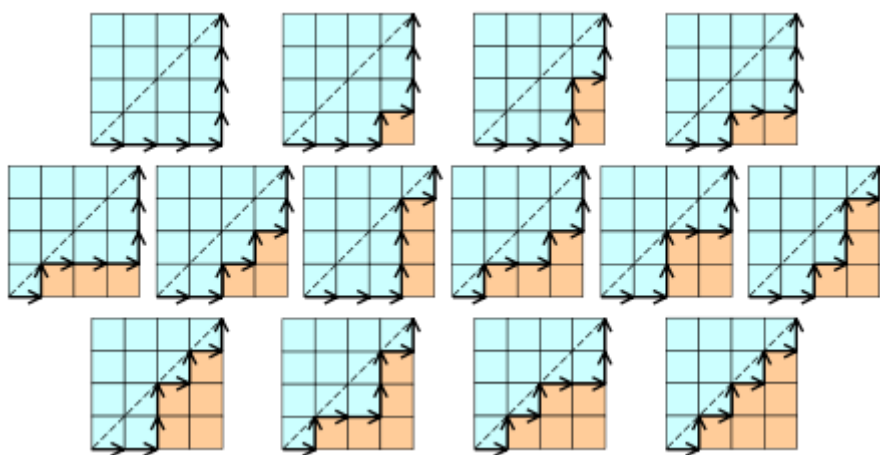
考虑一个含  $n$  个 1、 $n$  个 0 的  $2n$  位二进制数，扫描到第  $2m+1$  位上时有  $m+1$  个 0 和  $m$  个 1（容易证明一定存在这样的情况），则后面的 0-1 排列中必有  $n-m$  个 1 和  $n-m-1$  个 0。将  $2m+2$  及其以后的部分 0 变成 1、1 变成 0，则对应一个  $n+1$  个 0 和  $n-1$  个 1 的二进制数。反之亦然（相似的思路证明两者一一对应）。

从而

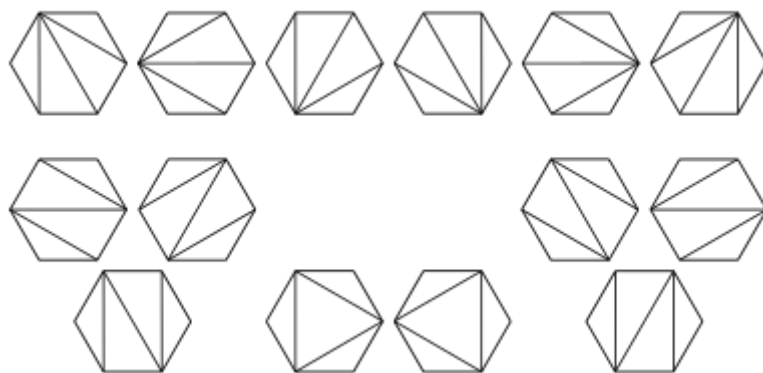
$$C_n = \binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1} \binom{2n}{n}$$

。证毕。

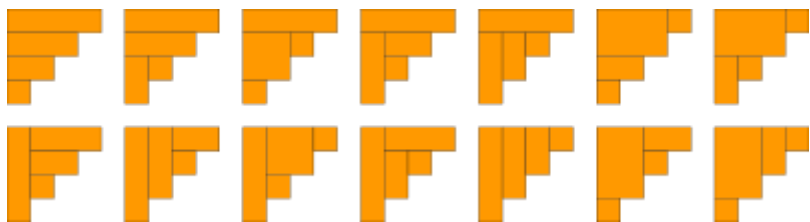
- $C_n$  表示所有在  $n \times n$  格点中不越过对角线的单调路径的个数。一个单调路径从格点左下角出发，在格点右上角结束，每一步均为向上或向右。计算这种路径的个数等价于计算 Dyck word 的个数：X 代表“向右”，Y 代表“向上”。下图为  $n = 4$  的情况：



- $C_n$  表示通过连结顶点而将  $n+2$  边的凸多边形分成三角形的方法个数。下图中为  $n = 4$  的情况：



- $C_n$  表示对  $\{1, \dots, n\}$  依序进出栈的置换个数。一个置换  $w$  是依序进出栈的当  $S(w) = (1, \dots, n)$ ，其中  $S(w)$  递归定义如下：令  $w = unv$ ，其中  $n$  为  $w$  的最大元素， $u$  和  $v$  为更短的数列；再令  $S(w) = S(u)S(v)n$ ，其中  $S$  为所有含一个元素的数列的单元元。
- $C_n$  表示集合  $\{1, \dots, n\}$  的不交叉划分的个数。那么， $C_n$  永远不大于第  $n$  项贝尔数。  $C_n$  也表示集合  $\{1, \dots, 2n\}$  的不交叉划分的个数，其中每个段落的长度为 2。综合这两个结论，可以用数学归纳法证明：在 魏格纳半圆分布定律 中度数大于 2 的情形下，所有 自由的累积量  $s$  为零。该定律在 自由概率论 和 随机矩阵 理论中非常重要。
- $C_n$  表示用  $n$  个长方形填充一个高度为  $n$  的阶梯状图形的方法个数。下图为  $n = 4$  的情况：



- $C_n$ 表示表为  $2 \times n$  的矩阵的标准杨氏矩阵的数量。也就是说，它是数字  $1, 2, \dots, 2n$  被放置在一个  $2 \times n$  的矩形中并保证每行每列的数字升序排列的方案数。同样的，该式可由勾长公式的一个特殊情形推导得出。
- $C_n$ 表示  $n$  个无标号物品的半序的个数。

汉克尔矩阵[编辑]

无论  $n$  的取值为多少， $n \times n$  的汉克尔矩阵:

$$A_{i,j} = C_{i+j-2}.$$

的行列式为 1。例如， $n = 4$  时我们有

$$\det \begin{bmatrix} 1 & 1 & 2 & 5 \\ 1 & 2 & 5 & 14 \\ 2 & 5 & 14 & 42 \\ 5 & 14 & 42 & 132 \end{bmatrix} = 1$$

。

进一步，无论  $n$  的取值为多少，如果矩阵被移动成

$$A_{i,j} = C_{i+j-1}.$$

，它的行列式仍然为 1。例如， $n = 4$  时我们有

$$\det \begin{bmatrix} 1 & 2 & 5 & 14 \\ 2 & 5 & 14 & 42 \\ 5 & 14 & 42 & 132 \\ 14 & 42 & 132 & 429 \end{bmatrix} = 1$$

。

同时，这两种情形合在一起唯一定义了卡特兰数。

来自 <<https://zh.wikipedia.org/wiki/%E5%8D%A1%E5%A1%94%E5%85%B0%E6%95%B0>>

## LCM 组合数

In this paper, we prove the identity

$$\text{lcm} \left\{ \binom{k}{0}, \binom{k}{1}, \dots, \binom{k}{k} \right\} = \frac{\text{lcm}(1, 2, \dots, k, k+1)}{k+1} \quad (\forall k \in \mathbb{N}).$$

$\text{Lcm}(1,2,\dots,n) = \pi p^{(\log(p)n)}$  其中  $p$  为小于  $n$  的所有素数

题意：给定一个正整数  $n$ ，求

$$\text{lcm}(1,2,3,\dots,n)$$

的值，输入数据有 10000 组，每组一个数  $n$ ， $1 \leq n \leq 10^8$ 。

分析：定义

$$L(n)$$

为  $1, 2, 3, \dots, n$  的最小公倍数。则可以发现有如下结论：

$$L(n+1) = \begin{cases} L(n) * p & \text{if } n+1 \text{ is a perfect power of prime } p \\ L(n) & \text{otherwise} \end{cases}$$

来自 <<http://blog.csdn.net/acdreamers/article/details/18364107>>

skywalkert 1 天前

← ftiasch LCMSUM有一个子问题是 $[1, R]$ 里与 $R$ 互质的数之和，为 $\frac{R\phi(R)}{2}$ ，该怎么快速得到呢？

$$\frac{R\phi(R)}{2}$$

分子第二项为欧拉函数

# Fwt ( 快速沃尔什变换 )

2015 年 10 月 2 日

18:10

## CodeForces #259 div1.D

The energy in Elements of Harmony is in constant movement. According to the ancient book, the energy of vertex  $u$  in time  $i$  ( $e_i[u]$ ) equals to:

$$e_i[u] = \sum_v e_{i-1}[v] \cdot b[f(u, v)].$$

Here  $b[]$  is the transformation coefficient — an array of  $m+1$  integers and  $f(u, v)$  is the number of ones in the binary representation of number  $(u \text{ xor } v)$ .

Given the transformation coefficient and the energy distribution at time 0 ( $e_0[]$ ).

Help Twilight Sparkle predict the energy distribution at time  $t$  ( $e_t[]$ ). The answer can be quite large, so output it modulo  $p$ .

### Input

The first line contains three integers  $m, t$  and  $p$  ( $1 \leq m \leq 20$ ;  $0 \leq t \leq 10^{18}$ ;  $2 \leq p \leq 10^9$ ).

The following line contains  $n$  ( $n = 2^m$ ) integers  $e_0[i]$  ( $1 \leq e_0[i] \leq 10^9$ ;  $0 \leq i < n$ ). The next line contains  $m+1$  integers  $b[i]$  ( $0 \leq b[i] \leq 10^9$ ;  $0 \leq i \leq m$ ).

### Output

Output  $n$  lines, the  $i$ -th line must contain a single integer  $e_t[i]$  modulo  $p$ .

来自 <<http://codeforces.com/contest/453/problem/D>>

```
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long int64;
const int N = (1 << 20) + 10;
int m, n;
int64 t, mod;
int b[22];
int64 A[N], B[N];
int64 aa[N];
```

```

int64 mul(int64 a, int64 b, int64 p) {
    return (a * b - (int64)((long double)a / p * b + 1e-3) * p + p) % p;
}

int count(int s) {
    int ret = 0;
    while (s) s -= s & -s, ++ret;
    return ret;
}

void arrMul(int64 *c, int64 *a, int64 *b) {
    for (int i = 0; i < n; ++i)
        c[i] = mul(a[i], b[i], mod);
}

void FWT(int64 *a, int n) {
    if (n == 1) return;
    int m = n >> 1;
    FWT(a, m);
    FWT(a + m, m);
    for (int i = 0; i < m; ++i) {
        int64 u = a[i], v = a[i + m];
        a[i] = (u + v) % mod;
        a[i + m] = (u - v + mod) % mod;
    }
}

void dFWT(int64 *a, int n) {
    if (n == 1) return;
    int m = n >> 1;
    for (int i = 0; i < m; ++i) {
        int64 u = a[i], v = a[i + m];
        a[i] = (u + v) % mod;
        a[i + m] = (u - v + mod) % mod;
    }
    dFWT(a, m);
    dFWT(a + m, m);
}

int main() {
    cin >> m >> t >> mod;
    n = 1 << m;
    mod *= n;
    for (int i = 0; i < n; ++i) {
        scanf("%I64d", A + i);
    }
}

```

```

    A[i] %= mod;
}
for (int i = 0; i <= m; ++i) {
    scanf("%d", b + i);
    b[i] %= mod;
}
for (int i = 0; i < n; ++i) {
    B[i] = b[count(i)];
}

FWT(A, n);
FWT(B, n);
for (; t >= 1, arrMul(B, B, B)) //类似快速幂
    if (t & 1) arrMul(A, A, B);
dFWT(A, n);
for (int i = 0; i < n; ++i) {
    printf("%l64d\n", A[i] >> m);
}
}

```

来自 <[http://codeforces.com/contest/453/status/D/page/1?order=BY\\_PROGRAM\\_LENGTH\\_ASC](http://codeforces.com/contest/453/status/D/page/1?order=BY_PROGRAM_LENGTH_ASC)>

## 2015 北京网络赛：

### 描述

Long long ago, there lived some rabbits in forest. In the last day of the year, rabbit kingdom would hold a celebration for new year. Of course, rabbit king and his guards would take part in this celebration.

The king had  $2n+1$  guards. At the beginning of celebration, just for fun, the king gave out some money to every guards. All guards were numbered from 1 to  $2n+1$ , and the  $i$ -th guard get  $a_i$  Yuan ( $a_i$  is an integer in  $[0, m]$ ).

After the celebration, in order to show his happiness, the king would give out  $x$  Yuan more to every guard. The king didn't want to give too much or too less, so he decided that  $x$  must be an integer in the interval  $[L, R]$ . The king liked number 0 very much, so he wanted that the xor sum of everybody's amount of money to be zero after the celebration. The king also wanted to

know, under such circumstances, how many different ways could he do to give out the money before celebration.

## 输入

There are no more than 12 test cases.

For each test case, there is only one line containing four integers  $n$ ,  $m$ ,  $L$  and  $R$ . Their meanings are already mentioned above.

( $1 \leq n$ ,  $m \leq 1000$ ,  $1 \leq L \leq R \leq 1000$ )

## 输出

For each test, print one line containing the answer modulo 1000000007

## 说明

For the sample input, here are valid solutions:

(0, 1, 2),  $x = 1$ ,  $(0+1)^{(1+1)}(2+1) = 0$

(0, 2, 1),  $x = 1$

(0, 2, 3),  $x = 3$

(0, 3, 2),  $x = 3$

(1, 0, 2),  $x = 1$

(1, 2, 0),  $x = 1$

(2, 0, 1),  $x = 1$

(2, 0, 3),  $x = 3$

(2, 1, 0),  $x = 1$

(2, 3, 0),  $x = 3$

(3, 0, 2),  $x = 3$

(3, 2, 0),  $x = 3$

**解法：**

这个题，我的想法是这样的：首先枚举 $x$ ，那么也就是需要在 $[x, x+m]$ 中找出 $n$ 个数，使得这 $n$ 个数异或和为0，然后方案数加起来即可。那么对于每个枚举的 $x$ 来说，我们记 $dp[i][j]$ 表示前 $i$ 个数，异或和为 $j$ 的方案数，那么 $dp[i][j] = \sum dp[i-1][k] * b[k \oplus j]$ 。其中 $b[i]=1$ 当且仅当 $x \leq i \leq x+m$ ，否则为0；那么上述式子其实和上一题CF的题就没多大区别了。然后就直接套用fwt，当然这里需要注意的是模数必须乘上做fwt的那个数组的大小 $len$ (这里的 $len$ 也和NTT,FFT一样必须是大于等于 $N$ 的最小二次幂数)。最后答案也需要除以这个 $len$ 。

下面的解法中，直接用 $a[i]$ 数组 $pow\_mod$ ，我认为是因为 $dp[0][j]$ 的初始值都与 $b[i]$ 数组相同；

UPD: 验证发现的确是这样。

#### D. The Celebration of Rabbits

题意：有 $2*n+1$ 个人，第一遍给每个人先发 $[0,m]$ 中任意值的钱，第二遍再给每个人发 $x$ 元， $x$ 的范围为 $[L,R]$ 。问有多少种第一遍发钱的方法使得：存在 $x$ 使得这些人的钱的异或值在第二遍发完之后为0。解法：首先暴力打表发现如果某一方案可行，则第二遍发钱的值 $x$ 一定唯一。（并不会证，但发现 $2*n+1$ 为奇数时有该性质）

然后就可以转化为：先枚举 $x$ ，然后求给每个人发 $[x, x+m]$ 元钱，使得他们异或值为0的方案数。

这个问题可以参照 TC SRM 518，1000 的题解。[链接](#)

（或参照 CF 259 div1 D）

直接枚举 $x$ 之后使用快速沃尔什变换 fwt 即可。

注意模数是 $1e9+7$ ，而 fwt 逆变换之后需要除掉 $2^n$ ，所以模数要改为 $(1e9+7) * 2^n$ ，导致乘法可能会超 long long。如果用快速乘太慢，可以参照下面的写法。

复杂度  $O((R-L)*M*(\log M + \log N))$

比赛的时候写了一下发现本地大数据跑的有点慢，但是交上去就直接过了。

-(听说本题有一个 $O(N*M)$ 复杂度的算法，并不能想到，orz)-

```
1 #include <bits/stdc++.h>
2 #include <ext/hash_map>
3 #include <ext/hash_set>
4 #include <ext/pb_ds/assoc_container.hpp>
5 #include <ext/pb_ds/tree_policy.hpp>
6 #include <ext/pb_ds/priority_queue.hpp>
7 using namespace std;
```



```

8 using namespace __gnu_cxx;
9 using namespace __gnu_pbds;
10 #define XINF INT_MAX
11 #define INF 0x3F3F3F3F
12 #define MP(X,Y) make_pair(X,Y)
13 #define PB(X) push_back(X)
14 #define REP(X,N) for(int X=0;X<N;X++)
15 #define REP2(X,L,R) for(int X=L;X<=R;X++)
16 #define DEP(X,R,L) for(int X=R;X>=L;X--)
17 #define CLR(A,X) memset(A,X,sizeof(A))
18 #define IT iterator
19 #define RIT reverse_iterator
20 typedef long long ll;
21 typedef unsigned long long ull;
22 typedef pair<int,int> PII;
23 typedef vector<PII> VII;
24 typedef vector<int> VI;
25 typedef tree<int, null_type, greater<int>, rb_tree_tag,
tree_order_statistics_node_update > rb_tree_set;
26 typedef tree<int, int, greater<int>, rb_tree_tag,
tree_order_statistics_node_update > rb_tree;
27 #define PQ std::priority_queue
28 #define HEAP __gnu_pbds::priority_queue
29 #define X first
30 #define Y second
31 #define lson(X) ((X)<<1)
32 #define rson(X) ((X)<<1|1)
33
34 ll mo;
35
36 ll mul(ll a,ll b){
37     return ((a*b-ll((((long double)a)/mo*b+1e-3)*mo)%mo+mo)%mo;
38 }
39 ll pow_mod(ll a, ll n) {
40     a%=mo;
41     ll r = 1;
42     while(n) {
43         if(n&1) r = mul(r,a);
44         a=mul(a,a);
45         n>>=1;

```

```

46 }
47 return r;
48 }
49 void fwt(ll *a,int l,int r){
50     if(l!=r){
51         int mid=l+r>>1,len=mid-l+1;
52         fwt(a,l,mid);
53         fwt(a,mid+1,r);
54         for (int i=l;i<=mid;i++){
55             ll u=a[i],v=a[i+len];
56             a[i]=(u+v)%mo;
57             a[i+len]=(u-v)%mo;
58         }
59     }
60 }
61 void ifwt(ll *a,int l,int r){
62     if(l!=r){
63         int mid=l+r>>1,len=mid-l+1;
64         for (int i=l;i<=mid;i++){
65             ll u=a[i],v=a[i+len];
66             a[i]=(u+v)%mo;
67             a[i+len]=(u-v)%mo;
68         }
69         ifwt(a,l,mid);
70         ifwt(a,mid+1,r);
71     }
72 }
73
74 const int M = 1000000007;
75 int N = 2048;
76
77 ll a[2048];
78
79 ll gao(int L, int R, int n) {
80     N = 1;
81     while(N<=R) N*=2;
82     mo = 1LL * M * N;
83     ll res = 0;
84     REP(i,N) a[i] = i<=R&& i>=L?1:0;
85     fwt(a,0,N-1);

```

```

86  REP(i,N) a[i] = pow_mod(a[i], n);
87  ifwt(a,0,N-1);
88  REP(i,N) a[i] = (a[i]+mo)%mo/N;
89  return (a[0]+M)%M;
90 }
91
92 int main()
93 {
94 #ifdef LOCAL
95  freopen("in.txt","r",stdin);
96 #endif // LOCAL
97  int n,m,L,R;
98  while(~scanf("%d%d%d%d",&n,&m,&L,&R)) {
99      n = n*2+1;
100      ll ans = 0;
101      REP2(i,L,R) {
102          ans+=gao(i,m+i,n);
103          ans%=M;
104      }
105      printf("%d\n", (int)ans);
106  }
107  return 0;
108 }

```

来自 <<http://www.cnblogs.com/curs0r/p/4827046.html>>

#### D-[The Celebration of Rabbits](#)

很容易想到可以枚举  $x$ ，然后就变成了  $n*2+1$  个在区间  $[x,x+m]$  中的数字异或为 0 的种数。裸的转移是  $n^3$  的 dp，但是可以用 FWT 加速复杂度变为  $m\log m$ 。

学 FWT 可以去[这里](#) NIM 那道题。

[cpp] [view plaincopyprint?](#)

```

1.  1. // #pragma comment(linker, "/STACK:1024000000,1024000000")
2.  #include<algorithm>
3.  #include<iostream>
4.  #include<cstring>
5.  #include<cstdio>
6.  #include<vector>
7.  #include<cmath>
8.  #include<queue>

```

```

9. #include<stack>
10. #include<set>
11. #include<map>
12. #define LL long long
13. #define MP make_pair
14. #define xx first
15. #define yy second
16. #define lson l, m, rt << 1
17. #define rson m + 1, r, rt << 1|1
18. #define CLR(a, b) memset(a, b, sizeof(a))
19. using namespace std;
20.
21. const int MOD = 1000000007;
22.
23. LL PowMod(LL a, LL n)
24. {
25.     LL ret = 1;
26.     while(n)
27.     {
28.         if(n & 1) ret = ret * a % MOD;
29.         a = a * a % MOD;
30.         n >>= 1;
31.     }
32.     return ret;
33. }
34.
35. LL a[1<<11];
36. LL Inv2 = PowMod(2, MOD - 2);
37.
38. void Transform(int l, int r, LL a[]) /// [l, r)
39. {
40.     if(l == r - 1) return ;
41.     int len = (r - l) >> 1;
42.     int m = l + len;
43.     Transform(l, m, a);
44.     Transform(m, r, a);
45.     for(int i = l; i < m; i++)
46.     {

```

```

47.     LL x1 = a[i], x2 = a[i + len];
48.     a[i] = (x1 - x2 + MOD) % MOD;
49.     a[i + len] = (x1 + x2) % MOD;
50. }
51. }
52.
53. void ReTransform(int l, int r, LL a[])
54. {
55.     if(l == r - 1) return ;
56.     int len = (r - l) >> 1;
57.     int m = l + len;
58.     for(int i = l; i < m; i++)
59.     {
60.         LL y1 = a[i], y2 = a[i + len];
61.         a[i] = (y1 + y2) * Inv2 % MOD;
62.         a[i + len] = (y2 - y1 + MOD) * Inv2 % MOD;
63.     }
64.     ReTransform(l, m, a);
65.     ReTransform(m, r, a);
66. }
67.
68. void Conv(LL a[], LL b[], int len)
69. {
70.     Transform(0, len, a);
71.     Transform(0, len, b);
72.     for(int i = 0; i < len; i++)
73.         a[i] = a[i] * b[i] % MOD;
74.     ReTransform(0, len, a);
75. }
76.
77. int n, m, L, R;
78.
79. void solve()
80. {
81.     LL ans = 0;
82.     for(int i = L; i <= R; i++)
83.     {
84.         CLR(a, 0);

```

```

85.     for(int j = i; j <= i + m; j++)
86.         a[j] = 1;
87.     Transform(0, 1 << 11, a);
88.     for(int j = 0; j < (1 << 11); j++)
89.     {
90.         a[j] = PowMod(a[j], n * 2 + 1);
91.     }
92.     ReTranform(0, 1 << 11, a);
93.     ans += a[0]; ans %= MOD;
94. }
95. printf("%lld\n", ans);
96. }
97.
98.int main()
99. {
100.     while(scanf("%d%d%d%d", &n, &m, &L, &R) != EOF)
101.     {
102.         solve();
103.     }
104.     return 0;
105. }

```

来自 <[http://blog.csdn.net/ok\\_again/article/details/48656505](http://blog.csdn.net/ok_again/article/details/48656505)>

## 特征方程求数列通项

2015 年 10 月 21 日  
17:33

② 为什么特征方程可以求数列通项?数列  $\{a(n)\}$ , 设递推公式为  $a(n+2)=p*a(n+1)+q*a(n)$ , 其特征方程为  $x^2-px-q=0$ . 若方程有两相异根  $A$ 、 $B$ 。为什么就有  $a(n)=c*A^n+d*B^n$ ?

为什么特征方程可以求数列通项?

数列  $\{a(n)\}$ , 设递推公式为  $a(n+2)=p*a(n+1)+q*a(n)$ , 其特征方程为  $x^2-px-q=0$ .

若方程有两相异根  $A$ 、 $B$ 。为什么就有  $a(n)=c*A^n+d*B^n$ ?

血刺心碎们w 数学 2014-10-16



### 优质解答

数列  $\{a(n)\}$ , 设递推公式为  $a(n+2)=p*a(n+1)+q*a(n)$ , 则其特征方程为  $x^2-px-q=0$ .

若方程有两相异根  $A$ 、 $B$ , 则  $a(n)=c*A^n+d*B^n$  ( $c$ 、 $d$ 可由初始条件确定,下同)

若方程有两等根  $A=B$ , 则  $a(n)=(c+nd)*A^n$

回答者SKY9314 的回答准确来说是以上部分内容的证明过程:

设  $r$ 、 $s$  使  $a(n+2)-r*a(n+1)=s[a(n+1)-r*a(n)]$

所以  $a(n+2)=(s+r)*a(n+1)-sr*a(n)$

即,  $s+r=p$ ,  $sr=-q$ , 由韦达定理可知,  $r$ 、 $s$  就是一元二次方程  $x^2-px-q=0$  的两根, 也就是刚才说的特征根.

然后进一步证明那个通项公式:

如果  $r=s$ , 那么数列  $\{a(n+1)-r*a(n)\}$  是以  $a(2)-r*a(1)$  为首项、 $r$  为公比的等比数列, 根据等比数列的性质可知:  $a(n+1)-r*a(n) = [a(2)-r*a(1)]*r^{(n-1)}$ ,

两边同时除以  $r^{(n+1)}$ , 得到  $a(n+1)/r^{(n+1)}-a(n)/r^n = a(2)/r^2-a(1)/r$

等号右边的是个常数, 说明数列  $\{a(n)/r^n\}$  是个等差数列. 显然等号右边那个就是公差, 首项也比较明显, 这里不重复了. 根据等差数列性质:  $a(n)/r^n = a(1)/r + (n-1)*[a(2)/r^2-a(1)/r]$  整理一下, 并设  $a(2)/r^2-a(1)/r = d$ , 再设  $2a(1)/r-a(2)/r^2 = c$ , 然后把那个  $r$  用  $A$  来代, 就可以得到  $a(n)=(c+nd)*A^n$  了.

例题:  
ZOJ 3857

很神的一道题。

题意：给定 $k, L$ ，求 $(L + \sqrt{L(L-1)})^k \% k$ （下取整）。 $k, L$ 属于 $\text{int}$ 范围内。

先输入 $k$ ，再输入 $L$ 。注意不要打反了。

题解：

将所求的 $L + \sqrt{L(L-1)}$ 看做一个根，另外一个根是 $L - \sqrt{L(L-1)}$ 。

令 $a_k = (L + \sqrt{L(L-1)})^k + (L - \sqrt{L(L-1)})^k$ 。

后者是一个极小的数，最大为1，最小为0.5，所以求得 $a_k$ 后直接-1就是答案。

直接求通项公式， $x^2 - 2Lx + L = 0$ 。

$a[n] = 2L \cdot a[n-1] - L \cdot a[n-2]$ 。

我构造的矩阵递推，事实上多项式乘法应该是更优的，只不过我掌握得太少。

矩阵：

$\begin{bmatrix} 0 & -L \end{bmatrix}$

$\begin{bmatrix} 1 & 2L \end{bmatrix}$

快速幂求解， $a_0, a_1$ ，直接算。

[cnn] 1 0

2015/10/21 17:36 - 屏幕剪辑

## 欧拉函数

2015 年 11 月 5 日  
15:17

先来介绍几个与欧拉函数有关的定理：

定理一：设  $m$  与  $n$  是互素的正整数，那么

$$\varphi(mn) = \varphi(m)\varphi(n)$$



定理二：当  $n$  为奇数时，有

$$\varphi(2n) = \varphi(n)$$

因为  $2n$  是偶数，偶数与偶数一定不互素，所以只考虑  $2n$  与小于它的奇数互素的情况，则恰好就等于  $n$  的欧拉函数值。

定理三：设  $p$  是素数， $a$  是一个正整数，那么

$$\varphi(p^a) = p^a - p^{a-1}$$

关于这个定理的证明用到容斥：

定理四：设

$$n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$$

为正整数  $n$  的素数幂分解，那么

$$\varphi(n) = n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_k}\right)$$

这个定理可以根据定理一和定理三证明，其实用到的就是容斥。如果对容斥熟悉，其实完全就可以直接容斥。

定理五：设  $n$  是一个正整数，那么

$$\sum_{d|n} \varphi(d) = n$$

这个其实可以看莫比乌斯反演就明白了。

定理六：设  $m$  是正整数， $(a, m) = 1$ ，则：

$$x \equiv ba^{\varphi(m)-1} \pmod{m}$$

是同于方程

$$ax \equiv b \pmod{m}$$

的解。

定理七：如果  $n$  大于 2，那么  $n$  的欧拉函数值是偶数。

来自 <<http://blog.csdn.net/acdreamers/article/details/8007991>>

## pollard\_rho 大数分解 Java 版

2015 年 11 月 5 日  
15:25

```
import java.math.BigInteger;
import java.security.SecureRandom;

class PollardRho
{
    private final static BigInteger ZERO = new BigInteger("0");
    private final static BigInteger ONE = new BigInteger("1");

    private final static BigInteger TWO = new BigInteger("2");

    private final static SecureRandom random = new SecureRandom();

    public static BigInteger rho(BigInteger N)
    {
        BigInteger divisor;
        BigInteger c = new BigInteger(N.bitLength(), random);

        BigInteger x = new BigInteger(N.bitLength(), random);

        BigInteger xx = x;

        if (N.mod(TWO).compareTo(ZERO) == 0) return TWO;

        do
        {
            x = x.multiply(x).mod(N).add(c).mod(N);
            xx = xx.multiply(xx).mod(N).add(c).mod(N);
            xx = xx.multiply(xx).mod(N).add(c).mod(N);
            divisor = x.subtract(xx).gcd(N);
        } while((divisor.compareTo(ONE)) == 0);

        return divisor;
    }

    public static void factor(BigInteger N)
    {
        if (N.compareTo(ONE) == 0) return;
    }
}
```

```

        if (N.isProbablePrime(20))
        {
            System.out.println(N);
            return;
        }
        BigInteger divisor = rho(N);
        factor(divisor);
        factor(N.divide(divisor));
    }

    public static void main(String[] args)
    {
        BigInteger N = BigInteger.valueOf(120);
        factor(N);
    }
}

```

来自 <<http://blog.csdn.net/acdreamers/article/details/9671633>>

## 等比数列二分求和

2015 年 11 月 5 日  
15:31

今天我们学习如何有效地求表达式

$$S_n = (a + a^2 + \dots + a^n) \bmod M$$

的值。对于这个问题，用二分解决比较好。

(1) 当

$$n == 0$$

时，

$$S_n = 1$$

(2) 当

$$n \% 2 == 0$$

时，那么有

$$\begin{aligned}
S_n &= a + a^2 + \dots + a^n \\
&= a + a^2 + \dots + a^{\frac{n}{2}} + a^{\frac{n}{2}+1} + \dots + a^{\frac{n}{2}+\frac{n}{2}} \\
&= (1 + a^{\frac{n}{2}})(a + a^2 + \dots + a^{\frac{n}{2}}) \\
&= (1 + a^{\frac{n}{2}})S_{\frac{n}{2}}
\end{aligned}$$

(3) 当  
 $n \% 2 == 1$   
 时，那么有

$$\begin{aligned}
S_n &= a + a^2 + \dots + a^n \\
&= a + a^2 + \dots + a^{\frac{n-1}{2}} + a^{\frac{n-1}{2}+1} + a^{\frac{n-1}{2}+2} + \dots + a^{\frac{n-1}{2}+\frac{n-1}{2}+1} \\
&= (a + a^2 + \dots + a^{\frac{n-1}{2}}) + a^{\frac{n-1}{2}+1}(a + a^2 + \dots + a^{\frac{n-1}{2}}) + a^{\frac{n-1}{2}+1} \\
&= (1 + a^{\frac{n-1}{2}+1})S_{\frac{n-1}{2}} + a^{\frac{n-1}{2}+1}
\end{aligned}$$

来自 <http://blog.csdn.net/acdreamers/article/details/7851144>

## LCA+倍增+带权并查集： POJ 3728 The Merchant

2015 年 7 月 27 日  
 2:22

### F - The merchant

**Time Limit:**3000MS    **Memory Limit:**65536KB    **64bit IO Format:**%l64d  
 & %l64u

[Submit](#) [Status](#)

### Description

There are  $N$  cities in a country, and there is one and only one simple path between each pair of cities. A merchant has chosen some paths and wants to earn as much money as possible in each path. When he move along a path, he can choose one city to buy some goods and sell them in a city after it. The goods in all cities are the same but the prices are different. Now your task is to calculate the maximum possible profit on each path.

### Input

The first line contains  $N$ , the number of cities.

Each of the next  $N$  lines contains  $w_i$  the goods' price in each city.

Each of the next  $N-1$  lines contains labels of two cities, describing a road between the two cities.

The next line contains  $Q$ , the number of paths.

Each of the next  $Q$  lines contains labels of two cities, describing a path. The cities are numbered from 1 to  $N$ .

$1 \leq N, w_i, Q \leq 50000$

## Output

The output contains  $Q$  lines, each contains the maximum profit of the corresponding path. If no positive profit can be earned, output 0 instead.

### 题意:

在一颗树上，每个节点都有一个 val，有很多个询问，每个询问 u、v，要求输出 u->v 的路径上 X-Y 的最大值，其中，在 u->v 的 simple path 中，X 必须在 Y 的前面。

### 做法：

一：

这个题我最开始 YY 了个想法，在求 LCA 的过程中同时记录四个与 par 数组对应的数组分别为 mi[k][u], ma, sub, subrev，表示从 u 节点开始到 u 的第  $2^k$  个父亲之间的最小值，最大值，儿子 val-父亲 val 的最大值，父亲 val-儿子 val 的最大值，对于每个询问来说在求 LCA 的过程中不断更新答案即可。AC

二：

后来看了一下别人的代码，发现别人用的都是 tarjan 离线 LCA.....，赶紧学习这种姿势又补一发，顺带弄懂了带权并查集。先说做法，就是对每个点记录一个 mi, ma, Uto, toU，表示 u 这个点到当前的根的最小值，最大值，儿子 val-父亲 val 的最大值，父亲 val-儿子 val 的最大值，所不同的是，这里的根由于一直在变，所以更新是需要在并查集压缩路径的过程中完成的。

对于带权并查集，我的理解就是，维护一个集合中每个点与祖宗的关系，这个和之前一直维护整个集合的信息不一样。

### 代码一：

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
// #include <unordered_map>
#define N 100010
// #define ID(x, y) ((x) * m + (y))
// #define CHECK(x, y) ((x) >= 0 && (x) < n && (y) >= 0 && (y) < m)
```

```

using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int n,m;
vector<int> g[N];
int root, par[20][N];
int ma[20][N];
int mi[20][N];
int sub[20][N];
int subrev[20][N];
int dep[N];
int sn[N];
void dfs(int v, int p, int d)
{
    par[0][v] = p, ma[0][v] = mi[0][v] = sn[p], sub[0][v] = 0,
    subrev[0][v] = 0, dep[v] = d;
    for(int i=0;i<g[v].size();i++) if(g[v][i] != p) dfs(g[v][i],
v, d+1);
}
void init()
{
    root = 1;
    dfs(root, -1, 0);
    for(int k=0;k<20;k++)
        for(int v = 1;v <= n;v++)
            if(par[k][v] < 0) par[k+1][v] = -1, ma[k+1][v] = 0,
mi[k+1][v] = INF, sub[k+1][v] = subrev[k+1][v] = 0;
            else {
                par[k+1][v] = par[k][par[k][v]], ma[k+1][v] =
max(ma[k][v], ma[k][par[k][v]]), mi[k+1][v] = min(mi[k][v],
mi[k][par[k][v]]);
                sub[k+1][v] = max(max(sub[k][v],
sub[k][par[k][v]]), ma[k][v] - mi[k][par[k][v]]);
                subrev[k+1][v] = max(max(subrev[k][v],
subrev[k][par[k][v]]), ma[k][par[k][v]] - mi[k][v]);
            }
}
int lca(int u,int v)
{
    int tmpu = u, tmpv = v;
    int curmi = sn[v], curma = sn[u];
    int ans = 0;
    if(dep[u] < dep[v]){
        for(int k=0;k<20;k++)
            if((dep[v]-dep[u])>>k&1){
                ans = max(ma[k][v] - curmi, max(ans,
subrev[k][v]));
                curmi = min(curmi, mi[k][v]);
                v = par[k][v];
            }
    }else {

```

```

        for(int k=0;k<20;k++)
            if((dep[u]-dep[v])>>k&1){
                ans = max(curma - mi[k][u], max(ans, sub[k][u]));
                curma = max(curma, ma[k][u]);
                u = par[k][u];
            }
    }
    //if(u == v) cout<<"pa: "<<u<<endl;
    if(u==v) return max(max(max(ans, curma - sn[tmpv]), sn[tmpu]
- curmi), sn[tmpu] - sn[tmpv]);
    for(int k=20-1;k>=0;k--)
    {
        if(par[k][u]!=par[k][v])
        {
            ans = max(ans, max(curma - mi[k][u], ma[k][v] -
curmi));
            curmi = min(curmi, mi[k][v]);
            curma = max(curma, ma[k][u]);
            ans = max(ans, max(sub[k][u],subrev[k][v]));
            u=par[k][u];
            v=par[k][v];
        }
    }
    int pa = par[0][u];
    ans = max(ans, max(curma - sn[pa], sn[pa] - curmi));
    ans = max(ans, curma - curmi);
    //cout<<"pa : "<<pa<<endl;
    //cout<<tmpu<<" "<<tmpv<<" : "<<curma<<","<<curmi<<endl;
    return ans;
}
int main()
{
    //Open();
    while(~scanf("%d", &n))
    {
        for(int i=1;i<=n;i++)
            scanf("%d", sn+i);
        for(int i=0;i<=n;i++)
            g[i].clear();
        for(int i=1;i<n;i++)
        {
            int x, y;
            scanf("%d%d", &x, &y);
            g[x].push_back(y);
            g[y].push_back(x);
        }
        init();
        scanf("%d", &m);
        while(m--)
        {
            int x, y;
            scanf("%d%d", &x, &y);
            //cout<<x<<" "<<y<<" : ";
            printf("%d\n", lca(y, x));
        }
    }
}

```

```

    }
}
return 0;
}

```

来自 <<http://acm.hust.edu.cn/vjudge/contest/viewSource.action?id=4137678>>

代码二:

```

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
// #include <unordered_map>
#define N 100010
// #define ID(x, y) ((x)*m+(y))
// #define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        // freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
struct info{
    int u, v, id;
    info(int u = 0, int v = 0, int id = 0):u(u), v(v), id(id){}
};
int n, m;
vector<int> g[N];
vector<PII> query[N];
vector<info> rt[N];
int Uto[N], toU[N], ma[N], mi[N], sn[N], ans[N], fa[N];
bool vis[N];
void init()
{
    for(int i=0; i<=n; i++)
    {
        vis[i] = 0;
        g[i].clear();
        query[i].clear();
        rt[i].clear();
        fa[i] = i;
        Uto[i] = toU[i] = 0;
        ma[i] = mi[i] = sn[i];
    }
}
int Find(int u)
{
    if(fa[u] == u) return u;
    int root = Find(fa[u]);
    Uto[u] = max(Uto[u], max(Uto[fa[u]], ma[fa[u]] - mi[u]));
}

```



```

    toU[u] = max(toU[u], max(toU[fa[u]], ma[u] - mi[fa[u]]));
    ma[u] = max(ma[u], ma[fa[u]]);
    mi[u] = min(mi[u], mi[fa[u]]);
    return fa[u] = root;
}

void lca(int u)
{
    vis[u] = 1;
    for(int i=0;i<query[u].size();i++)
    {
        int v = query[u][i].first, id = query[u][i].second;
        if(vis[v])
        {
            int root = Find(v);
            if(id < 0) rt[root].push_back(info(v, u, -id));
            else rt[root].push_back(info(u, v, id));
        }
    }
    for(int i=0;i<g[u].size();i++)
    {
        int v = g[u][i];
        if(vis[v]) continue;
        lca(v);
        fa[v] = u;
    }
    for(int i=0;i<rt[u].size(); i++)
    {
        int x = rt[u][i].u, y = rt[u][i].v, id = rt[u][i].id;
        //cout<<"lca: "<<u<<"("<<x<<","<<y<<")  ans:";
        Find(x), Find(y);
        ans[id] = max(Uto[x], max(toU[y], ma[y] - mi[x]));
        //cout<<id<<":"<<ans[id]<<endl;
    }
}

int main()
{
    Open();
    while(~scanf("%d", &n))
    {
        for(int i=1;i<=n;i++)
            scanf("%d", &sn+i);
        init();
        for(int i=1;i<=n;i++)
        {
            int x, y;
            scanf("%d%d", &x, &y);
            g[x].push_back(y);
            g[y].push_back(x);
        }
        scanf("%d", &m);
        for(int i=1;i<=m;i++)
        {
            int u, v;
            scanf("%d%d", &u, &v);
            query[u].push_back(PII(v, i));
        }
    }
}

```

```

        query[v].push_back(PII(u, -i));
    }
    lca(1);
    for(int i=1;i<=m;i++)
        printf("%d\n", ans[i]);
    //cout<<endl;
}
return 0;
}

```

来自 <http://acm.hust.edu.cn/vjudge/contest/viewSource.action?id=4143872>

## LCA: POJ 3694 Network

2015 年 7 月 27 日  
2:37

### G - Network

**Time Limit:**5000MS    **Memory Limit:**65536KB    **64bit IO Format:**%l64d & %l64u

[Submit Status](#)

### Description

A network administrator manages a large network. The network consists of  $N$  computers and  $M$  links between pairs of computers. Any pair of computers are connected directly or indirectly by successive links, so data can be transformed between any two computers. The administrator finds that some links are vital to the network, because failure of any one of them can cause that data can't be transformed between some computers. He call such a link a bridge. He is planning to add some new links one by one to eliminate all bridges.

You are to help the administrator by reporting the number of bridges in the network after each new link is added.

### Input

The input consists of multiple test cases. Each test case starts with a line containing two integers  $N$  ( $1 \leq N \leq 100,000$ ) and  $M$  ( $N - 1 \leq M \leq 200,000$ ).

Each of the following  $M$  lines contains two integers  $A$  and  $B$  ( $1 \leq A \neq B \leq N$ ), which indicates a link between computer  $A$  and  $B$ . Computers are numbered from 1 to  $N$ . It is guaranteed that any two computers are connected in the initial network.

The next line contains a single integer  $Q$  ( $1 \leq Q \leq 1,000$ ), which is the number of new links the administrator plans to add to the network one by one.

The  $i$ -th line of the following  $Q$  lines contains two integer  $A$  and  $B$  ( $1 \leq A \neq B \leq N$ ), which is the  $i$ -th added new link connecting computer  $A$  and  $B$ .

The last test case is followed by a line containing two zeros.

## Output

For each test case, print a line containing the test case number( beginning with 1) and  $Q$  lines, the  $i$ -th of which contains a integer indicating the number of bridges in the network after the first  $i$  new links are added. Print a blank line after the output for each test case.

## 题意：

在一个无向图中，存在一些桥，人们想要增加一些边使得桥消失，现在每次加边操作之后要询问你图中还存在多少桥。

## 做法：

首先桥肯定只可能减少，所以我们双联通缩点，之后得到一棵树，树边肯定都是桥，只需要每次求 lca 的过程中记录并计数即可

# DFS 序线段树： HDU 5039 Hilarity

2015 年 7 月 27 日  
14:50

## Hilarity

Time Limit: 12000/6000 MS (Java/Others) Memory Limit: 262144/262144 K (Java/Others)  
Total Submission(s): 492 Accepted Submission(s): 137

## Problem Description

After June 1st, elementary students of Ted Land are still celebrating "The Sacred Day of Elementary Students". They go to the streets and do some elementary students stuff. So we call them "elementary students". There are  $N$  cities in Ted Land. But for simplicity, the mayor Matt only built  $N - 1$  roads so that all cities can reach each other. Some of the roads are occupied by the "elementary students". They will put an celebration hat on everyone who goes through the road without one. But if someone goes through the road with a celebration hat on his head, "elementary students" will steal the hat for no reason. Since Matt doesn't have a celebration hat, he wants to know how many different paths in his land that he can end up with a hat. Two paths are considered to be different if and only if they have different start city or end city. As the counsellor of the mayor Matt, you have to answer this question for him. The celebration elementary students are not stable: sometimes a new crowd of elementary students go to an empty road; sometimes the elementary students on a road will go back home and remain the road empty. Matt will send you the monitor about the change of elementary students on road and ask you the question above. You will be fired if you answer wrong.

## Input

The first line of the input contains an integer  $T$ , denoting the number of testcases. Then  $T$  test cases follow.

For each test case, the first line contains  $N$  ( $1 \leq N \leq 30000$ ) describing the number of cities.

Then  $N$  lines follow. The  $i$ th line of these contains the name of the  $i$ th city, it's a string containing only letters and will not be longer than 10.

The following  $N - 1$  lines indicate the  $N - 1$  edges between cities. Each of these lines will contain two strings for the cities' name and an integer for the initial status of the edge (0 for empty, 1 for crowded).

Then an integer  $M$  ( $1 \leq M \leq 60000$ ) describes the number of queries. There are two kinds of query:

- "M  $i$ " means the status changing of the  $i$ th (starting from 1) road (0 to 1, 1 to 0);
- "Q" means that Matt asks you the number of different paths.

## Output

For each test case, first output one line "Case #x:", where  $x$  is the case number (starting from 1).

Then for each "Q" in input, output a line with the answer.

这一组题都是 DFS 序线段树的题，之前遇到过没做出来，现在刷一波专题

## 题意：

一棵树上有很多条边，每条边上可能有一个小学生，我经过每条有小学生的边时，小学生会给我带上帽子，或者偷掉我的帽子，最开始我没有帽子。两种操作，"M  $i$ " 表示更改第  $i$  条的小学生的状态（在不在这条边）"Q" 询问此时整棵树上有多少条路径能让我最后带上帽子。

## 做法：

我们将小学生所在的边标记为 1，否则标记为 0，那么能得到正确答案的路径的异或和一定为 1。首先我们考虑没有修改操作的询问。我们可以处理每个节点到根节点的异或和，统计整棵树上面有多少个 0，多少个 1，相乘再加上 1 的个数即是答案（想一下应该能想到，对了，还要乘 2，因为一条路径可以以两个方向来走）。那对于修改的操作来说，修改某条边的状态，即是将这条边对应的子树的状态全部翻转即可，酱紫的区间修改操作，那当然用线段树来做啦。

## 代码：

```

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <unordered_map>
#define N 100010
#define lson(x) x << 1
#define rson(x) x << 1 | 1
// #define ID(x, y) ((x)*m+(y))
// #define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        // freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
struct node
{
    int l, r, ans;
    bool flag;
    node(int l = 0, int r = 0, int a = 0):l(l), r(r), ans(a){flag
= 0;}
}lt[N*8];
int n,m;
PII e[N*2];
int val[N];
int st[N], ed[N], Tn;
vector<PII> G[N];
unordered_map<string, int> f;
void push_down(int x)
{
    if(lt[x].flag)
    {
        lt[x].flag = 0;
        lt[lson(x)].flag = !lt[lson(x)].flag;
        lt[lson(x)].ans = lt[lson(x)].r - lt[lson(x)].l + 1 -
lt[lson(x)].ans;
        lt[rson(x)].flag = !lt[rson(x)].flag;
        lt[rson(x)].ans = lt[rson(x)].r - lt[rson(x)].l + 1 -
lt[rson(x)].ans;
    }
}
void build(int l, int r, int x = 1)
{
    lt[x] = node(l, r);
    if(l == r)
    {
        lt[x].ans = val[l];
        return;
    }
    int mid = (l+r) / 2;

```

```

    build(l, mid, lson(x));
    build(mid+1, r, rson(x));
    lt[x].ans = lt[lson(x)].ans + lt[rson(x)].ans;
}
void update(int l, int r, int x = 1)
{
    //lt[x].ans = r - l + 1 - lt[x].ans;
    if(lt[x].l >= l && lt[x].r <= r)
    {
        lt[x].flag = !lt[x].flag;
        lt[x].ans = lt[x].r - lt[x].l + 1 - lt[x].ans;
        return ;
    }
    push_down(x);
    int mid = (lt[x].l + lt[x].r) / 2;
    if(r <= mid) update(l, r, lson(x));
    else if(l > mid) update(l, r, rson(x));
    else update(l, mid, lson(x)), update(mid+1, r, rson(x));
    lt[x].ans = lt[lson(x)].ans + lt[rson(x)].ans;
}
void dfs(int u, int prew)
{
    st[u] = ++Tn;
    val[Tn] = prew;
    for(int i=0; i<G[u].size(); i++)
    {
        int v = G[u][i].first;
        if(st[v]) continue;
        int w = G[u][i].second;
        e[w/10] = PII(v, w%10);
        dfs(v, prew^(w%10));
    }
    ed[u] = Tn;
}
char tmp[111];
int main()
{
    Open();
    int T; scanf("%d", &T);
    int cas = 1;
    while(T--)
    {
        Tn = 0;
        memset(st, 0, sizeof(st));
        memset(ed, 0, sizeof(ed));
        f.clear();
        scanf("%d", &n);
        for(int i=0; i<=n; i++)
            G[i].clear();
        for(int i=1; i<=n; i++)
        {
            scanf("%s", tmp);
            f[tmp] = i;
        }
    }
}

```

```

for(int i=1;i<n;i++)
{
    scanf("%s", tmp); int u = f[tmp];
    scanf("%s", tmp); int v = f[tmp];
    int w;scanf("%d", &w);
    G[u].push_back(PII(v, i*10+w));
    G[v].push_back(PII(u, i*10+w));
}
dfs(1, 0);
build(2, Tn);
scanf("%d", &m);
printf("Case #%%d:\n", cas++);
while(m--)
{
    scanf("%s", tmp);
    if(tmp[0] == 'Q'){
        printf("%d\n", 2*(lt[1].ans + lt[1].ans * (n -
lt[1].ans - 1)));
    }else {
        int x;
        scanf("%d", &x);
        update(st[e[x].first], ed[e[x].first]);
    }
}
}
return 0;
}

```

来自 <<http://acm.hust.edu.cn/vjudge/contest/viewSource.action?id=4161066>>

## DFS 序线段树 : CodeForces 258 E

2015 年 7 月 27 日  
15:09

### E. Little Elephant and Tree

The Little Elephant loves trees very much, he especially loves root trees.  
He's got a tree consisting of  $n$  nodes (the nodes are numbered from 1 to  $n$ ), with root at node number 1. Each node of the tree contains some list of numbers which initially is empty.

The Little Elephant wants to apply  $m$  operations. On the  $i$ -th operation ( $1 \leq i \leq m$ ) he first adds number  $i$  to lists of all nodes of a subtree with the root in node number  $a_i$ , and then he adds number  $i$  to lists of all nodes of the subtree with root in node  $b_i$ . After applying all operations the Little Elephant wants to count for each node  $i$  number  $C_i$  — the number of integers  $j$  ( $1 \leq j \leq n; j \neq i$ ), such that the lists of the  $i$ -th and the  $j$ -th nodes contain at least one common number. Help the Little Elephant, count numbers  $C_i$  for him.

### Input

The first line contains two integers  $n$  and  $m$  ( $1 \leq n, m \leq 10^5$ ) — the number of the tree nodes and the number of operations.

Each of the following  $n-1$  lines contains two space-separated integers,  $u_i$  and  $v_i$  ( $1 \leq u_i, v_i \leq n, u_i \neq v_i$ ), that mean that there is an edge between nodes number  $u_i$  and  $v_i$ .

Each of the following  $m$  lines contains two space-separated integers,  $a_i$  and  $b_i$  ( $1 \leq a_i, b_i \leq n, a_i \neq b_i$ ), that stand for the indexes of the nodes in the  $i$ -th operation.

It is guaranteed that the given graph is an undirected tree.

### Output

In a single line print  $n$  space-separated integers —  $C_1, C_2, \dots, C_n$ .

## 题意：

在一棵树上可做  $m$  个操作，对第  $i$  个操作来说，可以将  $i$  添加到  $u$  的所有子树的节点上，最后询问每个节点  $i$  有多少个节点  $j$  满足“节点  $i$  和节点  $j$  含有至少一个相同的数”

## 做法：

这个题简直是线段树的花式更新。。。可能是我太弱了，第一次看到这种姿势，被吓到，看到了 cxlove 的代码，比较神奇的是这一题懒标记不是在 `push_down` 的时候下放，而是根本不下放，只是在 `push_up` 的时候判断这个节点的 `cnt` 该如何更新。首先还是先处理出 `dfs` 序，然后在 `dfs` 的时候边搜边更新，更新完之后记录答案（Orz）。因为对某个点的操作一定会将他的整个子树都会有影响，所以在 `dfs` 之前先更新，在往下搜，搜完当前点的子树之后，需要还原整棵线段树，也就是反着更新一边就可以了。这里的懒标记记录的是对应区间被累加了多少次，还原的时候将懒标记-1，当懒标记为 0 的时候，这个子树上的所有节点不在计算到答案中。看代码应该有助于理解上面这段话

## 代码：

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <set>
```



```

#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
#define lson (x<<1)
#define rson (x<<1|1)
#define mid ((lt[x].r+lt[x].l)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
}
struct node{
    int l, r, cnt, flag;
    int length(){return r - l + 1;}
}lt[N*8];
vector<int> G[N], q[N];
int ans[N];
int n, m, st[N], ed[N], Tn;
void dfs_Tn(int u, int p = -1)
{
    st[u] = ++Tn;
    for(int i=0; i<G[u].size(); i++)
    {
        int v = G[u][i];
        if(v == p) continue;
        dfs_Tn(v, u);
    }
    ed[u] = Tn;
}
void build(int l, int r, int x = 1)
{
    lt[x] = (node){l, r, 0, 0};
    if(l == r) return ;
    build(l, mid, lson);
    build(mid+1, r, rson);
}
void push_up(int x)
{
    if(lt[x].flag)
        lt[x].cnt = lt[x].length();
    else {
        if(lt[x].length() == 1) lt[x].cnt = 0;
        else lt[x].cnt = lt[lson].cnt + lt[rson].cnt;
    }
}

```

```

void update(int l, int r, int val, int x = 1)
{
    if(lt[x].l >= l && lt[x].r <= r)
    {
        lt[x].flag += val;
        push_up(x);
        return ;
    }
    if(r <= mid) update(l, r, val, lson);
    else if(l > mid) update(l, r, val, rson);
    else update(l, mid, val, lson), update(mid+1, r, val, rson);
    push_up(x);
}

void dfs_ans(int u, int p = -1)
{
    for(int i=0;i<q[u].size();i++)
    {
        int v = q[u][i];
        update(st[v], ed[v], 1);
    }
    ans[u] = lt[1].cnt;
    if(ans[u]) ans[u]--;
    for(int i=0;i<G[u].size();i++)
    {
        int v = G[u][i];
        if(v == p) continue;
        dfs_ans(v, u);
    }
    for(int i=0;i<q[u].size();i++)
    {
        int v = q[u][i];
        update(st[v], ed[v], -1);
    }
}

int main()
{
    Open();
    scanf("%d%d", &n, &m);
    for(int i=1;i<n;i++)
    {
        int u,v;
        scanf("%d%d", &u, &v);
        G[u].push_back(v);
        G[v].push_back(u);
    }
    dfs_Tn(1);
    //cout<<Tn<<endl;
    for(int i=0;i<m;i++)
    {
        int u, v;
        scanf("%d%d", &u, &v);
        q[u].push_back(u), q[u].push_back(v);
        q[v].push_back(u), q[v].push_back(v);
    }
}

```

```

    }
    build(1, Tn);
    dfs_ans(1);
    for(int i=1;i<=n;i++)
        printf("%d%c", ans[i], (i == n ? '\n' : ' '));
    return 0;
}

```

来自 <<http://acm.hust.edu.cn/vjudge/contest/viewSource.action?id=4168556>>

## DFS 序线段树 : CodeForces 343 D

2015 年 7 月 27 日  
15:26

### D. Water Tree

Mad scientist Mike has constructed a rooted tree, which consists of  $n$  vertices. Each vertex is a reservoir which can be either empty or filled with water.

The vertices of the tree are numbered from 1 to  $n$  with the root at vertex 1. For each vertex, the reservoirs of its children are located below the reservoir of this vertex, and the vertex is connected with each of the children by a pipe through which water can flow downwards.

Mike wants to do the following operations with the tree:

1. Fill vertex  $v$  with water. Then  $v$  and all its children are filled with water.
2. Empty vertex  $v$ . Then  $v$  and all its ancestors are emptied.
3. Determine whether vertex  $v$  is filled with water at the moment.

Initially all vertices of the tree are empty.

Mike has already compiled a full list of operations that he wants to perform in order. Before experimenting with the tree Mike decided to run the list through a simulation. Help Mike determine what results will he get after performing all the operations.

#### Input

The first line of the input contains an integer  $n$  ( $1 \leq n \leq 500000$ ) — the number of vertices in the tree. Each of the following  $n-1$  lines contains two space-separated numbers  $a_i, b_i$  ( $1 \leq a_i, b_i \leq n, a_i \neq b_i$ ) — the edges of the tree.

The next line contains a number  $q$  ( $1 \leq q \leq 500000$ ) — the number of operations to perform. Each of the following  $q$  lines contains two space-separated numbers  $c_i$  ( $1 \leq c_i \leq 3$ ),  $v_i$  ( $1 \leq v_i \leq n$ ), where  $c_i$  is the operation type (according to the numbering given in the statement), and  $v_i$  is the vertex on which the operation is performed.

It is guaranteed that the given graph is a tree.

## Output

For each type 3 operation print 1 on a separate line if the vertex is full, and 0 if the vertex is empty. Print the answers to queries in the order in which the queries are given in the input.

来自 <<http://codeforces.com/problemset/problem/343/D>>

## 题意：

一棵树上有若干个水箱，每个节点的状态只有两种，有水或没水，有三种操作，  
一是给一个节点灌水，然后这个节点的子树上所有节点都会灌满水；  
二是将一个节点的水抽走，那么这个节点已经他的所有父亲全部都会被抽干；  
三是询问某个节点现在处于那种状态。

## 做法：

对于第一种操作，正常的区间修改整个子树就可以了，而第二种操作，只需要在节点对应的线段树上标记该点为 0 即可。对于询问每个点的状态，只需要检查这个节点对应的子树上有没有值为 0 的点即可。有就是空，没有就是 1. 由于这个题的只需要检测子树有没有 0，，可以不用写线段树，直接用 set 的话，可以简化很多操作。

## 代码：

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <set>
#include <algorithm>
#include <cmath>
// #include <unordered_map>
#define N 500010
// #define ID(x, y) ((x)*m+(y))
// #define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt", "r", stdin);
        //freopen("D:/my.txt", "w", stdout);
    #endif // ONLINE_JUDGE
```

```

}
set<int> S;
vector<int> G[N];
int n, m;
int st[N], ed[N], fa[N], Tn;
void dfs(int u, int p = -1)
{
    st[u] = ++Tn;
    for(int i=0; i<G[u].size(); i++)
    {
        int v = G[u][i];
        if(v == p) continue;
        fa[v] = u;
        dfs(v, u);
    }
    ed[u] = Tn;
    if(ed[u] == st[u]) S.insert(st[u]);
}
int main()
{
    Open();
    scanf("%d", &n);
    for(int i=1; i<n; i++)
    {
        int x, y;
        scanf("%d%d", &x, &y);
        G[x].push_back(y); G[y].push_back(x);
    }
    dfs(1);
    scanf("%d", &m);
    while(m--)
    {
        int ty, u;
        scanf("%d%d", &ty, &u);
        if(ty == 1){
            set<int>::iterator it = S.lower_bound(st[fa[u]]);
            if(fa[u] && it != S.end() && *it <= ed[u])
S.insert(st[fa[u]]);
            S.erase(S.lower_bound(st[u]), S.upper_bound(ed[u]));
        }else if(ty == 2){
            S.insert(st[u]);
        }else{
            set<int>::iterator it = S.lower_bound(st[u]);
            if(it != S.end() && *it <= ed[u]) printf("0\n");
            else printf("1\n");
        }
    }
    return 0;
}

```

来自 <<http://acm.hust.edu.cn/vjudge/contest/viewSource?cid=4165570>>

# DFS 序线段树 : CodeForces 384 E

2015 年 7 月 27 日  
15:39

## E. Propagating tree

Iahub likes trees very much. Recently he discovered an interesting tree named propagating tree. The tree consists of  $n$  nodes numbered from 1 to  $n$ , each node  $i$  having an initial value  $a_i$ . The root of the tree is node 1.

This tree has a special property: when a value  $val$  is added to a value of node  $i$ , the value  $-val$  is added to values of all the children of node  $i$ . Note that when you add value  $-val$  to a child of node  $i$ , you also add  $-(-val)$  to all children of the child of node  $i$  and so on. Look an example explanation to understand better how it works.

This tree supports two types of queries:

- "1  $x$   $val$ " —  $val$  is added to the value of node  $x$ ;
- "2  $x$ " — print the current value of node  $x$ .

In order to help Iahub understand the tree better, you must answer  $m$  queries of the preceding type.

### Input

The first line contains two integers  $n$  and  $m$  ( $1 \leq n, m \leq 200000$ ). The second line contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 1000$ ). Each of the next  $n-1$  lines contains two integers  $v_i$  and  $u_i$  ( $1 \leq v_i, u_i \leq n$ ), meaning that there is an edge between nodes  $v_i$  and  $u_i$ .

Each of the next  $m$  lines contains a query in the format described above. It is guaranteed that the following constraints hold for all queries:  $1 \leq x \leq n, 1 \leq val \leq 1000$ .

### Output

For each query of type two (print the value of node  $x$ ) you must print the answer to the query on a separate line. The queries must be answered in the order given in the input.

### 题意 :

树上有两种操作, 1 x val 表示给 x 节点上的值+val, 给 x 的儿子上的值-val, 给 x 的儿子的儿子上的值-(-val), 2 x 查询 x 节点上的值。

## 做法：

这个题居然是我以前做过的。。利用 DFS 序给树上的节点重新排序，使得奇数层的节点在前面，偶数层的节点在后面，那么对一个节点的子树的更新，可以拆为更新两个区间。再预处理一下更新每个节点分别需要更新那些区间即可。

## 代码：

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <algorithm>
#define N 200020
using namespace std;
struct ee
{
    int x1,y1,x2,y2;
}e[N];
vector<int> g[N];
int n,m,a[N],d[N],c[N],bef[N],index=1;
void dfs1(int x,int fa,int deep)
{
    d[x]=deep;
    for(int i=0;i<g[x].size();i++)
        if(g[x][i]!=fa) dfs1(g[x][i],x,1-deep);
}
void dfs2(int x,int fa,int deep)
{
    if(d[x]==deep) bef[x]=index++; //aft[index++]=x;
    for(int i=0;i<g[x].size();i++)
        if(g[x][i]!=fa) dfs2(g[x][i],x,deep);
}
void dfs3(int x,int fa)
{
    for(int i=0;i<g[x].size();i++)
        if(g[x][i]!=fa) dfs3(g[x][i],x);
    int ma1=bef[x],mi2=N,ma2=0;
    for(int i=0;i<g[x].size();i++)
    {
        if(g[x][i]==fa) continue;
        int cur=g[x][i];
        mi2=min(mi2,e[cur].x1);
        ma2=max(ma2,e[cur].y1);
        ma1=max(ma1,e[cur].y2);
    }
    e[x].x1=bef[x],e[x].y1=ma1,e[x].x2=mi2,e[x].y2=ma2;
}
int getnum(int x)
{
    int rnt=0;
    for(int i=x;i<=n;i+=(i&(-i)))
    {
        rnt+=c[i];
    }
    return rnt;
}
```

```

void add(int i,int a)
{
    while(i>=1)
    {
        c[i]+=a;
        i--=(i&(-i));
    }
}
int main()
{
    #ifndef ONLINE_JUDGE
        //freopen("D:/in.txt","r",stdin);
    #endif // ONLINE_JUDGE
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) scanf("%d",a+i);
    for(int i=0;i<n-1;i++)
    {
        int a,b;
        scanf("%d%d",&a,&b);
        g[a].push_back(b),g[b].push_back(a);
    }
    dfs1(1,0,1);
    dfs2(1,0,1);
    for(int i=0;i<g[1].size();i++)
        dfs2(g[1][i],1,0);
    dfs3(1,0);
    //    for(int i=1;i<=n;i++)
    //    {
    //        cout<<i<<": "<<e[i].x1<<" "<<e[i].y1<<" "<<e[i].x2<<"
    //        "<<e[i].y2<<endl;
    //    }
    while(m-->0)
    {
        int ty;
        scanf("%d",&ty);
        if(ty==1)
        {
            int x,y;
            scanf("%d%d",&x,&y);
            int l1=e[x].x1,r1=e[x].y1,l2=e[x].x2,r2=e[x].y2;
            add(r1,y),add(l1-1,-y);
            if(r2!=0) add(r2,-y),add(l2-1,y);
        }
        else
        {
            int x;
            scanf("%d",&x);
            cout<<getnum(bef[x])+a[x]<<endl;
        }
    }
    return 0;
}

```



# 基尔霍夫矩阵

2015 年 7 月 24 日  
1:38

基尔霍夫矩阵为图  $G$  的度矩阵-邻接矩阵

义满足刚才描述的性质。

- 有了Kirchhoff矩阵这个工具，我们可以引入  
**Matrix-Tree**定理：

- 对于一个无向图 $G$ ，它的生成树个数等于其Kirchhoff矩阵任何一个 $n-1$ 阶主子式的行列式的绝对值。
- 所谓 $n-1$ 阶主子式，就是对于任意一个 $r$ ，将 $C$ 的第 $r$ 行和第 $r$ 列同时删去后的新矩阵，用 $C_r$ 表示。

## 数位 DP-D. Nass Number

2015 年 4 月 5 日  
19:53

时间限制 1000 ms 内存限制 65536 KB

### 题目描述

We define two functions under base 10.  $f(x)$  indicates the number which is formed by operating bitwise exclusive or(xor) with each digit of  $x$ , and  $g(x)$  indicates the sum of all these digits. For instance,  $f(47019)=4 \text{ xor } 7 \text{ xor } 0 \text{ xor }$

1 xor 9=11,  $g(47019)=4+7+0+1+9=21$ . The Nass Number of  $x$  is defined to be an integer  $y$ , which satisfies the conditions below:

1.  $0 \leq y < x$ ;
2.  $f(y) \leq f(x)$ ;
3. Define Set  $S=\{y_0, y_1, \dots, y_k\}$ , where  $y_i$  satisfies the two conditions above. The Nass Number  $y \in S$ , meanwhile  $g(y) \geq g(y_i)$  for  $i$  from 0 to  $k$ . If there exists another  $y'$  so that  $g(y')=g(y)$ , it should be satisfied that  $y > y'$ .

Given  $x$ , you're required to answer  $Nass(x)$ .

## 输入格式

The first line contains an integer  $T(T \leq 100)$ , indicating the number of test cases.

The following  $T$  lines, each with an integer  $x(1 \leq x \leq 2 \times 10^9)$ , give the number that you are to calculate.

## 输出格式

Output  $Nass(x)$  for each given  $x$ .

## 输入样例

```
3
1
9
11
```

## 输出样例

```
0
8
0
```

来自 <<http://code.bupt.edu.cn/problem/contest/183/problem/D/>>

```
#include<bits/stdc++.h>
using namespace std;
const int INF=INT_MAX;
const int MAXN=20;
int ans, ansg, nf, n;
int b[MAXN];
void dfs(int pos, int f, int g, int x, bool v){
    if(g+pos*9<=ansg) return;
    if(pos==0){
        if(x<n&&f<=nf&&g>ansg){
            ans=x;
            ansg=g;
        }
    }
}
```

```

    }
    return;
}
for(int flag=true, i=v?b[pos-1]:9;i>=0;i--){
    dfs(pos-1,f^i,g+i,x*10+i,v&&flag);
    flag=false;
}
}
int main(){
    int t; cin>>t;
    for(int cs=1; cs<=t; cs++){
        ans=0; ansg=0;
        scanf("%d",&n);
        int m=n, cnt=0;
        nf=0;
        while(m){
            b[cnt]=m%10;
            nf^=b[cnt++];
            m/=10;
        }
        for(int i=b[cnt-1], flag=true; i>=0; i--){
            dfs(cnt-1,i,i,i,flag);
            flag=false;
        }
        printf("%d\n",ans);
    }
    return 0;
}

```

来自 <<http://paste.ubuntu.com/10743561/>>

## 数位 DP-B. Claire\_ 's problem

2015 年 4 月 5 日  
19:55

时间限制 5000 ms 内存限制 65536 KB

### 题目描述

As we all know,Claire\_ and ykwd are famous ACMers in BUPT ACM Group.They always play games together.Nowadays they play such a game:at first,Claire\_ writes down a number,and then she selects a special digit(from 0 to 9),she removes all the

special digits, then she get a group of non-special digits. Claire\_ can use the group of non-special digits and limitless special digit to create new numbers. She is interested in how many positive numbers she create are less than the former number. She asks ykwd to solve this problem as soon as possible, otherwise ykwd would stand on his knees on the keyboard. Be notice leading zeros are not allowed.

## 输入格式

At first line, there is a number  $T$  which is the number of cases;

For each case, there is a number  $N$  ( $length \leq 1000$ ) and a digit (from 0 to 9).

## 输出格式

For each case, you should print the answer in one line. Because the answer can be so large that you should print the answer mod 20130303.

## 输入样例

```
3
1020 0
1020 1
2110 1
```

## 输出样例

```
7
2
13
```

来自 <<http://code.bupt.edu.cn/problem/contest/183/problem/B/>>

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN=1005;
const int MOD=20130303;
char s[MAXN];
long long C[MAXN][MAXN];
int cnt[10], b[MAXN];
long long comb(int x, int y) {
    if(y>x || y<0) return 0;
    if(x==y || y==0) return 1;
    return C[x][y];
}
void init() {
    memset(C, 0, sizeof C);
    for(int i=0; i<MAXN; i++) {
        C[i][0]=1;
        for(int j=1; j<=i; j++) {
            C[i][j]=(C[i-1][j]+C[i-1][j-1])%MOD;
        }
    }
}
```

```

    }
}
}
int solve(int now) {
    long long ret=1;
    for(int i=0; i<10; i++) if(cnt[i]) {
        ret=ret*comb(now, cnt[i])%MOD;
        now-=cnt[i];
    }
    return ret;
}
int d;
int calc(int now, int x) {
    long long ret=0;
    bool flag=true;
    for(int i=0; i<10; i++) {
        if(i==d&&x==0) continue;
        if(i!=d&&cnt[i]==0) continue;
        if(flag) ret=1, flag=false;
        if(i!=d) {
            ret=ret*comb(now-(i==0), cnt[i])%MOD;
            now-=cnt[i];
        }
        else {
            ret=ret*comb(now-(i==0), x)%MOD;
            now-=x;
        }
    }
    return ret;
}
int n;
int main() {
    init();
    int t; cin>>t;
    while(t--) {
        scanf("%s%d", s, &d);
        n=strlen(s);
        memset(cnt, 0, sizeof cnt);
        int sum=0;
        for(int i=0; i<n; i++) {
            b[i]=s[i]-'0';
            cnt[b[i]]++;
            if(b[i]!=d) sum++;
        }
        long long ans=0;
        for(int i=0; i<cnt[d]; i++) {
            ans+=calc(sum+i, i);
            ans%=MOD;
        }
        sum+=cnt[d];
        for(int i=0; i<n; i++) {

```

```

        for(int j=i?0:1; j<b[i]; j++){
            if(!cnt[j]) continue;
            cnt[j]--;
            ans+=solve(sum-1-i);
            ans%=MOD;
            cnt[j]++;
        }
        cnt[b[i]]--;
    }
    cout<<ans<<endl;
}
return 0;
}

```

来自 <http://paste.ubuntu.com/10743588/>

## 威佐夫博弈

2015 年 10 月 21 日  
0:53

具体内容：[编辑](#)

威佐夫博弈（Wythoff Game）：有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。

这种情况下是颇为复杂的。我们用  $(a_k, b_k)$  ( $a_k \leq b_k, k=0, 1, 2, \dots, n$ ) 表示两堆物品的数量并称其为局势，如果甲面对  $(0, 0)$ ，那么甲已经输了，这种局势我们称为奇异局势。前几个奇异局势是： $(0, 0)$ 、 $(1, 2)$ 、 $(3, 5)$ 、 $(4, 7)$ 、 $(6, 10)$ 、 $(8, 13)$ 、 $(9, 15)$ 、 $(11, 18)$ 、 $(12, 20)$ 。

可以看出， $a_0=b_0=0$ ， $a_k$  是未在前面出现过的最小自然数，而  $b_k = a_k + k$ 。

奇异局势有如下性质：[编辑](#)

1. 任何自然数都包含在一个且仅有一个奇异局势中。

由于  $a_k$  是未在前面出现过的最小自然数，所以有  $a[k] > a[k-1]$ ，而  $b_k = a[k] + k > a[k-1] + k > a[k-1] + k - 1 = b[k-1] > a[k-1]$ 。所以性质 1 成立。

2. 任意操作都可将奇异局势变为非奇异局势。

事实上，若只改变奇异局势  $(a_k, b_k)$  的某一个分量，那么另一个分量不可能在其他奇异局势中，所以必然是非奇异局势。如果使  $(a_k, b_k)$  的两个分量同时减少，则由于其差不变，且不可能是其他奇异局势的差，因此也是非奇异局势。

3. 采用适当的方法，可以将非奇异局势变为奇异局势。

假设面对的局势是  $(a, b)$ ，若  $b = a$ ，则同时从两堆中取走  $a$  个物体，就变为了奇异局势  $(0, 0)$ ；如果  $a = a_k$ ， $b > b_k$  那么，取走  $b - b_k$  个物体，即变为奇异局势；如果  $a = a_k$ ， $b < b_k$  则同时从两堆中拿走  $a - a[b-a]$  个物体变为奇异局势  $(a[b-a], b-a+a[b-a])$ ；如果  $a > a_k$ ， $b = a_k + k$  则从第一堆中拿走多余的数量  $a - a_k$  即可；如果  $a < a_k$ ， $b = a_k + k$ ，分两种情况，第一种， $a = a_j$  ( $j < k$ ) 从第二堆里面拿走  $b - b_j$  即可；第二种， $a = b_j$  ( $j < k$ ) 从第二堆里面拿走  $b - a_j$  即可。

来自 <[http://baike.baidu.com/link?url=LhI1tkaimSiB\\_BD0tXtn03dCXchi7KJ-wwWF1IrlkzA24PKWgk4zA3xgHtoir72dF21o98RJvov-61EYXWqZ1a](http://baike.baidu.com/link?url=LhI1tkaimSiB_BD0tXtn03dCXchi7KJ-wwWF1IrlkzA24PKWgk4zA3xgHtoir72dF21o98RJvov-61EYXWqZ1a)>