ACM Template

UESTC_Lasagne

Last build at March 29, 2013

Contents

1 注意事项

输入输出格式？调试信息？初始化？算术溢出？数组大小？

左右端点范围？acos/asin/sqrt 函数定义域？精度问题？

二分答案？暴力？单调性？凸性？块状结构？函数式？对偶问题？

排序的时候注意一下是否需要记录排序前的位置！

使用 map 进行映射的时候，不要用下面这种不安全写法

```
1 if (mp.find(s) == mp.end())
2   mp[s] = mp.size()-1;//挂成狗
3
4 if (mp.find(s) == mp.end())
5 {
6   int tmp = mp.size();
7   mp[s] = tmp;//正确
8 }
```

10^6 数量级慎用后缀数组

TLE 的时候要冷静哟。。

思考的时候结合具体步骤来的话会体会到一些不同的东西

C++ 与 G++ 是很不一样的。。。

map 套字符串是很慢的。。。

栈会被记录内存。。。

浮点数最短路要注意取 $\leq$ 来判断更新。。。

注意 long long

不要相信.size()

重复利用数组时小心数组范围

先构思代码框架每当实际拍马框架变化时停手重新思考

有时候四边形不等式也是帮得上忙的 dp 优化是可以水的

结构体里面带数组会非常慢, 有时候 BFS 把数组压成数字会快很多。

```
1 void fun(int a[])
2 {
3   printf("%d\n",sizeof(a));
4 }
```

结果是 sizeof(a[0]), 如果传数组指针然后要清空的话不要用 sizeof。

sqrt 某些时候会出现 sqrt(-0.00) 的问题。

将 code::blocks 的默认终端改成 gnome-terminal

```
1 gnome-terminal -t $TITLE -x
```

最小割割集找法在残量网络中从源点出发能到的点集记为 S 原图中 S 到 S' 的边即是最小割集

double 全局变量初始值可能不是 0

2 字符串处理

2.1 *AC 自动机

2.1.1 指针

```
const int CHAR=26;
const int TOTLEN=500000;
const int MAXLEN=1000000;
struct Vertex
{
  Vertex *fail,*next[CHAR];
  Vertex(){}
  Vertex(bool flag)//为什么要这样写?
  {
    fail=0;
    memset(next,0,sizeof(next));
  }
};
int size;
Vertex vertex[TOTLEN+1];
void init()
{
  vertex[0]=Vertex(0);
  size=1;
}
void add(Vertex *pos,int cha)
{
  vertex[size]=Vertex(0);
  pos->next[cha]=&vertex[size++];
}
void add(vector<int> s)
{
  int l=s.size();
  Vertex *pos=&vertex[0];
  for (int i=0; i<l; i++)
  {
    if (pos->next[s[i]]==NULL)
      add(pos,s[i]);
    pos=pos->next[s[i]];
  }
}
void bfs()
{
  queue<Vertex *> que;
  Vertex *u=&vertex[0];
  for (int i=0; i<CHAR; i++)
    if (u->next[i]!=NULL)
    {
      que.push(u->next[i]);
      u->next[i]->fail=u;
    }
```

```
47      else
48        u->next[i]=u;
49    u->fail=NULL;
50    while (!que.empty())
51    {
52      u=que.front();
53      que.pop();
54      for (int i=0; i<CHAR; i++)
55        if (u->next[i]!=NULL)
56        {
57          que.push(u->next[i]);
58          u->next[i]->fail=u->fail->next[i];
59        }
60        else
61          u->next[i]=u->fail->next[i];
62    }
63  }
```

2.1.2 非指针

```
 1  struct Trie
 2  {
 3    int next[50][10],fail[50];
 4    bool end[50];
 5    int L,root;
 6
 7    int newNode()
 8    {
 9      for (int i = 0;i < 10;i++)
10        next[L][i] = -1;
11      end[L] = false;
12      return L++;
13    }
14
15    void Init()
16    {
17      L = 0;
18      root = newNode();
19    }
20
21    void Insert(char s[])
22    {
23      int now = root;
24      for (int i = 0;s[i] != 0;i++)
25      {
26        if (next[now][s[i]-'0'] == -1)
27          next[now][s[i]-'0'] = newNode();
28        now = next[now][s[i]-'0'];
29      }
30      end[now] = true;
31    }
32
```

```
33    void Build()
34    {
35      queue<int> Q;
36      for (int i = 0;i < 10;i++)
37        if (next[root][i] == -1)
38          next[root][i] = root;
39        else
40        {
41          fail[next[root][i]] = root;
42          Q.push(next[root][i]);
43        }
44      while (!Q.empty())
45      {
46        int now = Q.front();
47        Q.pop();
48        end[now] |= end[fail[now]];
49        for (int i = 0;i < 10;i++)
50          if (next[now][i] == -1)
51            next[now][i] = next[fail[now]][i];
52          else
53          {
54            fail[next[now][i]] = next[fail[now]][i];
55            Q.push(next[now][i]);
56          }
57      }
58    }
59  };
```

2.2 后缀数组

2.2.1 DC3

所有下标都是 0 n-1，height[0] 无意义。

```
1  //所有相关数组都要开三倍
2  const int maxn = 300010;
3  # define F(x) ((x)/3+((x)%3==1?0:tb))
4  # define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
5  int wa[maxn * 3], wb[maxn * 3], wv[maxn * 3], ws[maxn * 3];
6  int c0(int *r, int a, int b)
7  {
8    return
9    r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
10 }
11 int c12(int k, int *r, int a, int b)
12 {
13   if (k == 2)
14     return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b + 1);
15   else return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];
16 }
17 void sort(int *r, int *a, int *b, int n, int m)
18 {
19   int i;
```

```
20    for (i = 0; i < n; i++) wv[i] = r[a[i]];
21    for (i = 0; i < m; i++) ws[i] = 0;
22    for (i = 0; i < n; i++) ws[wv[i]]++;
23    for (i = 1; i < m; i++) ws[i] += ws[i - 1];
24    for (i = n - 1; i >= 0; i--) b[--ws[wv[i]]] = a[i];
25    return;
26  }
27  void dc3(int *r, int *sa, int n, int m)
28  {
29    int i, j, *rn = r + n;
30    int *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
31    r[n] = r[n + 1] = 0;
32    for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
33    sort(r + 2, wa, wb, tbc, m);
34    sort(r + 1, wb, wa, tbc, m);
35    sort(r, wa, wb, tbc, m);
36    for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
37      rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
38    if (p < tbc) dc3(rn, san, tbc, p);
39    else for (i = 0; i < tbc; i++) san[rn[i]] = i;
40    for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i] * 3;
41    if (n % 3 == 1) wb[ta++] = n - 1;
42    sort(r, wb, wa, ta, m);
43    for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
44    for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
45      sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
46    for (; i < ta; p++) sa[p] = wa[i++];
47    for (; j < tbc; p++) sa[p] = wb[j++];
48  }
49  //str 和 sa 也要三倍
50  void da(int str[],int sa[],int rank[],int height[],int n,int m)
51  {
52    for (int i = n; i < n * 3; i++)
53      str[i] = 0;
54    dc3 (str , sa , n + 1 , m);
55    int i, j, k;
56    for (i = 0; i < n; i++)
57    {
58      sa[i] = sa[i + 1];
59      rank[sa[i]] = i;
60    }
61    for (i = 0, j = 0, k = 0; i < n; height[rank[i ++]] = k)
62      if (rank[i] > 0)
63        for (k ? k-- : 0 , j = sa[rank[i] - 1];
64          i + k < n && j + k < n && str[i + k] == str[j + k];
65          k++);
66  }
```

2.2.2 DA

这份似乎就没啥要注意的了。

```
const int maxn = 200010;
int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];

bool cmp(int *r,int n,int a,int b,int l)
{
  return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
}
void da(int str[],int sa[],int rank[],int height[],int n,int m)
{
  int *s = str;
  int *x=wx,*y=wy,*t,p;
  int i,j;
  for(i=0; i<m; i++)wss[i]=0;
  for(i=0; i<n; i++)wss[x[i]=s[i]]++;
  for(i=1; i<m; i++)wss[i]+=wss[i-1];
  for(i=n-1; i>=0; i--)sa[--wss[x[i]]]=i;
  for(j=1,p=1; p<n && j<n; j*=2,m=p)
  {
    for(i=n-j,p=0; i<n; i++)y[p++]=i;
    for(i=0; i<n; i++)if(sa[i]-j>=0)y[p++]=sa[i]-j;
    for(i=0; i<n; i++)wv[i]=x[y[i]];
    for(i=0; i<m; i++)wss[i]=0;
    for(i=0; i<n; i++)wss[wv[i]]++;
    for(i=1; i<m; i++)wss[i]+=wss[i-1];
    for(i=n-1; i>=0; i--)sa[--wss[wv[i]]]=y[i];
    for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
      x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
  }
  for(int i=0; i<n; i++) rank[sa[i]]=i;
  for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
    if(rank[i]>0)
      for(k?k--:0,j=sa[rank[i]-1];
        i+k < n && j+k < n && str[i+k]==str[j+k];
        k++);
}
```

2.2.3 调用

注意几个数组的下标是不同的

```
char s[maxn];
int str[maxn],sa[maxn],rank[maxn],height[maxn];

int main()
{
  scanf("%s",s);
  int len = strlen(s);
  for (int i = 0;i <= len;i++)
    str[i] = s[i];
  da(str,sa,rank,height,len,128);

  for (int i = 0;i < len;i++)
```

```
13    {
14      printf("sa= %d,height= %d,s= %s\n",sa[i],height[i],s+sa[i]);
15    }
16    return 0;
17  }
```

2.2.4 最长公共前缀

记得不要忘记调用 lcpinit !

```
 1  int f[maxn][20];
 2  int lent[maxn];
 3  void lcpinit()
 4  {
 5    int i,j;
 6    int n = len,k = 1,l = 0;
 7    for (i = 0; i < n; i++)
 8    {
 9      f[i][0] = height[i];
10      if (i+1 > k*2)
11      {
12        k *= 2;
13        l++;
14      }
15      lent[i+1] = l;
16    }
17    for (j = 1; (1<<j)-1<n; j++)
18      for (i = 0; i+(1<<j)-1<n; i++)
19        f[i][j] = min(f[i][j-1],f[i+(1<<(j-1))][j-1]);
20  }
21  int lcp(int x,int y)
22  {
23    if (x > y)  swap(x,y);
24    if (x == y)
25      return x-sa[x];//自己和自己当然是自己的长度啦lcp
26    x++;
27    int k = lent[y-x+1];
28    return min(f[x][k],f[y-(1<<k)+1][k]);
29  }
```

2.2.5 最长公共前缀大于等于某个值的区间

```
1  void getinterv(int pos,int comlen,int& pl,int& pr)
2  {
3    int l,r,mid,cp;
4    l = 0;
5    r = pos;
6    while (l < r)
7    {
8      mid = l+r>>1;
9      cp = lcp(mid,pos);
```

```
10      if (cp < comlen)
11        l = mid+1;
12      else
13        r = mid;
14    }
15    pl = l;
16
17    l = pos;
18    r = len-1;
19    while (l < r)
20    {
21      mid = l+r+1>>1;
22      cp = lcp(pos,mid);
23      if (cp < comlen)
24        r = mid-1;
25      else
26        l = mid;
27    }
28    pr = l;
29  }
```

2.3 KMP

求 A[0..i] 的一个后缀最多能匹配 B 的前缀多长。先对 B 进行自匹配然后与 A 匹配。KMP[i] 就是对应答案，p[i]+1 是 B[0..i] 的一个后缀最多能匹配 B 的前缀多长。

```
1  //自匹配过程
2  int j;
3  p [0] = j = -1;
4  for ( int i = 1; i < lb; i++)
5  {
6    while (j >= 0 && b[j + 1] != b[i]) j = p[j];
7    if (b[j + 1] == b[i]) j ++;
8    p[i] = j;
9  }
10 //下面是匹配过程
11 j = -1;
12 for ( int i = 0; i < la; i++)
13 {
14   while (j >= 0 && b[j + 1] != a[i]) j = p[j];
15   if (b[j + 1] == a[i]) j ++;
16   KMP[i] = j + 1;
17 }
```

2.4 Manacher

```
1  const int maxn = 110000;
2
3  char Ma[maxn*2];
4  int Mp[maxn*2];
5  void Manacher(char s[],int len)
6  {
```

```
  int l = 0;
  Ma[l++] = '.';
  Ma[l++] = ',';
  for (int i = 0;i < len;i++)
  {
    Ma[l++] = s[i];
    Ma[l++] = ',';
  }
  Ma[l] = 0;
  int pnow = 0,pid = 0;
  for (int i = 1;i < l;i++)
  {
    if (pnow > i)
      Mp[i] = min(Mp[2*pid-i],pnow-i);
    else
      Mp[i] = 1;
    for (;Ma[i-Mp[i]] == Ma[i+Mp[i]];Mp[i]++);
    if (i+Mp[i] > pnow)
    {
      pnow = i+Mp[i];
      pid = i;
    }
  }
}
/*
abaaba
 .  ,  a  ,  b  ,  a  ,  a  ,  b  ,  a  ,
 0  1  2  1  4  1  2  7  2  1  4  1  2  1
*/
```

2.5 不同回文串

往 hash 表中插入新东西的时候就说明找到了一个新回文字串
一共 $O(n)$ 个

```
typedef unsigned int uint;

const int maxn = 110000;

char Ma[maxn*2];
int Mp[maxn*2];
void Manacher(char s[],int len)
{
  int l = 0;
  Ma[l++] = '.';
  Ma[l++] = ',';
  for (int i = 0; i < len; i++)
  {
    Ma[l++] = s[i];
    Ma[l++] = ',';
  }
  Ma[l] = 0;
  int pnow = 0,pid = 0;
```

```
19    for (int i = 1; i < l; i++)
20    {
21      if (pnow > i)
22        Mp[i] = min(Mp[2*pid-i],pnow-i);
23      else
24        Mp[i] = 1;
25      for (; Ma[i-Mp[i]] == Ma[i+Mp[i]]; Mp[i]++);
26      if (i+Mp[i] > pnow)
27      {
28        pnow = i+Mp[i];
29        pid = i;
30      }
31    }
32  }
33
34  char s[maxn*2];
35  int len;
36  int p[maxn*2];
37  const int muts = 129;
38  uint sum[maxn];
39  uint mutpower[maxn];
40
41  struct hash_map
42  {
43    const static int mod = 300007;
44    int head[mod];
45    struct hash_tables
46    {
47      uint key1;
48      int key2;
49      int next;
50    } ele[maxn*10];
51    int N;
52    void init()
53    {
54      memset(head,-1,sizeof(head));
55      N = 0;
56    }
57    int totlen[mod];
58    void clear()
59    {
60      for (int i = 0; i < N; i++)
61        head[ele[i].key1%mod] = -1;
62      N = 0;
63    }
64    int find(uint x,int len)
65    {
66      int hashcode = x%mod;
67      for (int i = head[hashcode]; i != -1; i = ele[i].next)
68        if (ele[i].key1 == x && ele[i].key2 == len)
69          return i;
```

```
70      return -1;
71    }
72    void insert(uint x,int len)
73    {
74      int tmp = x%mod;
75      ele[N].key1 = x;
76      ele[N].key2 = len;
77      ele[N].next = head[tmp];
78      head[tmp] = N++;
79    }
80  };
81
82  hash_map hash;
83
84  uint gethashcode(int l,int r)
85  {
86    uint ret;
87    ret = sum[r];
88    if (l)
89      ret -= sum[l-1]*mutpower[r-l+1];
90    return ret;
91  }
92
93  int calc(char s[])
94  {
95    len = strlen(s);
96    Manacher(s,len);
97
98    sum[0] = s[0];
99    for (int i = 1; i < len; i++)
100     sum[i] = sum[i-1]*muts+s[i];
101
102   int res = 0;
103   uint tmp;
104   int nt = 0;
105   hash.clear();
106   //odd
107   for (int i = 0; i < len; i++)
108     if (Mp[i*2+2]%2 == 0)
109     {
110       int pl = Mp[i*2+2]/2;
111       if (i+pl < nt || pl == 0) continue;
112       for (int j = i-pl+1; j <= i; j++)
113       {
114         tmp = gethashcode(j,i);
115         if (hash.find(tmp,i-j+1) != -1) break;
116         hash.insert(tmp,i-j+1);
117       }
118       nt = i+pl;
119     }
120   res += hash.N;
```

```
121
122   nt = 0;
123   hash.clear();
124   //even
125   for (int i = 0; i < len; i++)
126     if (Mp[i*2+3] > 1)
127     {
128       int pl = Mp[i*2+3]/2;
129       if (i+pl < nt || pl == 0) continue;
130       for (int j = i-pl+1; j <= i; j++)
131       {
132         tmp = gethashcode(j,i);
133         if (hash.find(tmp,i-j+1) != -1) break;
134         hash.insert(tmp,i-j+1);
135       }
136       nt = i+pl;
137     }
138   res += hash.N;
139   return res;
140 }
141
142 int main()
143 {
144   mutpower[0] = 1;
145   for (int i = 1; i < maxn; i++)
146     mutpower[i] = mutpower[i-1]*muts;
147   hash.init();
148
149   int totcas;
150   scanf("%d",&totcas);
151   for (int cas = 1; cas <= totcas; cas++)
152   {
153     scanf("%s",s);
154
155     printf("Case #%d: %d\n",cas,calc(s));
156   }
157   return 0;
158 }
```

2.6 * 字符串最小表示法

```
1  int Gao(char a[],int len)
2  {
3    int i = 0,j = 1,k = 0;
4    while (i < len && j < len && k < len)
5    {
6      int cmp = a[(j+k)%len]-a[(i+k)%len];
7      if (cmp == 0)
8        k++;
9      else
10     {
```

```
11        if (cmp > 0)
12          j += k+1;
13        else
14          i += k+1;
15        if (i == j) j++;
16        k = 0;
17      }
18    }
19    return min(i,j);
20  }
```

2.7 带 * 通配符的匹配

```
 1  #include <iostream>
 2  #include <algorithm>
 3  #include <cstdio>
 4  #include <cstring>
 5  using namespace std;
 6
 7  char a[110],b[110],sp[110][110],tot,place[110];
 8  int n,la,lb,ll;
 9
10  bool check(int id,int pos)
11  {
12    for (int i = 0;sp[id][i] != 0;i++)
13      if (b[pos+i] != sp[id][i])
14        return false;
15    return true;
16  }
17
18  bool check()
19  {
20    lb = strlen(b);
21    int pre = 0;
22    for (int i = 0;i < tot;i++)
23    {
24      bool find = false;
25      for (int j = pre;j < lb;j++)
26        if (check(i,j) == true)
27        {
28          place[i] = j;
29          pre = place[i]+1;
30          find = true;
31          break;
32        }
33      if (find == false)  return false;
34    }
35    if (a[0] != '*')
36      if (place[0] != 0)
37        return false;
38    if (a[la-1] != '*')
39      if (check(tot-1,lb-ll) == false)
```

```
40        return false;
41    return true;
42  }
43
44  int main()
45  {
46    while (scanf("%s",a) != EOF)
47    {
48      tot = 0;
49      for (int i = 0;a[i] != 0;i++)
50        if (a[i] != '*')
51        {
52          int j;
53          for (j = i;a[j] != 0 && a[j] != '*';j++)
54            sp[tot][j-i] = a[j];
55          sp[tot++][j-i] = 0;
56          i = j;
57        }
58      la = strlen(a);
59      ll = strlen(sp[tot-1]);
60      scanf("%d",&n);
61      for (int i = 0;i < n;i++)
62      {
63        scanf("%s",b);
64        if (check() == true)
65          puts(b);
66      }
67    }
68    return 0;
69  }
70  /*
71  Sample Input 1
72  *.*
73  4
74  main.c
75  a.out
76  readme
77  yacc
78
79  Sample Input 2
80  *a*a*a
81  4
82  aaa
83  aaaaa
84  aaaaax
85  abababa
86
87  Sample Output 1
88  main.c
89  a.out
90
```

```
Sample Output 2
aaa
aaaaa
abababa
*/
```

3 数学

3.1 扩展 GCD

求 ax+by=gcd(a,b) 的一组解

```
long long ex_gcd(long long a,long long b,long long &x,long long &y)
{
  if (b)
  {
    long long ret = ex_gcd(b,a%b,x,y),tmp = x;
    x = y;
    y = tmp-(a/b)*y;
    return ret;
  }
  else
  {
    x = 1;
    y = 0;
    return a;
  }
}
```

3.2 模线性方程组

```
//有更新
int m[10],a[10];//模数m 余数a
bool solve(int &m0,int &a0,int m,int a)//模线性方程组
{
  int y,x;
  int g=ex_gcd(m0,m,x,y);
  if (abs(a-a0)%g) return 0;
  x*=(a-a0)/g;
  x%=m/g;
  a0=(x*m0+a0);
  m0*=m/g;
  a0%=m0;
  if (a0<0) a0+=m0;
  return 1;
}
int MLES()
{
  bool flag=1;
  int m0=1,a0=0;
  for (int i=0; i<n; i++)
    if (!solve(m0,a0,m[i],a[i]))
    {
      flag=0;
      break;
    }
  if (flag)
```

```
27      return a0;
28    else
29      return -1;
30  }
```

3.3 矩阵

乘法的时候将 B 数组转置一下然后 $C[i][j] = \sum A[i][k] \times B[j][k]$ 会有奇效。

```
1   struct Matrix
2   {
3     int a[52][52];
4     void clear()
5     {
6       memset(a,0,sizeof(a));
7     }
8     int det(int n)//求行列式的值模上一个数，需要预处理逆元
9     {
10      for (int i = 0;i < n;i++)
11        for (int j = 0;j < n;j++)
12          a[i][j] = (a[i][j]%mod+mod)%mod;
13      int res = 1;
14      for (int i = 0;i < n;i++)
15      {
16        for (int j = i;j < n;j++)
17          if (a[j][i] != 0)
18          {
19            for (int k = i;k < n;k++)
20              swap(a[i][k],a[j][k]);
21            if (i != j)
22              res = (res+mod)%mod;
23            break;
24          }
25        if (a[i][i] == 0)
26        {
27          res = -1;//不存在
28          break;
29        }
30        for (int j = i+1;j < n;j++)
31        {
32          int mut = (a[j][i]*inv[a[i][i]])%mod;
33          for (int k = i;k < n;k++)
34            a[j][k] = (a[j][k]-(a[i][k]*mut)%mod+mod)%mod;
35        }
36        res = (res*a[i][i])%mod;
37      }
38      return res;
39    }
40    Matrix operator * (const Matrix &b)const
41    {
42      Matrix res;
43      for (int i = 0; i < 52; i++)
44        for (int j = 0; j < 52; j++)
```

```
45        {
46          res.a[i][j] = 0;
47          for (int k = 0; k < 52; k++)
48            res.a[i][j] += a[i][k] * b.a[k][j];
49        }
50      return res;
51    }
52    Matrix operator ^ (int y)const
53    {
54      Matrix res, x;
55      for (int i = 0; i < 52; i++)
56      {
57        for (int j = 0; j < 52; j++)
58          res.a[i][j] = 0, x.a[i][j] = a[i][j];
59        res.a[i][i] = 1;
60      }
61      for (; y; y >>= 1, x = x * x)
62        if (y & 1)
63          res = res * x;
64      return res;
65    }
66  };
```

3.4 FFT

```
1  const double PI= acos(-1.0);
2  struct vir
3  {
4    double re,im; //实部和虚部
5    vir(double a=0,double b=0)
6    {
7      re=a;
8      im=b;
9    }
10   vir operator +(const vir &b)
11   {return vir(re+b.re,im+b.im);}
12   vir operator -(const vir &b)
13   {return vir(re-b.re, im-b.im);}
14   vir operator *(const vir &b)
15   {return vir(re*b.re-im*b.im , re*b.im+im*b.re);}
16 };
17 vir x1[200005],x2[200005];
18 void change(vir *x,int len,int loglen)
19 {
20   int i,j,k,t;
21   for(i=0;i<len;i++)
22   {
23     t=i;
24     for(j=k=0; j<loglen; j++,t>>=1)
25       k= (k<<1)|(t&1);
26     if(k<i)
```

```
27      {
28      //  printf("%d %d\n",k,i);
29        vir wt=x[k];
30        x[k]=x[i];
31        x[i]=wt;
32      }
33    }
34  }
35  void fft(vir *x,int len,int loglen)
36  {
37    int i,j,t,s,e;
38    change(x,len,loglen);
39    t=1;
40    for(i=0;i<loglen;i++,t<<=1)
41    {
42      s=0;
43      e=s+t;
44      while(s<len)
45      {
46        vir a,b,wo(cos(PI/t),sin(PI/t)),wn(1,0);
47        for(j=s;j<s+t;j++)
48        {
49          a=x[j];
50          b=x[j+t]*wn;
51          x[j]=a+b;
52          x[j+t]=a-b;
53          wn=wn*wo;
54        }
55        s=e+t;
56        e=s+t;
57      }
58    }
59  }
60  void dit_fft(vir *x,int len,int loglen)
61  {
62    int i,j,s,e,t=1<<loglen;
63    for(i=0;i<loglen;i++)
64    {
65      t>>=1;
66      s=0;
67      e=s+t;
68      while(s<len)
69      {
70        vir a,b,wn(1,0),wo(cos(PI/t),-sin(PI/t));
71        for(j=s;j<s+t;j++)
72        {
73          a=x[j]+x[j+t];
74          b=(x[j]-x[j+t])*wn;
75          x[j]=a;
76          x[j+t]=b;
77          wn=wn*wo;
```

```
78        }
79        s=e+t;
80        e=s+t;
81      }
82    }
83    change(x,len,loglen);
84    for(i=0;i<len;i++)
85      x[i].re/=len;
86  }
87  int main()
88  {
89    char a[100005],b[100005];
90    int i,len1,len2,len,loglen;
91    int t,over;
92    while(scanf("%s%s",a,b)!=EOF)
93    {
94      len1=strlen(a)<<1;
95      len2=strlen(b)<<1;
96      len=1;loglen=0;
97      while(len<len1)
98      {
99        len<<=1;  loglen++;
100     }
101     while(len<len2)
102     {
103       len<<=1;  loglen++;
104     }
105     for(i=0;a[i];i++)
106     {
107       x1[i].re=a[i]-'0';
108       x1[i].im=0;
109     }
110     for(;i<len;i++)
111       x1[i].re=x1[i].im=0;
112     for(i=0;b[i];i++)
113     {
114       x2[i].re=b[i]-'0';
115       x2[i].im=0;
116     }
117     for(;i<len;i++)
118       x2[i].re=x2[i].im=0;
119     fft(x1,len,loglen);
120     fft(x2,len,loglen);
121     for(i=0;i<len;i++)
122       x1[i] = x1[i]*x2[i];
123     dit_fft(x1,len,loglen);
124     for(i=(len1+len2)/2-2,over=len=0;i>=0;i--)
125     {
126       t=(int)(x1[i].re+over+0.5);
127       a[len++]= t%10;
128       over = t/10;
```

```
129     }
130     while(over)
131     {
132       a[len++]=over%10;
133       over/=10;
134     }
135     for(len--;len>=0&&!a[len];len--);
136       if(len<0)
137       putchar('0');
138       else
139         for(;len>=0;len--)
140           putchar(a[len]+'0');
141     putchar('\n');
142   }
143   return 0;
144 }
```

3.5 分解质因数

3.5.1 米勒拉宾 + 分解因数

```
1  #include<ctime>
2  #include<iostream>
3  #define bint long long
4  using namespace std;
5  const int TIME = 8;//测试次数，够了8~10
6  int factor[100],fac_top = -1;
7
8  //计算两个数的gcd
9  bint gcd(bint small,bint big)
10 {
11   while(small)
12   {
13     swap(small,big);
14     small%=big;
15   }
16   return abs(big);
17 }
18
19 //ret = (a*b)%n (n<2^62)
20 bint muti_mod(bint a,bint b,bint n)
21 {
22   bint exp = a%n, res = 0;
23   while(b)
24   {
25     if(b&1)
26     {
27       res += exp;
28       if(res>n) res -= n;
29     }
30     exp <<= 1;
31     if (exp>n) exp -= n;
```

```
32      b>>=1;
33    }
34    return res;
35  }
36
37  // ret = (a^b)%n
38  bint mod_exp(bint a,bint p,bint m)
39  {
40    bint exp=a%m, res=1; //
41    while(p>1)
42    {
43      if(p&1)
44        res=muti_mod(res,exp,m);
45      exp = muti_mod(exp,exp,m);
46      p>>=1;
47    }
48    return muti_mod(res,exp,m);
49  }
50
51  //miller-法测试素数rabin, time 测试次数
52  bool miller_rabin(bint n, int times)
53  {
54    if(n==2)return 1;
55    if(n<2||!(n&1))return 0;
56    bint a, u=n-1, x, y;
57    int t=0;
58    while(u%2==0)
59    {
60      t++;
61      u/=2;
62    }
63    srand(time(0));
64    for(int i=0; i<times; i++)
65    {
66      a = rand() % (n-1) + 1;
67      x = mod_exp(a, u, n);
68      for(int j=0; j<t; j++)
69      {
70        y = muti_mod(x, x, n);
71        if ( y == 1 && x != 1 && x != n-1 )
72          return false; //must not
73        x = y;
74      }
75      if( y!=1) return false;
76    }
77    return true;
78  }
79
80  bint pollard_rho(bint n,int c)//找出一个因子
81  {
82    bint x,y,d,i = 1,k = 2;
```

```
  srand(time(0));
  x = rand()%(n-1)+1;
  y = x;
  while(true)
  {
    i++;
    x = (muti_mod(x,x,n) + c) % n;
    d = gcd(y-x, n);
    if (1 < d && d < n) return d;
    if( y == x) return n;
    if(i == k)
    {
      y = x;
      k <<= 1;
    }
  }
}

void findFactor(bint n,int k)//二分找出所有质因子，存入factor
{
  if(n==1)return;
  if(miller_rabin(n, TIME))
  {
    factor[++fac_top] = n;
    return;
  }
  bint p = n;
  while(p >= n)
    p = pollard_rho(p,k--);//值变化，防止死循环k
  findFactor(p,k);
  findFactor(n/p,k);
}

int main()
{
  bint cs,n,min;
  cin>>cs;
  while (cs--)
  {
    cin>>n;
    fac_top = min = -1;
    if(miller_rabin(n,TIME)) cout<<"Prime"<<endl;
    else
    {
      findFactor(n,107);
      for(int i=0; i<=fac_top; i++)
      {
        if(min<0||factor[i]<min)
          min = factor[i];
      }
      cout<<min<<endl;
```

```
134     }
135   }
136   return 0;
137 }
```

3.5.2 暴力版本

```
 1 int N;
 2 int num[30],fac[30];
 3 void getFactor(int x)
 4 {
 5   N=0;
 6   memset(num,0,sizeof(num));
 7   for (int i=0; prime[i]*prime[i]<=x && i<L; i++)
 8   {
 9     if (x%prime[i]==0)
10     {
11       while (x%prime[i]==0)
12       {
13         x/=prime[i];
14         num[N]++;
15       }
16       fac[N++]=prime[i];
17     }
18   }
19   if (x>1)
20   {
21     num[N]=1;
22     fac[N++]=x;
23   }
24 }
```

3.6 逆元

```
 1 void getInv2(int x)
 2 {
 3   inv[1]=1;
 4   for (int i=2; i<=x; i++)
 5     inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
 6 }
 7 int getInv(int x)//为素数mod
 8 {
 9   return power(x,mod-2);
10 }
```

3.7 卢卡斯

卢卡斯， $num[i]$ 阶乘也

```
 1 int comLucus(int n,int m,int p)
 2 {
 3   int ans=1;
 4   for (; n && m && ans; n/=p,m/=p)
 5   {
```

```
6       if (n%p>=m%p)
7         ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p
8           *getInv(num[n%p-m%p])%p;
9       else
10        ans=0;
11    }
12    return ans;
13  }
```

3.8 组合数求模

模是质数

```
1  #include<cstdio>
2  #include<cstring>
3  #include<iostream>
4  using namespace std;
5  int mod;
6  long long num[100000];
7  int ni[100],mi[100];
8  int len;
9  void init(int p)
10 {
11   mod=p;
12   num[0]=1;
13   for (int i=1; i<p; i++)
14     num[i]=i*num[i-1]%p;
15 }
16 void get(int n,int ni[],int p)
17 {
18   for (int i = 0; i < 100; i++)
19     ni[i] = 0;
20   int tlen = 0;
21   while (n != 0)
22   {
23     ni[tlen++] = n%p;
24     n /= p;
25   }
26   len = tlen;
27 }
28 long long power(long long x,long long y)
29 {
30   long long ret=1;
31   for (long long a=x%mod; y; y>>=1,a=a*a%mod)
32     if (y&1)
33       ret=ret*a%mod;
34   return ret;
35 }
36 long long getInv(long long x)//mod 为素数
37 {
38   return power(x,mod-2);
39 }
```

```
long long calc(int n,int m,int p)//C(n,m)%p
{
  init(p);
  long long ans=1;
  for (; n && m && ans; n/=p,m/=p)
  {
    if (n%p>=m%p)
      ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p
        *getInv(num[n%p-m%p])%p;
    else
      ans=0;
  }
  return ans;
}
int main()
{
  int t;
  scanf("%d",&t);
  while (t--)
  {
    int n,m,p;
    scanf("%d%d%d",&n,&m,&p);
    printf("%I64d\n",calc(n+m,m,p));
  }
  return 0;
}
```

3.9 高斯消元

```
const double eps = 1e-8;

void Guess(int n)
{
  for (int i = 0; i < n; i++)
  {
    for (int j = i; j < n; j++)
      if (fabs(a[j][i]) > eps)
      {
        for (int k = i; k <= n; k++)
          swap(a[i][k],a[j][k]);
        break;
      }

    if (fabs(a[i][i]) < eps)  continue;

    for (int j = 0; j < n; j++)
      if (i != j && fabs(a[j][i]) > eps)
      {
        double det = a[j][i]/a[i][i];
        for (int k = i; k <= n; k++)
          a[j][k] -= a[i][k]*det;
```

```
23        }
24    }
25
26    for (int i = 0; i < n; i++)
27    {
28      if (fabs(a[i][i]) < eps)
29      {
30        if (fabs(a[i][n]) > eps)
31        {
32          //无解
33          puts("Fuck");
34        }
35        //否则 x_i 可以是任意解
36      }
37      else
38      {
39        a[i][n] /= a[i][i];
40        if (fabs(a[i][n]) < eps)
41          a[i][n] = 0;
42      }
43    }
44
45  }
```

3.10 其它公式

3.10.1 Polya

设 G 是 p 个对象的一个置换群，用 k 种颜色去染这 p 个对象，若一种染色方案在群 G 的作用下变为另一种方案，则这两个方案当作是同一种方案，这样的不同染色方案数为：
$L = \frac{1}{|G|} \times \Sigma(k^{C(f)}), f \in G$
$C(f)$ 为循环节， $|G|$ 表示群的置换方法数

对于有 n 个位置的手镯，有 n 种旋转置换和 n 种翻转置换

对于旋转置换：
$C(f_i) = gcd(n,i)$,i 表示一次转过 i 颗宝石， $i = 0$ 时 $c = n$;

对于翻转置换：

如果 n 为偶数： 则有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2}$，有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2} + 1$

如果 n 为奇数： $C(f) = \frac{n}{2} + 1$

3.10.2 拉格朗日插值法

已知 $y = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ 曲线上的 n 个点 $(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots (x_n, y_n)$ 用拉格朗日插值法可以不求系数可知任意 x 对应的 y 值。

$$
\begin{aligned}
y = & y_1 \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)} \\
& + y_2 \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)} \\
& + \cdots \\
& + y_n \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})}
\end{aligned}
$$

特别的，如果 $x_1 \sim x_n$ 为连续自然数，那么对于下一个自然数对应的 y 值为：

$$y_{n+1} = (-1)^{n-1}C_n^0 y_1 + (-1)^{n-2}C_n^1 y_2 + \cdots + (-1)^0 C_n^{n-1} y_n$$

这个组合系数可以通过高斯消元暴出来，前提是要猜到它满足递推关系。

3.10.3 正多面体顶点着色

正四面体： $N = \frac{(n^4+11\times n^2)}{12}$
正六面体： $N = \frac{(n^8+17\times n^4+6\times n^2)}{24}$
正八面体： $N = \frac{(n^6+3\times n^4+12\times n^3+8\times n^2)}{24}$
正十二面体： $N = \frac{(n^{20}+15\times n^{10}+20\times n^8+24\times n^4)}{60}$
正二十面体： $N = \frac{(n^{12}+15\times n^6+44\times n^4)}{60}$

3.10.4 求和公式

$\sum k = \frac{n\times(n+1)}{2}$
$\sum 2k-1 = n^2$
$\sum k^2 = \frac{n\times(n+1)\times(2n+1)}{6}$
$\sum (2k-1)^2 = \frac{n\times(4n^2-1)}{3}$
$\sum k^3 = (\frac{n\times(n+1)}{2})^2$
$\sum (2k-1)^3 = n^2\times(2n^2-1)$
$\sum k^4 = \frac{n\times(n+1)\times(2n+1)\times(3n^2+3n-1)}{30}$
$\sum k^5 = \frac{n^2\times(n+1)^2\times(2n^2+2n-1)}{12}$
$\sum k\times(k+1) = \frac{n\times(n+1)\times(n+2)}{3}$
$\sum k\times(k+1)\times(k+2) = \frac{n\times(n+1)\times(n+2)\times(n+3)}{4}$
$\sum k\times(k+1)\times(k+2)\times(k+3) = \frac{n\times(n+1)\times(n+2)\times(n+3)\times(n+4)}{5}$

3.10.5 几何公式

球扇形：
全面积： $T = \pi r(2h+r_0)$， h 为球冠高， r_0 为球冠底面半径
体积： $V = \frac{2\pi r^2 h}{3}$

3.10.6 小公式

Pick 公式： $A = E \times 0.5 + I - 1$（A 是多边形面积， E 是边界上的整点， I 是多边形内部的整点）

海伦公式： $S = \sqrt{p(p-a)(p-b)(p-c)}$，其中 $p = \frac{(a+b+c)}{2}$， abc 为三角形的三条边长

求 $\binom{n}{k}$ 中素因子 P 的个数：

1. 把 n 转化为 P 进制，并记它每个位上的和为 $S1$
2. 把 $n-k$， k 做同样的处理，得到 $S2$， $S3$

则 $\binom{n}{k}$ 中素因子 P 的个数： $\frac{S2+S3-S1}{P-1}$

部分错排公式：
$n+m$ 个数中 m 个数必须错排求排列数

```
dp[i] = n*dp[i-1]+(i-1)*(dp[i-1]+dp[i-2]);
dp[0] = n!;
dp[1] = n*n!;
```

$dp[m]$ 为所求解

3.10.7 马步问题

任意步长 (p,q) 无限棋盘可达性判定

```
bool check(int dx,int dy,int p,int q)
{
  if (p < 0) p = -p;
  if (q < 0) q = -q;
  LL g = gcd(p,q);
  if (dx % g || dy % g) return false;
  dx /= g,dy /= g,p = (p / g) & 1,q = (q / g) & 1;
  return !(p == q && ((dx ^ dy) & 1));
}
```

拓展:
若可选马步可以有 N 种 (p_i,q_i), 令 $g = gcd(p_1,q_1,p_2,q_2 \cdots p_N,q_N)$, 则不在 g 的整数倍点上的节点肯定不可达。坐标除 $2g$, 同时将可选马步除 g 之后放缩到 2×2 之内, 即 $(\frac{p_i}{g} \mod 2, \frac{q_i}{g} \mod 2)$。若放缩后马步中有 $(1,0)$ 或 $(0,1)$, 则全放缩后全棋盘可达, 否则只可达偶点。

(2,1) 马步无限棋盘最小距离

```
int dis(int dx,int dy)
{
  if (dx < 0) dx = -dx;
  if (dy < 0) dy = -dy;
  if (dx < dy) swap(dx,dy);
  if (dx & 1)
  {
    if (dy & 1) return dis(dx+1,dy-1);
```

```
 9       if (dx == 1 && dy == 0) return 3;
10       return dis(dx+3,dy)-1;
11     }
12     if (dy & 1)
13     {
14       if (dx == 4 && dy == 3) return 3;
15       return dis(dx-2,dy-1)+1;
16     }
17     if (dx == 0 && dy == 0) return 0;
18     if (dx == 2 && dy == 2) return 4;
19     int c = (((dx-1) / 4)+1)*2;
20     if (dx & 2) dy -= 2;
21     if (dy <= c) return c;
22     dy -= c;
23     return c+(dy-2) / 6*2+2;
24 }
```

4 数据结构

4.1 树链剖分

4.1.1 点权

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <algorithm>
using namespace std;
const int MAX = 12000;
const int LOG = 15;
const int oo = 0x3f3f3f3f;
struct Edge
{
    int to, w, id;
    Edge* next;
} memo[MAX<<1],*cur,*g[MAX],*pree[MAX],*solid[MAX],*valid[MAX];
int dp[MAX][LOG], pos[MAX], lst[MAX], dep[MAX], cnt[MAX], h[MAX], K
   , n;
void init()
{
  for (int i = 1; i <= n; i++)
  {
    g[i] = NULL;
    valid[i] = NULL;
    solid[i] = NULL;
    pree[i] = NULL;
  }
  for (int i = 0; i < LOG; i++)
  {
    dp[1][i] = 1;
  }
  cur = memo;
  K = 0;
}
void add(int u, int v, int w, int id)
{
  cur->to = v;
  cur->w = w;
  cur->id = id;
  cur->next = g[u];
  g[u] = cur++;
}
void dfsLCA(int d, int u, int f)
{
  dep[u] = d;
  dp[u][0] = f;
  cnt[u] = 1;
  for (int i = 1; i < LOG; i++)
  {
```

```
46      dp[u][i] = dp[dp[u][i - 1]][i - 1];
47    }
48    for (Edge* it = g[u]; it; it = it->next)
49    {
50      int v = it->to;
51      if (v != f)
52      {
53        pree[v] = it;
54        valid[it->id] = it;
55        dfsLCA(d + 1, v, u); //RE
56        cnt[u] += cnt[v];
57        if (solid[u] == NULL || cnt[solid[u]->to] < cnt[v])
58        {
59          solid[u] = it;
60        }
61      }
62    }
63  }
64  void dfsChain(int u, int head)
65  {
66    h[u] = head;
67    if (solid[u])
68    {
69      lst[pos[u] = K++] = u;
70      dfsChain(solid[u]->to, head);
71    }
72    else
73    for (Edge* it = g[u]; it; it = it->next)
74    {
75      int v = it->to;
76      if (it != solid[u] && v != dp[u][0])
77      {
78        dfsChain(v, v);
79      }
80    }
81  }
82  int getLCA(int u, int v)
83  {
84    if (dep[u] < dep[v])
85      swap(u, v);
86    for (int st = 1 << (LOG - 1), i = LOG - 1; i >= 0; i--, st >>= 1)
87    {
88      if (st <= dep[u] - dep[v])
89      {
90        u = dp[u][i];
91      }
92    }
93    if (u == v)
94      return u;
95    for (int i = LOG - 1; i >= 0; i--)
96    {
```

```
97      if (dp[u][i] != dp[v][i])
98      {
99        u = dp[u][i];
100       v = dp[v][i];
101     }
102   }
103   return dp[u][0];
104 }
105 struct Node
106 {
107     int l, r, ma, mi;
108     bool rev;
109 } seg[MAX << 2];
110 void reverse(int k)
111 {
112   seg[k].mi *= -1;
113   seg[k].ma *= -1;
114   seg[k].rev ^= 1;
115   swap(seg[k].mi, seg[k].ma);
116 }
117 void pushdown(int k)
118 {
119   if (seg[k].rev)
120   {
121     reverse(k << 1);
122     reverse(k << 1 | 1);
123     seg[k].rev = false;
124   }
125 }
126 void update(int k)
127 {
128   seg[k].mi = min(seg[k << 1].mi, seg[k << 1 | 1].mi);
129   seg[k].ma = max(seg[k << 1].ma, seg[k << 1 | 1].ma);
130 }
131 void init(int k, int l, int r)
132 {
133   seg[k].l = l;
134   seg[k].r = r;
135   seg[k].rev = false;
136   if (l == r)
137   {
138     seg[k].mi = seg[k].ma = solid[lst[l]]->w; //solid WA
139     return;
140   }
141   int mid = l + r >> 1;
142   init(k << 1, l, mid);
143   init(k << 1 | 1, mid + 1, r);
144   update(k);
145 }
146 void update(int k, int id, int v)
147 {
```

```
148   if (seg[k].l == seg[k].r)
149   {
150     seg[k].mi = seg[k].ma = solid[lst[id]]->w = v;
151     return;
152   }
153   pushdown(k);
154   int mid = seg[k].l + seg[k].r >> 1;
155   if (id <= mid)
156     update(k << 1, id, v);
157   else
158     update(k << 1 | 1, id, v);
159   update(k);
160 }
161 void reverse(int k, int l, int r)
162 {
163   if (seg[k].l > r || seg[k].r < l)
164     return;
165   if (seg[k].l >= l && seg[k].r <= r)
166   {
167     reverse(k);
168     return;
169   }
170   pushdown(k);
171   reverse(k << 1, l, r);
172   reverse(k << 1 | 1, l, r);
173   update(k);
174 }
175 int read(int k, int l, int r)
176 {
177   if (seg[k].l > r || seg[k].r < l)
178     return -oo;
179   if (seg[k].l >= l && seg[k].r <= r)
180     return seg[k].ma;
181   pushdown(k);
182   return max(read(k << 1, l, r), read(k << 1 | 1, l, r));
183 }
184 void setEdge(int id, int v)
185 {
186   Edge* it = valid[id];
187   if (h[it->to] != it->to)
188   {
189     update(1, pos[dp[it->to][0]], v);
190   }
191   else
192   {
193     it->w = v;
194   }
195 }
196 void negateLCA(int t, int u)
197 {
198   while (t != u)
```

```
199   {
200     int tmp = h[u];
201     if (dep[tmp] < dep[t])
202       tmp = t;
203     if (h[u] == u)
204     {
205       pree[u]->w *= -1;
206       u = dp[u][0];
207     }
208     else
209     {
210       reverse(1, pos[tmp], pos[dp[u][0]]);
211       u = tmp;
212     }
213   }
214 }
215 void negate(int u, int v)
216 {
217   int t = getLCA(u, v);
218   negateLCA(t, u);
219   negateLCA(t, v);
220 }
221 int maxLCA(int t, int u)
222 {
223   int ret = -oo;
224   while (t != u)
225   {
226     int tmp = h[u];
227     if (dep[tmp] < dep[t])
228       tmp = t;
229     if (h[u] == u)
230     {
231       ret = max(ret, pree[u]->w);
232       u = dp[u][0];
233     }
234     else
235     {
236       ret = max(ret, read(1, pos[tmp], pos[dp[u][0]]));
237       u = tmp;
238     }
239   }
240   return ret;
241 }
242 int query(int u, int v)
243 {
244   int t = getLCA(u, v);
245   return max(maxLCA(t, u), maxLCA(t, v));
246 }
247 int main()
248 {
249   int T;
```

```
250   int u, v, w;
251   char op[15];
252   scanf("%d", &T);
253   while (T--)
254   {
255     scanf("%d", &n);
256     init();
257     for (int i = 1; i < n; i++)
258     {
259       scanf("%d%d%d", &u, &v, &w);
260       add(u, v, w, i);
261       add(v, u, w, i);
262     }
263     dfsLCA(0, 1, 1);
264     dfsChain(1, 1);
265     init(1, 0, K - 1);
266     while (scanf("%s", op), op[0] != 'D')
267     {
268       scanf("%d%d", &u, &v);
269       if (op[0] == 'C')
270       {
271         setEdge(u, v);
272       }
273       else if (op[0] == 'N')
274       {
275         negate(u, v);
276       }
277       else
278       {
279         printf("%d\n", query(u, v));
280       }
281     }
282   }
283   return 0;
284 }
```

4.1.2 边权

```
 1 #include <cstdio>
 2 #include <iostream>
 3 #include <cstdlib>
 4 #include <algorithm>
 5 #include <cmath>
 6 #include <cstring>
 7 using namespace std;
 8 int n,m,sum,pos;
 9 int head[50005],e;
10 int s[50005],from[50005];
11 int fa[50005][20],deep[50005],num[50005];
12 int solid[50005],p[50005],fp[50005];
13 struct N
14 {
```

```
15    int l,r,mid;
16    int add,w;
17  }nod[50005*4];
18  struct M
19  {
20    int v,next;
21  }edge[100005];
22  void addedge(int u,int v)
23  {
24    edge[e].v=v;
25    edge[e].next=head[u];
26    head[u]=e++;
27
28    edge[e].v=u;
29    edge[e].next=head[v];
30    head[v]=e++;
31  }
32  void LCA(int st,int f,int d)
33  {
34    deep[st]=d;
35    fa[st][0]=f;
36    num[st]=1;
37    int i,v;
38    for(i=1;i<20;i++)
39      fa[st][i]=fa[fa[st][i-1]][i-1];
40    for(i=head[st];i!=-1;i=edge[i].next)
41    {
42      v=edge[i].v;
43      if(v!=f)
44      {
45        LCA(v,st,d+1);
46        num[st]+=num[v];
47        if(solid[st]==-1||num[v]>num[solid[st]])
48          solid[st]=v;
49      }
50    }
51  }
52  void getpos(int st,int sp)
53  {
54    from[st]=sp;
55    if(solid[st]!=-1)
56    {
57      p[st]=pos++;
58      fp[p[st]]=st;
59      getpos(solid[st],sp);
60    }
61    else
62    {
63      p[st]=pos++;
64      fp[p[st]]=st;
65      return;
```

```
  }
  int i,v;
  for(i=head[st];i!=-1;i=edge[i].next)
  {
    v=edge[i].v;
    if(v!=solid[st]&&v!=fa[st][0])
      getpos(v,v);
  }
}
int getLCA(int u,int v)
{
  if(deep[u]<deep[v])
    swap(u,v);
  int d=1<<19,i;
  for(i=19;i>=0;i--)
  {
    if(d<=deep[u]-deep[v])
      u=fa[u][i];
    d>>=1;
  }
  if(u==v)
    return u;
  for(i=19;i>=0;i--)
    if(fa[u][i]!=fa[v][i])
    {
      u=fa[u][i];
      v=fa[v][i];
    }
  return fa[u][0];
}
void init(int p,int l,int r)
{
  nod[p].l=l;
  nod[p].r=r;
  nod[p].mid=(l+r)>>1;
  nod[p].add=0;
  if(l==r)
    nod[p].w=s[fp[l]];
  else
  {
    init(p<<1,l,nod[p].mid);
    init(p<<1|1,nod[p].mid+1,r);
  }
}
void lazy(int p)
{
  if(nod[p].add!=0)
  {
    nod[p<<1].add+=nod[p].add;
    nod[p<<1|1].add+=nod[p].add;
    nod[p].add=0;
```

```
117   }
118 }
119 void update(int p,int l,int r,int v)
120 {
121   if(nod[p].l==l&&nod[p].r==r)
122   {
123     nod[p].add+=v;
124     return;
125   }
126   lazy(p);
127   if(nod[p].mid<l)
128     update(p<<1|1,l,r,v);
129   else if(nod[p].mid>=r)
130     update(p<<1,l,r,v);
131   else
132   {
133     update(p<<1,l,nod[p].mid,v);
134     update(p<<1|1,nod[p].mid+1,r,v);
135   }
136 }
137 int read(int p,int l,int r)
138 {
139   if(nod[p].l==l&&nod[p].r==r)
140     return nod[p].w+nod[p].add;
141   lazy(p);
142   if(nod[p].mid<l)
143     return read(p<<1|1,l,r);
144   else if(nod[p].mid>=r)
145     return read(p<<1,l,r);
146 }
147 void jump(int st,int ed,int val)
148 {
149   while(deep[st]>=deep[ed])
150   {
151     int tmp=from[st];
152     if(deep[tmp]<deep[ed])
153       tmp=ed;
154     update(1,p[tmp],p[st],val);
155     st=fa[tmp][0];
156   }
157 }
158 void change(int st,int ed,int val)
159 {
160   int lca=getLCA(st,ed);
161   jump(st,lca,val);
162   jump(ed,lca,val);
163   jump(lca,lca,-val);
164 }
165 int main()
166 {
167   while(scanf("%d%d%d",&n,&m,&sum)==3)
```

```
168   {
169     int i;
170     s[0]=0;pos=0;deep[0]=-1;
171     memset(fa,0,sizeof(fa));
172     for(i=1;i<=n;i++)
173     {
174       solid[i]=-1;
175       scanf("%d",&s[i]);
176     }
177     memset(head,-1,sizeof(head));
178     e=0;
179     for(i=0;i<m;i++)
180     {
181       int a,b;
182       scanf("%d%d",&a,&b);
183       addedge(a,b);
184     }
185     LCA(1,0,0);
186     getpos(1,1);
187     init(1,0,pos-1);
188     for(i=0;i<sum;i++)
189     {
190       char que[5];
191       scanf("%s",que);
192       if(que[0]!='Q')
193       {
194         int a,b,c;
195         scanf("%d%d%d",&a,&b,&c);
196         if(que[0]=='D')
197           c=-c;
198         change(a,b,c);
199       }
200       else
201       {
202         int a;
203         scanf("%d",&a);
204         printf("%d\n",read(1,p[a],p[a]));
205       }
206     }
207   }
208   return 0;
209 }
```

4.2 树状数组

```
1 int read(int k)
2 {
3   int sum = 0;
4   for (; k; k^=k&-k)
5     sum+=tree[k];
6   return sum;
7 }
```

```
 8 void update(int k, int v)
 9 {
10   for (; k<=MaxN; k+=k&-k)
11     tree[k]+=v;
12 }
13 int find_Kth(int k)
14 {
15   int idx = 0;
16   for(int i=20; i>=0; i--)
17   {
18     idx |= 1 << i;
19     if(idx <= MaxN && tree[idx] < k)
20       k -= tree[idx];
21     else  idx ^= 1 << i;
22   }
23   return idx + 1;
24 }
```

5 图论

5.1 优先队列优化的 dijkstra

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
using namespace std;
const int MAXN=100;
const int MAXM=1000;
int N,L;
int head[MAXN];
struct edges
{
  int to,next,cost;
} edge[MAXM];
int dist[MAXN];
class states
{
public:
  int cost,id;
};
class cmp
{
public:
  bool operator ()(const states &i,const states &j)
  {
    return i.cost>j.cost;
  }
};
void init(int n)
{
  N=n;
  L=0;
  for (int i=0; i<n; i++)
    head[i]=-1;
}
void add_edge(int x,int y,int cost)
{
  edge[L].to=y;
  edge[L].cost=cost;
  edge[L].next=head[x];
  head[x]=L++;
}
int dijkstra(int s,int t)
{
  memset(dist,63,sizeof(dist));
  states u;
```

```
48    u.id=s;
49    u.cost=0;
50    dist[s]=0;
51    priority_queue<states,vector<states>,cmp> q;
52    q.push(u);
53    while (!q.empty())
54    {
55      u=q.top();
56      q.pop();
57      if (u.id==t) return dist[t];
58      if (u.cost!=dist[u.id]) continue;
59      for (int i=head[u.id]; i!=-1; i=edge[i].next)
60      {
61        states v=u;
62        v.id=edge[i].to;
63        if (dist[v.id]>dist[u.id]+edge[i].cost)
64        {
65          v.cost=dist[v.id]=dist[u.id]+edge[i].cost;
66          q.push(v);
67        }
68      }
69    }
70    return -1;
71  }
72  int main()
73  {
74    int n,m;
75    scanf("%d%d",&n,&m);
76    init(n);
77    for (int i=0; i<m; i++)
78    {
79      int x,y,z;
80      scanf("%d%d%d",&x,&y,&z);
81      add_edge(x,y,z);
82      add_edge(y,x,z);
83    }
84    int s,t;
85    scanf("%d%d",&s,&t);
86    printf("%d\n",dijkstra(s,t));
87    return 0;
88  }
```

5.2 SAP 四版

```
1  const int MAXEDGE=20400;
2  const int MAXN=400;
3  const int inf=0x3fffffff;
4  struct edges
5  {
6    int cap,to,next,flow;
7  } edge[MAXEDGE+100];
8  struct nodes
```

```
{
  int head,label,pre,cur;
} node[MAXN+100];
int L,N;
int gap[MAXN+100];
void init(int n)
{
  L=0;
  N=n;
  for (int i=0; i<N; i++)
    node[i].head=-1;
}
void add_edge(int x,int y,int z,int w)
{
  edge[L].cap=z;
  edge[L].flow=0;
  edge[L].to=y;
  edge[L].next=node[x].head;
  node[x].head=L++;
  edge[L].cap=w;
  edge[L].flow=0;
  edge[L].to=x;
  edge[L].next=node[y].head;
  node[y].head=L++;
}
int maxflow(int s,int t)
{
  memset(gap,0,sizeof(gap));
  gap[0]=N;
  int u,ans=0;
  for (int i=0; i<N; i++)
  {
    node[i].cur=node[i].head;
    node[i].label=0;
  }
  u=s;
  node[u].pre=-1;
  while (node[s].label<N)
  {
    if (u==t)
    {
      int min=inf;
      for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
        if (min>edge[i].cap-edge[i].flow)
          min=edge[i].cap-edge[i].flow;
      for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
      {
        edge[i].flow+=min;
        edge[i^1].flow-=min;
      }
      u=s;
```

```
60        ans+=min;
61        continue;
62      }
63      bool flag=false;
64      int v;
65      for (int i=node[u].cur; i!=-1; i=edge[i].next)
66      {
67        v=edge[i].to;
68        if (edge[i].cap-edge[i].flow &&
69          node[v].label+1==node[u].label)
70        {
71          flag=true;
72          node[u].cur=node[v].pre=i;
73          break;
74        }
75      }
76      if (flag)
77      {
78        u=v;
79        continue;
80      }
81      node[u].cur=node[u].head;
82      int min=N;
83      for (int i=node[u].head; i!=-1; i=edge[i].next)
84        if (edge[i].cap-edge[i].flow && node[edge[i].to].label<min)
85          min=node[edge[i].to].label;
86      gap[node[u].label]--;
87      if (!gap[node[u].label]) return ans;
88      node[u].label=min+1;
89      gap[node[u].label]++;
90      if (u!=s) u=edge[node[u].pre^1].to;
91    }
92    return ans;
93  }
```

5.3 费用流三版

T 了可以改成栈。

```
1  const int MAXM=60000;
2  const int MAXN=400;
3  const int inf=0x3fffffff;
4  int L,N;
5  int K;
6  struct edges
7  {
8    int to,next,cap,flow,cost;
9  } edge[MAXM];
10 struct nodes
11 {
12   int dis,pre,head;
13   bool visit;
```

```
} node[MAXN];
void init(int n)
{
  N=n;
  L=0;
  for (int i=0; i<N; i++)
    node[i].head=-1;
}
void add_edge(int x,int y,int cap,int cost)
{
  edge[L].to=y;
  edge[L].cap=cap;
  edge[L].cost=cost;
  edge[L].flow=0;
  edge[L].next=node[x].head;
  node[x].head=L++;
  edge[L].to=x;
  edge[L].cap=0;
  edge[L].cost=-cost;
  edge[L].flow=0;
  edge[L].next=node[y].head;
  node[y].head=L++;
}
bool spfa(int s,int t)
{
  queue <int> q;
  for (int i=0; i<N; i++)
  {
    node[i].dis=0x3fffffff;
    node[i].pre=-1;
    node[i].visit=0;
  }
  node[s].dis=0;
  node[s].visit=1;
  q.push(s);
  while (!q.empty())
  {
    int u=q.front();
    node[u].visit=0;
    for (int i=node[u].head; i!=-1; i=edge[i].next)
    {
      int v=edge[i].to;
      if (edge[i].cap>edge[i].flow &&
          node[v].dis>node[u].dis+edge[i].cost)
      {
        node[v].dis=node[u].dis+edge[i].cost;
        node[v].pre=i;
        if (!node[v].visit)
        {
          node[v].visit=1;
          q.push(v);
```

```
65          }
66        }
67      }
68      q.pop();
69    }
70    if (node[t].pre==-1)
71      return 0;
72    else
73      return 1;
74  }
75  int mcmf(int s,int t,int &cost)
76  {
77    int flow=0;
78    while (spfa(s,t))
79    {
80      int max=inf;
81      for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
82      {
83        if (max>edge[i].cap-edge[i].flow)
84          max=edge[i].cap-edge[i].flow;
85      }
86      for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
87      {
88        edge[i].flow+=max;
89        edge[i^1].flow-=max;
90        cost+=edge[i].cost*max;
91      }
92      flow+=max;
93    }
94    return flow;
95  }
```

5.4 一般图最大加权匹配

注意 G 初始化

```
1  #define N 229
2  int G[N][N];
3  int cnt_node;
4  int dist[N];
5  int rec[N],cr,M[N],P[N];
6  bool vst[N];
7  const int inf = 0x3f3f3f3f;
8  bool spfa(int u)
9  {
10    rec[cr++]=u;
11    if(vst[u]) return true;
12    vst[u]=true;
13    int v;
14    for(v=0; v<cnt_node; v++)
15    {
```

```
16        if(v!=u&&M[u]!=v&&!vst[v])
17        {
18          int w=M[v];
19          if(dist[w]<dist[u]+G[u][v]-G[v][w])
20          {
21             dist[w]=dist[u]+G[u][v]-G[v][w];
22             if(spfa(w))
23             {
24                return true;
25             }
26          }
27        }
28     }
29     cr--;
30     vst[u]=false;
31     return false;
32  }
33  int match()
34  {
35     int i;
36     for(i=0; i<cnt_node; i++) P[i]=i;
37     for(i=0; i<cnt_node; i+=2) M[i]=i+1,M[i+1]=i;
38     int cnt=0;
39     while(1)
40     {
41        memset(dist,0,sizeof(dist));
42        cr=0;
43        int i;
44        bool fd=false;
45        memset(vst,0,sizeof(vst));
46        for(i=0; i<cnt_node; i++)
47        {
48          if(spfa(P[i]))
49          {
50             fd=true;
51             int j;
52             int nx=M[rec[cr-1]];
53             for(j=cr-2; rec[j]!=rec[cr-1]; j--)
54             {
55                M[nx]=rec[j];
56                int tmp=nx;
57                nx=M[rec[j]];
58                M[rec[j]]=tmp;
59             }
60             M[nx]=rec[j];
61             M[rec[j]]=nx;
62             break;
63          }
64        }
65        if(!fd)
66        {
```

```
67          cnt++;
68          if(cnt>=3) break;
69          random_shuffle(P,P+cnt_node);
70        }
71    }
72    int sum=0;
73    for(i=0; i<cnt_node; i++)
74    {
75       int v=M[i];
76       if(i<v)
77       {
78         sum+=G[i][v];
79       }
80    }
81    return sum;
82 }
```

5.5 一般图匹配带花树

```
 1 const int MaxN = 222;
 2 int N;
 3 bool Graph[MaxN+1][MaxN+1];
 4 int Match[MaxN+1];
 5 bool InQueue[MaxN+1],InPath[MaxN+1],InBlossom[MaxN+1];
 6 int Head,Tail;
 7 int Queue[MaxN+1];
 8 int Start,Finish;
 9 int NewBase;
10 int Father[MaxN+1],Base[MaxN+1];
11 int Count;
12 void CreateGraph()
13 {
14   int u,v;
15   memset(Graph,false,sizeof(Graph));
16   scanf("%d",&N);
17   while (scanf("%d%d",&u,&v) != EOF)
18     Graph[u][v] = Graph[v][u] = true;
19 }
20 void Push(int u)
21 {
22   Queue[Tail] = u;
23   Tail++;
24   InQueue[u] = true;
25 }
26 int Pop()
27 {
28   int res = Queue[Head];
29   Head++;
30   return res;
31 }
32 int FindCommonAncestor(int u,int v)
```

```
33 {
34   memset(InPath,false,sizeof(InPath));
35   while (true)
36   {
37     u = Base[u];
38     InPath[u] = true;
39     if (u == Start) break;
40     u = Father[Match[u]];
41   }
42   while (true)
43   {
44     v = Base[v];
45     if (InPath[v]) break;
46     v = Father[Match[v]];
47   }
48   return v;
49 }
50 void ResetTrace(int u)
51 {
52   int v;
53   while (Base[u] != NewBase)
54   {
55     v = Match[u];
56     InBlossom[Base[u]] = InBlossom[Base[v]] = true;
57     u = Father[v];
58     if (Base[u] != NewBase) Father[u] = v;
59   }
60 }
61 void BlossomContract(int u,int v)
62 {
63   NewBase = FindCommonAncestor(u,v);
64   memset(InBlossom,false,sizeof(InBlossom));
65   ResetTrace(u);
66   ResetTrace(v);
67   if (Base[u] != NewBase) Father[u] = v;
68   if (Base[v] != NewBase) Father[v] = u;
69   for (int tu = 1; tu <= N; tu++)
70     if (InBlossom[Base[tu]])
71     {
72       Base[tu] = NewBase;
73       if (!InQueue[tu]) Push(tu);
74     }
75 }
76 void FindAugmentingPath()
77 {
78   memset(InQueue,false,sizeof(InQueue));
79   memset(Father,0,sizeof(Father));
80   for (int i = 1; i <= N; i++)
81     Base[i] = i;
82   Head = Tail = 1;
83   Push(Start);
```

```
  Finish = 0;
  while (Head < Tail)
  {
    int u = Pop();
    for (int v = 1; v <= N; v++)
      if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v))
      {
        if ((v == Start) ||
          ((Match[v] > 0) && (Father[Match[v]] > 0)))
          BlossomContract(u,v);
        else if (Father[v] == 0)
        {
          Father[v] = u;
          if (Match[v] > 0)
            Push(Match[v]);
          else
          {
            Finish = v;
            return;
          }
        }
      }
  }
}
void AugmentPath()
{
  int u,v,w;
  u = Finish;
  while (u > 0)
  {
    v = Father[u];
    w = Match[v];
    Match[v] = u;
    Match[u] = v;
    u = w;
  }
}
void Edmonds()
{
  memset(Match,0,sizeof(Match));
  for (int u = 1; u <= N; u++)
    if (Match[u] == 0)
    {
      Start = u;
      FindAugmentingPath();
      if (Finish > 0) AugmentPath();
    }
}
void PrintMatch()
{
  for (int u = 1; u <= N; u++)
```

```
135     if (Match[u] > 0)
136       Count++;
137   printf("%d\n",Count);
138   for (int u = 1; u <= N; u++)
139     if (u < Match[u])
140       printf("%d %d\n",u,Match[u]);
141 }
142 int main()
143 {
144   CreateGraph();
145   Edmonds();
146   PrintMatch();
147 }
```

5.6 KM

5.6.1 最大加权匹配

```
1  bool visx[N],visy[N];//x,y 中的点是否被访问
2  int lx[N],ly[N];//x,y 中的点的标号
3  int matchy[N];//y 中各点匹配状态
4  int map[N][N];//二分图描述 [x][y]
5  bool find(int x)
6  {
7    visx[x]=true;
8    int t;
9    for (int y=0;y<ycnt;y++)
10   {
11     if (!visy[y])
12     {
13       t=lx[x]+ly[y]-map[x][y];
14       if (t==0)
15       {
16         visy[y]=true;
17         if (matchy[y]==-1 || find(matchy[y]))
18         {
19           matchy[y]=x;
20           return true;
21         }
22       }
23       else if (lack>t) lack=t;
24     }
25   }
26   return false;
27 }
28 void KM()
29 {
30   memset(lx,0,sizeof(lx));
31   memset(ly,0,sizeof(ly));
32   memset(matchy,-1,sizeof(matchy));
33   for (int i=0;i<xcnt;i++)
34     for (int j=0;j<ycnt;j++)
```

```
35        if (map[i][j]>lx[i])
36          lx[i]=map[i][j];
37    for (int x=0;x<xcnt;x++)
38    {
39      while (true)
40      {
41        memset(visx,false,sizeof(visx));
42        memset(visy,false,sizeof(visy));
43        lack=INFI;
44        if (find(x)) break;
45        for (int i=0;i<xcnt;i++)
46        {
47          if (visx[i]) lx[i]-=lack;
48          if (visy[i]) ly[i]+=lack;
49        }
50      }
51    }
52    int cost=0;
53    for (int i=0;i<ycnt;i++)
54      cost+=map[matchy[i]][i];
55  }
```

5.6.2 自认为正确的 Kuhn_Munkras

未验证

```
 1  #include<cstdio>
 2  #include<cstring>
 3  #include<algorithm>
 4  using namespace std;
 5  const int MAXN=100;
 6  const int inf=0x3f3f3f3f;
 7  bool visitx[MAXN],visity[MAXN];
 8  int labx[MAXN],laby[MAXN],matx[MAXN],maty[MAXN],slack[MAXN];
 9  int ma[MAXN][MAXN];
10  bool check(int x,int n)
11  {
12    visitx[x]=1;
13    for (int i=0; i<n; i++)
14      if (!visity[i])
15        if (labx[x]+laby[i]==ma[x][i])
16        {
17          visity[i]=1;
18          if (maty[i]==-1 || check(maty[i],n))
19          {
20            matx[x]=i;
21            maty[i]=x;
22            return 1;
23          }
24        }
25        else
26          slack[i]=min(slack[i],labx[x]+laby[i]-ma[x][i]);
27
```

```
28    return 0;
29  }
30  void maintain(int n)
31  {
32    int diff=inf;
33    for (int i=0; i<n; i++)
34      if (!visity[i])
35        diff=min(diff,slack[i]);
36    for (int i=0; i<n; i++)
37    {
38      if (visitx[i])
39        labx[i]-=diff;
40      if (visity[i])
41        laby[i]+=diff;
42      else
43        slack[i]-=diff;
44    }
45  }
46  int Kuhn_Munkras(int n)
47  {
48    for (int i=0; i<n; i++)
49    {
50      labx[i]=-inf;
51      for (int j=0; j<n; j++)
52        labx[i]=max(labx[i],ma[i][j]);
53    }
54    memset(laby,0,4*n);
55    memset(matx,-1,4*n);
56    memset(maty,-1,4*n);
57    for (int i=0; i<n; i++)
58    {
59      memset(visitx,0,n);
60      memset(visity,0,n);
61      memset(slack,63,4*n);
62      while (!check(i,n))
63      {
64        maintain(n);
65        memset(visitx,0,n);
66        memset(visity,0,n);
67      }
68    }
69    int ret=0;
70    for (int i=0;i<n;i++)
71      ret+=labx[i]+laby[i];
72    return ret;
73  }
74  int main()
75  {
76    int n,m;
77    scanf("%d%d",&m,&n);
78    for (int i=m; i<n; i++)
```

```
79      for (int j=0; j<n; j++)
80        ma[i][j]=0;
81    for (int i=0; i<m; i++)
82      for (int j=0; j<n; j++)
83        scanf("%d",&ma[i][j]);
84    printf("%d\n",Kuhn_Munkras(n));
85    printf("%d",matx[0]+1);
86    for (int i=1;i<m;i++)
87      printf(" %d",matx[i]+1);
88    puts("");
89    return 0;
90  }
```

5.7 强联通

```
 1  int dfsnum[2000];
 2  int low[2000];
 3  int stack[2000];
 4  int top;
 5  int ans;
 6  int an;
 7  int be[2000];
 8  int flag[2000];
 9  void dfs(int x)
10  {
11    dfsnum[x] = low[x] = ans++;
12    stack[++top] = x;
13    flag[x] = 1;
14    for (int i = head[x];i != -1;i = edge[i].next)
15    {
16      int y = edge[i].to;
17      if (dfsnum[y] == -1)
18      {
19        dfs(y);
20        low[x] = min(low[x],low[y]);
21      }
22      else if (flag[y] == 1)
23        low[x] = min(low[x],dfsnum[y]);
24    }
25    if (dfsnum[x] == low[x])
26    {
27      while (stack[top] != x)
28      {
29        flag[stack[top]] = 0;
30        be[stack[top]] = an;
31        top--;
32      }
33      flag[x] = 0;
34      be[x] = an++;
35      top--;
36    }
```

```
37 }
```

调用：

```
1  void SC()
2  {
3    memset(dfsnum,-1,sizeof(dfsnum));
4    memset(flag,0,sizeof(flag));
5    top = 0;
6    an = 0;
7    ans = 0;
8    for (int i = 0;i < n;i++)
9      if (dfsnum[i] == -1)
10       dfs(i);
11 }
```

5.8 最大团以及相关知识

独立集： 独立集是指图的顶点集的一个子集, 该子集的导出子图不含边. 如果一个独立集不是任何一个独立集的子集, 那么称这个独立集是一个极大独立集. 一个图中包含顶点数目最多的独立集称为最大独立集。最大独立集一定是极大独立集，但是极大独立集不一定是最大的独立集。

支配集： 与独立集相对应的就是支配集，支配集也是图顶点集的一个子集，设 S 是图 G 的一个支配集，则对于图中的任意一个顶点 u，要么属于集合 s, 要么与 s 中的顶点相邻。在 s 中除去任何元素后 s 不再是支配集，则支配集 s 是极小支配集。称 G 的所有支配集中顶点个数最少的支配集为最小支配集，最小支配集中的顶点个数成为支配数。

最小点的覆盖： 最小点的覆盖也是图的顶点集的一个子集，如果我们选中一个点，则称这个点将以他为端点的所有边都覆盖了。将图中所有的边都覆盖所用顶点数最少，这个集合就是最小的点的覆盖。

最大团： 图 G 的顶点的子集，设 D 是最大团，则 D 中任意两点相邻。若 u， v 是最大团，则 u,v 有边相连，其补图 u,v 没有边相连，所以图 G 的最大团 $=$ 其补图的最大独立集。给定无向图 $G=(V,E)$，如果 U 属于 V，并且对于任意 u,v 包含于 U 有 $<u,v>$ 包含于 E，则称 U 是 G 的完全子图， G 的完全子图 U 是 G 的团，当且仅当 U 不包含在 G 的更大的完全子图中， G 的最大团是指 G 中所含顶点数目最多的团。如果 U 属于 V，并且对于任意 u,v 包含于 U 有 $<u,v>$ 不包含于 E，则称 U 是 G 的空子图， G 的空子图 U 是 G 的独立集，当且仅当 U 不包含在 G 的更大的独立集， G 的最大团是指 G 中所含顶点数目最多的独立集。

一些性质： 最大独立集 $+$ 最小覆盖集 $=V$，最大团 $=$ 补图的最大独立集，最小覆盖集 $=$ 最大匹配

```
1  #include <cstdio>
2  bool am[100][100];
3  int ans;
4  int c[100];
5  int U[100][100];
6  int n;
7  bool dfs(int rest,int num)
8  {
9    if (!rest)
```

```
10    {
11      if (num>=ans)
12        return 1;
13      else
14        return 0;
15    }
16    int pre=-1;
17    for (int i=0;i<rest && rest-i+num>=ans;i++)
18    {
19      int idx=U[num][i];
20      if (num+c[idx]<ans)
21        return 0;
22      int nrest=0;
23      for (int j=i+1; j<rest; j++)
24        if (am[idx][U[num][j]])
25          U[num+1][nrest++]=U[num][j];
26      if (dfs(nrest,num+1))
27        return 1;
28    }
29    return 0;
30  }
31  int main()
32  {
33    while (scanf("%d",&n),n)
34    {
35      for (int i=0;i<n;i++)
36        for (int j=0;j<n;j++)
37          scanf("%d",&am[i][j]);
38      ans=0;
39      for (int i=n-1; i>=0; i--)
40      {
41        int rest=0;
42        for (int j=i+1; j<n; j++)
43          if (am[i][j])
44            U[0][rest++]=j;
45        ans+=dfs(rest,0);
46        c[i]=ans;
47      }
48      printf("%d\n",ans);
49    }
50    return 0;
51  }
```

5.9 双连通分量

标号从 0 起

```
1  #include<cstdio>
2  #include<cstring>
3  #include<stack>
4  #include<queue>
5  #include<algorithm>
```

```
 6 using namespace std;
 7 const int MAXN=100000*2;
 8 const int MAXM=200000;
 9 struct edges
10 {
11   int to,next;
12   bool cut,visit;
13 } edge[MAXM<<1];
14 int head[MAXN],low[MAXN],dpt[MAXN],L;
15 bool visit[MAXN],cut[MAXN];
16 void init(int n)
17 {
18   L=0;
19   memset(head,-1,4*n);
20   memset(visit,0,n);
21 }
22 void add_edge(int u,int v)
23 {
24   edge[L].cut=edge[L].visit=0;
25   edge[L].to=v;
26   edge[L].next=head[u];
27   head[u]=L++;
28 }
29 int idx;
30 stack<int> st;
31 int bcc[MAXM];
32 void dfs(int u,int fu,int deg)
33 {
34   cut[u]=0;
35   visit[u]=1;
36   low[u]=dpt[u]=deg;
37   int tot=0;
38   for (int i=head[u]; i!=-1; i=edge[i].next)
39   {
40     int v=edge[i].to;
41     if (edge[i].visit)
42       continue;
43     st.push(i/2);
44     edge[i].visit=edge[i^1].visit=1;
45     if (visit[v])
46     {
47       low[u]=dpt[v]>low[u]?low[u]:dpt[v];
48       continue;
49     }
50     dfs(v,u,deg+1);
51     edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
52     if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
53     if (low[v]>=dpt[u] || u==fu)
54     {
55       while (st.top()!=i/2)
56       {
```

```
57          int x=st.top()*2,y=st.top()*2+1;
58          bcc[st.top()]=idx;
59          st.pop();
60        }
61        bcc[i/2]=idx++;
62        st.pop();
63      }
64      low[u]=low[v]>low[u]?low[u]:low[v];
65      tot++;
66    }
67    if (u==fu && tot>1) cut[u]=1;
68  }
69  int main()
70  {
71    int n,m;
72    while (scanf("%d%d",&n,&m)!=EOF)
73    {
74      init(n);
75      for (int i=0; i<m; i++)
76      {
77        int u,v;
78        scanf("%d%d",&u,&v);
79        add_edge(u,v);
80        add_edge(v,u);
81      }
82      idx=0;
83      for (int i=0; i<n; i++)
84        if (!visit[i])
85          dfs(i,i,0);
86    }
87    return 0;
88  }
```

5.10 割点与桥

```
1  #include<cstdio>
2  #include<cstring>
3  const int MAXN=10000;
4  struct edges
5  {
6    int to,next;
7    bool cut,visit;
8    int from;
9  } edge[MAXN-1<<1];
10 int head[MAXN],low[MAXN],dfn[MAXN],L;
11 bool visit[MAXN],cut[MAXN];
12 void init(int n)
13 {
14   L=0;
15   memset(head,-1,4*n);
16   memset(cut,0,4*n);
```

```
17    memset(visit,0,4*n);
18  }
19  void add_edge(int u,int v)
20  {
21    edge[L].from=u;
22    edge[L].cut=edge[L].visit=0;
23    edge[L].to=v;
24    edge[L].next=head[u];
25    head[u]=L++;
26  }
27  int idx;
28  void dfs(int u,int fu)
29  {
30    visit[u]=1;
31    low[u]=dfn[u]=idx++;
32    int tot=0;
33    for (int i=head[u]; i!=-1; i=edge[i].next)
34    {
35      int v=edge[i].to;
36      if (edge[i].visit)
37        continue;
38      edge[i].visit=edge[i^1].visit=1;
39      if (visit[v])
40      {
41        low[u]=dfn[v]>low[u]?low[u]:dfn[v];
42        continue;
43      }
44      dfs(v,u);
45      edge[i].cut=edge[i^1].cut=low[v]>dfn[u] || edge[i].cut;
46      if (u!=fu) cut[u]=low[v]>=dfn[u]?1:cut[u];
47      low[u]=low[v]>low[u]?low[u]:low[v];
48      tot++;
49    }
50    if (u==fu && tot>1) cut[u]=1;
51  }
52  int main()
53  {
54    int t;
55    scanf("%d",&t);
56    while (t--)
57    {
58      int n,m;
59      scanf("%d%d",&n,&m);
60      init(n);
61      for (int i=0; i<m; i++)
62      {
63        int u,v;
64        scanf("%d%d",&u,&v);
65        add_edge(--u,--v);
66        add_edge(v,u);
67      }
```

```
    for (int i=0; i<n; i++)
      if (!visit[i])
      {
        idx=0;
        dfs(i,i);
      }
  }
  return 0;
}
```

5.11 LCA

在线 LCA，bfs

```
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
const int NSIZE = 50000;
const int DEG = 20;
struct trees
{

  int fa[DEG];
  int head,deg;
} tree[NSIZE];
struct edges
{
  int to , next;
} edge[NSIZE];
struct states
{
  int u,fu,deg;
};
int L;
void add_edge(int x, int y)
{
  edge[L].to = y;
  edge[L].next = tree[x].head;
  tree[x].head = L++;
}
int Root;
void BFS(int s)
{
  queue<states> que;
  states st;
  st.deg=0;
  st.fu=st.u=s;
  que.push(st);
  while(!que.empty())
  {
    states st=que.front();
    que.pop();
```

```
40      tree[st.u].deg = st.deg;
41      tree[st.u].fa[0] = st.fu;
42      for (int i=1;i<DEG;i++)
43        tree[st.u].fa[i]=s;
44      for (int tmp=st.fu,num=1;tree[tmp].deg;tmp=tree[st.u].fa[num
           ++])
45        tree[st.u].fa[num]=tree[tmp].fa[num-1];
46      for(int i = tree[st.u].head ; i != -1; i = edge[i].next)
47      {
48        int v = edge[i].to;
49        if (v == st.fu) continue;
50        states nst;
51        nst.u=v;
52        nst.fu=st.u;
53        nst.deg=st.deg+1;
54        que.push(nst);
55      }
56    }
57  }
58  int LCA(int x, int y)
59  {
60    if(tree[x].deg > tree[y].deg) swap(x,y);
61    int hx=tree[x].deg,hy=tree[y].deg;
62    int tx=x,ty=y;
63    for (int det=hy-hx,i=0; det; det>>=1,i++)
64      if (det&1)
65        ty=tree[ty].fa[i];
66    if(tx == ty) return tx;
67    for (int i=DEG-1; i>=0; i--)
68    {
69      if(tree[tx].fa[i] == tree[ty].fa[i])
70        continue;
71      tx = tree[tx].fa[i];
72      ty = tree[ty].fa[i];
73    }
74    return tree[tx].fa[0];
75  }
76  int main()
77  {
78    int t;
79    scanf("%d",&t);
80    while(t--)
81    {
82      int n;
83      scanf("%d",&n);
84      L = 0;
85      for(int i = 0 ; i < n ; i++)
86        tree[i].head = -1;
87      for(int i = 0 ; i < n-1 ; i++)
88      {
89        int a,b;
```

```
90        scanf("%d%d",&a ,&b);
91        add_edge(a-1,b-1);
92        add_edge(b-1,a-1);
93      }
94      Root=0;
95      BFS(Root);
96      int a,b;
97      scanf("%d%d",&a,&b);
98      int lca=LCA(a-1,b-1)+1;
99      printf("%d\n",lca);
100   }
101   return 0;
102 }
```

5.12 稳定婚姻

假定有 n 个男生和 个女生，理想的拍拖状态就是对于每对情侣 (a,b)，找不到另一对情侣 (c,d) 使得 c 更喜欢 b,b 也更喜欢 c，同理，对 a 来说也没有 (e,f) 使得 a 更喜欢 e 而 e 更喜欢 a, 当然最后会有一些人落单。这样子一个状态可以称为理想拍拖状态，它也有一个专业的名词叫稳定婚姻。

求解这个问题可以用一个专有的算法，延迟认可算法，其核心就是让每个男生按自己喜欢的顺序逐个向女生表白，例如 leokan 向一个女生求爱，这个过程中，若这个女生没有男朋友，那么这个女生就暂时成为 leokan 的女朋友，或这个女生喜欢她现有男朋友的程度没有喜欢 leokan 高，这个女生也暂时成为 leokan 的女朋友，而她原有的男朋友则再将就找下一个次喜欢的女生来当女朋友。

```
1  #include<string.h>
2  #include<stdio.h>
3  #define N 1050
4  int boy[N][N];
5  int girl[N][N];
6  int ans[N];
7  int cur[N];
8  int n;
9  void getMarry(int g)
10 {
11   for (int i=ans[g]+1;i<n;i++)
12   {
13     int b=girl[g][i]-1;
14     if (cur[b]<0)
15     {
16       ans[g]=i;
17       cur[b]=g;
18       return;
19     }
20     int og=cur[b];
21     if (boy[b][og] > boy[b][g])
22     {
23       cur[b]=g;
24       ans[g]=i;
25       getMarry(og);
26       return;
```

```
27      }
28    }
29  };
30  int main()
31  {
32    int t,a;
33    scanf("%d",&t);
34    while(t--)
35    {
36      memset(girl,0,sizeof(girl));
37      memset(boy,0,sizeof(boy));
38      scanf("%d",&n);
39      for (int i=0;i<n;i++)
40        for (int j=0;j<n;j++)
41          scanf("%d",&girl[i][j]);
42      for (int i=0;i<n;i++)
43        for (int j=0;j<n;j++)
44        {
45          scanf("%d",&a);
46          boy[i][a-1]=j;
47        }
48      memset(cur,0xff,sizeof(cur));
49      memset(ans,0xff,sizeof(ans));
50      for (int i=0;i<n;i++)
51        getMarry(i);
52      for (int i=0;i<n;i++)
53        printf("%d\n",girl[i][ans[i]]);
54    }
55    return 0;
56  }
```

5.13 最小树形图

```
1  const int inf = 19921005;
2  int n,m,u,v,cost,dis[1001][1001],L;
3
4  void init(int n)
5  {
6    L = 0;
7    for (int i = 0; i < n; i++)
8      for (int j = 0; j < n; j++)
9        dis[i][j] = inf;
10 }
11
12 struct Edge
13 {
14   int u,v,cost;
15 };
16
17 Edge e[1001*1001];
18
```

```
19 int pre[1001],id[1001],visit[1001],in[1001];
20
21 int zhuliu(int root,int n,int m,Edge e[])
22 {
23   int res = 0,u,v;
24   while (true)
25   {
26     for (int i = 0; i < n; i++)
27       in[i] = inf;
28     for (int i = 0; i < m; i++)
29       if (e[i].u != e[i].v && e[i].cost < in[e[i].v])
30       {
31         pre[e[i].v] = e[i].u;
32         in[e[i].v] = e[i].cost;
33       }
34     for (int i = 0; i < n; i++)
35       if (i != root)
36         if (in[i] == inf)   return -1;
37     int tn = 0;
38     memset(id,-1,sizeof(id));
39     memset(visit,-1,sizeof(visit));
40     in[root] = 0;
41     for (int i = 0; i < n; i++)
42     {
43       res += in[i];
44       v = i;
45       while (visit[v] != i && id[v] == -1 && v != root)
46       {
47         visit[v] = i;
48         v = pre[v];
49       }
50       if(v != root && id[v] == -1)
51       {
52         for(int u = pre[v] ; u != v ; u = pre[u])
53           id[u] = tn;
54         id[v] = tn++;
55       }
56     }
57     if(tn == 0) break;
58     for (int i = 0; i < n; i++)
59       if (id[i] == -1)
60         id[i] = tn++;
61     for (int i = 0; i < m;)
62     {
63       int v = e[i].v;
64       e[i].u = id[e[i].u];
65       e[i].v = id[e[i].v];
66       if (e[i].u != e[i].v)
67         e[i++].cost -= in[v];
68       else
69         swap(e[i],e[--m]);
```

```
70      }
71      n = tn;
72      root = id[root];
73    }
74    return res;
75  }
76
77  int main()
78  {
79    freopen("in.txt","r",stdin);
80    while (scanf("%d%d",&n,&m) != EOF)
81    {
82      init(n);
83      for (int i = 0; i < m; i++)
84      {
85        scanf("%d%d%d",&u,&v,&cost);
86        if (u == v) continue;
87        dis[u][v] = min(dis[u][v],cost);
88      }
89      L = 0;
90      for (int i = 0; i < n; i++)
91        for (int j = 0; j < n; j++)
92          if (dis[i][j] != inf)
93          {
94            e[L].u = i;
95            e[L].v = j;
96            e[L++].cost = dis[i][j];
97          }
98      printf("%d\n",zhuliu(0,n,L,e));
99    }
100   return 0;
101 }
```

6 计算几何

6.1 注意事项

如果用整数小心越界（多次乘法？）
如果用浮点数判断的时候一定要用 eps！

6.2 基本函数

6.2.1 Point 定义

```
struct Point
{
  double x, y;
  Point() {}
  Point(double _x, double _y)
  {
    x = _x, y = _y;
  }
  Point operator -(const Point &b)const
  {
    return Point(x-b.x, y-b.y);
  }
  double operator *(const Point &b)const
  {
    return x*b.y-y*b.x;
  }
  double operator &(const Point &b)const
  {
    return x*b.x+y*b.y;
  }
  void transXY(double B)
  {
    double tx = x,ty = y;
    x = tx*cos(B)-ty*sin(B);
    y = tx*sin(B)+ty*cos(B);
  }
};
```

6.2.2 Line 定义

```
struct Line
{
  Point s, e;
  double k;
  Line() {}
  Line(Point _s, Point _e)
  {
    s = _s, e = _e;
    k = atan2(e.y-s.y, e.x-s.x);
  }
```

```
11    Point operator &(const Line &b)const
12    {
13      Point res = s;
14      //注意：有些题目可能会有直线相交或者重合情况
15      //可以把返回值改成 pair<Point,int> 来返回两直线的状态。
16      double t = ((s-b.s)*(b.s-b.e))/((s-e)*(b.s-b.e));
17      res.x += (e.x-s.x)*t;
18      res.y += (e.y-s.y)*t;
19      return res;
20    }
21  };
```

6.2.3 距离：点到直线距离

result: 点到直线最近点

```
1  Point NPT(Point P, Line L)
2  {
3    Point result;
4    double a, b, t;
5
6    a = L.e.x-L.s.x;
7    b = L.e.y-L.s.y;
8    t = ((P.x-L.s.x)*a+(P.y-L.s.y)*b)/(a*a+b*b);
9
10   result.x = L.s.x+a*t;
11   result.y = L.s.y+b*t;
12   return dist(P, result);
13 }
```

6.2.4 距离：点到线段距离

res：点到线段最近点

```
1  Point NearestPointToLineSeg(Point P, Line L)
2  {
3    Point result;
4    double a, b, t;
5
6    a = L.e.x-L.s.x;
7    b = L.e.y-L.s.y;
8    t = ( (P.x-L.s.x)*a+(P.y-L.s.y)*b )/(a*a+b*b);
9
10   if (t >= 0 && t <= 1)
11   {
12     result.x = L.s.x+a*t;
13     result.y = L.s.y+b*t;
14   }
15   else
16   {
17     if (dist(P,L.s) < dist(P,L.e))
18       result = L.s;
19     else
20       result = L.e;
```

```
21    }
22    return result;
23  }
```

旧版

```
1  double CalcDis(Point a,Point s,Point e) //点到线段距离
2  {
3    if (sgn((e-s)*(a-s)) < 0 || sgn((s-e)*(a-e)) < 0)
4      return min(dist(a,s),dist(a,e));
5    return abs(((s-a)*(e-a))/dist(s-e));
6  }
```

6.2.5 面积：多边形

点按逆时针排序。

```
1  double CalcArea(Point p[], int n)
2  {
3    double res = 0;
4    for (int i = 0; i < n; i++)
5      res += (p[i]*p[(i+1) % n])/2;
6    return res;
7  }
```

6.2.6 判断：线段相交

```
1  bool inter(Line l1,Line l2)
2  {
3    return
4    max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
5    max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
6    max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
7    max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
8    sgn((l2.s-l1.s)*(l1.e-l1.s))*sgn((l2.e-l1.s)*(l1.e-l1.s)) <= 0 &&
9    sgn((l1.s-l2.s)*(l2.e-l2.s))*sgn((l1.e-l2.s)*(l2.e-l2.s)) <= 0;
10 }
```

6.2.7 判断：点在线段上

```
1  bool OnSeg(Line a,Point b)
2  {
3    return ((a.s-b)*(a.e-b) == 0 &&
4        (b.x-a.s.x)*(b.x-a.e.x) <= 0 &&
5        (b.y-a.s.y)*(b.y-a.e.y) <= 0);
6  }
```

6.2.8 判断：点在多边形内

凸包且按逆时针排序

```
1  bool inPoly(Point a,Point p[],int n)
2  {
3    for (int i = 0;i < n;i++)
4      if ((p[i]-a)*(p[(i+1)%n]-a) < 0)
```

```
5       return false;
6     return true;
7   }
```

射线法, 多边形可以是凸的或凹的
poly 的顶点数目要大于等于 3
返回值为：
0 – 点在 poly 内
1 – 点在 poly 边界上
2 – 点在 poly 外

```
1   int inPoly(Point p,Point poly[], int n)
2   {
3     int i, count;
4     Line ray, side;
5
6     count = 0;
7     ray.s = p;
8     ray.e.y  = p.y;
9     ray.e.x  = -1;//-INF，注意取值防止越界！
10
11    for (i = 0; i < n; i++)
12    {
13      side.s = poly[i];
14      side.e = poly[(i+1)%n];
15
16      if(OnSeg(p, side))
17        return 1;
18
19      // 如果平行轴则不作考虑sidex
20      if (side.s.y == side.e.y)
21        continue;
22
23      if (OnSeg(side.s, ray))
24      {
25        if (side.s.y > side.e.y) count++;
26      }
27      else if (OnSeg(side.e, ray))
28      {
29        if (side.e.y > side.s.y) count++;
30      }
31      else if (inter(ray, side))
32      {
33        count++;
34      }
35    }
36    return ((count % 2 == 1) ? 0 : 2);
37  }
```

6.2.9 判断：两凸包相交

需要考虑这几个：一个凸包的点在另外一个图包内（包括边界）；一个凸包的某条边与另一个凸包某条边相交；如果凸包可能退化成点线还需要判断点在线段上和点和点重合。

6.2.10 排序：叉积极角排序

```
bool cmp(const Point& a,const Point& b)
{
  if (a.y*b.y <= 0)
  {
    if (a.y > 0 || b.y > 0) return a.y < b.y;
    if (a.y == 0 && b.y == 0) return a.x < b.x;
  }
  return a*b > 0;
}
```

6.3 三维几何

6.3.1 Point 定义

```
struct Point3D
{
  double x,y,z;
  Point3D() {}
  Point3D(double _x,double _y,double _z)
  {
    x = _x;
    y = _y;
    z = _z;
  }
  Point3D operator -(const Point3D& b)const
  {
    return Point3D(x-b.x,y-b.y,z-b.z);
  }
  Point3D operator *(const Point3D& b)const
  {
    return Point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
  }
  double operator &(const Point3D& b)const
  {
    return x*b.x+y*b.y+z*b.z;
  }
};
//模
double Norm(Point3D p)
{
  return sqrt(p&p);
}
//绕单位向量 V 旋转 θ 角度
Point3D Trans(Point3D pa,Point3D V,double theta)
{
  double s = sin(theta);
  double c = cos(theta);
  double x,y,z;
  x = V.x;
  y = V.y;
```

```
37    z = V.z;
38    Point3D pp =
39    Point3D(
40      (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z*(1-c)+y*s)*pa.z,
41      (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-x*s)*pa.z,
42      (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c)+c)*pa.z);
43    return pp;
44  }
```

6.3.2 经度纬度转换

直角坐标系与极坐标系转换：

$$\begin{cases} x = r \times sin\theta \times cos\varphi \\ y = r \times sin\theta \times sin\varphi \\ z = r \times cos\theta \end{cases} \quad \begin{cases} r = \sqrt{x \times 2 + y \times 2 + z \times 2} \\ \varphi = arctan(\frac{y}{x}) \\ \theta = arccos(\frac{z}{r}) \end{cases} \quad r \in [0, +\infty), \varphi \in [0, 2\pi], \theta \in [0, \pi]$$

经度维度转换（$lat1 \in (-\frac{\pi}{2}, \frac{\pi}{2}), lng1 \in (-\pi, \pi)$）

```
1  Point3D getPoint3D(double lat,double lng,double r)
2  {
3    lat += pi/2;
4    lng += pi;
5    return
6      Point3D(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),r*cos(lat));
7  }
```

6.3.3 判断：直线相交

```
1  bool LineIntersect(Line3D L1, Line3D L2)
2  {
3    Point3D s = L1.s-L1.e;
4    Point3D e = L2.s-L2.e;
5    Point3D p  = s*e;
6    if (ZERO(p)) return false;    //是否平行
7    p = (L2.s-L1.e)*(L1.s-L1.e);
8    return ZERO(p&L2.e);     //是否共面
9  }
```

6.3.4 判断：线段相交

需要先判断是否在一个平面上：

```
1  bool inter(Point a,Point b,Point c,Point d)
2  {
3    Point ret = (a-b)*(c-d);
4    Point t1 = (b-a)*(c-a);
5    Point t2 = (b-a)*(d-a);
6    Point t3 = (d-c)*(a-c);
7    Point t4 = (d-c)*(b-c);
8    return sgn(t1&ret)*sgn(t2&ret) < 0 &&
9           sgn(t3&ret)*sgn(t4&ret) < 0;
10 }
```

6.3.5 判断：三维向量是否为 0

```
inline bool ZERO(Point3D p)
{
  return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
}
```

6.3.6 判断：点在直线上

```
bool OnLine(Point3D p, Line3D L)
{
  return ZERO((p-L.s)*(L.e-L.s));
}
```

6.3.7 判断：点在线段上

```
bool OnSeg(Point3D p, Line3D L)
{
  return (ZERO((L.s-p)*(L.e-p)) &&
    EQ(Norm(p-L.s)+Norm(p-L.e),Norm(L.e-L.s)));
}
```

6.3.8 距离：点到直线

```
double Distance(Point3D p, Line3D L)
{
  return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
}
```

6.3.9 夹角

返回值是 $[0,\pi]$ 之间的弧度

```
double Inclination(Line3D L1, Line3D L2)
{
  Point3D u = L1.e - L1.s;
  Point3D v = L2.e - L2.s;
  return acos( (u & v) / (Norm(u)*Norm(v)) );
}
```

6.4 圆

6.4.1 面积：两圆相交

圆不可包含

```
double dis(int x,int y)
{
  return sqrt((double)(x*x+y*y));
}
double area(int x1,int y1,int x2,int y2,double r1,double r2)
{
  double s=dis(x2-x1,y2-y1);
  if(r1+r2<s) return 0;
  else if(r2-r1>s) return PI*r1*r1;
  else if(r1-r2>s) return PI*r2*r2;
```

```
11    double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
12    double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
13    return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
14  }
```

6.4.2 三角形外接圆

```
 1  void CircumscribedCircle()
 2  {
 3    for (int i = 0; i < 3; i++)
 4      scanf("%lf%lf",&p[i].x,&p[i].y);
 5    tp = Point((p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
 6    l[0] = Line(tp,Point(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x)));
 7    tp = Point((p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
 8    l[1] = Line(tp,Point(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x)));
 9    tp = LineToLine(l[0],l[1]);
10    r = Point(tp,p[0]).Length();
11    printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
12  }
```

6.4.3 三角形内切圆

```
 1  void InscribedCircle()
 2  {
 3    for (int i = 0; i < 3; i++)
 4      scanf("%lf%lf",&p[i].x,&p[i].y);
 5    if (xmult(Point(p[0],p[1]),Point(p[0],p[2])) < 0)
 6      swap(p[1],p[2]);
 7    for (int i = 0; i < 3; i++)
 8      len[i] = Point(p[i],p[(i+1)%3]).Length();
 9    tr = (len[0]+len[1]+len[2])/2;
10    r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/tr);
11    for (int i = 0; i < 2; i++)
12    {
13      v = Point(p[i],p[i+1]);
14      tv = Point(-v.y,v.x);
15      tr = tv.Length();
16      tv = Point(tv.x*r/tr,tv.y*r/tr);
17      tp = Point(p[i].x+tv.x,p[i].y+tv.y);
18      l[i].s = tp;
19      tp = Point(p[i+1].x+tv.x,p[i+1].y+tv.y);
20      l[i].e = tp;
21    }
22    tp = LineToLine(l[0],l[1]);
23    printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
24  }
```

6.4.4 点对圆的两个切点

```
1  void calc_qie(Point poi,Point o,double r,Point &result1,Point &
     result2)
2  {
```

```
  double line = sqrt((poi.x-o.x)*(poi.x-o.x)+(poi.y-o.y)*(poi.y-o.y
     ));
  double angle = acos(r/line);
  Point unitvector,lin;
  lin.x = poi.x-o.x;
  lin.y = poi.y-o.y;
  unitvector.x = lin.x/sqrt(lin.x*lin.x+lin.y*lin.y)*r;
  unitvector.y = lin.y/sqrt(lin.x*lin.x+lin.y*lin.y)*r;
  result1 = unitvector.Rotate(-angle);
  result2 = unitvector.Rotate(angle);
  result1.x += o.x;
  result1.y += o.y;
  result2.x += o.x;
  result2.y += o.y;
}
```

6.4.5 两圆公切点

```
void Gao()
{
  tn = 0;
  Point a,b,vab;
  double tab,tt,dis,theta;
  for (int i = 0; i < tc; i++)
    for (int j = 0; j < tc; j++)
      if (i != j)
      {
        a = c[i];
        b = c[j];
        vab = Point(a,b);
        tab = atan2(vab.y,vab.x);
        dis = sqrt(vab.x*vab.x+vab.y*vab.y);
        if (b.r > a.r)
          tt = asin((b.r-a.r)/dis);
        else
          tt = -asin((a.r-b.r)/dis);
        theta = tab+pi/2+tt;
        tp[tn++] = Point(a.x+a.r*cos(theta),a.y+a.r*sin(theta));
        tp[tn++] = Point(b.x+b.r*cos(theta),b.y+b.r*sin(theta));
      }
}
```

6.4.6 两圆交点

```
lab = Point(p[j].x-p[i].x,p[j].y-p[i].y);
AB = lab.Length();
AC = cr[i];
BC = cr[j];

if (cmp(AB+AC,BC) <= 0) continue;//包含
if (cmp(AB+BC,AC) <= 0) continue;
if (cmp(AB,AC+BC) > 0)  continue;//相离

```

```
theta = atan2(lab.y,lab.x);
fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
a0 = theta-fai;
if (cmp(a0,-pi) < 0)  a0 += 2*pi;
a1 = theta+fai;
if (cmp(a1,pi) > 0) a1 -= 2*pi;
//答案
xp[totp++] = Point(p[i].x+cr[i]*cos(a0),p[i].y+cr[i]*sin(a0));
xp[totp++] = Point(p[i].x+cr[i]*cos(a1),p[i].y+cr[i]*sin(a1));
```

6.5 矩阵

6.5.1 基本矩阵

按向量 $\overrightarrow{(x,y,z)}$ 平移：

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

按比例 (x,y,z) 缩放：

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

绕单位向量 $\overrightarrow{(x,y,z)}$ 旋转 $angle$ 角度：

$$\begin{pmatrix} x^2\times(1-c)+c & x\times y\times(1-c)-z\times s & x\times z\times(1-c)+y\times s & 0 \\ y\times x\times(1-c)+z\times s & y^2\times(1-c)+c & y\times z\times(1-c)-x\times s & 0 \\ x\times z\times(1-c)-y\times s & y\times z\times(1-c)+x\times s & z^2\times(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{cases} s=sin(angle) \\ c=cos(angle) \end{cases}$$

以上矩阵变换都把点当作列向量，旋转角度的正负由右手定则决定

6.5.2 刘汝佳的几何教室

```
const double pi = acos(-1.0);

int n,m,q;
struct Point
{
  double a,b,c,d;
};
Point p[50000],f[50000];

double a,b,c,theta,mt[4][4],tmp[4][4],tmt[4][4],rmt[4][8];
char com[20];

void TRANSLATE()
```

```
14 {
15   memset(tmt,0,sizeof(tmt));
16   tmt[0][0] = tmt[1][1] = tmt[2][2] = tmt[3][3] = 1;
17   tmt[3][0] = a;
18   tmt[3][1] = b;
19   tmt[3][2] = c;
20   memset(tmp,0,sizeof(tmp));
21   for (int i = 0; i < 4; i++)
22     for (int j = 0; j < 4; j++)
23       for (int k = 0; k < 4; k++)
24         tmp[i][j] += mt[i][k]*tmt[k][j];
25   for (int i = 0; i < 4; i++)
26     for (int j = 0; j < 4; j++)
27       mt[i][j] = tmp[i][j];
28 }
29
30 void ROTATE()
31 {
32   theta = -theta*pi/180;
33   memset(tmt,0,sizeof(tmt));
34   tmt[3][3] = 1;
35   tmt[0][0] = cos(theta)+(1-cos(theta))*a*a;
36   tmt[1][0] = (1-cos(theta))*a*b+c*sin(theta);
37   tmt[2][0] = (1-cos(theta))*a*c-b*sin(theta);
38   tmt[0][1] = (1-cos(theta))*a*b-c*sin(theta);
39   tmt[1][1] = cos(theta)+(1-cos(theta))*b*b;
40   tmt[2][1] = (1-cos(theta))*b*c+a*sin(theta);
41   tmt[0][2] = (1-cos(theta))*a*c+b*sin(theta);
42   tmt[1][2] = (1-cos(theta))*b*c-a*sin(theta);
43   tmt[2][2] = cos(theta)+(1-cos(theta))*c*c;
44   memset(tmp,0,sizeof(tmp));
45   for (int i = 0; i < 4; i++)
46     for (int j = 0; j < 4; j++)
47       for (int k = 0; k < 4; k++)
48         tmp[i][j] += mt[i][k]*tmt[k][j];
49   for (int i = 0; i < 4; i++)
50     for (int j = 0; j < 4; j++)
51       mt[i][j] = tmp[i][j];
52 }
53
54 void SCALE()
55 {
56   memset(tmt,0,sizeof(tmt));
57   tmt[0][0] = a;
58   tmt[1][1] = b;
59   tmt[2][2] = c;
60   tmt[3][3] = 1;
61   memset(tmp,0,sizeof(tmp));
62   for (int i = 0; i < 4; i++)
63     for (int j = 0; j < 4; j++)
64       for (int k = 0; k < 4; k++)
```

```
65          tmp[i][j] += mt[i][k]*tmt[k][j];
66    for (int i = 0; i < 4; i++)
67      for (int j = 0; j < 4; j++)
68        mt[i][j] = tmp[i][j];
69  }
70
71  void solvep(Point p)
72  {
73    memset(tmt,0,sizeof(tmt));
74    tmt[0][0] = p.a;
75    tmt[0][1] = p.b;
76    tmt[0][2] = p.c;
77    tmt[0][3] = 1;
78    memset(tmp,0,sizeof(tmp));
79    for (int i = 0; i < 1; i++)
80      for (int j = 0; j < 4; j++)
81        for (int k = 0; k < 4; k++)
82          tmp[i][j] += tmt[i][k]*mt[k][j];
83    printf("%.2f %.2f %.2f\n",tmp[0][0],tmp[0][1],tmp[0][2]);
84  }
85
86  void solvef(Point f)
87  {
88    memset(tmt,0,sizeof(tmt));
89    tmt[0][0] = f.a;
90    tmt[1][0] = f.b;
91    tmt[2][0] = f.c;
92    tmt[3][0] = 0;
93    memset(tmp,0,sizeof(tmp));
94    for (int i = 0;i < 4;i++)
95      for (int j = 0;j < 1;j++)
96        for (int k = 0;k < 4;k++)
97          tmp[i][j] += mt[i][k]*tmt[k][j];
98    tmp[3][0] += f.d;
99    double kk = tmp[0][0]*tmp[0][0]+tmp[1][0]*tmp[1][0]+tmp[2][0]*tmp
        [2][0];
100   kk = sqrt(1/kk);
101   for (int i = 0;i < 4;i++)
102     printf("%.2f ",tmp[i][0]*kk);
103   printf("\n");
104 }
105
106 void solvermt()
107 {
108   memset(rmt,0,sizeof(rmt));
109   for (int i = 0;i < 4;i++)
110     for (int j = 0;j < 4;j++)
111       rmt[i][j] = mt[i][j];
112   rmt[0][4] = rmt[1][5] = rmt[2][6] = rmt[3][7] = 1;
113   for (int i = 0;i < 4;i++)
114   {
```

```
115     for (int j = i;j < 4;j++)
116       if (fabs(rmt[j][i]) > 1e-8)
117       {
118         for (int k = i;k < 8;k++)
119           swap(rmt[i][k],rmt[j][k]);
120         break;
121       }
122     double tt = rmt[i][i];
123     for (int j = i;j < 8;j++)
124       rmt[i][j] /= tt;
125     for (int j = 0;j < 4;j++)
126       if (i != j)
127       {
128         tt = rmt[j][i];
129         for (int k = i;k < 8;k++)
130           rmt[j][k] -= rmt[i][k]*tt;
131       }
132   }
133   for (int i = 0;i < 4;i++)
134     for (int j = 0;j < 4;j++)
135       mt[i][j] = rmt[i][4+j];
136 }
137
138 int main()
139 {
140   scanf("%d%d%d",&n,&m,&q);
141   for (int i = 0; i < n; i++)
142     scanf("%lf%lf%lf",&p[i].a,&p[i].b,&p[i].c);
143   for (int i = 0; i < m; i++)
144     scanf("%lf%lf%lf%lf",&f[i].a,&f[i].b,&f[i].c,&f[i].d);
145   memset(mt,0,sizeof(mt));
146   mt[0][0] = mt[1][1] = mt[2][2] = mt[3][3] = 1;
147   for (int i = 0; i < q; i++)
148   {
149     scanf("%s",com);
150     if (strcmp(com,"TRANSLATE") == 0)
151     {
152       scanf("%lf%lf%lf",&a,&b,&c);
153       TRANSLATE();
154     }
155     else if (strcmp(com,"ROTATE") == 0)
156     {
157       scanf("%lf%lf%lf%lf",&a,&b,&c,&theta);
158       ROTATE();
159     }
160     else if (strcmp(com,"SCALE") == 0)
161     {
162       scanf("%lf%lf%lf",&a,&b,&c);
163       SCALE();
164     }
165   }
```

```
166   //处理点
167   for (int i = 0; i < n; i++)
168     solvep(p[i]);
169   //处理面
170   solvermt();
171   for (int i = 0; i < m; i++)
172     solvef(f[i]);
173   return 0;
174 }
```

6.6 凸包

得到的凸包按照逆时针方向排序。

```
1  //判断是否是共点或者共线用
2  bool conPoint(Point p[],int n)
3  {
4    for (int i = 1;i < n;i++)
5      if (p[i].x != p[0].x || p[i].y != p[0].y)
6        return false;
7    return true;
8  }
9  bool conLine(Point p[],int n)
10 {
11   for (int i = 2;i < n;i++)
12     if ((p[i]-p[0])*(p[1]-p[0]) != 0)
13       return false;
14   return true;
15 }
16
17 bool GScmp(Point a, Point b)
18 {
19   if (fabs(a.x - b.x) < eps)
20     return a.y < b.y - eps;
21   return a.x < b.x - eps;
22 }
23
24 void GS(Point p[],int n,Point res[],int &resn)
25 {
26   resn = 0;
27   int top = 0;
28   sort(p,p+n,GScmp);
29
30   if (conPoint(p,n))
31   {
32     res[resn++] = p[0];
33     return;
34   }
35   if (conLine(p,n))
36   {
37     res[resn++] = p[0];
38     res[resn++] = p[n-1];
39     return;
```

```
40    }
41
42    for (int i = 0;i < n;)
43      if (resn < 2 ||
44        (res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
45        res[resn++] = p[i++];
46      else
47        --resn;
48    top = resn-1;
49    for (int i = n-2;i >= 0;)
50      if (resn < top+2 ||
51        res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
52        res[resn++] = p[i--];
53      else
54        --resn;
55    resn--;
56  }
```

6.7 精度问题

6.7.1 浮点数为啥会有精度问题

浮点数 (以 C/C++ 为准)，一般用的较多的是 float、double。

	占字节数	数值范围	十进制精度位数
float	4	$-3.4e-38 \sim 3.4e38$	$6 \sim 7$
double	8	$-1.7e-308 \sim 1.7e308$	$14 \sim 15$

如果内存不是很紧张或者精度要求不是很低，一般选用 double。 14 位的精度 (是有效数字位，不是小数点后的位数) 通常够用了。注意，问题来了，数据精度位数达到了 14 位，但有些浮点运算的结果精度并达不到这么高，可能准确的结果只有 $10 \sim 12$ 位左右。那低几位呢？自然就是不可预料的数字了。这给我们带来这样的问题：即使是理论上相同的值，由于是经过不同的运算过程得到的，他们在低几位有可能 (一般来说都是) 是不同的。这种现象看似没太大的影响，却会一种运算产生致命的影响： ==。恩，就是判断相等。注意，C/C++ 中浮点数的 == 需要完全一样才能返回 true。

6.7.2 eps

eps 缩写自 epsilon，表示一个小量，但这个小量又要确保远大于浮点运算结果的不确定量。eps 最常见的取值是 $1e-8$ 左右。引入 eps 后，我们判断两浮点数 a、b 相等的方式如下:

```
1  int sgn(double a){return a < -eps ? -1 : a < eps ? 0 : 1;}
```

这样，我们才能把相差非常近的浮点数判为相等；同时把确实相差较大 (差值大于 eps) 的数判为不相等。
养成好习惯，尽量不要再对浮点数做 == 判断。

6.7.3 eps 带来的函数越界

如果 sqrt(a), asin(a), acos(a) 中的 a 是你自己算出来并传进来的，那就得小心了。
如果 a 本来应该是 0 的，由于浮点误差，可能实际是一个绝对值很小的负数（比如 $-1e-12$），这样 sqrt(a) 应得 0 的，直接因 a 不在定义域而出错。
类似地，如果 a 本来应该是 ± 1, 则 asin(a)、acos(a) 也有可能出错。
因此，对于此种函数，必需事先对 a 进行校正。

6.7.4 输出陷阱 I

现在考虑一种情况，题目要求输出保留两位小数。有个 case 的正确答案的精确值是 0.005, 按理应该输出 0.01，但你的结果可能是 0.005000000001(恭喜)，也有可能是 0.004999999999(悲剧)，如果按照 `printf("%.2lf", a)` 输出，那你的遭遇将和括号里的字相同。
解决办法是，如果 a 为正，则输出 $a + eps$, 否则输出 $a - eps$

6.7.5 输出陷阱 II

ICPC 题目输出有个不成文的规定 (有时也成文)，不要输出： -0.000
那我们首先要弄清，什么时候按 `printf("%.3lf", a)` 输出会出现这个结果。
直接给出结果好了： $a \in (-0.000499999\cdots, -0.000\cdots 1)$
所以，如果你发现 a 落在这个范围内，请直接输出 0.000。更保险的做法是用 `sprintf` 直接判断输出结果是不是 -0.000 再予处理。

6.7.6 范围越界

请注意，虽然 double 可以表示的数的范围很大，却不是不穷大，上面说过最大是 $1e308$。所以有些时候你得小心了，比如做连乘的时候，必要的时候要换成对数的和。

6.7.7 关于 set

经观察,set 不是通过 == 来判断相等的，是通过 $<$ 来进行的，具体说来，只要 $a < b$ 和 $b < a$ 都不成立，就认为 a 和 b 相等，可以发现，如果将小于定义成：

```
1 bool operator < (const Dat dat)const{return val < dat.val - eps;}
```

就可以解决问题了。(基本类型不能重载运算符，所以封装了下)

6.7.8 输入值波动过大

这种情况不常见，不过可以帮助你更熟悉 eps。假如一道题输入说，给一个浮点数 a, $1e-20 < a < 1e20$。那你还敢用 $1e-8$ 做 eps 么？合理的做法是把 eps 按照输入规模缩放到合适大小。

6.7.9 一些建议

容易产生较大浮点误差的函数有 asin、acos。欢迎尽量使用 atan2。
另外，如果数据明确说明是整数，而且范围不大的话，使用 int 或者 long long 代替 double 都是极佳选择，因为就不存在浮点误差了

7 搜索

7.1 Dancing Links

7.1.1 估价函数

```
int h()
{
  bool vis[100];
  memset(vis,false,sizeof(vis));
  int i,j,k,res=0,mi,col;
  while(1)
  {
    mi=inf;
    for(i=R[head]; i!=head&&i<=2*n; i=R[i])
      if(mi>nk[i]&&!vis[i])
      {
        mi=nk[i];
        col=i;
      }
    if(mi==inf)
      break;
    res++;
    vis[col]=true;
    for(j=D[col]; j!=col; j=D[j])
      for(k=R[j]; k!=j; k=R[k])
      {
        if(C[k]>2*n)
          continue;
        vis[C[k]]=true;
      }
  }
  return res;
}
```

7.1.2 DLX

```
void remove1(int col)
{
  int i,j;
  L[R[col]]=L[col];
  R[L[col]]=R[col];
  for(i=D[col];i!=col;i=D[i])
  {
    L[R[i]]=L[i];
    R[L[i]]=R[i];
  }
}
void remove2(int col)
{
  int i,j;
  L[R[col]]=L[col];
```

```
16    R[L[col]]=R[col];
17    for(i=D[col];i!=col;i=D[i])
18    {
19      for(j=R[i];j!=i;j=R[j])
20      {
21        U[D[j]]=U[j];
22        D[U[j]]=D[j];
23        --nk[C[j]];
24      }
25    }
26  }
27  void resume1(int col)
28  {
29    int i,j;
30    for(i=U[col];i!=col;i=U[i])
31    {
32      L[R[i]]=i;
33      R[L[i]]=i;
34    }
35    L[R[col]]=col;
36    R[L[col]]=col;
37  }
38  void resume2(int col)
39  {
40    int i,j;
41    for(i=U[col];i!=col;i=U[i])
42    {
43      for(j=L[i];j!=i;j=L[j])
44      {
45        ++nk[C[j]];
46        U[D[j]]=j;
47        D[U[j]]=j;
48      }
49    }
50    L[R[col]]=col;
51    R[L[col]]=col;
52  }
53  int h()
54  {
55    bool vis[100];
56    memset(vis,false,sizeof(vis));
57    int i,j,k,res=0,mi,col;
58    while(1)
59    {
60      mi=inf;
61      for(i=R[head];i!=head&&i<=2*n;i=R[i])
62        if(mi>nk[i]&&!vis[i])
63        {
64          mi=nk[i];
65          col=i;
66        }
```

```
67      if(mi==inf)
68        break;
69      res++;vis[col]=true;
70      for(j=D[col];j!=col;j=D[j])
71        for(k=R[j];k!=j;k=R[k])
72        {
73          if(C[k]>2*n)
74            continue;
75          vis[C[k]]=true;
76        }
77    }
78    return res;
79  }
80  bool DLX(int d,int deep)
81  {
82    if(d+h()>deep) return false;
83    if(R[head]==head||R[head]>2*n)
84      return true;
85    if(d>=deep)
86      return false;
87    int col,ma=inf;
88    int i,j;
89    for(i=R[head];i!=head&&i<=2*n;i=R[i])
90      if(nk[i]<ma)
91      {
92        col=i;
93        ma=nk[i];
94      }
95    remove1(col);
96    for(i=D[col];i!=col;i=D[i])
97    {
98      int flag=1;
99      for(j=R[i];;j=R[j])
100     {
101       if(j==R[i]&&!flag)
102         break;
103       U[D[j]]=U[j];
104       D[U[j]]=D[j];
105       if(C[j]>2*n)
106         remove2(C[j]);
107       else
108         remove1(C[j]);
109       flag=0;
110     }
111     if(DLX(d+1,deep))
112       return true;
113     flag=1;
114     for(j=L[i];;j=L[j])
115     {
116       if(j==L[i]&&!flag)
117         break;
```

```
118       if(C[j]>2*n)
119         resume2(C[j]);
120       else
121         resume1(C[j]);
122       U[D[j]]=j;
123       D[U[j]]=j;
124       flag=0;
125     }
126   }
127   resume1(col);
128   return false;
129 }
```

8 动态规划

8.1 斜率优化

```
#include<cstdio>
#include<algorithm>
using namespace std;
int a[1000],sum[1001],dp[1000][1000];
int deque[1000];
const int inf=0x7fffffff;
int N,s,t;
int calc(int i,int l,int j)//决策值计算
{
  return dp[j][l-1]-(sum[i]-sum[j])*(sum[N]-sum[i]);
}
bool check(int i,int l)//尾端判断
{
  int k1=deque[t-1],k2=deque[t-2];
  return (long long)(dp[k1][l]-dp[k2][l])*(sum[i]-sum[k1])>(long
     long)(dp[i][l]-dp[k1][l])*(sum[k1]-sum[k2]);
}
int main()
{
  int n,m;
  while (scanf("%d%d",&n,&m),n)
  {
    for (int i=0; i<n; i++)
      scanf("%d",&a[i]);
    N=n;
    sum[0]=0;
    for (int i=0; i<n; i++)
      sum[i+1]=sum[i]+a[i];
    dp[0][0]=0;
    for (int i=0; i<n; i++)
      for (int j=i+1; j<n; j++)
        dp[0][0]+=a[i]*a[j];
    for (int i=1; i<n; i++)
      dp[i][0]=inf;
    for (int i=1; i<n; i++)
    {
      dp[i][1]=inf;
      for (int j=0; j<i; j++)
        dp[i][1]=min(dp[i][1],calc(i,1,j));
    }
    for (int l=2; l<=m; l++)
    {
      s=t=0;//双端队列清空
      for (int i=l; i<n; i++)
      {
        while (t-s>1 && check(i-1,l-1)) t--;
        deque[t++]=i-1;//决策加入
```

```
47          while (t-s>1 && calc(i,l,deque[s])>calc(i,l,deque[s+1])) s
               ++;
48          dp[i][l]=calc(i,l,deque[s]);
49        }
50      }
51      int ans=0x7fffffff;
52      for (int i=m; i<n; i++)
53        ans=min(ans,dp[i][m]);
54      printf("%d\n",ans);
55    }
56    return 0;
57  }
```

8.2 RMQ 二版

```
1  void init()
2  {
3    int i,j;
4    int n=N,k=1,l=0;
5    for (i=0; i<n; i++)
6    {
7      f[i][0]=ele[i].num;
8      if (i+1>k*2)
9      {
10       k*=2;
11       l++;
12     }
13     lent[i+1]=l;
14   }
15   for (j=1; (1<<j)-1<n; j++)
16     for (i=0; i+(1<<j)-1<n; i++)
17       f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
18 }
19 int fint(int x,int y)
20 {
21   int k=lent[y-x+1];
22   return max(f[x][k],f[y-(1<<k)+1][k]);
23 }
```

9 杂物

9.1 Java

9.1.1 文件操作

```
import java.io.*;
import java.util.*;
import java.math.*;
import java.text.*;

public class Main
{

  public static void main(String args[]) throws
     FileNotFoundException, IOException
  {
    Scanner sc = new Scanner(new FileReader("a.in"));
    PrintWriter pw = new PrintWriter(new FileWriter("a.out"));
    int n,m;
    n=sc.nextInt();//读入下一个INT
    m=sc.nextInt();

    for(ci=1; ci<=c; ++ci)
    {
      pw.println("Case #"+ci+": easy for output");
    }

    pw.close();//关闭流并释放，这个很重要，否则是没有输出的
    sc.close();//关闭流并释放
  }
}
```

9.1.2 优先队列

```
PriorityQueue queue = new PriorityQueue( 1, new Comparator()
{
  public int compare( Point a, Point b )
  {
  if( a.x < b.x || a.x == b.x && a.y < b.y )
    return -1;
  else if( a.x == b.x && a.y == b.y )
    return 0;
  else
    return 1;
  }
});
```

9.1.3 Map

```
Map map = new HashMap();
map.put("sa","dd");
String str = map.get("sa").toString;
```

```

for(Object obj : map.keySet()){
  Object value = map.get(obj );
}
```

9.1.4 sort

```
static class cmp implements Comparator
{
  public int compare(Object o1,Object o2)
  {
  BigInteger b1=(BigInteger)o1;
  BigInteger b2=(BigInteger)o2;
  return b1.compareTo(b2);
  }
}
public static void main(String[] args) throws IOException
{
  Scanner cin = new Scanner(System.in);
  int n;
  n=cin.nextInt();
  BigInteger[] seg = new BigInteger[n];
  for (int i=0;i<n;i++)
  seg[i]=cin.nextBigInteger();
  Arrays.sort(seg,new cmp());
}
```

9.2 C++&STL 常用函数

9.2.1 lower_bound/upper_bound

不解释

```
iterator lower_bound(const key_type &key )
\\返回一个迭代器，指向键值 >= key 的第一个元素。
iterator upper_bound(const key_type &key )
\\返回一个迭代器，指向键值 > key 的第一个元素。

#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int main () {
  int myints[] = {10,20,30,30,20,10,10,20};
  vector<int> v(myints,myints+8);
  // 10 20 30 30 20 10 10 20
  vector<int>::iterator low,up;

  sort (v.begin(), v.end());
  // 10 10 10 20 20 20 30 30

  low=lower_bound (v.begin(), v.end(), 20);
```

```
21    // 10 10 10 20 20 20 30 30
22    //          ^
23    up= upper_bound (v.begin(), v.end(), 20);
24    // 10 10 10 20 20 20 30 30
25    //                   ^
26
27    cout << "lower_bound at position " << int(low- v.begin()) << endl
         ;
28    cout << "upper_bound at position " << int(up - v.begin()) << endl
         ;
29
30    return 0;
31  }
```

Output:

```
1  lower_bound at position 3
2  upper_bound at position 6
```

9.2.2 rotate

把数组后一半搬到前面

```
1  template <class ForwardIterator>
2    void rotate ( ForwardIterator first, ForwardIterator middle,
3                  ForwardIterator last );
```

9.2.3 nth_element

```
1  template <class RandomAccessIterator>
2    void nth_element ( RandomAccessIterator first,
       RandomAccessIterator nth,
3                       RandomAccessIterator last );
4
5  template <class RandomAccessIterator, class Comapre>
6    void nth_element ( RandomAccessIterator first,
       RandomAccessIterator nth,
7                       RandomAccessIterator last, Compare comp );
```

9.2.4 bitset

取用

```
1  bitset<4> mybits;
2
3  mybits[1]=1;              // 0010
4  mybits[2]=mybits[1];      // 0110
```

翻转

```
bitset<4> mybits (string("0001"));

cout << mybits.flip(2) << endl;     // 0101
cout << mybits.flip() << endl;      // 1010
```

运算

```
bitset<4> first (string("1001"));
bitset<4> second (string("0011"));

cout << (first^=second) << endl;          // 1010 (XOR,assign)
cout << (first&=second) << endl;          // 0010 (AND,assign)
cout << (first|=second) << endl;          // 0011 (OR,assign)

cout << (first<<=2) << endl;              // 1100 (SHL,assign)
cout << (first>>=1) << endl;              // 0110 (SHR,assign)

cout << (~second) << endl;                // 1100 (NOT)
cout << (second<<1) << endl;              // 0110 (SHL)
cout << (second>>1) << endl;              // 0001 (SHR)

cout << (first==second) << endl;          // false (0110==0011)
cout << (first!=second) << endl;          // true  (0110!=0011)

cout << (first&second) << endl;           // 0010
cout << (first|second) << endl;           // 0111
cout << (first^second) << endl;           // 0101
```

9.2.5 multimap

遍历

```
multimap<char,int> mymm;
multimap<char,int>::iterator it;
char c;

mymm.insert(pair<char,int>('x',50));
mymm.insert(pair<char,int>('y',100));
mymm.insert(pair<char,int>('y',150));
mymm.insert(pair<char,int>('y',200));
mymm.insert(pair<char,int>('z',250));
mymm.insert(pair<char,int>('z',300));

for (c='x'; c<='z'; c++)
{
  cout << "There are " << (int)mymm.count(c);
  cout << " elements with key " << c << ":";
  for (it=mymm.equal_range(c).first; it!=mymm.equal_range(c).second
     ; ++it)
    cout << " " << (*it).second;
  cout << endl;
```

```
}
/*
Output:

There are 1 elements with key x: 50
There are 3 elements with key y: 100 150 200
There are 2 elements with key z: 250 300
*/
```

二分查找

```
multimap<char,int> mymultimap;
multimap<char,int>::iterator it,itlow,itup;

mymultimap.insert(pair<char,int>('a',10));
mymultimap.insert(pair<char,int>('b',121));
mymultimap.insert(pair<char,int>('c',1001));
mymultimap.insert(pair<char,int>('c',2002));
mymultimap.insert(pair<char,int>('d',11011));
mymultimap.insert(pair<char,int>('e',44));

itlow=mymultimap.lower_bound ('b');  // itlow points to b
itup=mymultimap.upper_bound ('d');   // itup points to e (not d)

// print range [itlow,itup):
for ( it=itlow ; it != itup; it++ )
  cout << (*it).first << " => " << (*it).second << endl;

/*
Output:

b => 121
c => 1001
c => 2002
d => 11011
*/
```

删除

```
multimap<char,int> mymultimap;
multimap<char,int>::iterator it;

// insert some values:
mymultimap.insert(pair<char,int>('a',10));
mymultimap.insert(pair<char,int>('b',20));
mymultimap.insert(pair<char,int>('b',30));
mymultimap.insert(pair<char,int>('c',40));
mymultimap.insert(pair<char,int>('d',50));
mymultimap.insert(pair<char,int>('d',60));
mymultimap.insert(pair<char,int>('e',70));
mymultimap.insert(pair<char,int>('f',80));

```

```
14 it=mymultimap.find('b');
15 mymultimap.erase (it);
16 // erasing by iterator (1 element)
17
18 mymultimap.erase ('d');
19 // erasing by key (2 elements)
20
21 it=mymultimap.find ('e');
22 mymultimap.erase ( it, mymultimap.end() );
23 // erasing by range
24
25 // show content:
26 for ( it=mymultimap.begin() ; it != mymultimap.end(); it++ )
27   cout << (*it).first << " => " << (*it).second << endl;
28
29 /*
30 Output:
31
32 a => 10
33 b => 30
34 c => 40
35 */
```

9.3 位运算

9.3.1 基本操作

注意括号

功能	示例	位运算
去掉最后一位	$(101101 \to 10110)$	x shr 1
在最后加一个 0	$(101101 \to 1011010)$	x shl 1
在最后加一个 1	$(101101 \to 1011011)$	x shl 1+1
把最后一位变成 1	$(101100 \to 101101)$	x or 1
把最后一位变成 0	$(101101 \to 101100)$	x or 1-1
最后一位取反	$(101101 \to 101100)$	x xor 1
把右数第 k 位变成 1	$(101001 \to 101101, k = 3)$	x or (1 shl (k-1))
把右数第 k 位变成 0	$(101101 \to 101001, k = 3)$	x and not (1 shl (k-1))
右数第 k 位取反	$(101001 \to 101101, k = 3)$	x xor (1 shl (k-1))
取末三位	$(1101101 \to 101)$	x and 7
取末 k 位	$(1101101 \to 1101, k = 5)$	x and (1 shl k-1)
取右数第 k 位	$(1101101 \to 1, k = 4)$	x shr (k-1) and 1
把末 k 位变成 1	$(101001 \to 101111, k = 4)$	x or (1 shl k-1)
末 k 位取反	$(101001 \to 100110, k = 4)$	x xor (1 shl k-1)
把右边连续的 1 变成 0	$(100101111 \to 100100000)$	x and (x+1)
把右起第一个 0 变成 1	$(100101111 \to 100111111)$	x or (x+1)
把右边连续的 0 变成 1	$(11011000 \to 11011111)$	x or (x-1)
取右边连续的 1	$(100101111 \to 1111)$	(x xor (x+1)) shr 1
去掉右起第一个 1 的左边	$(100101000 \to 1000)$	x and (x xor (x-1))

9.3.2 枚举长为 n 含 k 个 1 的 01 串

```
int n = 5,k = 3;
for (int s = (1 << k)-1,u = 1 << n; s < u;)
{
  for (int i = 0;i < n;i++)
    printf("%d",(((s>>(n-1-i))&1) == 1));
  printf("\n");

  int b = s & -s;
  s = (s+b)|(((s^(s+b))>>2)/b);
}
```

9.3.3 枚举 x 的二进制非空子集

```
void subsets(int x) {
    for(int i = x; i; i = x&(i-1)) {
        //printbin(i);
    }
}
```

9.4 其它

9.4.1 对跑脚本

```
while true; do
  ./gen > input
  ./sol < input > output.sol
  ./bf < input > output.bf

  diff output.sol output.bf
  if [ $? -ne 0 ] ; then break; fi
done
```