T1 围圈圈

最终的圈可能本来就是一个环.

另外所有的两个点的环所能延伸的最长链都可以互相组合形成一个大环

```cpp
#include <cstdio>
#include <iostream>
#include <ctime>
#include <iomanip>
#include <cstdlib>
#include <algorithm>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <cstring>
using namespace std;


int Case, ans;
```

```cpp
int T, n, x[10000], a[10000];
vector <int> ve[10000];
bool used[10000];

void dfs(int now, int tar, int l) {
  if (now == tar) {
    ans = max(ans, l);
  }
  if (used[now])
    return ;
  used[now] = true;
  if (tar == now)
    return ;
  dfs(a[now], tar, l + 1);
}

int dfs(int k, int fa, int l, int xx) {
  x[xx] = max(x[xx], l);
  for (int i = 0; i < (int) ve[k].size(); i++)
    if (ve[k][i] != fa)
      dfs(ve[k][i], k, l + 1, xx);
}

int main() {
  scanf("%d", &T);
  while (T--) {
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
      ve[i].clear();
```

```
45      for (int i = 1; i <= n; i++)
46        scanf("%d", &a[i]),
    ve[a[i]].push_back(i);
47      ans = 0;
48          //找环
49      for (int i = 1; i <= n; i++) {
50        used[i] = true;
51        dfs(a[i], i, 1);
52        memset(used, false, sizeof used);
53      }
54
55          //所有的两个点的环所能延伸的最长链都可以互
    相组合形成一个大环
56      int tot = 0;
57      for (int i = 1; i <= n; i++)
58        if (a[a[i]] == i) {
59          x[i] = 0;
60          dfs(i, a[i], 0, i);
61          tot += 1;
62          tot += x[i];
63        }else x[i] = -10000;
64      ans = max(ans, tot);
65      printf("Case #%d: %d\n", ++Case, ans);
66    }
67  }
68
```

T2 比赛

枚举第一个不同的数位, 两个数各自是多少, 一旦大小确立, 大的那边肯定后面全部是0, 小的那边肯定后面全部是9

```cpp
#include <algorithm>
#include <cstdio>
#include <iostream>
#include <string>
#include <utility>

typedef long long Long;

char match(char p, int d)
{
    if (p == '?') {
        return true;
    }
    return p - '0' == d;
}

std::string fill(std::string s, char d)
{
    for (auto& c : s) {
        if (c == '?') {
            c = d;
        }
    }
    return s;
```

```cpp
}

std::string format(Long n, int l)
{
    auto s = std::to_string(n);
    while (s.size() < l) {
        s = "0" + s;
    }
    return s;
}

int main()
{
    int T;
    scanf("%d", &T);
    for (int t = 1; t <= T; ++ t) {
        std::string a, b;
        std::cin >> a >> b;
        int n = a.size();
        std::pair<Long, std::pair<Long, Long>> result{1e18, {0, 0}};
        bool gg = false;
        for (int i = 0; i < n; ++ i) {
            for (int x = 0; x < 10; ++ x) {
                for (int y = 0; y < 10; ++ y) {
                    if (x != y && match(a[i], x) && match(b[i], y)) {
                        std::string aa = a;
                        aa[i] = '0' + x;
```

```
                        std::string bb = b;
                        bb[i] = '0' + y;
                        if (x > y) {
                            aa = fill(aa, '0');
                            bb = fill(bb, '9');
                        } else {
                            aa = fill(aa, '9');
                            bb = fill(bb, '0');
                        }
                        auto na =
std::stoll(aa);
                        auto nb =
std::stoll(bb);
                        result =
std::min(result, {std::abs(na - nb), {na,
nb}});
                    }
                }
            }
            char d = '0';
            if (a[i] != '?') {
                d = a[i];
            }
            if (b[i] != '?') {
                if (a[i] != '?' && a[i] !=
b[i]) {
                    gg = true;
                    break;
                }
```

```
76                  d = b[i];
77              }
78              a[i] = b[i] = d;
79          }
80          if (!gg) {
81              auto na = std::stoll(a);
82              auto nb = std::stoll(b);
83              result = std::min(result,
    {std::abs(na - nb), {na, nb}});
84          }
85          std::cout << "Case #" << t << ": " <<
    format(result.second.first, n) << " " <<
    format(result.second.second, n) << std::endl;
86      }
87  }
```

T3

首先将每个数分解成$a*2^t*5^f$,易知末尾0的个数只与t和f有关，那么考虑dp

$dp[i][j]$表示i个元素，j个5能获得的最大的2的数量

那么，$dp[i][j]=max(dp[i-1][j-a[i].f]+a[i].t)$

$ans=max(ans,max(i,dp[k][i]))$

$Code$:

```
#include<bits/stdc++.h>
#define ll long long
#define MAX_N 210
using namespace std;
int n,m;
struct node
{
    int t,f;
}x[MAX_N];
int st,sf;
int ans;
int dp[MAX_N][7000];
int main()
{
    cin>>n>>m;
    for(int i=1;i<=n;++i)
    {
        ll a;
        cin>>a;
        while(a%2==0)a/=2,x[i].t++,st++;
        while(a%5==0)a/=5,x[i].f++,sf++;
    }
    memset(dp,0xc0,sizeof dp);
    dp[0][0]=0;
    for(int j=1;j<=n;++j)
```

```
        for(int i=m;i>=1;--i)
            for(int k=sf;k>=x[j].f;--k)
                dp[i][k]=max(dp[i][k],dp[i-1][k-x[j].f]+x[j].t);
    for(int i=1;i<=sf;++i)
        ans=max(ans,min(i,dp[m][i]));
    cout<<ans<<endl;
    return 0;
}
```

T4

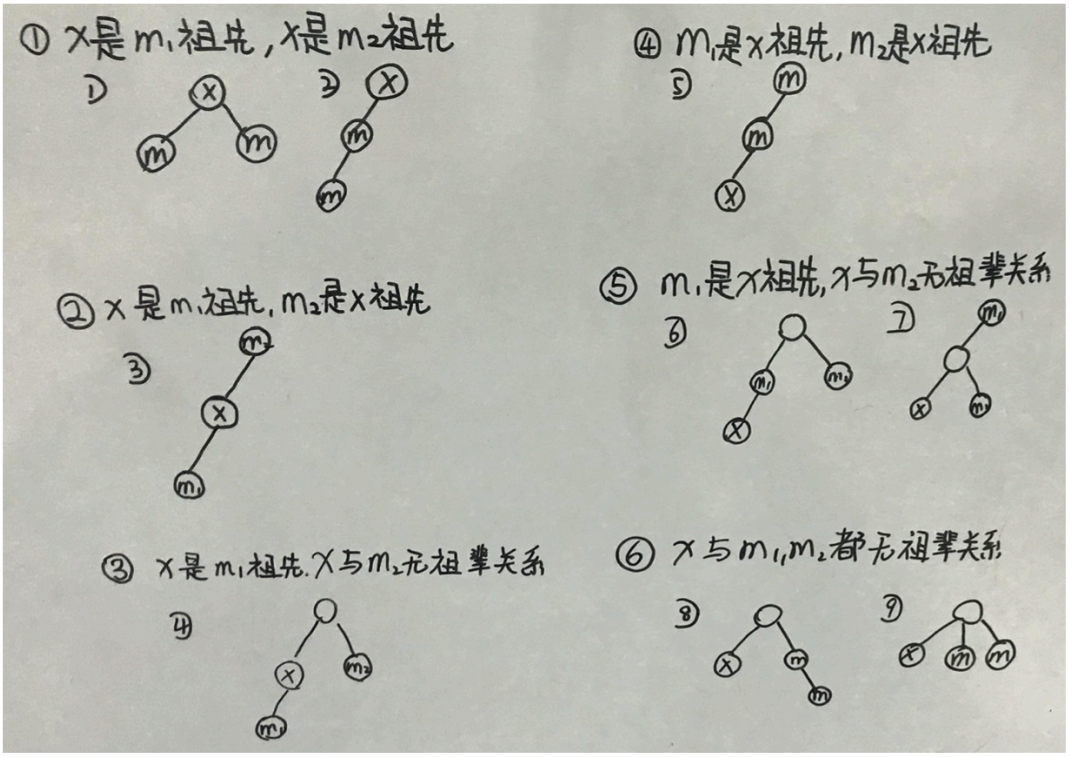
T3. 战场　　算法：大力分类讨论　　时间复杂度：$O(q\log_2 n)$

这个题只要判断，当前节点和另两个节点的中点位置关系就可慢慢推算推出 n 种情况

当我们在计算 x 点的值时，设 m1 为 x 与 y 的中点，m2 为 x 与 z 的中点

x 与 m1、m2 之间分别有 3 种中关系，但有 m1、m2 其实是等价的，只需要通过某种神奇的转换，即可得让 9 种情况转换成 6 种情况。（比老师的 3 种情况麻烦一点点，但码量好像差不多（捂脸））

（借用了一下基地的白板嗝 r，为避免反光拍照角度有点诡异+字丑勿嫌 qaq）



剩下答案什么的就不赘述了，给大家有自己自己思考的空间，其实代码很详细的了

蒟蒻不会用什么神仙剪切板，只会用洛谷云剪切板见谅 qaq

```cpp
#include<bits/stdc++.h>
#define N 100005
using namespace std;
int n, q, ln, sz[N], dep[N], fa[N][20];
int tot = 0, son[N<<1], nxt[N<<1], lnk[N];

inline int read() {
  int red = 0, f_f = 1; char ch = getchar();
  while(ch>'9'||ch<'0') {if(ch == '-') f_f = -1; ch = getchar();}
```

```
  while(ch>='0'&&ch<='9') red = red * 10+ch-'0', ch = getchar();
  return red * f_f;
}

void add(int x, int y) { son[++tot] = y, nxt[tot] = lnk[x], lnk[x] = tot; }

void dfs(int u, int f) {
  fa[u][0] = f, sz[u] = 1;
  for(int i = 1; i <= ln; i++) fa[u][i] = fa[fa[u][i-1]][i-1];
  for(int i = lnk[u]; i; i = nxt[i]) {
    int v = son[i];
    if(v == f) continue;
    dep[v] = dep[u]+1;
    dfs(v, u);
    sz[u] += sz[v];
  }
}

int lca(int x, int y) {
  if(dep[x] < dep[y]) swap(x, y);
  for(int i = ln; i >= 0; i--)
    if(dep[fa[x][i]] >= dep[y]) x = fa[x][i];
  if(x == y) return x;
  for(int i = ln; i >= 0; i--)
    if(fa[x][i] != fa[y][i]) x = fa[x][i], y = fa[y][i];
```

```
35    return fa[x][0];
36  }
37
38  int jump(int x, int wdep) {
39    for(int i = ln; i >= 0; i--)
40      if(dep[fa[x][i]] >= wdep) x = fa[x][i];
41    return x;
42  }
43
44  int get_mid(int x, int y) {
45    int xy = lca(x, y);
46    if(xy == x) return jump(y,
    (dep[x]+dep[y]+1)/2);
47    else if(xy == y) return jump(x,
    (dep[x]+dep[y])/2);
48    else if(dep[x] > dep[y]) return jump(x,
    dep[x]-((dep[x]+dep[y]-2*dep[xy])+1)/2);
49    else return jump(y, dep[y]-
    (dep[x]+dep[y]-2*dep[xy])/2);
50  }
51
52  int calc(int x, int y, int z) { //我好像根本没
    有用到ab ac
53    int m1 = get_mid(x, y), am1 = lca(x, m1);
54    int m2 = get_mid(x, z), am2 = lca(x, m2);
55
56    if((m1 == am1 && x == am2) || (m1 != am1 &&
    x != am1 && (m2 == am2 || x == am2))) {
57      swap(m1, m2); swap(am1, am2);
```

```
58    } //可以少枚举点状态 9种->6种 还是很多的亚子/kk
59
60    if(x == am1) { //x 是m1的祖先
61      if(x == am2) { //x是m2是祖先
62        int m3 = lca(m1, m2);
63        if(m1 == m3) return n - sz[m1];
64        if(m2 == m3) return n - sz[m2];
65        else return n-sz[m1]-sz[m2];
66      }
67      else if(m2 == am2) { //m2是x的祖先
68        int tanao = jump(x, dep[m2]+1);
69        return sz[tanao] - sz[m1];
70      }
71      else { //m2和x没有祖先关系
72        return n-sz[m1]-sz[m2];
73      }
74    }
75    else if(m1 == am1) { //m1 是x的祖先
76      if(m2 == am2) { //m2 也是x的祖先
77        int tanao = jump(x, max(dep[m1],
    dep[m2])+1);
78        return sz[tanao];
79      }
80      else { //m2和x莫得祖先关系
81        int tanao = jump(x, dep[m1]+1), m3 =
    lca(tanao, m2);
82        if(m3 == tanao) return sz[tanao]-sz[m2];
83        else return sz[tanao];
84      }
```

```
  }
  else { //m1和x没有祖先关系 m2和x也没有祖先关系
    int m3 = lca(m1, m2);
    if(m3 == m1 || m3 == m2) {
      if(dep[m1] < dep[m2]) return n-sz[m1];
      else return n-sz[m2];
    }
    else return n-sz[m1]-sz[m2];
  }
}

int main()
{
  n = read(), ln = log2(n);
  for(int i = 1; i < n; i++) {
    int x = read(), y = read();
    add(x, y); add(y, x);
  }
  dfs(1, 1);
  q = read();
  while(q--) {
    int x = read(), y = read(), z = read();
    printf("%d %d %d\n", calc(x, y, z),
calc(y, x, z), calc(z, x, y));
  }
  return 0;
}
```