2020-05-09 洛谷多校Day 1

空游君买下的陈年老题，需付费。~~但据说比较水，适合新手~~

开始时间：19:00。坚持了约三个半小时

通过情况 3/7 200min

题号	A	B	C	D	E	F	G	H
状态		125/3			86/1	这是道错题，不在下文列出	119/2	

原题面都是纯英文的。为方便wiki阅读，现用中文翻译机写。不通见谅——

B

题面

“好吧，但如果我马上给了一大笔钱，那就不是什么好交易了。所以，我今天会给你0美元，但我会每天增加1美元。也就是说，我明天给你1美元，后天给你2美元……”

“然而，只有聪明的人才可能富有，所以当你成为亿万富翁时，你应该立即回答我。”

对温妮来说并不难，因为她善于计算。目前，她在银行账户上有M个美元，她准确地告诉lamp□如果她不使用账户上的钱，她将成为亿万富翁的日期。

现在你也是个聪明人。所以你会知道这个问题的答案：给定M和今天的日期，她成为亿万富翁的日期是什么？

第一行包含整数T□1≤T≤105□-测试用例数。

每个测试用例描述只包含四个整数M□y□m□d□0≤M<109□1900≤y≤2050□1≤m≤12□1≤d≤31□-她现在拥有的钱，以及现在的时间。

保证y□m□d形成相应的日期。

对于每个测试用例，在单独的行上打印三个表示答案日期的整数。

请不要打印月份和日期的前导零。请参阅示例输出以了解清楚信息。

从今天开始，每天拿到的 有

日子 今天 明天

钱数 0 1 2 3 4 5 6 7 8

每天增加 1

输入：第一行组数，每组第一个是现在已经有的钱数、后面是现在的年月日

输出：到有1 000 000 000块钱的时候的日期

样例

输入

2

999999999 2020 2 29

114514 1919 8 10

输出

2020 3 1

2042 1 15

解答

水题，直接解方程就行了。唯一的难点在于日期查找。非常适合出成OJ程设题。

当时的通关代码如下。

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>

int daytable[2][13]= {{0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
                      {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}};

int leapyear(int year)
{
    return ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0);
}

long long solve(long long need)
{
    if(need<=0)
    {
        return 0;
    }
    return (long long)ceil(0.5*(sqrt(1+8*(double)need)-1));
}

long long money;
long long int y,m,d,yy,mm,dd,count,leap,q;

int main(void)
{
    scanf("%d",&q);
```

```
    while(q--)
    {
        scanf("%lld%lld%lld%lld",&money,&yy,&mm,&dd);
        count=solve(1e9-money);
        while(count>366)
        {
            if(mm>=3)
            {
                yy++;
                int tmp = 365+leapyear(yy);
                count-=tmp;
            }
            else if(mm==2&&dd==29)
            {
                if(leapyear(yy+1))
                {
                    count-=366;
                    yy++;
                }
                else
                {
                    count-=365;
                    mm=3;
                    dd=1;
                    yy++;
                }
            }
            else
            {
                int tmp = 365+leapyear(yy);
                count-=tmp;
                yy++;
            }
        }
        while (count)
        {
            count--;
            dd++;
            leap= leapyear(yy);
            if(dd>daytable[leap][mm])
            {
                mm++;
                dd = 1;
            }
            if(mm> 12)
            {
                yy++;
                mm = 1;
            }
        }
        printf("%d %d %d \n", yy,mm,dd);
```

```
    }
}
```

E

题面

在熊王国，国王有一个非常奇怪的爱好：他不喜欢所有长度小于10的名字！事实上，每次他看到谁的名字很短，他都会非常生气，以至于在接下来的998244353天内拒绝发言！

但是，国王对熊王国来说是必不可少的，所以民政部长，肺炎支原体矽肺孢子虫病，宣布：每只熊都应该有一个长长的名字！熊很难改名，因此肺孢子虫病用一系列方法来显示改名规则如下：

（下面是一段编排整齐的长文，超长的规则，至少1500词了。比如）

If first letter of the name is not 'g', reverse the name, else append 'h' after the name.

（翻译成代码大约如下）

if (first letter/last letter/length ==/!= 条件) {

,操作,

} else {

操作.

}

（继续题面）

但是对于熊来说，精确的变换仍然是非常困难的。所以他们要求你——一个可以使用计算机程序的人——来解决这个问题。你能帮他们吗？

输入包含单个字符串s□1≤| s |≤50□由小写英文字母组成-您的程序需要操作的原始名称。

在一行中输出所需的名称。

word翻译题（或者手搓？）

解答

这是一道杀时间的屑题，真的实在不能再屑了。不说了……

冗长无味的AC代码

G

题面

大于2的偶数可以表示为两个素数之和。

2019年初，普罗万“ 证明 ”了戈德巴赫的猜想，并将其发表在智湖。

不管他的证明多么荒谬，你——数学天才——都知道戈德巴赫的猜想在你那神圣的数学直觉中是正确的。你知道这很难证明，但是你可以用你非凡的编程技巧来证明这一点。你决定写一个程序来证明它的正确性，你也要感谢普罗姆先生勇敢地向真理迈进了一步，尽管他失败了，被许多人嘲笑。

请注意，我们不能像您那样提供超级计算机，这样您的程序将使用109以下的数字进行测试。

输入包含一个偶数整数n（2<n≤109）-你需要证明满足戈德巴赫猜想的数字。

输出两个素数，在一行中用一个空格隔开。素数之和应该等于n。

如果有多个答案，请打印所有答案。

哥德巴赫猜想：大于2的每个偶数整数都可以表示为两个素数之和。

输入

单个偶数整数n（2 <n≤1e9）

输出

在一行中输出两个素数，以一个空格分隔。 素数之和应等于n。

如果有多个答案，请打印任何答案。

样例

输入

4

输出

2 2

输入

6

输出

3 3

输入

8

输出

3 5

第三个输出5 3也是正确的。

解答

这题一开始我理解错意思了，以为要输出所有结果。写了个线性筛，疯狂RE电脑差点炸掉。

直到调试的时候，发现1e5就已经在以肉眼可见的速度输出，幡然醒悟。遂搬了个素性测试的板子过来就AC了。

~~赛后看题解这题暴力分解质因数判定都能过~~

```
#include<stdio.h>
#include<stdlib.h>

long long n;

long long test_time=10;

long long QPow(long long bas,long long t,long long MOD)
{
    long long ret=1;
    for(;t;t>>=1,bas=bas*bas%MOD)
    {
        if(t&1LL)
        {
            ret=ret*bas%MOD;
        }
    }
    return ret;
}

bool millerRabbin(long long n)
{
    if(n < 3)
    {
        return n==2;
    }
    long long a = n - 1, b = 0;
    while(a % 2 == 0)
    {
        a /= 2, ++b;
    }
    long long i,j;
    for(i = 1; i <= test_time; ++i)
    {
        long long x = rand() % (n - 2) + 2, v =QPow(x, a, n);
        if(v == 1 || v == n - 1)
        {
            continue;
```

```
        }
        for(j = 0; j < b; ++j)
        {
            v =v * v % n;
            if (v == n - 1)
            {
                break;
            }
        }
        if(j >= b)
        {
            return 0;
        }
    }
    return 1;
}

int main()
{
    scanf("%lld",&n);
    if(n==4)
    {
        printf("2 2\n");
        return 0;
    }
    long long i;
    for(i=3;i<=n/2;i++)
    {
        if(millerRabbin(i)&&millerRabbin(n-i))
        {
            printf("%lld %lld\n",i,n-i);
            return 0;
        }
    }
}
```

A

题面

给出了一个具有n个顶点和m条边的加权连通无向图。你知道有人在顶点1点燃了火，它会立即将当前位置烧成灰尘，并以每秒1英里的速度扩展到相邻的地方。火将在顶点处分裂到所有未点亮的边上，并在至少两个火在同一点相遇时引发爆炸。革命者喜欢爆炸。他们想让你数一数图表上发生的爆炸次数。

第一行包含整数n□m□1≤n≤3×105□0≤m≤106□-顶点数。

接下来的m行中的每一行包含3个整数ui□vi□wi□1≤ui□vi≤n□1≤wi≤9□-第i条边的端点和第i条边的权重。

保证图是连通的。

图可以包含自循环和多条边。示例1显示了处理它们的方法。

输入文件的大小可能很大。请不要读得太慢。

输出将在图形上以单行方式发生的爆炸数。

样例

输入

2 3

1 1 1

1 2 1

1 2 1

输出

2

输入

4 5

1 2 1

1 3 1

2 3 1

2 4 1

3 4 1

输出

2

题解

原题题解如下：

在点上爆炸意味着存在至少两个点可以是他最短路上的前驱；在边上爆炸意味着这条边不在最短路里。因此求出最短路即可。

边权≤ 9并不意味着可以使用SPFA□本题的本意是使用O(nw) 的桶排序Dijkstra, 但是由于vector实现的桶排Dijkstra 和邻接表+ priority_queue 实现的O(mlogm) Dijkstra 速度在使用输入随机数生成器的方法开大数据后没法拉开差距，因此为了避免卡常数都放了过去。

本题真的不卡常数，测出来时限不宽是因为读入太慢了。

（以上）

总之大概的意思是，这东西恰好是最短图标记图后直接算出来。

我们总共有三门课讲到最短路：数据结构、离散II□算法。大概就是把BFS里面的queue换成priority_queue就行了。也就是说，如果所有的边权都为1□priority_queue也可以用queue替代。

（推荐任韩图论，一本跨越了数竞和计竞的高中数竞书）

~~然而就是没想到这层。我一开始直觉是统计圈的个数（破圈），但后来发现当同时到达顶点的时候，只统计一次爆炸而不是多次，最后只好打算模拟bfs……~~

去年的水专栏：Dijkstra的板子分析

```
#include<stdio.h>
#include<string.h>
#include<queue>

using namespace std;

struct node
{
    int to,next,val;
};

struct node e[10000050];

int head[1000005],cnt,n,m,K;

void add(int first,int second,int z)
{
    e[cnt]=(struct node){second,head[first],z};
    head[first]=cnt++;
}

priority_queue<pair<int,int> > q;

int dis[1000005],vis[1000005],ans,used[1000005];

void Dijkstra()
{
    q.push(make_pair(0,1));
    memset(dis,0x3f,sizeof(dis));
    dis[1]=0;
    while(!q.empty())
    {
        int first = q.top().second;
        q.pop();
        if(vis[first])
        {
            continue;
        }
```

```
        vis[first] = 1;
        int i;
        for(i=head[first];i!=-1;i=e[i].next)
        {
            int to1 = e[i].to;
            if(dis[to1]>dis[first]+e[i].val)
            {
                dis[to1]=dis[first]+e[i].val;
                used[to1]=1;
                q.push(make_pair(-dis[to1],to1));
            }
            else if(dis[to1]==dis[first]+e[i].val)
            {
                used[to1]++;
            }
        }
    }
}

int main()
{
    scanf("%d%d",&n,&m);
    memset(head,-1,sizeof(head));
    int i,x,y,z;
    for(i=1;i<=m;i++)
    {
        scanf("%d%d%d",&x,&y,&z);
        add(x,y,z);
        add(y,x,z);
    }
    Dijkstra();
    for(i=1;i<=n;i++)
    {
        if(used[i]>=2)
        {
            used[i]--;
        }
        m-=used[i];
    }
    printf("%d\n",m);
}
```

C

题面

k进制数是以k为基数的数，表示为□A□k□例如□□15□10=□f□16=□17□8=□1111□2个。大于9的数字用小写英文字母a…f表示，表示10<k≤16□

回文数是一个非负数，在某些确定的基址k□它向前或向后读取相同的数据。例如□□121□10□□abba□16□□0□2是回文数字，但是（00）2，（010）10不是。

今天汤米太伤心了，连他最喜欢的k元回文数都数不出来，这些回文的长度是n□而且可以被p整除。

你能帮他数一数汤米最喜欢的数字吗？

注意，答案可能非常大，您只需要告诉答案模块998244353-另一个汤米最喜欢的素数。

输入包含3个整数n□p和k□1≤n≤1018□2≤p≤1000□2≤k≤16□-回文长度、质数和基数。

可以保证p是质数。

输出应答模块998244353。

样例

输入

1 2 10

输出

5

输入

2 7 16

输出

2

输入

16 7 10

输出

12866400

输入

16 7 16

输出

575218836

题解

这是本套题最难的题目。难点在于细节特别多。

原题题解如下：

首先考虑p 与k 不互素的情况。这种情况下，整除只与最后一位有关，且有$$\frac{k}{p}-1$$种取法，其余位任取。注意k = p 时答案为[n = 1]可能会导致部分没有特判的程序挂掉。

现在考虑(p, k) = 1 的情况。考虑暴力dpi,j 表示前i 位和后i 位填好了，其余位取0 时mod p = j的方案数。更新一位的复杂度是O(kp) 的，总复杂度是O(nkp) 的。

考虑优化，由费马小定理ki mod p 存在长度为p − 1 循环节，因此$$k^{i} + k^{n-i-1}\quad mod \ p$$ 也有长度为p − 1 的循环节。按p − 1 分块后，每块的贡献是相同的，而块的背包合并可以通过类似多项式乘法的循环卷积来实现，因此就可以快速幂了。

预处理一个块和处理边缘的零散部分的复杂度是O(kp2), 一次块的暴力卷积是O(p2) 的，因此总复杂度为O(kp2 + p2(log n − log p))

标程使用vector 实现，不开O2 也只跑了一秒，应该是不卡常的。

以上。

好。我可以明确地宣布，到“ 快速幂 ”之前的部分还是能看懂的。之后就不知道怎么个递推法了。

标程代码如下：

```
#include<iostream>
#include<vector>

using namespace std;

long long MOD=998244353;

long long power(long long first, long long second, long long z)
{
    long long ret = 1;
    for(first %= z; second; second >>= 1)
    {
        if(second & 1)
        {
            ret = ret * first % z;
        }
        first = first * first % z;
    }
    return ret;
}

long long n;
int p, K;

void init(vector <int> & a)
{
    a.clear();
    a.resize(p);
    a[0] = 1;
}
```

```
void upd(vector <int> &a, long long b, int st = 0)
{
    int first;
    if (b == n - b - 1)
    {
        first = int(power(K, b, p)) % p;
    }
    else
    {
        first = int(power(K, b, p) + power(K, n - b - 1, p)) % p;
    }
    vector <int> c;
    c.resize(p);
    int i;
    for(i = st; i < K; i++)
    {
        int j;
        for(j = 0; j < p; j++)
        {
            (c[(j + i * first) % p] += a[j]) %= MOD;
        }
    }
    swap(a, c);
}

void mul(vector <int> &a, vector <int> &b)
{
    vector <int> c;
    c.resize(p);
    int i;
    for(i = 0; i < p; i++)
    {
        int j;
        for(j = 0; j < p; j++)
        {
            int first = (i + j) % p;
            c[first] = int((c[first] + (long long)a[i] * b[j]) % MOD);
        }
    }
    swap(a, c);
}

void power(vector <int> &a, long long b)
{
    vector <int> ret;
    init (ret);
    for(;b;b>>=1)
    {
        if(b & 1)
        {
            mul(ret, a);
```

```
        }
        mul(a, a);
    }
    swap(a, ret);
}

vector <int> ans, tr;

int main()
{
    scanf("%lld%d%d", &n, &p, &K);
    if(n == 1)
    {
        int aa = 0;
        int i;
        for(i = 0; i < K; i++)
        {
            if (i % p == 0)
            {
                aa ++;
            }
        }
        cout << aa << endl;
        return 0;
    }
    long long sz = n / 2 - 1;
    if (K % p == 0)
    {
        cout << (K / p - 1) * power(K, sz + (n & 1), MOD) % MOD << endl;
        return 0;
    }
    init(ans);
    init(tr);
    long long tot = sz / (p - 1);
    if(tot)
    {
        int i;
        for(i = 2; i <= p; i++)
        {
            upd(tr, i - 1);
        }
        power(tr, tot);
        mul(ans, tr);
    }
    upd(ans, 0, 1);
    long long i;
    for(i = 2 + tot * (p - 1); i <= (n + 1) / 2; i++)
    {
        upd(ans, i - 1);
    }
```

```
    printf("%d\n", ans[0]);
}
```

总之这是道值得好好研究的题目。

D

题面

双城是一个一切都是双胞胎的地方，除了他们的硬币。他们使用两种硬币，其价值相当于A和B克黄金。那里的人很了解欧几里德算法，所以我们保证A和B的最大公约数是1。他们会告诉你为什么硬币系统是有效的：通过求解线性不定方程Ax+by=C□每一个可能的整数C都会有一个结果。

但当它进入现实时，事情变得更加复杂。事实上，改变-换句话说，负的x或y-是很麻烦的，在双土地上的人根本不喜欢它。所以当有必要改变的时候，他们总是多付一点钱。

一个叫伊尼的骗子，住在一个地方，恨恶两个地方的人。他知道当没有合适的非负x□y作为C的价格时，人们会付出更多的代价，于是决定用这样不方便的价格来骗钱。他买了很多货，把它们送到了双人间土地，并以各种价格定价-当然，所有这些价格都是不方便的人在双重土地。

但是艾尼不是很聪明-也许这就是他现在不得不住在一个地方的原因。他发现很难计算出第K个最小的不公平价格。你能帮他吗？他愿意和你分享他从双面人那里骗取的钱！

第一行包含整数T□1≤T≤10□-测试用例数。

每个测试用例描述只包含一行三个整数□A□B□K□1≤A□B≤107□1≤K≤1018□-两种硬币的值，以及所需的K□

保证了gcd□A□B□=1□且存在第K个最小不公平价格。

对于每个测试用例，在单独的一行上打印一个表示第K个最小不公平价格的整数。

样例

输入

2

2 3 1

314159 233333 123456789

输出

1

123570404

题解

本题找到了“ 逆变换 ”结论：给定值，判断是第几个。因此最终就是一个二分查找。

我想新开一个页面来讲解本题。题解也先放进去了。

裴蜀定理

H

题面

为了准备潜在的病毒传入H国需要调查每个行政部门的危害评分。抽象地说H国的等级构成了一棵根茎树，每个顶点都是一个行政级别，如省、市、县、镇等。根茎是顶点1。每个顶点都有自己的颜色，这表明当地居民的生活习惯-人们更可能在同一颜色的顶点之间进行交通。 每个顶点的危险度得分计算为具有相同颜色的顶点子树中所有不同顶点对的距离之和。两个顶点之间的距离计算为它需要在两个顶点之间传输的最小边数。

时间就是生命。你被雇来尽可能快地计算H国每一个vectex的危险分数。

第一行包含整数n（1≤n≤105）-顶点数。

第二行包含n个整数，第i个数ci（1≤ci≤n）表示顶点i的颜色。

接下来的n-1行中的每一行包含2个整数ui，vi（1≤ui，vi≤n）-第i条边的末端。

它保证了边形成一个包含n个顶点的树。

输出n行，在第i行打印一个整数，表示子树i的危险分数。

样例

输入

5

1 2 1 1 2

1 2

2 3

2 4

5 1

输出

8 2 0 0 0

题解

这个题解就很高端了……

采用支持合并的数据结构，下标维护标号，信息维护深度总和与点数，在每个点合并子树信息并维护答案。合并的两端的某个颜色的信息分别为(dep1, cnt1), (dep2, cnt2)，当前点深度为d，则合并时产生贡献：

$$dep_1cnt_2 + dep_2cnt_1 - 2d \ cnt_1cnt_2$$

采用线段树合并等的时间复杂度为O(n log n)，采用启发式合并的set(map) 也能够以O(n log2 n) 的复杂度通过。

给了两份代码：

```
#include<stdio.h>
#include<vector>

using namespace std;

struct Node
{
    long long dep;
    int l, r, cnt;
};

struct Node t[201111 * 19];

vector <int> E[201111];

int n, tcnt;
int a[201111], rt[201111], dep[201111];

long long sc[201111], delta;
int curdep;

int Merge(int x, int y, int l, int r)
{
    if (!y) return x;
    if (!x) return y;
    if (l == r)
    {
        delta += t[x].dep * t[y].cnt + t[y].dep * t[x].cnt - 2ll * curdep *
t[x].cnt * t[y].cnt;
        t[x].dep += t[y].dep;
        t[x].cnt += t[y].cnt;
    }
    else
    {
        int mid = (l + r) / 2;
        t[x].l = Merge(t[x].l, t[y].l, l, mid);
        t[x].r = Merge(t[x].r, t[y].r, mid + 1, r);
```

```
    }
    return x;
}

int init(int p, int l, int r)
{
    int x = ++tcnt;
    if (l < r)
    {
        int mid = (l + r) / 2;
        if (p <= mid) t[x].l = init(p, l, mid);
        else          t[x].r = init(p, mid + 1, r);
    }
    else t[x].dep = curdep, t[x].cnt = 1;
    return x;
}



void dfs(int x, int fa)
{
    curdep = dep[x];
    rt[x] = init(a[x], 1, n);
    for (auto v : E[x])
    {
        if (v != fa)
        {
            dep[v] = dep[x] + 1;
            dfs(v, x);
            sc[x] += sc[v];
            delta = 0;
            curdep = dep[x];
            rt[x] = Merge(rt[x], rt[v], 1, n);
            sc[x] += delta;
        }
    }
}

int main()
{
    scanf("%d", &n);
    int i;
    for(i = 1; i <= n; i++)
    {
        scanf("%d", a + i);
    }
    for(i = 1; i < n; i++)
    {
        int u, v;
        scanf("%d%d", &u, &v);
```

```
        E[u].push_back(v);
        E[v].push_back(u);
    }
    dfs(1, -1);
    for(i = 1; i <= n; i++)
    {
        printf("%lld ", sc[i]);
    }
}
```

```
#include<stdio.h>
#include<vector>
#include<map>

using namespace std;

vector <int> E[101111];

int n;
int a[101111], dep[101111];

map<int, pair<long long, int> > rt[101111];

long long sc[101111], delta;
int curdep;

void Merge(map<int, pair<long long, int> >& a, map<int, pair<long long, int>
>& b)
{
    if (a.size() < b.size()) swap(a, b);
    for (auto& v : b)
    {
        int col = v.first;
        auto it = a.find(col);
        if (it != a.end())
        {
            long long d1 = it->second.first, c1 = it->second.second;
            long long d2 = v.second.first, c2 = v.second.second;
            delta += d1 * c2 + d2 * c1 - 2LL* curdep * c1 * c2;
            it->second.first += d2;
            it->second.second += int(c2);
        }
        else
        {
            a.insert(v);
        }
    }
}

void init(map<int, pair<long long, int>>& r, int p)
{
```

```
    r[p] = { curdep, 1 };
}



void dfs(int x, int fa)
{
    curdep = dep[x];
    init(rt[x], a[x]);
    for (auto v : E[x])
    {
        if (v != fa)
        {
            dep[v] = dep[x] + 1;
            dfs(v, x);
            sc[x] += sc[v];
            delta = 0;
            curdep = dep[x];
            Merge(rt[x], rt[v]);
            sc[x] += delta;
        }
    }
}

int main()
{
    scanf("%d", &n);
    int i;
    for(i = 1; i <= n; i++)
    {
        scanf("%d", a + i);
    }
    for(i = 1; i < n; i++)
    {
        int u, v;
        scanf("%d%d", &u, &v);
        E[u].push_back(v);
        E[v].push_back(u);
    }
    dfs(1, -1);
    for(i = 1; i <= n; i++)
    {
        printf("%lld ", sc[i]);
    }
}
```