

Standard Code Library

Boboge

Zhejiang University City College

November 12, 2021

Contents

一切的开始	2
fread 快速读	2
getchars() 快读	2
dbg 宏	3
mt19937	3
重载哈希用于 unordered_set	3
数据结构	3
主席树	3
数学	4
倍增求矩阵幂和	4
二次剩余	5
高斯消元	6
矩阵快速幂	7
快速判断能否被一些数整除	8
欧拉函数	9
欧拉降幂	10
判断第二类斯特林数奇偶性	10
三分	10
凸函数	10
凹函数	11
线性基	11
exBSGS	11
exCRT	12
Miller-Rabin & Pollard-Rho	13
FFT	14
NTT	16
图论	17
点分治	17
轻重链剖分	18
虚树	21
网络流	23
Tarjan	23
倍增 LCA	23
dsu on tree	24
几何板子	25
pick 定理	25
字符串	25
AC 自动机	25
Z 函数 (扩展 KMP)	27
马拉车	28
双哈希方便写法	28
杂项	29
日期	29
子集枚举	29
数位 DP	30
背包	30

一切的开始

fread 快速读

```
1 inline char nc() {
2     static char buf[100000], *p1 = buf, *p2 = buf;
3     return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ? EOF : *p1++;
4 }
5 template <typename T>
6 bool rn(T& v) {
7     static char ch;
8     while (ch != EOF && !isdigit(ch)) ch = nc();
9     if (ch == EOF) return false;
10    for (v = 0; isdigit(ch); ch = nc())
11        v = v * 10 + ch - '0';
12    return true;
13 }
14
15 template <typename T>
16 void o(T p) {
17     static int stk[70], tp;
18     if (p == 0) { putchar('0'); return; }
19     if (p < 0) { p = -p; putchar('-'); }
20     while (p) stk[++tp] = p % 10, p /= 10;
21     while (tp) putchar(stk[tp--] + '0');
22 }
```

- 需要初始化
- 需要一次读入
- 不支持负数

```
1 const int MAXS = 100 * 1024 * 1024;
2 char buf[MAXS];
3 template<typename T>
4 inline bool read(T& x) {
5     static char* p = buf;
6     x = 0;
7     while (*p && !isdigit(*p)) ++p;
8     if (!*p) return false;
9     while (isdigit(*p)) x = x * 10 + *p++ - 48;
10    return true;
11 }
12
13 fread(buf, 1, MAXS, stdin);
```

getchars() 快读

```
1 //quick read
2 template<typename T>
3 inline void read(T &x) {
4     int s = 1;
5     x = 0;
6     char ch = getchar();
7     while (ch < '0' || ch > '9') {
8         if (ch == '-') s = -1;
9         ch = getchar();
10    }
11    while (ch >= '0' && ch <= '9') {
12        x = (x << 3) + (x << 1) + (ch ^ 48);
13        ch = getchar();
14    }
15    x *= s;
16 }
17
18 template<typename T, typename... Args>
19 inline void read(T &x, Args &... args) {
20     read(x);
21     read(args...);
22 }
```

dbg 宏

```
1  #define dbg(x...) \
2      do { \
3          cout << #x << " -> "; \
4          err(x); \
5      } while (0)
6
7  void err() {
8      cout << endl;
9  }
10
11 template<class T, class... Ts>
12 void err(T arg, Ts &... args) {
13     cout << arg << ' ';
14     err(args...);
15 }
```

mt19937

```
1  mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());
2  ll rng(ll l, ll r) {
3      uniform_int_distribution<ll> uni(l, r);
4      return uni(mt);
5  }
```

重载哈希用于 unordered_set

```
1  class my_hash {
2  public:
3      ull operator()(const pair<ll, ll> &p) const {
4          return (ull) p.first * P + (ull) p.second;
5      }
6  };
7
8  //unordered_set<pair<ll, ll>, my_hash> s;
```

数据结构

主席树

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  const int maxn = 2e5 + 10;
5  int tot;
6  int sum[maxn << 5], id[maxn], ls[maxn << 5], rs[maxn << 5];
7  int a[maxn], len;
8
9
10 int build(int l, int r) {
11     int root = ++tot;
12     if (l == r) return root;
13     int mid = l + r >> 1;
14     ls[root] = build(l, mid);
15     rs[root] = build(mid + 1, r);
16     return root;
17 }
18
19 int update(int k, int l, int r, int root) {
20     int dir = ++tot;
21     ls[dir] = ls[root], rs[dir] = rs[root], sum[dir] = sum[root] + 1;
22     if (l == r) return dir;
23     int mid = l + r >> 1;
24     if (k <= mid)
25         ls[dir] = update(k, l, mid, ls[dir]);
26     else
27         rs[dir] = update(k, mid + 1, r, rs[dir]);
28     return dir;
```

```

29 }
30
31 int query(int u, int v, int l, int r, int k) {
32     int mid = l + r >> 1, x = sum[ls[v]] - sum[ls[u]];
33     if (l == r) return l;
34     if (k <= x)
35         return query(ls[u], ls[v], l, mid, k);
36     else
37         return query(rs[u], rs[v], mid + 1, r, k - x);
38 }
39
40 vector<int> ind;
41
42 int getid(int val) {
43     return lower_bound(ind.begin(), ind.end(), val) - ind.begin() + 1;
44 }
45
46 int main() {
47     int n, m;
48     scanf("%d%d", &n, &m);
49     for (int i = 1; i <= n; ++i) scanf("%d", &a[i]), ind.push_back(a[i]);
50     sort(ind.begin(), ind.end());
51     ind.erase(unique(ind.begin(), ind.end()), ind.end());
52     len = ind.size();
53     id[0] = build(1, len);
54     for (int i = 1; i <= n; ++i) id[i] = update(getid(a[i]), 1, len, id[i - 1]);
55     while (m--) {
56         int l, r, k;
57         scanf("%d%d%d", &l, &r, &k);
58         printf("%d\n", ind[query(id[l - 1], id[r], 1, len, k) - 1]); // 回答询问
59     }
60     return 0;
61 }

```

数学

倍增求矩阵幂和

```

1 struct mat {
2     int data[N][N] = {};
3     int size{};
4
5     int *operator[](int index) {
6         return data[index];
7     }
8 } G, e;
9
10 mat operator+(const mat &a, const mat &b) {
11     mat ret;
12     ret.size = a.size;
13     for (int i = 1; i <= a.size; ++i) {
14         for (int j = 1; j <= a.size; ++j) {
15             ret.data[i][j] = a.data[i][j] + b.data[i][j];
16         }
17     }
18     return ret;
19 }
20
21 mat mul(mat &A, mat &B) {
22     mat C;
23     C.size = A.size;
24     for (int i = 1; i <= A.size; i++) {
25         for (int k = 1; k <= A.size; k++) {
26             for (int j = 1; j <= A.size; j++) {
27                 C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
28             }
29         }
30     }
31     return C;
32 }

```

```

33
34 mat matpow(mat A, int n) {
35     mat B;
36     B.size = A.size;
37     for (int i = 1; i <= A.size; i++) {
38         B[i][i] = 1;
39     }
40     while (n) {
41         if (n & 1) B = mul(B, A);
42         A = mul(A, A);
43         n >>= 1;
44     }
45     return B;
46 }
47
48 /* 倍增法求解  $A^1 + A^2 + \dots + A^n$  */
49 mat pow_sum(const mat &a, int n) {
50     if (n == 1) return a;
51     mat tmp = pow_sum(a, n / 2);
52     mat tt = matpow(a, n / 2);
53     mat sum = tmp + mul(tmp, tt);
54     //若 n 为奇数,  $n/2 + n/2 = n-1$ , 因此 sum 需要加上  $A^n$  这一项
55     if (n & 1) sum = sum + matpow(a, n);
56     return sum;
57 }

```

二次剩余

```

1  /*
2   * 二次剩余, mod 为奇素数时有解
3   * 解最多两个, 为相反数 x, (mod-x)
4   * n 为 0 特判
5   * 算法返回-1 则无解
6   * mod 为 2 返回 n
7   */
8  #include <bits/stdc++.h>
9
10 using namespace std;
11 typedef long long ll;
12 ll w;
13 struct num {
14     ll x, y;
15 };
16
17 num mul(num a, num b, ll p) {
18     num ans = {0, 0};
19     ans.x = ((a.x * b.x % p + a.y * b.y % p * w % p) % p + p) % p;
20     ans.y = ((a.x * b.y % p + a.y * b.x % p) % p + p) % p;
21     return ans;
22 }
23
24 ll qpow_real(ll a, ll b, ll p) {
25     ll ans = 1;
26     while (b) {
27         if (b & 1) ans = mul(ans, a, p);
28         a = a % p * a % p;
29         b >>= 1;
30     }
31     return ans % p;
32 }
33
34 ll qpow_imag(num a, ll b, ll p) {
35     num ans = {1, 0};
36     while (b) {
37         if (b & 1) ans = mul(ans, a, p);
38         a = mul(a, a, p);
39         b >>= 1;
40     }
41     return ans.x % p;
42 }
43

```

```

44 ll solve(ll n, ll p) {
45     n %= p;
46     if (p == 2) return n;
47     if (qpow_real(n, (p - 1) / 2, p) == p - 1) return -1; //不存在
48     ll a;
49     while (1) {
50         a = rand() % p;
51         w = ((a * a % p - n) % p + p) % p;
52         if (qpow_real(w, (p - 1) / 2, p) == p - 1) break;
53     }
54     num x = {a, 1};
55     return qpow_imag(x, (p + 1) / 2, p);
56 }
57
58 int main() {
59     srand(time(0));
60     int t;
61     scanf("%d", &t);
62     while (t--) {
63         ll n, p;
64         scanf("%lld%lld", &n, &p);
65         if (!n) {
66             printf("0\n");
67             continue;
68         }
69         ll ans1 = solve(n, p), ans2;
70         if (ans1 == -1) {
71             printf("Hola!\n");
72         } else {
73             ans2 = p - ans1;
74             if (ans1 > ans2) swap(ans1, ans2);
75             if (ans1 == ans2) printf("%lld\n", ans1);
76             else printf("%lld %lld\n", ans1, ans2);
77         }
78     }
79 }

```

高斯消元

```

1  #include <iostream>
2  #include <iomanip>
3  #include <cmath>
4
5  using namespace std;
6
7  const int maxn = 10;
8  const int maxq = 100;
9
10 int n;
11 double a[maxn][maxn + 1];
12 double res[maxn] = {};
13 double query[maxq];
14
15 void solve() {
16     int now;
17     double temp;
18     for (int j = 0; j < n; j++) {
19         now = j;
20         for (int i = j; i < n; i++)
21             if (fabs(a[i][j]) > fabs(a[now][j]))
22                 now = i;
23         if (now != j)
24             for (int i = 0; i < n + 1; i++) {
25                 double t = a[j][i];
26                 a[j][i] = a[now][i];
27                 a[now][i] = t;
28             }
29         // 0 0 0
30         for (int i = j + 1; i < n; i++) {
31             temp = a[i][j] / a[j][j];
32             for (int k = j; k <= n; k++)

```

```

33         a[i][k] -= a[j][k] * temp;
34     }
35 }
36 for (int i = n - 1; i >= 0; i--) {
37     if (a[i][i] == 0) {
38         res[i] = 0;
39         continue;
40     }
41     res[i] = a[i][n];
42     for (int j = i + 1; j < n; j++) {
43         res[i] -= a[i][j] * res[j];
44     }
45     res[i] /= a[i][i];
46 }
47 }
48
49 int main() {
50     cin >> n;
51     if (n == 1) cin >> a[0][0] >> a[0][1];
52     else {
53         for (int i = 0; i < n; i++) {
54             double x, y;
55             cin >> x >> y;
56             a[i][n - 1] = 1;
57             a[i][n] = y;
58             for (int j = 0; j < n - 1; j++) {
59                 a[i][j] = pow(x, n - j - 1);
60             }
61         }
62     }
63     int q;
64     cin >> q;
65     for (int i = 0; i < q; i++) cin >> query[i];
66     if (n == 1) {
67         double k = a[0][1] / a[0][0];
68         for (int i = 0; i < q; i++) {
69             double ans = query[i] * k;
70             if (ans >= -0.005 && ans < 0) ans = 0;
71             cout << fixed << setprecision(2) << ans << '\n';
72         }
73     } else {
74         solve();
75         for (int i = 0; i < q; i++) {
76             double ans = 0;
77             for (int j = 0; j < n; j++) {
78                 ans += pow(query[i], n - j - 1) * res[j];
79             }
80             if (ans >= -0.005 && ans < 0) ans = 0;
81             cout << fixed << setprecision(2) << ans << '\n';
82         }
83     }
84     return 0;
85 }

```

矩阵快速幂

```

1 //矩阵快速幂
2 #include <vector>
3 #include <iostream>
4
5 using namespace std;
6 typedef long long ll;
7 const int maxn = 250;
8
9 struct mat {
10     ll data[maxn][maxn] = {};
11     int size;
12
13     ll *operator[](int index) {
14         return data[index];
15     }

```



```

16 };
17
18 const ll mod = 1e9 + 7;
19
20 mat mul(mat &A, mat &B) { //矩阵乘法
21     mat C;
22     C.size = A.size;
23     for (int i = 0; i < A.size; i++) {
24         for (int k = 0; k < A.size; k++) {
25             for (int j = 0; j < A.size; j++) {
26                 C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
27             }
28         }
29     }
30     return C;
31 }
32
33 mat matpow(mat A, ll n) { //矩阵快速幂
34     mat B;
35     B.size = A.size;
36     for (int i = 0; i < A.size; i++) {
37         B[i][i] = 1;
38     }
39     while (n) {
40         if (n & 1) B = mul(B, A);
41         A = mul(A, A);
42         n >>= 1;
43     }
44     return B;
45 }
46
47 int main() {
48     int n;
49     ll k;
50     cin >> n >> k;
51     mat A;
52     A.size = n;
53     for (int i = 0; i < n; i++) {
54         for (int j = 0; j < n; j++) {
55             cin >> A[i][j];
56         }
57     }
58     A = matpow(A, k);
59     for (int i = 0; i < n; i++) {
60         for (int j = 0; j < n; j++) {
61             cout << A[i][j] << " ";
62         }
63         cout << endl;
64     }
65     return 0;
66 }

```

快速判断能否被一些数整除

1. 被 2 整除的数的特征：一个整数的末位是偶数（0、2、4、6、8）的数能被 2 整除。
2. 被 3 整除的数的特征：一个整数的数字和能被 3 整除，则这个数能被 3 整除。
3. 被 4 整除的数的特征：一个整数的末尾两位数能被 4 整除则这个数能被 4 整除。可以这样快速判断：最后两位数，要是十位是单数，个位就是 2 或 6，要是十位是双数，个位就是 0、4、8。
4. 被 5 整除的数的特征：一个整数的末位是 0 或者 5 的数能被 5 整除。
5. 被 6 整除的数的特征：一个整数能被 2 和 3 整除，则这个数能被 6 整除。
6. 被 7 整除的数的特征：“割减法”。若一个整数的个位数字截去，再从余下的数中，减去个位数的 2 倍，这样，一次次下去，直到能清楚判断为止，如果差是 7 的倍数（包括 0），则这个数能被 7 整除。过程为：截尾、倍大、相减、验差。例如，判断 133 是否 7 的倍数的过程如下： $13 - 3 \times 2 = 7$ ，所以 133 是 7 的倍数；又例如判断 6139 是否 7 的倍数的过程如下： $613 - 9 \times 2 = 595$ ， $59 - 5 \times 2 = 49$ ，所以 6139 是 7 的倍数，余类推。

7. 被 8 整除的数的特征：一个整数的末尾三位数能被 8 整除，则这个数能被 8 整除。
8. 被 9 整除的数的特征：一个整数的数字和能被 9 整除，则这个数能被 9 整除。
9. 被 10 整除的数的特征：一个整数的末位是 0，则这个数能被 10 整除。
10. 被 11 整除的数的特征：“奇偶位差法”。一个整数的奇位数字之和与偶位数字之和的差是 11 的倍数（包括 0），则这个数能被 11 整除。（隔位和相减）。例如，判断 491678 能不能被 11 整除的过程如下：奇位数字的和 $9+6+8=23$ ，偶位数字的和 $4+1+7=12$ 。 $23-12=11$ 。因此 491678 能被 11 整除。
11. 被 12 整除的数的特征：一个整数能被 3 和 4 整除，则这个数能被 12 整除。
12. 被 13 整除的数的特征：若一个整数的个位数字截去，再从余下的数中，加上个位数的 4 倍，这样，一次次下去，直到能清楚判断为止，如果是 13 的倍数（包括 0），则这个数能被 13 整除。过程为：截尾、倍大、相加、验差。
13. 被 17 整除的数的特征：若一个整数的个位数字截去，再从余下的数中，减去个位数的 5 倍，这样，一次次下去，直到能清楚判断为止，如果差是 17 的倍数（包括 0），则这个数能被 17 整除。过程为：截尾、倍大、相减、验差。
14. 被 19 整除的数的特征：若一个整数的个位数字截去，再从余下的数中，加上个位数的 2 倍，这样，一次次下去，直到能清楚判断为止，如果是 19 的倍数（包括 0），则这个数能被 19 整除。过程为：截尾、倍大、相加、验差。
15. 被 7、11、13 整除的数的共同特征：若一个整数的末 3 位与末 3 位以前的数字所组成的数之差（以大减小）能被 7、11、13 整除，则这个数能被 7、11、13 整除。例如：128114，由于 $128-114=14$ ，14 是 7 的倍数，所以 128114 能被 7 整除。64152，由于 $152-64=88$ ，88 是 11 的倍数，所以 64152 能被 11 整除。94146，由于 $146-94=52$ ，52 是 13 的倍数，所以 94146 能被 13 整除。

欧拉函数

```

1 //sqrt 求单个欧拉函数
2 ll ph(ll x) {
3     ll ret = x, tmp = x;
4     for (ll i = 2; i * i <= x; i++) {
5         if (tmp % i == 0) {
6             ret = ret / i * (i - 1);
7             while (tmp % i == 0) tmp /= i;
8         }
9     }
10    if (tmp > 1) ret = ret / tmp * (tmp - 1);
11    return ret;
12 }

13
14
15 //线性筛欧拉函数
16 int phi[N];
17 vector<int> prime;
18 bool isprime[N];
19 void init() {
20     memset(isprime, 1, sizeof(isprime));
21     phi[1] = 1, isprime[1] = false;
22     for (int i = 2; i < N; ++i) {
23         if (isprime[i]) {
24             prime.push_back(i);
25             phi[i] = i - 1;
26         }
27         for (int p : prime) {
28             if (1ll * i * p >= N) break;
29             int now = i * p;
30             isprime[now] = false;
31             if (i % p == 0) {
32                 phi[now] = phi[i] * p;
33                 break;
34             } else {
35                 phi[now] = phi[i] * (p - 1);
36             }
37         }
38     }
39 }

```

$$a^b \equiv \begin{cases} a^{b \% \varphi(p)} & (\gcd(a, p) = 1) \\ a^b & (\gcd(a, p) \neq 1, b < \varphi(p)) \\ a^{b \% \varphi(p) + \varphi(p)} & (\gcd(a, p) \neq 1, b \geq \varphi(p)) \end{cases}$$

Figure 1: 欧拉降幂

欧拉降幂

判断第二类斯特林数奇偶性

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  typedef long long ll;
5
6  int solve(ll n) {
7      int ret = 0;
8      while (n) {
9          n >>= 1;
10         ret += n;
11     }
12     return ret;
13 }
14
15 int main() {
16     ll n, m;
17     cin >> n >> m;
18     ll a = n - m;
19     ll b = (m - 1) / 2;
20     ll ans1 = solve(a + b);
21     ll ans2 = solve(a);
22     ll ans3 = solve(b);
23     if (ans1 == ans2 + ans3) {
24         printf("1\n");
25     } else {
26         printf("0\n");
27     }
28     return 0;
29 }

```

三分

凸函数

整数

```

1  //L, R 保证在凸函数两端
2  while (l + 1 < r) {
3      int lm = (l + r) >> 1, rm = (lm + r) >> 1;
4      if (calc(lm) > calc(rm))
5          r = rm;
6      else
7          l = lm;
8  }
9  //答案取 L

```

double

```

1  while (l + eps < r) {
2      double lm = (l + r) / 2, rm = (lm + r) / 2;
3      if (calc(lm) > calc(rm))

```

```

4         r = rm;
5     else
6         l = lm;
7 }
8 //答案取 (L + R) / 2

```

凹函数

只需要将 check 时的符号互换。

线性基

```

1 //线性基 build
2 typedef long long ll;
3 const int maxn = 50;
4 ll d[maxn + 1];
5
6 bool add(ll x) {
7     for (int i = maxn; i >= 0; --i) {
8         if ((x >> i)) {
9             if (d[i] ^ x ^ d[i];
10                 else {
11                     d[i] = x;
12                     return true;
13                 }
14         }
15     }
16     return false;
17 }

```

exBSGS

```

1 /* exBSGS 算法
2  *  $N^x = M \pmod P$ , 其中  $N, P$  互质
3  * 返回的  $x$  为最小解
4  */
5 #include <bits/stdc++.h>
6
7 using namespace std;
8 typedef long long ll;
9 unordered_map<ll, int> H;
10
11 ll gcd(ll a, ll b) {
12     if (!b) return a;
13     return gcd(b, a % b);
14 }
15
16 ll expow(ll a, ll b, ll mod) {
17     ll res = 1;
18     while (b) res = ((b & 1) ? res * a % mod : res), a = a * a % mod, b >>= 1;
19     return res;
20 }
21
22 ll exgcd(ll &x, ll &y, ll a, ll b) {
23     if (!b) {
24         x = 1, y = 0;
25         return a;
26     }
27     ll t = exgcd(y, x, b, a % b);
28     y -= x * (a / b);
29     return t;
30 }
31
32 ll BSGS(ll a, ll b, ll mod, ll q) {
33     H.clear();
34     ll Q, p = ceil(sqrt(mod)), x, y;
35     exgcd(x, y, q, mod), b = (b * x % mod + mod) % mod,
36         Q = expow(a, p, mod), exgcd(x, y, Q, mod), Q = (x % mod + mod) % mod;
37     for (ll i = 1, j = 0; j <= p; ++j, i = i * a % mod) if (!H.count(i)) H[i] = j;

```

```

38     for (ll i = b, j = 0; j <= p; ++j, i = i * Q % mod) if (H[i]) return j * p + H[i];
39     return -1;
40 }
41
42 ll exBSGS(ll N, ll M, ll P) {
43     ll q = 1;
44     ll k = 0, ret;
45     if (M == 1) return 0;
46     while ((ret = gcd(N, P)) > 1) {
47         if (M % ret) return -1;
48         ++k, M /= ret, P /= ret, q = q * (N / ret) % P;
49         if (q == M) return k;
50     }
51     return (ret = BSGS(N, M, P, q)) == -1 ? -1 : ret + k;
52 }
53
54 int main() {
55     while (true) {
56         int N, M, P;
57         scanf("%d%d%d", &N, &P, &M);
58         if (!N && !M && !P) break;
59         N %= P, M %= P;
60         int ans = exBSGS(N, M, P);
61         if (ans == -1)
62             printf("No Solution\n");
63         else
64             printf("%d\n", ans);
65     }
66 }

```

exCRT

```

1  //x % A[i] = B[i] O(nlogn) x 为最小解 x + k * lcm 都可行
2  #include <bits/stdc++.h>
3
4  using namespace std;
5  typedef long long ll;
6  const int N = 1e5 + 10;
7
8  ll mul(ll a, ll b, ll mod) {
9      ll ret = 0;
10     while (b) {
11         if (b & 1) ret = (ret + a) % mod;
12         a = (a + a) % mod;
13         b >>= 1;
14     }
15     return ret;
16 }
17
18 ll exgcd(ll a, ll b, ll &x, ll &y) {
19     ll ret, tmp;
20     if (!b) {
21         x = 1;
22         y = 0;
23         return a;
24     }
25     ret = exgcd(b, a % b, x, y);
26     tmp = x;
27     x = y;
28     y = tmp - a / b * y;
29     return ret;
30 }
31
32 ll A[N], B[N];
33
34 ll excrt(int n) {
35     ll x, y;
36     ll M = A[1], ans = B[1];
37     for (int i = 2; i <= n; ++i) {
38         ll a = M, b = A[i], c = (B[i] - ans % b + b) % b;
39         ll g = exgcd(a, b, x, y), bg = b / g;

```

```

40         if (c % g) return -1;
41         x = mul(x, c / g, bg); //可能溢出
42         ans += x * M;
43         M *= bg;
44         ans = (ans % M + M) % M;
45     }
46     return (ans % M + M) % M;
47 }
48
49 int main() {
50     int n;
51     cin >> n;
52     for (int i = 1; i <= n; ++i) {
53         cin >> A[i] >> B[i];
54     }
55     cout << exCRT(n);
56     return 0;
57 }

```

Miller-Rabin & Pollard-Rho

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  typedef long long ll;
5
6  const int Times = 10;
7  const int N = 5500;
8
9  ll ct;
10 ll fac[N];
11
12 ll gcd(ll a, ll b) {
13     return b ? gcd(b, a % b) : a;
14 }
15
16 ll multi(ll a, ll b, ll m) {
17     ll ret = 0;
18     a %= m;
19     while (b) {
20         if (b & 1) {
21             ret = (ret + a) % m;
22         }
23         b >>= 1;
24         a = (a + a) % m;
25     }
26     return ret;
27 }
28
29 ll qpow(ll a, ll b, ll m) {
30     ll ret = 1;
31     a %= m;
32     while (b) {
33         if (b & 1) {
34             ret = ret * a % m;
35         }
36         b >>= 1;
37         a = a * a % m;
38     }
39     return ret;
40 }
41
42 bool Miller_Rabin(ll n) {
43     if (n == 2) return true;
44     if (n < 2 || !(n & 1)) return false;
45     ll m = n - 1;
46     int k = 0;
47     while ((m & 1) == 0) {
48         k++;
49         m >>= 1;
50     }

```

```

51     for (int i = 0; i < Times; ++i) {
52         ll a = rand() % (n - 1) + 1;
53         ll x = qpow(a, m, n);
54         ll y = 0;
55         for (int j = 0; j < k; ++j) {
56             y = multi(x, x, n);
57             if (y == 1 && x != 1 && x != n - 1) return false;
58             x = y;
59         }
60         if (y != 1) return false;
61     }
62     return true;
63 }
64
65 ll pollard_rho(ll n, ll c) {
66     ll i = 1, k = 2;
67     ll x = rand() % (n - 1) + 1;
68     ll y = x;
69     while (true) {
70         i++;
71         x = (multi(x, x, n) + c) % n;
72         ll d = gcd((y - x + n) % n, n);
73         if (1 < d && d < n) return d;
74         if (y == x) return n;
75         if (i == k) {
76             y = x;
77             k <<= 1;
78         }
79     }
80 }
81
82 void find(ll n, int c) {
83     if (n == 1) return;
84     if (Miller_Rabin(n)) {
85         fac[ct++] = n;
86         return;
87     }
88     ll p = n;
89     ll k = c;
90     while (p >= n) p = pollard_rho(p, c--);
91     find(p, k);
92     find(n / p, k);
93 }
94
95 int main() {
96     ll n;
97     cin >> n;
98     find(n, 120);
99     sort(fac, fac + ct);
100     //排好序的所有质因子 例如 60 被拆解为 2 2 3 5
101 }

```

FFT

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  typedef long long ll;
5  const int N = (1 << 21) + 10;
6  const double PI = acos(-1.0);
7
8  struct Complex {
9      double x, y;
10
11      Complex(double _x = 0.0, double _y = 0.0) {
12          x = _x;
13          y = _y;
14      }
15
16      Complex operator-(const Complex &b) const {
17          return Complex(x - b.x, y - b.y);

```

```

18     }
19
20     Complex operator+(const Complex &b) const {
21         return Complex(x + b.x, y + b.y);
22     }
23
24     Complex operator*(const Complex &b) const {
25         return Complex(x * b.x - y * b.y, x * b.y + y * b.x);
26     }
27 };
28
29 int rev[N];
30
31
32 void change(Complex x[], int len) {
33     for (int i = 0; i < len; ++i) {
34         if (i < rev[i]) {
35             swap(x[i], x[rev[i]]);
36         }
37     }
38 }
39
40 void fft(Complex x[], int len, int opt) {
41     change(x, len);
42     for (int h = 2; h <= len; h <= 1) {
43         Complex wn(cos(2 * PI / h), sin(opt * 2 * PI / h));
44         for (int j = 0; j < len; j += h) {
45             Complex w(1, 0);
46             for (int k = j; k < j + h / 2; k++) {
47                 Complex u = x[k];
48                 Complex t = w * x[k + h / 2];
49                 x[k] = u + t;
50                 x[k + h / 2] = u - t;
51                 w = w * wn;
52             }
53         }
54     }
55     if (opt == -1) {
56         for (int i = 0; i < len; i++) {
57             x[i].x /= len;
58         }
59     }
60 }
61
62 Complex A[N], B[N], C[N];
63
64 ll d[N];
65
66 int main() {
67     string s, t;
68     cin >> s >> t;
69     reverse(s.begin(), s.end());
70     reverse(t.begin(), t.end());
71     int n = (int) s.length(), m = (int) t.length();
72     for (int i = 0; i < n; ++i) A[i].x = s[i] - '0';
73     for (int i = 0; i < m; ++i) B[i].x = t[i] - '0';
74     int len = 1;
75     while (len < (n < 1)) len <= 1;
76     while (len < (m < 1)) len <= 1;
77     for (int i = 0; i < len; ++i) {
78         rev[i] = rev[i >> 1] >> 1;
79         if (i & 1) rev[i] |= len >> 1;
80     }
81     fft(A, len, 1);
82     fft(B, len, 1);
83     for (int i = 0; i < len; ++i) C[i] = A[i] * B[i];
84     fft(C, len, -1);
85     for (int i = 0; i < len; ++i) d[i] = round(C[i].x);
86     for (int i = 0; i < len; ++i) {
87         d[i + 1] += d[i] / 10;
88         d[i] %= 10;

```



```

89     }
90     string out;
91     for (int i = len - 1, f = 0; i >= 0; --i) {
92         if (d[i]) f = 1;
93         if (f) out += (char) ('0' + d[i]);
94     }
95     cout << out << '\n';
96     return 0;
97 }

```

NTT

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  typedef long long ll;
5
6  const int N = (1 << 21) + 10;
7  const int mod = 998244353;
8
9  ll qpow(ll x, ll y) {
10     ll ret = 1;
11     x %= mod;
12     while (y) {
13         if (y & 1) ret = ret * x % mod;
14         x = x * x % mod;
15         y >>= 1;
16     }
17     return ret;
18 }
19
20 int r[N];
21
22 //opt == -1, point to num. otherwise num to point.
23 void ntt(ll *x, int lim, int opt) {
24     for (int i = 0; i < lim; ++i)
25         if (r[i] < i) swap(x[i], x[r[i]]);
26     for (int m = 2; m <= lim; m <= 1) {
27         int k = m >> 1;
28         ll gn = qpow(3, (mod - 1) / m);
29         for (int i = 0; i < lim; i += m) {
30             ll g = 1;
31             for (int j = 0; j < k; ++j, g = g * gn % mod) {
32                 ll tmp = x[i + j + k] * g % mod;
33                 x[i + j + k] = (x[i + j] - tmp + mod) % mod;
34                 x[i + j] = (x[i + j] + tmp) % mod;
35             }
36         }
37     }
38     if (opt == -1) {
39         reverse(x + 1, x + lim);
40         ll inv = qpow(lim, mod - 2);
41         for (int i = 0; i < lim; ++i) x[i] = x[i] * inv % mod;
42     }
43 }
44
45 ll A[N], B[N], C[N];
46
47 int main() {
48     string s, t;
49     cin >> s >> t;
50     int n, m;
51     n = (int) s.length();
52     m = (int) t.length();
53     reverse(s.begin(), s.end());
54     reverse(t.begin(), t.end());
55     for (int i = 0; i < n; ++i) A[i] = s[i] - '0';
56     for (int i = 0; i < m; ++i) B[i] = t[i] - '0';
57     int lim = 1;
58     while (lim < (n << 1)) lim <= 1;
59     while (lim < (m << 1)) lim <= 1;

```

```

60     for (int i = 0; i < lim; ++i) r[i] = (i & 1) * (lim >> 1) + (r[i >> 1] >> 1);
61     ntt(A, lim, 1);
62     ntt(B, lim, 1);
63     for (int i = 0; i < lim; ++i) C[i] = A[i] * B[i] % mod;
64     ntt(C, lim, -1);
65     for (int i = 0; i < lim; ++i) {
66         C[i + 1] += C[i] / 10;
67         C[i] %= 10;
68     }
69     string out;
70     for (int i = lim - 1, f = 0; i >= 0; --i) {
71         if (C[i]) f = 1;
72         if (f) out += (char) ('0' + C[i]);
73     }
74     cout << out << '\n';
75     return 0;
76 }

```

图论

点分治

```

1 //点分治
2 #include <bits/stdc++.h>
3
4 using namespace std;
5 typedef long long ll;
6 const int maxn = 20005;
7 const int inf = 0x3f3f3f3f;
8 const int mod = 1e9 + 7;
9
10 struct edge {
11     int to, val;
12 };
13 vector<edge> mp[maxn];
14
15 int mini, rt, totSZ;
16 int sz[maxn], dis[maxn];
17 bool vis[maxn];
18
19 int l, r, q[maxn]; //q 为每次得到的距离合集
20
21 int n, ans; //ans 记录合法点对
22
23 void getRT(int u, int pre) { //每次调用 getRT() 前使 mini=inf, totSZ = sz[v];
24     sz[u] = 1;
25     int mxSub = 0;
26     for (auto it: mp[u]) {
27         int v = it.to;
28         if (v == pre || vis[v]) continue;
29         getRT(v, u);
30         sz[u] += sz[v];
31         mxSub = max(mxSub, sz[v]);
32     }
33     int mx = max(mxSub, totSZ - sz[u]);
34     if (mx < mini) {
35         mini = mx;
36         rt = u;
37     }
38 }
39
40 void getDIS(int u, int pre) {
41     q[++r] = dis[u];
42     for (auto it: mp[u]) {
43         int v = it.to, val = it.val;
44         if (v == pre || vis[v]) continue;
45         dis[v] = dis[u] + val;
46         getDIS(v, u);
47     }
48 }

```

```

49
50 int calc(int u, int val) {
51     l = 1, r = 0;
52     dis[u] = val;
53     getDIS(u, 0);
54     //按照题意处理 q
55     return sum;
56 }
57
58 void dfs(int u) {
59     vis[u] = true;
60     ans += calc(u, 0);
61     for (auto it: mp[u]) {
62         int v = it.to, val = it.val;
63         if (vis[v]) continue;
64         ans -= calc(v, val);
65         mini = inf;
66         totSZ = sz[v];
67         getRT(v, 0);
68         dfs(v);
69     }
70 }
71
72 int main() {
73     while (~scanf("%d", &n)) {
74         for (int i = 1; i <= n; i++) {
75             mp[i].clear();
76             vis[i] = false;
77         }
78         ans = 0;
79         for (int i = 1; i < n; i++) {
80             int u, v, val;
81             scanf("%d%d%d", &u, &v, &val);
82             mp[u].push_back(edge{v, val});
83             mp[v].push_back(edge{u, val});
84         }
85
86         mini = inf;
87         totSZ = n;
88         getRT(1, 0);
89         dfs(1);
90         printf("%d\n", ans);
91     }
92     return 0;
93 }

```

轻重链剖分

```

1 // Problem: P3384 【模板】轻重链剖分
2 // Contest: Luogu
3 // URL: https://www.luogu.com.cn/problem/P3384
4 // Memory Limit: 125 MB
5 // Time Limit: 1000 ms
6 // 已知一棵包含 N 个结点的树（连通且无环），每个结点上包含一个数值，需要支持以下操作：
7 // 操作 1: 格式: 1 x y z 表示将树从 x 到 y 结点最短路径上所有节点的值都加上 z
8 // 操作 2: 格式: 2 x y 表示求树从 x 到 y 结点最短路径上所有节点的值之和
9 // 操作 3: 格式: 3 x z 表示将以 x 为根节点的子树内所有节点值都加上 z
10 // 操作 4: 格式: 4 x 表示求以 x 为根节点的子树内所有节点值之和
11 // Powered by CP Editor (https://cpeditor.org)
12
13 #include <bits/stdc++.h>
14
15 using namespace std;
16 typedef long long ll;
17 const int N = 1e5 + 10;
18
19 int n, q, rt;
20 int dp[N], fa[N], son[N], ord[N], dep[N], a[N], top[N];
21 ll val[N], mod;
22 vector<int> G[N];
23

```

```

24 ll T[N << 2];
25 ll lazy[N << 2];
26
27 void push_down(int l, int r, int id) {
28     if (lazy[id] == 0) return;
29     int m = (l + r) >> 1;
30     T[id << 1] += (m - l + 1) * lazy[id] % mod;
31     T[id << 1] %= mod;
32     T[id << 1 | 1] += (r - m) * lazy[id] % mod;
33     T[id << 1 | 1] %= mod;
34     lazy[id << 1] += lazy[id];
35     lazy[id << 1] %= mod;
36     lazy[id << 1 | 1] += lazy[id];
37     lazy[id << 1 | 1] %= mod;
38     lazy[id] = 0;
39 }
40
41 void push_up(int id) {
42     T[id] = T[id << 1] + T[id << 1 | 1];
43     T[id] %= mod;
44 }
45
46 void update(int ql, int qr, int l, int r, ll k, int id) {
47     if (ql <= l && r <= qr) {
48         lazy[id] += k;
49         lazy[id] %= mod;
50         T[id] += (r - l + 1) * k % mod;
51         T[id] %= mod;
52         return;
53     }
54     push_down(l, r, id);
55     int m = (l + r) >> 1;
56     if (ql <= m) update(ql, qr, l, m, k, id << 1);
57     if (qr > m) update(ql, qr, m + 1, r, k, id << 1 | 1);
58     push_up(id);
59 }
60
61 ll query(int ql, int qr, int l, int r, int id) {
62     if (ql <= l && r <= qr) {
63         return T[id] % mod;
64     }
65     push_down(l, r, id);
66     ll ret = 0;
67     int m = (l + r) >> 1;
68     if (ql <= m) ret += query(ql, qr, l, m, id << 1) % mod, ret %= mod;
69     if (qr > m) ret += query(ql, qr, m + 1, r, id << 1 | 1) % mod, ret %= mod;
70     return ret;
71 }
72
73 void build(int l, int r, int id) {
74     if (l == r) {
75         T[id] = val[l] % mod;
76         return;
77     }
78     int m = (l + r) >> 1;
79     build(l, m, id << 1);
80     build(m + 1, r, id << 1 | 1);
81     push_up(id);
82 }
83
84 void init(int now, int father, int depth) {
85     fa[now] = father;
86     dp[now] = 1;
87     dep[now] = depth;
88     int mx = 0;
89     for (int i : G[now]) {
90         if (i == father) continue;
91         init(i, now, depth + 1);
92         dp[now] += dp[i];
93         mx = max(mx, dp[i]);
94     }

```

```

95     for (int i : G[now]) {
96         if (i == father) continue;
97         if (dp[i] == mx) {
98             son[now] = i;
99             break;
100     }
101 }
102 }
103
104 int cnt;
105
106 void dfs(int now, int father, int st) {
107     top[now] = st;
108     ord[now] = ++cnt;
109     val[ord[now]] = a[now];
110     if (son[now]) dfs(son[now], now, st);
111     for (int i : G[now]) {
112         if (i == father) continue;
113         if (i == son[now]) continue;
114         dfs(i, now, i);
115     }
116 }
117
118 int main() {
119     ios_base::sync_with_stdio(false);
120     cin >> n >> q >> rt >> mod;
121     for (int i = 1; i <= n; ++i) cin >> a[i];
122     for (int i = 1; i < n; ++i) {
123         int u, v;
124         cin >> u >> v;
125         G[u].push_back(v);
126         G[v].push_back(u);
127     }
128     init(rt, -1, 1);
129     dfs(rt, -1, rt);
130     build(1, n, 1);
131     while (q--) {
132         int op;
133         cin >> op;
134         if (op == 1) {
135             int x, y, z;
136             cin >> x >> y >> z;
137             z %= mod;
138             while (top[y] != top[x]) {
139                 if (dep[top[x]] > dep[top[y]]) swap(x, y);
140                 update(ord[top[y]], ord[y], 1, n, z, 1);
141                 y = fa[top[y]];
142             }
143             if (dep[x] > dep[y]) swap(x, y);
144             update(ord[x], ord[y], 1, n, z, 1);
145         } else if (op == 2) {
146             int x, y;
147             cin >> x >> y;
148             ll ans = 0;
149             while (top[y] != top[x]) {
150                 if (dep[top[x]] > dep[top[y]]) swap(x, y);
151                 ans += query(ord[top[y]], ord[y], 1, n, 1) % mod;
152                 ans %= mod;
153                 y = fa[top[y]];
154             }
155             if (dep[x] > dep[y]) swap(x, y);
156             ans += query(ord[x], ord[y], 1, n, 1);
157             ans %= mod;
158             cout << ans << '\n';
159         } else if (op == 3) {
160             int x, z;
161             cin >> x >> z;
162             z %= mod;
163             update(ord[x], ord[x] + dp[x] - 1, 1, n, z, 1);
164         } else {
165             int x;

```

```

166         cin >> x;
167         ll ans = query(ord[x], ord[x] + dp[x] - 1, 1, n, 1) % mod;
168         cout << ans << '\n';
169     }
170 }
171 return 0;
172 }

```

虚树

```

1 // Problem: P2495 [SDOI2011] 消耗战
2 // 1 为根，每次可以炸毁一个边，花费为边权。
3 // m 次询问，每次要求让 k 个关键点无法到达 1，问最小花费
4 // 对于每次询问单独建树，dfs 序 + 单调栈优化
5
6 #include <bits/stdc++.h>
7
8 using namespace std;
9 typedef long long ll;
10 const int N = 3e5 + 10;
11 vector<pair<int, ll>> G[N];
12 vector<int> g[N];
13 int ord[N]; //dfs 序
14 int stk[N];
15 int dep[N];
16 int parent[N][30];
17 bool inq[N]; //是否为关键点
18 ll minV[N]; //从 1 到 i 不联通的最小花费
19 ll dp[N]; //ans
20 int now;
21 int maxx = 0;
22
23 void dfs1(int u, int fa) {
24     ++now;
25     ord[u] = now;
26     dep[u] = dep[fa] + 1;
27     maxx = max(maxx, dep[u]);
28     parent[u][0] = fa;
29     for (auto pii : G[u]) {
30         int v = pii.first;
31         ll w = pii.second;
32         if (v == fa) continue;
33         minV[v] = min(minV[u], w);
34         dfs1(v, u);
35     }
36 }
37
38 void dfs2(int u) {
39     ll sum = 0;
40     for (int v : g[u]) {
41         dfs2(v);
42         sum += dp[v];
43     }
44     if (inq[u]) {
45         dp[u] = minV[u];
46     } else {
47         dp[u] = min(minV[u], sum);
48     }
49     inq[u] = false;
50     g[u].clear();
51 }
52
53 int n;
54
55 void init() {
56     memset(minV, 0x3f, sizeof minV);
57     dfs1(1, 0);
58     for (int i = 1; (1 << i) <= n; ++i) {
59         for (int j = 1; j <= n; ++j) {
60             if (parent[j][i - 1] == 0) parent[j][i] = 0;
61             else parent[j][i] = parent[parent[j][i - 1]][i - 1];

```

```

62     }
63 }
64 }
65
66 int LCA(int ta, int tb) {
67     if (dep[ta] < dep[tb]) swap(ta, tb);
68     for (int i = 0; dep[ta] != dep[tb]; ++i) {
69         if ((dep[ta] - dep[tb]) >> i & 1)
70             ta = parent[ta][i];
71     }
72     if (ta == tb) return ta;
73     for (int i = log2(maxx); i >= 0; --i) {
74         if (parent[ta][i] != parent[tb][i]) {
75             ta = parent[ta][i];
76             tb = parent[tb][i];
77         }
78     }
79     return parent[ta][0];
80 }
81
82 void build() {
83     vector<pair<int, int>> v;
84     int k;
85     cin >> k;
86     for (int i = 0; i < k; ++i) {
87         int id;
88         cin >> id;
89         inq[id] = true;
90         v.emplace_back(ord[id], id);
91     }
92     sort(v.begin(), v.end());
93     now = 0;
94     stk[++now] = 1;
95     g[1].clear();
96     for (int i = 0; i < k; ++i) {
97         int id = v[i].second;
98         if (id != 1) {
99             int lca = LCA(id, stk[now]);
100             if (lca != stk[now]) {
101                 while (ord[lca] < ord[stk[now - 1]]) {
102                     g[stk[now - 1]].push_back(stk[now]);
103                     now--;
104                 }
105                 if (ord[lca] > ord[stk[now - 1]]) {
106                     g[lca].clear();
107                     g[lca].push_back(stk[now]);
108                     now--;
109                     stk[++now] = lca;
110                 } else {
111                     g[lca].push_back(stk[now]);
112                     now--;
113                 }
114             }
115         }
116         g[id].clear();
117         stk[++now] = id;
118     }
119     for (int i = 1; i < now; ++i) {
120         g[stk[i]].push_back(stk[i + 1]);
121     }
122 }
123
124 void solve() {
125     cin >> n;
126     for (int i = 1; i < n; ++i) {
127         int u, v, w;
128         cin >> u >> v >> w;
129         G[u].emplace_back(v, w);
130         G[v].emplace_back(u, w);
131     }
132     init();

```

```

133     int m;
134     cin >> m;
135     while (m--) {
136         build();
137         dfs2(stk[1]);
138         printf("%lld\n", dp[stk[1]]);
139     }
140 }
141
142 int main() {
143     ios_base::sync_with_stdio(false);
144     int T = 1;
145     // cin >> T;
146     while (T--) {
147         solve();
148     }
149     return 0;
150 }

```

网络流

感觉不如 awei template ... 画质

Tarjan

```

1  const int N = 1e5 + 10;
2  vector<int> G[N];
3  int dfn[N], low[N]; // dfn[u] -> u 被搜索的次序 low[u] -> u 子树中 dfn 最小值 (包括自身)
4  int index; // 搜索序号
5  stack<int> S;
6  bool ins[N]; // 是否进栈
7  int col[N], num_color; // 染色
8
9  void Tarjan(int u) {
10     dfn[u] = low[u] = ++index;
11     S.push(u); // 进栈
12     ins[u] = true;
13     for (int i: G[u]) {
14         int v = i;
15         if (!dfn[v]) { // 未被访问过
16             Tarjan(v);
17             low[u] = min(low[u], low[v]); // 找爸爸 (环开头) 最小的
18         } else if (ins[v]) { // 已被访问过且在栈内, 则需要处理; 若不在栈内说明对应强连通分量处理完成
19             low[u] = min(low[u], dfn[v]); // 判断谁是爸爸
20         }
21     }
22     if (dfn[u] == low[u]) { // 发现更新完一轮自己是爸爸 (某强连通分量中仅第一个被访问的结点满足 dfn[u] == low[u])
23         num_color++;
24         int tmp;
25         do {
26             tmp = S.top();
27             col[tmp] = num_color; // 出栈, 染色
28             ins[tmp] = false;
29             S.pop();
30         } while (tmp != u);
31         col[u] = num_color;
32         ins[u] = false;
33     }
34 }

```

倍增 LCA

```

1  vector<int> G[N];
2  int st[N][20];
3  int dep[N];
4  int maxx = -1;
5
6  void dfs(int u, int fa = -1) {
7     if (fa == -1) dep[u] = 0;

```



```

8     else dep[u] = dep[fa] + 1;
9     st[u][0] = fa;
10    for (int v: G[u]) {
11        if (v == fa) continue;
12        dfs(v, u);
13    }
14 }
15
16 void init(int n) {
17     for (int i = 1; i <= n; ++i) memset(st[i], -1, sizeof st[i]);
18     dfs(rt);
19     for (int j = 1; j < 20; ++j) {
20         for (int i = 1; i <= n; ++i) {
21             if (st[i][j - 1] != -1) st[i][j] = st[st[i][j - 1]][j - 1];
22         }
23     }
24 }
25
26 int lca(int u, int v) {
27     if (dep[u] < dep[v]) swap(u, v);
28     for (int i = 0; dep[u] != dep[v]; ++i) {
29         if ((dep[u] - dep[v]) >> i & 1) {
30             u = st[u][i];
31         }
32     }
33     if (u == v) return u;
34     for (int i = 19; i >= 0; --i) {
35         if (st[u][i] != st[v][i]) {
36             u = st[u][i];
37             v = st[v][i];
38         }
39     }
40     return st[u][0];
41 }

```

dsu on tree

1. 先遍历轻儿子，并计算答案，但 不保留遍历后它对答案的影响；
2. 访问重儿子，保留它对答案的影响。
3. 再次遍历轻儿子，合并轻重儿子之间的答案。

```

1 void dfs(int u, int fa, bool keep) {
2     // 计算轻儿子的答案
3     for (int v: G[u])
4         if (v != fa && v != big[u]) {
5             dfs(v, u, false);
6         }
7     // 计算重儿子答案并保留计算过程中的数据（用于继承）
8     if (big[u]) {
9         dfs(big[u], u, true);
10    }
11    for (int v: G[u])
12        if (v != fa && v != big[u]) {
13            // 子树结点的 DFS 序构成一段连续区间，可以直接遍历
14            for (int i = L[v]; i <= R[v]; i++) {
15                add(Node[i]);
16            }
17        }
18    add(u);
19    ans[u] = getAns();
20    if (!keep) {
21        for (int i = L[u]; i <= R[u]; i++) {
22            del(Node[i]);
23        }
24    }
25 }

```

几何板子

pick 定理

```
1 //计算凸多边形格点数
2 #include <iostream>
3 #include <algorithm>
4
5 using namespace std;
6
7 typedef long long ll;
8
9 const int maxn = 1e5;
10
11 ll x[maxn + 1] = {};
12 ll y[maxn + 1] = {};
13
14 ll calc(int a, int b) {
15     if (y[a] == y[b]) return abs(x[a] - x[b]) - 1;
16     if (x[a] == x[b]) return abs(y[a] - y[b]) - 1;
17     return __gcd(abs(y[a] - y[b]), abs(x[a] - x[b])) - 1;
18 }
19
20 int main() {
21     int n;
22     scanf("%d", &n);
23     for (int i = 0; i < n; i++) scanf("%lld%lld", &x[i], &y[i]);
24     for (int i = 1; i < n; i++) {
25         x[i] -= x[0];
26         y[i] -= y[0];
27     }
28     x[n] = y[n] = x[0] = y[0] = 0;
29     //2c=2s-a-b+2
30     ll ans = 0;
31     ll sum = 0;
32     for (int i = 1; i <= n; i++) {
33         sum += x[i] * y[i + 1] - y[i] * x[i + 1];
34     }
35     ll b = 0;
36     for (int i = 0; i < n; i++) {
37         b += calc(i, i + 1);
38     }
39     cout << (abs(sum) - n - b + 2) / 2;
40     return 0;
41 }
```

其他的。。。感觉不如 kuangbin hdu

字符串

AC 自动机

```
1 // Problem: P5357 【模板】AC 自动机 (二次加强版)
2 // Contest: Luogu
3 // URL: https://www.luogu.com.cn/problem/P5357
4 // Memory Limit: 256 MB
5 // Time Limit: 1000 ms
6 //
7 // Powered by CP Editor (https://cpeditor.org)
8
9 #include <bits/stdc++.h>
10
11 using namespace std;
12
13 #define dbg(x...) \
14     do { \
15         cout << #x << " -> "; \
16         err(x); \
17     } while (0)
```

```

18
19 void err() {
20     cout << endl;
21 }
22
23 template<class T, class... Ts>
24 void err(T arg, Ts &... args) {
25     cout << arg << ' ';
26     err(args...);
27 }
28
29 typedef long long ll;
30 typedef unsigned long long ull;
31 typedef long double ld;
32 typedef __int128 i128;
33 const int N = 2e5 + 10;
34 const ll mod = 1e9 + 7;
35
36 mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());
37
38 ll rng(ll l, ll r) {
39     uniform_int_distribution<ll> uni(l, r);
40     return uni(mt);
41 }
42
43 struct AC {
44     int t[N][26], tot, cnt;
45     int ed[N], fail[N];
46     vector<int> G[N];
47     vector<int> top;
48
49     void init() {
50         memset(ed, -1, sizeof ed);
51     }
52
53     void insert(const string &s) {
54         int u = 0;
55         for (char ch: s) {
56             int to = ch - 'a';
57             if (!t[u][to]) {
58                 t[u][to] = ++tot;
59             }
60             u = t[u][to];
61         }
62         ed[cnt++] = u;
63     }
64
65     void build() {
66         queue<int> q;
67         for (int i = 0; i < 26; ++i) {
68             if (t[0][i]) q.push(t[0][i]);
69         }
70         while (!q.empty()) {
71             int u = q.front();
72             q.pop();
73             for (int i = 0; i < 26; ++i) {
74                 if (t[u][i]) {
75                     fail[t[u][i]] = t[fail[u]][i];
76                     q.push(t[u][i]);
77                 } else {
78                     t[u][i] = t[fail[u]][i];
79                 }
80             }
81         }
82         for (int i = 1; i <= tot; ++i) {
83             G[fail[i]].push_back(i);
84         }
85         q.push(0);
86         while (!q.empty()) {
87             int u = q.front();

```

```

89         q.pop();
90         top.push_back(u);
91         for (int v: G[u]) {
92             q.push(v);
93         }
94     }
95     reverse(top.begin(), top.end());
96 }
97
98 void query(const string &s) {
99     int u = 0;
100     vector<int> c(tot + 1, 0);
101     vector<int> d(tot + 1, 0);
102     for (char ch: s) {
103         int to = ch - 'a';
104         u = t[u][to];
105         d[u]++;
106     }
107     for (int i: top) {
108         c[i] += d[i];
109         d[fail[i]] += d[i];
110     }
111     for (int i = 0; i < cnt; ++i) {
112         cout << c[ed[i]] << '\n';
113     }
114 }
115 } ac;
116
117 void solve(int tCase) {
118     int n;
119     cin >> n;
120     ac.init();
121     for (int i = 0; i < n; ++i) {
122         string s;
123         cin >> s;
124         ac.insert(s);
125     }
126     ac.build();
127     string t;
128     cin >> t;
129     ac.query(t);
130 }
131
132 int main() {
133     ios_base::sync_with_stdio(false);
134     int T = 1;
135     // cin >> T;
136     for (int t = 1; t <= T; ++t) {
137         solve(t);
138     }
139     return 0;
140 }

```

Z函数 (扩展 KMP)

```

1 //z[i] 为 s 本身和 s.substr(i) 的最长公共前缀长度, 但是 z[0] = 0
2 vector<int> z_function(string s) {
3     int n = (int) s.length();
4     vector<int> z(n);
5     for (int i = 1, l = 0, r = 0; i < n; ++i) {
6         if (i <= r && z[i - l] < r - i + 1) {
7             z[i] = z[i - l];
8         } else {
9             z[i] = max(0, r - i + 1);
10            while (i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
11        }
12        if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
13    }
14    return z;
15 }

```

马拉车

```
1 //Manacher
2 string Manacher(string s)
3 {
4     /* 改造字符串 */
5     string res="$#";
6     for(int i=0;i<s.size();++i)
7     {
8         res+=s[i];
9         res+="#";
10    }
11
12    /* 数组 */
13    vector<int> P(res.size(),0);
14    int mi=0,right=0; //mi 为最大回文串对应的中心点, right 为该回文串能达到的最右端的值
15    int maxLen=0,maxPoint=0; //maxLen 为最大回文串的长度, maxPoint 为记录中心点
16
17    for(int i=1;i<res.size();++i)
18    {
19        P[i]=right>i ?min(P[2*mi-i],right-i):1; //关键句, 文中对这句以详细讲解
20
21        while(res[i+P[i]]==res[i-P[i]])
22            ++P[i];
23
24        if(right<i+P[i]) //超过之前的最右端, 则改变中心点和对应的最右端
25        {
26            right=i+P[i];
27            mi=i;
28        }
29
30        if(maxLen<P[i]) //更新最大回文串的长度, 并记下此时的点
31        {
32            maxLen=P[i];
33            maxPoint=i;
34        }
35    }
36    return s.substr((maxPoint-maxLen)/2,maxLen-1);
37 }
```

双哈希方便写法

```
1 const pii mod = {1e9 + 7, 1e9 + 9};
2
3 const pii base = {131, 251};
4
5 pll pw[maxn * 2];
6
7 pll operator * (const pll &p1, const pll &p2) {
8     return {p1.first * p2.first % mod.first, p1.second * p2.second % mod.second};
9 }
10
11 pll operator + (const pll &p1, const pll &p2) {
12     return {(p1.first + p2.first) % mod.first, (p1.second + p2.second) % mod.second};
13 }
14
15 pll operator - (const pll &p1, const pll &p2) {
16     return {(p1.first - p2.first + mod.first) % mod.first, (p1.second - p2.second + mod.second) % mod.second};
17 }
18
19 struct Hash {
20     string s;
21     vector<pll> f;
22     int n;
23     void init(char ss[]) {
24         s = " ";
25         s += string(ss);
26         n = (int) s.length() - 1;
27         f.resize(n + 1, {0, 0});
28         for (int i = 1; i <= n; i++) {
29             int ch = s[i] - 'a';
```

```

30         f[i] = f[i - 1] * base + pll{ch, ch};
31     }
32 }
33 pll ask(int l, int r) { //[l + 1, r]
34     return f[r] - f[l] * pw[r - l];
35 }
36 } s, t[10007];

```

杂项

日期

```

1 // Routines for performing computations on dates. In these routines,
2 // months are expressed as integers from 1 to 12, days are expressed
3 // as integers from 1 to 31, and years are expressed as 4-digit
4 // integers.
5
6 string dayOfWeek[] = {"Mo", "Tu", "We", "Th", "Fr", "Sa", "Su"};
7
8 // converts Gregorian date to integer (Julian day number)
9
10 int DateToInt (int m, int d, int y){
11     return
12         1461 * (y + 4800 + (m - 14) / 12) / 4 +
13         367 * (m - 2 - (m - 14) / 12 * 12) / 12 -
14         3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +
15         d - 32075;
16 }
17
18 // converts integer (Julian day number) to Gregorian date: month/day/year
19
20 void IntToDate (int jd, int &m, int &d, int &y){
21     int x, n, i, j;
22
23     x = jd + 68569;
24     n = 4 * x / 146097;
25     x -= (146097 * n + 3) / 4;
26     i = (4000 * (x + 1)) / 1461001;
27     x -= 1461 * i / 4 - 31;
28     j = 80 * x / 2447;
29     d = x - 2447 * j / 80;
30     x = j / 11;
31     m = j + 2 - 12 * x;
32     y = 100 * (n - 49) + i + x;
33 }
34
35 // converts integer (Julian day number) to day of week
36
37 string IntToDay (int jd){
38     return dayOfWeek[jd % 7];
39 }

```

子集枚举

- 枚举真子集

```

1 for (int s = (S - 1) & S; s; s = (s - 1) & S)

```

- 枚举大小为 k 的子集

```

1 template<typename T>
2 void subset(int k, int n, T&& f) {
3     int t = (1 << k) - 1;
4     while (t < 1 << n) {
5         f(t);
6         int x = t & -t, y = t + x;
7         t = ((t & ~y) / x >> 1) | y;
8     }
9 }

```

数位 DP

```
1 LL dfs(LL base, LL pos, LL len, LL s, bool limit) {
2     if (pos == -1) return s ? base : 1;
3     if (!limit && dp[base][pos][len][s] != -1) return dp[base][pos][len][s];
4     LL ret = 0;
5     LL ed = limit ? a[pos] : base - 1;
6     FOR (i, 0, ed + 1) {
7         tmp[pos] = i;
8         if (len == pos)
9             ret += dfs(base, pos - 1, len - (i == 0), s, limit && i == a[pos]);
10        else if (s && pos < (len + 1) / 2)
11            ret += dfs(base, pos - 1, len, tmp[len - pos] == i, limit && i == a[pos]);
12        else
13            ret += dfs(base, pos - 1, len, s, limit && i == a[pos]);
14    }
15    if (!limit) dp[base][pos][len][s] = ret;
16    return ret;
17 }
18
19 LL solve(LL x, LL base) {
20     LL sz = 0;
21     while (x) {
22         a[sz++] = x % base;
23         x /= base;
24     }
25     return dfs(base, sz - 1, sz - 1, 1, true);
26 }
```

背包

01 背包

有 N 件物品和一个容量为 V 的背包。第 i 件物品的费用是 $c[i]$ ，价值是 $w[i]$ 。求解将哪些物品装入背包可使价值总和最大。

```
1 for i=1..N
2     for v=V..0
3         f[v]=max{f[v],f[v-c[i]]+w[i]};
```

完全背包问题

有 N 种物品和一个容量为 V 的背包，每种物品都有无限件可用。第 i 种物品的费用是 $c[i]$ ，价值是 $w[i]$ 。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

```
1 for i=1..N
2     for v=0..V
3         f[v]=max{f[v],f[v-cost]+weight}
```

多重背包问题

有 N 种物品和一个容量为 V 的背包。第 i 种物品最多有 $n[i]$ 件可用，每件费用是 $c[i]$ ，价值是 $w[i]$ 。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

$$f[i][v] = \max(f[i-1][v-k*c[i]] + k*w[i] | 0 \leq k \leq n[i])$$

```
1 procedure MultiplePack(cost,weight,amount)
2     if cost*amount>=V
3         CompletePack(cost,weight)
4     return
5     integer k=1
6     while k<amount
7         ZeroOnePack(k*cost,k*weight)
8         amount=amount-k
9         k=k*2
10    ZeroOnePack(amount*cost,amount*weight)
```

暴力 $O(V \sum n_i)$

单调队列可以优化到 $O(VN)$

混合三种背包问题

有的物品只可以取一次（01 背包），有的物品可以取无限次（完全背包），有的物品可以取的次数有一个上限（多重背包）

01 背包与完全背包的混合

如果只有两类物品：一类物品只能取一次，另一类物品可以取无限次，那么只需在对每个物品应用转移方程时，根据物品的类别选用顺序或逆序的循环即可，复杂度是 $O(VN)$

```
1 for i=1..N
2   if 第i 件物品属于 01 背包
3     for v=V..0
4       f[v]=max{f[v],f[v-c[i]]+w[i]};
5   else if 第i 件物品属于完全背包
6     for v=0..V
7       f[v]=max{f[v],f[v-c[i]]+w[i]};
```

加上多重背包

```
1 for i=1..N
2   if 第i 件物品属于 01 背包
3     ZeroOnePack(c[i],w[i])
4   else if 第i 件物品属于完全背包
5     CompletePack(c[i],w[i])
6   else if 第i 件物品属于多重背包
7     MultiplePack(c[i],w[i],n[i])
```

可以用单调队列优化

二维费用的背包问题

对于每件物品，具有两种不同的费用；选择这件物品必须同时付出这两种代价；对于每种代价都有一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。设这两种代价分别为代价 1 和代价 2，第 i 件物品所需的两种代价分别为 $a[i]$ 和 $b[i]$ 。两种代价可付出的最大值（两种背包容量）分别为 V 和 U 。物品的价值为 $w[i]$ 。

费用加了一维，只需状态也加一维即可。设 $f[i][v][u]$ 表示前 i 件物品付出两种代价分别为 v 和 u 时可获得的最大价值。状态转移方程就是：

$$f[i][v][u] = \max(f[i-1][v][u], f[i-1][v-a[i]][u-b[i]] + w[i])$$

可以滚动优化

分组背包

有 N 件物品和一个容量为 V 的背包。第 i 件物品的费用是 $c[i]$ ，价值是 $w[i]$ 。这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

```
1 for 所有的组k
2   for v=V..0
3     for 所有的i 属于组 k
4       f[v]=max{f[v],f[v-c[i]]+w[i]}
```