ACM 模板 *for 北湖咸鱼*

三条咸鱼——zhber、strawberry、miamiao

目录

一、头文件

```
#include <bits/stdc++.h>
using namespace std;
#define eps 1e-6
#define e exp(1.0)
#define pi acos(-1.0)
#define fi first
#define se second
#define pb push_back
#define mp make_pair
#define SZ(x) ((int)(x).size())
#define All(x) (x).begin(),(x).end()
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define Close() ios::sync_with_stdio(0),cin.tie(0)
#define INF 0x3f3f3f3f
typedef vector<int> VI;
typedef pair<int, int> PII;
typedef long long ll;
typedef long double ld;
typedef unsigned long long ull;
const int maxn = 1e5 + 10;
const int maxm = 3e3 + 5;
```

```
const int mod = 1e9 + 7;
```

注：①部分模板可能会重新定义新类型；②不同模板的 maxn、maxm 大小取决于题目要求

二、算法

1.快速读入

①普通快速读

```
inline ll read() {
	ll x=0,f=1;char ch=getchar();
	while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
	while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
	return x*f;
}
```

②文件读入流

```
namespace fastIO {
#define BUF_SIZE 100000
	//fread -> read
	bool IOerror = 0;
	inline char nc() {
		static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf
+ BUF_SIZE;
		if(p1 == pend) {
			p1 = buf;
			pend = buf + fread(buf, 1, BUF_SIZE, stdin);
			if(pend == p1) {
				IOerror = 1;
				return -1;
			}
		} return *p1++;
	}
	inline bool blank(char ch) {
		return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';
	}
	inline void read(ll &x) {
		char ch;
		ll f = 1;
		while(blank(ch = nc()));
		while (ch == '-')f = -f, ch = nc();
		if(IOerror)return;
		for(x = ch - '0'; (ch = nc()) >= '0' && ch <= '9'; x = x * 10
+ ch - '0');
		x *= f;
	}
	inline void read(int &x) {
		char ch;
		int f = 1;
```

```
33                while(blank(ch = nc()));
34                while (ch == '-')f = -f, ch = nc();
35                if(IOerror)return;
36                for(x = ch - '0'; (ch = nc()) >= '0' && ch <= '9'; x = x * 10
37 + ch - '0');
38                x *= f;
39        }
40        inline void read(char *s) {
41                char ch;
42                while(blank(ch = nc()));
43                if(IOerror)return;
44                int i = 0;
45                for (; !blank(ch); s[i++] = ch, ch = nc());
46                s[i] = '\0';
47        }
48 #undef BUF_SIZE
49 };
50 using namespace fastIO;
```

2.快速幂

```
1  ll quickmod(ll a, ll b, ll m) {
2         ll ans = 1;
3         while(b) { //用一个循环从右到左便利 b 的所有二进制位
4                 if(b & 1) { //判断此时 b[i]的二进制位是否为 1
5                         ans = (ans * a) % m; //乘到结果上，这里 a 是 a^(2^i)%m
6                         b--;//把该为变 0
7                 }
8                 b /= 2;
9                 a = a * a % m;
10        }
11        return ans;
12 }
```

3.在线求 LCA

```
1  void RMQ() {
2         for (int j = 1; (1 << j) <= n; j++)
3                 for (int i = 1; i <= n; i++)
4                         if (~pa[i][j - 1])
5                                 pa[i][j] = pa[pa[i][j - 1]][j - 1];
6  }
7  int LCA(int x, int y) {
8         if (deep[x] < deep[y]) swap(x, y);
9         int i, j;
10        for (i = 0; (1 << i) <= deep[x]; i++); i--;
11        for (j = i; j >= 0; j--) // 把深度统一
12                if (deep[x] - (1 << j) >= deep[y])
13                        x = pa[x][j];
14        if (x == y) return x;
15        for (j = i; j >= 0; j--) //去找 LCA
```

```
                if (pa[x][j] != -1 && pa[x][j] != pa[y][j]) {
                        x = pa[x][j];
                        y = pa[y][j];
                }
        return pa[x][0];
}
```

4.二维 RMQ

```
int num[maxn][maxn];
int st[maxn][maxn][9][9];
void build_st(int n, int m) {
        for (int i = 1; i <= n ; ++i)
                for (int j = 1; j <= m ; ++j)
                        st[i][j][0][0] = num[i][j];
        for (int g = 0; g < 10 ; ++g) {
                for (int t = 0; t < 10 ; ++t) {
                        if(g == 0 && t == 0)
                                continue;
                        for (int i = 1; i <= n; ++i) {
                                for (int j = 1; j <= m ; ++j) {
                                        if(i + (1 << g) - 1 > n || j + (1 << t)
- 1 > m)
                                                break;
                                        if(g == 0)
                                                st[i][j][g][t] =
max(st[i][j][g][t - 1], st[i][j + (1 << (t - 1))][g][t - 1]);
                                        else
                                                st[i][j][g][t] = max(st[i][j][g -
1][t], st[i + (1 << (g - 1))][j][g - 1][t]);
                                }
                        }
                }
        }
}
int query(int a, int b, int c, int d) {
        int k = (c - a + 1);
        int t = (d - b + 1);
        k = log2(1.0 * k) + eps;
        t = log2(1.0 * t) + eps;
        return max(max(st[a][b][k][t], st[c - (1 << k) + 1][b][k][t]),
max(st[a][d - (1 << t) + 1][k][t], st[c - (1 << k) + 1][d - (1 << t) +
1][k][t]));
}
```

5.Kmp

①Char 版本

```
char n[N], m[N];
int f[N];
void compute(char *P) {
```

```
        int len = strlen(P);
        memset(f, 0, sizeof(f));
        for (int i = 1; i < len; i++) {
                int j = f[i];
                while (j && P[i] != P[j])
                        j = f[j];
                f[i + 1] = (P[i] == P[j]) ? j + 1 : 0;
        }
}
bool kmp(char *T, char *P) {
        int n = strlen(T);
        int m = strlen(P);
        compute(P);
        int q = 0;
        for (int i = 0; i < n; i++) {
                while (q && P[q] != T[i])
                        q = f[q];
                if (P[q] == T[i])
                        q++;
                if (q == m)
                        return 1;      //修改这里可以实现很多功能
        }
        return 0;
}
```

②String 版本

```
string n, m;
int f[maxn];
void compute(string &P) {
        int len = P.length();
        memset(f, 0, sizeof(f));
        for (int i = 1; i < len; i++) {
                int j = f[i];
                while (j && P[i] != P[j])
                        j = f[j];
                f[i + 1] = (P[i] == P[j]) ? j + 1 : 0;
        }
}
bool kmp(string &T, string &P) {
        int n = T.length();
        int m = P.length();
        compute(P);
        int q = 0;
        for (int i = 0; i < n; i++) {
                while (q && P[q] != T[i])
                        q = f[q];
                if (P[q] == T[i])
                        q++;
                if (q == m)
```

```
                        return 1;
        }
        return 0;
}
```

6.背包

```
int dp[maxn];
//0-1 背包，代价为 cost,获得的价值为 weight
void ZeroOnePack(int cost, int weight, int nValue) {
        for (int i = nValue; i >= cost; i--)
                dp[i] = max(dp[i], dp[i - cost] + weight);
}
//完全背包，代价为 cost,获得的价值为 weight
void CompletePack(int cost, int weight, int nValue) {
        for (int i = cost; i <= nValue; i++)
                dp[i] = max(dp[i], dp[i - cost] + weight);
}
//多重背包
void MultiplePack(int cost, int weight, int amount, int nValue) {
        if (cost * amount >= nValue)
                CompletePack(cost, weight, nValue);
        else {
                int k = 1;
                while (k < amount) {
                        ZeroOnePack(k * cost, k * weight, nValue);
                        amount -= k;
                        k <<= 1;
                }
                ZeroOnePack(amount * cost, amount * weight, nValue);
        }
}
```

三、数据结构

1.单调栈

```
int n, a[N], l[N];
int st[N], t;
void monotone() {
        t = 0;
        for (int i = 1; i <= n; i++) {
                while (t > 0 && a[st[t - 1]] ? = a[i])
                        t--;
                l[i] = t == 0 ? 0 : st[t - 1];
                st[t++] = i;
        }
}
```

2.线段树

注意：这里每个节点存了左和右边界，在**极限情况**可以省略。

①结构体版本

```
//这是一个区间加和 RMQ
class SegmentTree {
public:
#define lson (root << 1)
#define rson (root << 1 | 1)
#define lent (t[root].r - t[root].l + 1)
#define lenl (t[lson].r - t[lson].l + 1)
#define lenr (t[rson].r - t[rson].l + 1)
	struct Tree {
		int l, r, val, lazy;
	} t[maxn << 4];

	void pushup(int root) {
		t[root].val = t[lson].val + t[rson].val;
	}

	void pushdown(int root) {
		if (t[root].lazy) {
			t[lson].lazy += t[root].lazy;
			t[rson].lazy += t[root].lazy;
			t[lson].val += lenl * t[root].lazy;
			t[rson].val += lenr * t[root].lazy;
			t[root].lazy = 0;
		}
	}

	void build(int l, int r, int root) {
		t[root].l = l;
		t[root].r = r;
		t[root].lazy = 0;
		if (l == r) {
			t[root].val = 0;
			return;
		}
		int mid = l + r >> 1;
		build(l, mid, lson);
		build(mid + 1, r, rson);
		pushup(root);
	}

	void update(int l, int r, int val, int root) {
		if (l <= t[root].l && t[root].r <= r) {
			t[root].val += lent * val;
			t[root].lazy += val;
			return;
```

```
                }
                pushdown(root);
                int mid = t[root].l + t[root].r >> 1;
                if (l <= mid) update(l, r, val, lson);
                if (r > mid) update(l, r, val, rson);
                pushup(root);
        }

        int query(int l, int r, int root) {
                if (l <= t[root].l && t[root].r <= r)
                        return t[root].val;
                pushdown(root);
                int mid = t[root].l + t[root].r >> 1;
                int ans = 0;
                if (l <= mid) ans += query(l, r, lson);
                if (r > mid) ans += query(l, r, rson);
                return ans;
        }
#undef lenr
#undef lenl
#undef lent
#undef rson
#undef lson
};
```

②数组版本（占坑）

3.树状数组

①一维树状数组

```
int c[maxn];
inline int lowbit(int x) { return x & (-x); }
inline void modify(int x, int y, int data) {
        while (x <= y) {
                c[x] += data;
                x += lowbit(x);
        }
}
inline int query(int x) {
        int sum = 0;
        while (x > 0) {
                sum += c[x];
                x -= lowbit(x);
        }
        return sum;
}
```

②二维树状数组——记得更新四个点

```
class BTree {
        ll a[N][N];
        int n;
        void init() {
```

```
                n = 0;
                memset(a, 0, sizeof a);
        }
        void lowbit(int x) {
                return x & (-x);
        }
        void modify(int x, int y, int d) {
                for (int i = x; i <= n; i += lowbit(i))
                        for (int j = y; j <= n; j += lowbit(j))
                                a[i][j] += d;
        }
        ll query(int x, int y) {
                ll res = 0;
                for (int i = x; i >= 0; i -= lowbit(i))
                        for (int j = y; j >= 0; j -= lowbit(j))
                                res += a[i][j];
                return res;
        }
};
```

4.树链剖分

①点权式

```
// 点权式 - 配合线段树的单点修改+区间查询
struct Edge {
        int to, next;
        Edge() {}
        Edge(int a, int b) { to = a; next = b; }
} E[maxn << 1];
int head[maxn], cnt, tot;
int top[maxn], son[maxn], size[maxn], deep[maxn], pa[maxn], id[maxn];
int a[maxn];
void init() {
        memset(head, -1, sizeof head);
        tot = cnt = 0;
}
void addedge(int u, int v) {
        E[cnt].to = v;
        E[cnt].next = head[u];
        head[u] = cnt++;
}
void dfs1(int u, int fa, int d) {
        size[u] = 1; deep[u] = d; son[u] = 0;
        for (int i = head[u]; ~i; i = E[i].next) {
                int v = E[i].to;
                if (v != fa) {
                        dfs1(v, u, d + 1);
                        pa[v] = u;
                        size[u] += size[v];
                        if (size[v] > size[son[u]]) son[u] = v;
```

```
                }
        }
}
void dfs2(int u, int first) {
        top[u] = first;
        id[u] = ++tot;
        if (son[u]) dfs2(son[u], first);
        for (int i = head[u]; ~i; i = E[i].next) {
                int v = E[i].to;
                if (v != pa[u] && v != son[u]) dfs2(v, v);
        }
}
void solve(int u, int v) { //注意改的是这里
        int x = top[u], y = top[v], res = 0;
        while (x != y) {
                if (deep[x] < deep[y]) {
                        swap(u, v);
                        swap(x, y);
                }
                res += T.query(id[x], id[u], 1);
                u = pa[x];
                x = top[u];
        }
        if (deep[u] > deep[v]) swap(u, v);
        res += T.query(id[u], id[v], 1);
        printf("%d\n", res);
}
```

②边权式

```
// 边权式：把边权偏移到深度较大的点，化成点权式
struct Point {
        int from, to, val;
} p[maxn];
struct Edge {
        int to, next;
} E[maxn << 1];
int head[maxn], cnt, tot;
int top[maxn], son[maxn], size[maxn], deep[maxn], pa[maxn], id[maxn];
int a[maxn];
void init() {
        memset(head, -1, sizeof head);
        tot = cnt = 0;
}
void addedge(int u, int v) {
        E[cnt].to = v;
        E[cnt].next = head[u];
        head[u] = cnt++;
}
void dfs1(int u, int fa, int d) {
        size[u] = 1; deep[u] = d; son[u] = 0;
```

```
        for (int i = head[u]; ~i; i = E[i].next) {
                int v = E[i].to;
                if (v != fa) {
                        dfs1(v, u, d + 1);
                        pa[v] = u;
                        size[u] += size[v];
                        if (size[v] > size[son[u]]) son[u] = v;
                }
        }
}
void dfs2(int u, int first) {
        top[u] = first;
        id[u] = ++tot;
        if (son[u]) dfs2(son[u], first);
        for (int i = head[u]; ~i; i = E[i].next) {
                int v = E[i].to;
                if (v != pa[u] && v != son[u]) dfs2(v, v);
        }
}
int solve(int u, int v) {
        int x = top[u], y = top[v];
        int ans = 0;
        while (x != y) {
                if (deep[x] < deep[y]) {
                        swap(u, v);
                        swap(x, y);
                }
                ans += T.query(id[x], id[u], 1);
                u = pa[x];
                x = top[u];
        }
        //      printf("%d : %d - %d : ", ans, u, v);
        if (deep[u] > deep[v])
                swap(u, v);
        if (u != v) ans += T.query(id[son[u]], id[v], 1);
        return ans;
}
```

5.Trie

```
struct Trie {
#define sigma_size 26
        int ch[maxn][sigma_size];
        int val[maxn];
        int sz;

        void init() {
                sz = 1;
                memset(ch[0], 0, sizeof ch[0]);
        }
```

```
11
12          int idx(char c) {
13                  return c - 'a';
14          }
15
16          void insert(char *s, int v) {
17                  int u = 0, len = strlen(s);
18                  rep(i, 0, len) {
19                          int c = idx(s[i]);
20                          if (!ch[u][c]) {
21                                  memset(ch[sz], 0, sizeof ch[sz]);
22                                  val[sz] = 0;
23                                  ch[u][c] = sz++;
24                          }
25                          u = ch[u][c];
26                  }
27                  val[u] = v;
28          }
29  #undef sigma_size
30  };
```

6.Treap

```
1   class Treap{
2   public:
3       struct treap_point{
4           int l,r,x,rep,son,rnd;
5       }t[N];
6       int treesz,root;
7       inline void update(int k){t[k].son=t[t[k].l].son+t[t[k].r].son+t[k].rep;}
8       inline void right_rotate(int &k){int
9   tt=t[k].l;t[k].l=t[tt].r;t[tt].r=k;update(k);update(tt);k=tt;}
10      inline void left_rotate(int &k){int
11  tt=t[k].r;t[k].r=t[tt].l;t[tt].l=k;update(k);update(tt);k=tt;}
12      inline void insert_to_tree(int &k,int x)
13      {
14          if
15  (!k){k=++treesz;t[k].x=x;t[k].rnd=rand();t[k].son=t[k].rep=1;t[k].l=t[k].r=0;ret
16          t[k].son++;
17          if (x==t[k].x){t[k].rep++;return;}
18          if (x<t[k].x){insert_to_tree(t[k].l,x);if
19  (t[t[k].l].rnd<t[k].rnd)right_rotate(k);}
20          if (x>t[k].x){insert_to_tree(t[k].r,x);if
21  (t[t[k].r].rnd<t[k].rnd)left_rotate(k);}
22      }
23      inline bool delete_from_tree(int &k,int x)
24      {
25          if (!k)return 0;
26          if (x==t[k].x)
27          {
```

```
            if (t[k].rep>1){t[k].rep--;t[k].son--;return 1;}
            if (!t[k].l||!t[k].r){k=t[k].l+t[k].r;return 1;}
            if (t[t[k].l].rnd<t[t[k].r].rnd){right_rotate(k);return
delete_from_tree(k,x);}
            left_rotate(k);return delete_from_tree(k,x);
        }
        int res;
        if (x<t[k].x){res=delete_from_tree(t[k].l,x);if (res)t[k].son--;return re
        res=delete_from_tree(t[k].r,x);if (res)t[k].son--;return res;
    }
    inline int get_succ_in_tree(int k,int x)
    {
        if (!k)return -1;
        int sv;
        if (x<t[k].x){sv=get_succ_in_tree(t[k].l,x);return sv==-1?t[k].x:sv;}
        return get_succ_in_tree(t[k].r,x);
    }
    inline int get_pred_in_tree(int k,int x)
    {
        if(!k)return -1;
        int sv;
        if (x>t[k].x){sv=get_pred_in_tree(t[k].r,x);return sv==-1?t[k].x:sv;}
        return get_pred_in_tree(t[k].l,x);
    }
    inline int ask_kth_in_tree(int k,int x)
    {
        if (!k)return -1;
        if (x<=t[t[k].l].son)return ask_kth_in_tree(t[k].l,x);
        if (x>t[t[k].l].son+t[k].rep)return ask_kth_in_tree(t[k].r,x-t[t[k].l].so
t[k].rep);
        return t[k].x;
    }
    inline int ask_rank_in_tree(int k,int x)
    {
        if (!k)return -1;
        if (x==t[k].x)return t[t[k].l].son+1;
        if (x<t[k].x)return ask_rank_in_tree(t[k].l,x);
        return t[t[k].l].son+t[k].rep+ask_rank_in_tree(t[k].r,x);
    }
    /*
    inline void bianli(int k)//for debug
    {
        if (!k)return;
        bianli(t[k].l);
        printf("i=%d l=%d r=%d x=%d rep=%d
son=%d\n",k,t[k].l,t[k].r,t[k].x,t[k].rep,t[k].son);
        bianli(t[k].r);
    }

```

```
    inline void put(){puts("\n----------------------------------------
");bianli(root);puts("----------------------------------------\n");}//for debug
    */
    inline void init(){treesz=root=0;}
    inline void insert(int x){insert_to_tree(root,x);}
    inline int del(int x){return delete_from_tree(root,x);}
    inline int succ(int x){return get_succ_in_tree(root,x);}
    inline int pred(int x){return get_pred_in_tree(root,x);}
    inline int ask_kth(int x){return ask_kth_in_tree(root,x);}
    inline int ask_rank(int x){return ask_rank_in_tree(root,x);}
}T;
int main()
{
    int n,x,op;read(n);
    T.init();
    for (int i=1;i<=n;i++)
    {
        read(op);read(x);
        if (op==1)T.insert(x);
        if (op==2)T.del(x);
        if (op==3)printf("%d\n",T.ask_rank(x));
        if (op==4)printf("%d\n",T.ask_kth(x));
        if (op==5)printf("%d\n",T.pred(x));
        if (op==6)printf("%d\n",T.succ(x));
    }
};
```

7.后缀数组

①Da

②Dc3

四、图论

1.最短路

①Floyd

```
#define inf 0x3fffffff      //注意，太大会溢出
#define M                           //最大点数
int n, dist[M][M];          //n：实际点数

void init ()                //有时候需要初始化
{
    int i, j;
    for (i = 1; i <= n; i++)
        for (j = i + 1; j <= n; j++)
            dist[i][j] = dist[j][i] = inf;
```

```
}

void floyd ()
{
    int i, j, k;
    for (k = 1; k <= n; k++)
        for (i = 1; i <= n; i++)
            for (j = 1; j <= n; j++)    //有的题目会溢出就要自己变通了
                if (dist[i][k] + dist[k][j] < dist[i][j])
                            dist[i][j] = dist[i][k] + dist[k][j];
}
```

②Dijkstra

- 普通:

```
#define inf 0x3fffffff
#define M 105
int dist[M], map[M][M], n;
bool mark[M];
void init () {
       int i, j;
       for (i = 1; i <= n; i++) //i==j 的时候也可以初始化为 0，只是有时候不合适
              for (j = 1; j <= n; j++)
                     map[i][j] = inf;
}
void dijkstra (int u) {
       int i, j, mins, v;
       for (i = 1; i <= n; i++) {
              dist[i] = map[u][i];
              mark[i] = false;
       }
       mark[u] = true;
       dist[u] = 0;    //既然上面的 map 当 i==j 时不是 0，就要这句
       while (1) {
              mins = inf;
              for (j = 1; j <= n; j++)
                     if (!mark[j] && dist[j] < mins)
                            mins = dist[j], v = j;
              if (mins == inf)
                     break;
              mark[v] = true;
              for (j = 1; j <= n; j++)
                     if (!mark[j] && dist[v] + map[v][j] < dist[j])
                            dist[j] = dist[v] + map[v][j];
       }
}
```

- 堆优化:

```
struct node {
       int to, val;
```

```
        friend bool operator< (const node &a, const node &b) {
                return a.val > b.val;
        }
};
vector<node> g[maxn];
int dis[maxn];
int dijkstra(int s, int t, int n) {
        fill(dis, dis + n + 1, INF);
        priority_queue<node> q;
        dis[s] = 0;
        q.push({s, 0});
        while (!q.empty()) {
                int u = q.top().to; q.pop();
                int sz = SZ(g[u]);
                rep(i, 0, sz) {
                        int v = g[u][i].to, w = g[u][i].val;
                        if (dis[v] > dis[u] + w)
                                q.push({v, dis[v] = dis[u] + w});
                }
        }
        return dis[t];
}
```

③Spfa

- 普通队列:
- 双端队列:

2.二分图匹配

①匈牙利算法

- 链表:

```
vector<int> g[maxn];
int match[maxn];
bool vis[maxn];
int n;
bool dfs(int u) {
        vis[u] = 1;
        int sz = g[u].size();
        rep(i, 0, sz) {
                int v = g[u][i];
                if (match[v] == -1 || !vis[v] && dfs(v)) {
                        match[v] = u;
                        match[u] = v;
                        return 1;
                }
        }
        return 0;
}
int bipartite_matching() {
        int ans = 0;
```

```
        memset(match, -1, sizeof match);
        rep(u, 1, n + 1) {
                if (match[u] == -1) {
                        memset(vis, 0, sizeof vis);
                        if (dfs(u))
                                ans++;
                }
        }
        return ans;
}
```

- 邻接矩阵：

```
int g[N][N], match[N];
bool vis[N];
int n, m; // n 个点 m 个边
bool dfs(int u) {
        rep(v, 1, n + 1) {
                if (g[u][v] && !vis[v]) {
                        vis[v] = 1;
                        if (match[v] == -1 || dfs(match[v])) {
                                match[v] = u;
                                return 1;
                        }
                }
        }
        return 0;
}
int hungary() {
        int ans = 0;
        memset(match, -1, sizeof match);
        rep(u, 1, m + 1) {
                memset(vis, 0, sizeof vis);
                if (dfs(u))
                        ans++;
        }
        return ans;
}
```

3.KM——计算最大权匹配

```
int nx, ny; //两边的点数
int g[N][N];//二分图描述
int match[N], lx[N], ly[N]; //y 中各点匹配状态，x,y 中的点标号
int slack[N];
bool visx[N], visy[N];

bool DFS(int x) {
        visx[x] = true;
        for(int y = 0; y < ny; y++) {
                if(visy[y])continue;
```

```
11              int tmp = lx[x] + ly[y] - g[x][y];
12              if(tmp == 0) {
13                      visy[y] = true;
14                      if(match[y] == -1 || DFS(match[y])) {
15                              match[y] = x;
16                              return true;
17                      }
18              } else if(slack[y] > tmp)
19                      slack[y] = tmp;
20      }
21      return false;
22  }
23  int KM() {
24      memset(match, -1, sizeof(match));
25      memset(ly, 0, sizeof(ly));
26      for(int i = 0; i < nx; i++) {
27              lx[i] = -INF;
28              for(int j = 0; j < ny; j++)
29                      if(g[i][j] > lx[i])
30                              lx[i] = g[i][j];
31      }
32      for(int x = 0; x < nx; x++) {
33              for(int i = 0; i < ny; i++)
34                      slack[i] = INF;
35              while(true) {
36                      memset(visx, false, sizeof(visx));
37                      memset(visy, false, sizeof(visy));
38                      if(DFS(x))break;
39                      int d = INF;
40                      for(int i = 0; i < ny; i++)
41                              if(!visy[i] && d > slack[i])
42                                      d = slack[i];
43                      for(int i = 0; i < nx; i++)
44                              if(visx[i])
45                                      lx[i] -= d;
46                      for(int i = 0; i < ny; i++) {
47                              if(visy[i])ly[i] += d;
48                              else slack[i] -= d;
49                      }
50              }
51      }
52      int res = 0;
53      for(int i = 0; i < ny; i++)
54              if(match[i] != -1)
55                      res += g[match[i]][i];
56      return res;
57  }
```

4.网络流

①Dinic

```
struct Edge {
	int to, cap, next;
} E[maxn];
int head[N], pa[N], vis[N], cnt;
void init() {
	memset(head, -1, sizeof head);
	cnt = 0;
}
void addedge(int u, int v, int w) {
	E[cnt].to = v; E[cnt].cap = w; E[cnt].next = head[u]; head[u] =
cnt++;
	E[cnt].to = u; E[cnt].cap = 0; E[cnt].next = head[v]; head[v] =
cnt++; // 0 or w
}
bool bfs(int s, int t) {
	memset(vis, -1, sizeof vis);
	queue<int> q;
	vis[s] = 0;
	q.push(s);
	while (!q.empty()) {
		int u = q.front();
		q.pop();
		for (int i = head[u]; i != -1; i = E[i].next) {
			int v = E[i].to;
			if (E[i].cap && vis[v] == -1) {
				vis[v] = vis[u] + 1;
				q.push(v);
			}
		}
	}
	return vis[t] != -1;
}
int dfs(int u, int t, int flow) {
	if (u == t) return flow;
	for (int &i = pa[u]; i != -1; i = E[i].next) {
		int v = E[i].to;
		if (E[i].cap && vis[v] == vis[u] + 1) {
			int res = dfs(v, t, min(flow, E[i].cap));
			if (res) {
				E[i].cap -= res;
				E[i ^ 1].cap += res;
				return res;
			}
		}
	}
	return 0;
```

```
}
int Dinic(int s, int t) {
	int max_flow = 0;
	while (bfs(s, t)) {
		memcpy(pa, head, sizeof head);
		int res;
		do {
			res = dfs(s, t, INF);
			max_flow += res;
		} while (res);
	}
	return max_flow;
}
```

②ISAP

```
struct Edge {
	int from, to, cap, flow;
};
class ISAP {
public:
#define max_V 202
	int n, m, s, t;
	vector<Edge> edges;
	vector<int> g[max_V];
	bool vis[max_V];
	int d[max_V], cur[max_V], p[max_V], num[max_V];

	void Addedge(int u, int v, int w) {
		edges.pb((Edge) {
			u, v, w, 0
		});
		edges.pb((Edge) {
			v, u, 0, 0
		});
		m = SZ(edges);
		g[u].pb(m - 2);
		g[v].pb(m - 1);
	}

	bool Bfs() {
		memset(vis, 0, sizeof vis);
		queue<int> q;
		q.push(t);
		vis[t] = 1;
		d[t] = 0;
		while (!q.empty()) {
			int u = q.front(); q.pop();
			int sz = SZ(g[u]);
			rep(i, 0, sz) {
```

```
                        Edge &E = edges[g[u][i] ^ 1];
                        if (!vis[E.from] && E.cap > E.flow) {
                                vis[E.from] = 1;
                                d[E.from] = d[u] + 1;
                                q.push(E.from);
                        }
                }
        }
        return vis[s];
}

void ClearAll(int n) {
        this->n = n;
        rep(i, 0, n) g[i].clear();
        edges.clear();
}

void ClearFlow() {
        int sz = SZ(edges);
        rep(i, 0, sz) edges[i].flow = 0;
}

int Augment() {
        int x = t, a = INF;
        while (x != s) {
                Edge &E = edges[p[x]];
                a = min(a, E.cap - E.flow);
                x = edges[p[x]].from;
        }
        x = t;
        while (x != s) {
                edges[p[x]].flow += a;
                edges[p[x] ^ 1].flow -= a;
                x = edges[p[x]].from;
        }
        return a;
}

int MaxFlow(int s, int t) {
        this->s = s; this->t = t;
        int flow = 0;
        Bfs();
        memset(num, 0, sizeof num);
        rep(i, 0, n) num[d[i]]++;
        int x = s;
        memset(cur, 0, sizeof cur);
        while (d[s] < n) {
                if (x == t) {
                        flow += Augment();
```

```
84                              x = s;
85                      }
86                      bool ok = 0;
87                      int sz = SZ(g[x]);
88                      rep(i, cur[x], sz) {
89                              Edge &E = edges[g[x][i]];
90                              if (E.cap > E.flow && d[x] == d[E.to] + 1) {
91                                      ok = 1;
92                                      p[E.to] = g[x][i];
93                                      cur[x] = i;
94                                      x = E.to;
95                                      break;
96                              }
97                      }
98                      if (!ok) {
99                              int m = n - 1;
100                             rep(i, 0, sz) {
101                                     Edge &E = edges[g[x][i]];
102                                     if (E.cap > E.flow)
103                                             m = min(m, d[E.to]);
104                             }
105                             if (--num[d[x]] == 0) break; // gap
106                             num[d[x] = m + 1]++;
107                             cur[x] = 0;
108                             if (x != s) x = edges[p[x]].from;
109                     }
110             }
111             return flow;
112     }
113
114     vector<int> Mincut() {
115             Bfs();
116             vector<int> ans;
117             int sz = SZ(edges);
118             rep(i, 0, sz) {
119                     Edge &E = edges[i];
120                     if (!vis[E.from] && vis[E.to] && E.cap > 0)
121                             ans.pb(i);
122             }
123             return ans;
124     }
125
126     void Reduce() {
127             int sz = SZ(edges);
128             rep(i, 0, sz) {
129                     edges[i].cap -= edges[i].flow;
130             }
131     }
132
```

```
        void Print() {
                puts("Graph:");
                int sz = SZ(edges);
                rep(i, 0, sz) {
                        printf("%d->%d, %d, %d\n", edges[i].from,
edges[i].to, edges[i].cap, edges[i].flow);
                }
        }
#undef max_V
};
```

5.费用流

```
struct Edge {
        int from, to, cap, flow, cost;
        Edge(int a, int b, int c, int d, int E) { from = a; to = b; cap = c;
flow = d; cost = E; }
};
class MCMF {
public:
#define max_V 202
        int n, m, s, t;
        vector<Edge> edges;
        vector<int> g[max_V];
        bool vis[max_V];
        int d[max_V], p[max_V], a[max_V];

        void Init(int n) {
                this->n = n;
                rep(i, 0, n) g[i].clear();
                edges.clear();
        }

        void Addedge(int u, int v, int c, int w) {
                edges.pb(Edge(u, v, c, 0, w));
                edges.pb(Edge(v, u, 0, 0, -w));
                int m = SZ(edges);
                g[u].pb(m - 2);
                g[v].pb(m - 1);
        }

        bool Spfa(int s, int t, int &flow, int &cost) {
                fill(d, d + n, INF);
                memset(vis, 0, sizeof vis);
                d[s] = 0; vis[s] = 1; p[s] = 0; a[s] = INF;
                queue<int> q;
                q.push(s);
                while (!q.empty()) {
                        int u = q.front(); q.pop();
                        vis[u] = 0;
```

```
                        int sz = SZ(g[u]);
                        rep(i, 0, sz) {
                                Edge& E = edges[g[u][i]];
                                if (E.cap > E.flow && d[E.to] > d[u] + E.cost)
{
                                        d[E.to] = d[u] + E.cost;
                                        p[E.to] = g[u][i];
                                        a[E.to] = min(a[u], E.cap - E.flow);
                                        if (!vis[E.to]) {
                                                vis[E.to] = 1;
                                                q.push(E.to);
                                        }
                                }
                        }
                }
                if (d[t] == INF) return 0;
                flow += a[t];
                cost += d[t] * a[t];
                int u = t;
                while (u != s) {
                        edges[p[u]].flow += a[t];
                        edges[p[u] ^ 1].flow -= a[t];
                        u = edges[p[u]].from;
                }
                return 1;
        }

        int MincostMaxflow(int s, int t) {
                int flow = 0, cost = 0;
                while (Spfa(s, t, flow, cost));
                return cost;
        }
#undef max_V
};
```

五、数论

1.质因子分解

Miller Rabin and Pollard Rho

```
//质因子分解，小数据用筛法直接判，大数据用 pollard_rho
//map[i]是含有多少个质因子 i
//map<LL,int>::iterator c,c->first 表示质因子，c->second 表示次方

#define maxn_for_division 1000001
bool is_prime[maxn_for_division];
vector<int>prime;
map<LL, int>factor;
```

```
inline void get_prime() {
	for(int i = 0; i < maxn_for_division; i++)is_prime[i] = 1;
	is_prime[0] = is_prime[1] = 0;
	for(int i = 2; i < maxn_for_division; i++)
		if (is_prime[i]) {
			prime.push_back(i);
			for (int j = i; j < maxn_for_division; j +=
i)is_prime[j] = 0;
		}
}
inline LL gcd(LL a, LL b) {
	if (!b)return a;
	return gcd(b, a % b);
}
inline LL mod_mul(LL a, LL b, LL p) {
	LL ans = 0ll;
	a %= p, b %= p;
	if (a > b)swap(a, b);
	while (b) {
		if (b & 1)ans = (ans + a) % p;
		a = (a + a) % p;
		b >>= 1;
	}
	return ans;
}
LL mod_pow(LL a, LL b, LL p) {
	LL ans = 1ll;
	a %= p;
	while (b) {
		if (b & 1)ans = mod_mul(ans, a, p);
		a = mod_mul(a, a, p);
		b >>= 1;
	}
	return ans;
}
bool witness(LL a, LL n) {
	LL m = n - 1;
	int j = 0;
	while(!(m & 1))j++, m >>= 1;
	LL x = mod_pow(a, m, n);
	if (x == 1 || x == n - 1)return 0;
	while(j--) {
		x = mod_mul(x, x, n);
		if(x == n - 1)return 0;
	}
	return 1;
}
#define rep_times 20
bool Miller_Rabin(LL n) { //判断 n 是否为素数
```

```
58          srand(time(0));
59          if(n < 2)return 0;
60          if(n == 2)return 1;
61          if (!(n & 1))return 0;
62          for(int i = 0; i < rep_times; i++) {
63                  LL a = rand() % (n - 1) + 1;
64                  if (witness(a, n))return 0;
65          }
66          return 1;
67  }
68  #undef rep_times
69  LL Pollard_Rho(LL n, int c) {
70          LL x = 2, y = 2, d = 1;
71          while (d == 1) {
72                  x = mod_mul(x, x, n) + c;
73                  y = mod_mul(y, y, n) + c;
74                  y = mod_mul(y, y, n) + c;
75                  d = gcd((x - y >= 0 ? x - y : y - x), n);
76          }
77          if (d == n)return Pollard_Rho(n, c + 1);
78          return d;
79  }
80  bool Is_Prime(LL n) {
81          return n < maxn_for_division && is_prime[n] || n >=
82  maxn_for_division && Miller_Rabin(n);
83  }
84  void Find_Factor(LL n) {
85          if (Is_Prime(n)) {
86                  factor[n]++;
87                  return;
88          }
89          for (int i = 0; i < prime.size() && prime[i] <= n; i++)
90                  if (n % prime[i] == 0) {
91                          while (n % prime[i] == 0) {
92                                  factor[prime[i]]++;
93                                  n /= prime[i];
94                          }
95                  }
96          if (n != 1) {
97                  if (Is_Prime(n))factor[n]++;
98                  else {
99                          LL p = Pollard_Rho(n, 1);
100                         Find_Factor(p);
101                         Find_Factor(n / p);
102                 }
103         }
104 }
105 int main() {
106         LL n;
```

```
107         get_prime();
108         while(read(n), !IOerror) {
109                 factor.clear();
110                 Find_Factor(n);
111                 //----------------------------output-----------------------
112 -------
113                 for(map<LL, int>::iterator c = factor.begin(); c !=
114 factor.end();;) {
115                         printf("%lld^%d", c->first, c->second);
116                         if((++c) != factor.end())printf("*");
117                 }
118                 //-------------------------------------------------------------
119 -------
120                 puts("");
121         }
122 }
```

六、代数

七、组合数学