图

SPFA

```
const int MAXN = 1000+5, MAXM = 1000+5;
const int INF = 0x3f3f3f3f;
int n, m, e, s;
int v[MAXM], next[MAXM], head[MAXN];
int w[MAXM], d[MAXN];
int inq_cnt[MAXN]; //存在负权回路时需要
bool inq[MAXN];
queue<int> Q;
void addedge(int x, int y, int z)
{
    v[e] = y; w[e] = z;
    next[e] = head[x]; head[x] = e;
    e++;
}
bool spfa()
{
    for (int i = 1; i <= n; i++)
        d[i] = (i == s ? 0 : INF);
    memset(inq, 0, sizeof(inq));
    memset(inq_cnt, 0, sizeof(inq_cnt));
    while (!Q.empty()) Q.pop();
    Q.push(s);
    inq[s] = 1;
    inq_cnt[s]++;
    while (!Q.empty())
    {
        int u = Q.front(); Q.pop();
        inq[u] = 0;
        for(int e = head[u]; e != -1; e = next[e])
            if(d[v[e]] > d[u]+w[e])
            {
                d[v[e]] = d[u]+w[e];
                if(!inq[v[e]])
                {
                    Q.push(v[e]);
                    inq[v[e]] = 1;
                    inq_cnt[v[e]]++;
                    if (inq_cnt[v[e]] > n)
                        return 0;
                }
            }
    }
    return 1;
}
int main()
{
    memset(head, -1, sizeof(head));
    e = 0;

    return 0;
}
```

Floyd 求最小环（poj 1734）

```
const int INF = 0x3f3f3f3f;
const int MAXN = 100+5;
int n, m; // n:节点个数, m:边的个数
int w[MAXN][MAXN], d[MAXN][MAXN]; // 无向图, 最短路径
int cnt, out[MAXN], r[MAXN][MAXN]; // 记录最小环路径,
r[i][j]: j 到i 的最短路径的第一步
int make_ans(int i, int j, int k)
{// 记录最小环路径
    cnt = 0;
    while (j != i)
    {
        out[++cnt] = j;
        j = r[i][j];
    }
    out[++cnt] = i; out[++cnt] = k;
    return 0;
}
int main()
{
    while (scanf("%d%d", &n, &m) != EOF)
    {
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
            {
                w[i][j] = INF;
                r[i][j] = i;
            }
        for (int i = 1; i <= m; i++)
        {
            int x, y, l;
            scanf("%d%d%d", &x, &y, &l);
            if (l < w[x][y])
                w[x][y] = w[y][x] = l;
        }
        memcpy(d, w, sizeof(w));
        int ans = INF; // 最小环
        for (int k = 1; k <= n; k++)
        {//Floyd
            for (int i = 1; i < k; i++)// 一个环中的最大
结点为k(编号最大)
                if (w[k][i] < INF)
                    for (int j = i+1; j < k; j++)
                        if (d[i][j] < INF && w[k][j] < INF
&& ans > d[i][j]+w[k][i]+w[k][j])
                        {
                            ans =
d[i][j]+w[k][i]+w[k][j];
                            make_ans(i, j, k); // 记录最
小环
                        }
            for (int i = 1; i <= n; i++)
                if (d[i][k] < INF)
                    for (int j = 1; j <= n; j++)
                        if (d[k][j] < INF && d[i][j] >
d[i][k]+d[k][j])
                        {
                            d[i][j] = d[i][k]+d[k][j];
                            r[i][j] = r[k][j];
                        }
        }
        if (ans < INF)
        {
            for (int i = cnt; i >= 1; i--)
            {
                if (i < cnt)
                    printf(" ");
                printf("%d", out[i]);
            }
            printf("\n");
        }
        else
            printf("No solution.\n");
    }
    return 0;
}
```

最大流-邻接表

```
const int MAXN = 1000+5, MAXM = 1000+5;
const int INF = 0x3f3f3f3f;
int e, s, t, n;
int v[MAXM], next[MAXM], head[MAXN];
int cap[MAXM], a[MAXN], f;
int pv[MAXN], pe[MAXN];
queue<int> Q;
void addedge(int u_, int v_, int c_)
{
    v[e] = v_; cap[e] = c_;
    next[e] = head[u_]; head[u_] = e;
    e++;
```

```
    v[e] = u_; cap[e] = 0;
    next[e] = head[v_]; head[v_] = e;
    e++;
}
void maxflow()
{
    f = 0;
    for (;;)
    {
        memset(a, 0, sizeof(a));
        a[s] = INF;
        Q.push(s);
        while (!Q.empty())
        {
            int u = Q.front(); Q.pop();
            for (int e = head[u]; e != -1; e = next[e])
                if(!a[v[e]] && cap[e])
                {
                    Q.push(v[e]);
                    a[v[e]] = min(a[u], cap[e]);
                    pv[v[e]] = u; pe[v[e]] = e;
                }
        }
        if (!a[t]) break;
        for (int v = t; v != s; v = pv[v])
        {
            cap[pe[v]] -= a[t];
            cap[pe[v]^1] += a[t];
        }
        f += a[t];
    }
}
int main()
{
    memset(cap, 0, sizeof(cap));
    memset(head, -1, sizeof(head));
    e = 0;

    return 0;
}
```

最小费用最大流-邻接表

```
const int MAXN = 1000+5, MAXM = 1000+5;
const int INF = 0x3f3f3f3f;
int e, s, t, n;
int v[MAXM], next[MAXM], head[MAXN];
int cap[MAXM], f;
int cost[MAXM], d[MAXN], c;
int pv[MAXN], pe[MAXN];
bool inq[MAXN];
queue<int> Q;
void addedge(int u_, int v_, int c_, int w_)
{
    v[e] = v_; cap[e] = c_; cost[e] = w_;
    next[e] = head[u_]; head[u_] = e;
    e++;
    v[e] = u_; cap[e] = 0; cost[e] = -w_;
    next[e] = head[v_]; head[v_] = e;
    e++;
}
void mincostflow()
{
    f = 0; c = 0;
    for (;;)
    {
        memset(inq, 0, sizeof(inq));
        for (int i = 1; i <= n; i++)
            d[i] = (i == s ? 0 : INF);
        Q.push(s); inq[s] = 1;
        while (!Q.empty())
        {
            int u = Q.front(); Q.pop();
            inq[u] = 0;
            for (int e = head[u]; e != -1; e = next[e])
                if(cap[e] && d[v[e]] > d[u]+cost[e])
                {
                    d[v[e]] = d[u]+cost[e];
                    if (!inq[v[e]])
                        Q.push(v[e]), inq[v[e]] = 1;
                    pv[v[e]] = u; pe[v[e]] = e;
                }
        }
        if (d[t] == INF) break;
        int a = INF;
        for (int v = t; v != s; v = pv[v])
            a = min(a, cap[pe[v]]);
        for (int v = t; v != s; v = pv[v])
        {
            cap[pe[v]] -= a;
            cap[pe[v]^1] += a;
        }
        f += a;
        c += d[t]*a;
    }
}
int main()
{
    memset(cap, 0, sizeof(cap));
    memset(cost, 0, sizeof(cost));
    memset(head, -1, sizeof(head));
    e = 0;

    return 0;
}
```

Tarjan 求强连通分量+缩点（poj 2186）

```
const int MAXN = 10000+5, MAXM = 50000+5;
int N, M;
int outdgr[MAXN];
int v[MAXM], next[MAXM], head[MAXN], e;
int index, cnt;
int dfn[MAXN], low[MAXN];
int belong[MAXN], amount[MAXN]; //对强连通分量染色（缩点）、记录包含节点数
bool instack[MAXN];
stack<int> S;
void addedge(int x, int y)
{
    v[e] = y;
    next[e] = head[x]; head[x] = e;
    e++;
}
void tarjan(int u)
{
    dfn[u] = low[u] = ++index;
    S.push(u);
    instack[u] = 1;
    for (int i = head[u]; i != -1; i = next[i])
    {
        if (dfn[v[i]] == -1)
        {
            tarjan(v[i]);
            low[u] = min(low[u], low[v[i]]);
        }
        else if (instack[v[i]])
            low[u] = min(low[u], dfn[v[i]]);
    }
    if (low[u] == dfn[u])
    {
        cnt++;
        for (int c = 1; ; c++)
        {
            int x = S.top(); S.pop();
            instack[x] = 0;
            belong[x] = cnt;
            if (x == u)
            {
                amount[cnt] = c;
                break;
            }
```

```
        }
    }
}
int main()
{
    memset(head, -1, sizeof(head));
    e = 0;
    scanf("%d%d", &N, &M);
    for (int i = 0; i < M; i++)
    {
        int A, B;
        scanf("%d%d", &A, &B);
        addedge(A, B);
    }
    memset(dfn, -1, sizeof(dfn)); //Tarjan 初始化
    memset(instack, 0, sizeof(instack));
    index = 0; cnt = 0;
    for (int u = 1; u <= N; u++)
        if (dfn[u] == -1)
            tarjan(u);
    memset(outdgr, 0, sizeof(outdgr));
    for (int u = 1; u <= N; u++) //统计缩点后的出度
        for (int i = head[u]; i != -1; i = next[i])
            if (belong[u] != belong[v[i]])
                outdgr[belong[u]]++;
    int ans = 0;
    for (int i = 1; i <= cnt; i++) if (!outdgr[i])
    {
        if (!ans)
            ans = amount[i];
        else
        {
            ans = 0;
            break;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

基础

bign-bint（比较高效的大数）

```
const int base = 10000; // (base^2) fit into int
const int width = 4; // width = log base
const int maxn = 1000; // n*width: 可表示的最大位数
struct bint
{
    int len, s[maxn];
    bint (int r = 0)
    { // r 应该是字符串!
        for (len = 0; r > 0; r /= base)
            s[len++] = r%base;
    }
    bint &operator = (const bint &r)
    {
        memcpy(this, &r, (r.len+1)*sizeof(int));// !
        return *this;
    }
};
bool operator < (const bint &a, const bint &b)
{
    int i;
    if (a.len != b.len) return a.len < b.len;
    for (i = a.len-1; i >= 0 && a.s[i] == b.s[i]; i--);
    return i < 0 ? 0 : a.s[i] < b.s[i];
}
bool operator <= (const bint &a, const bint &b)
{
    return !(b < a);
}
bint operator + (const bint &a, const bint &b)
{
    bint res; int i, cy = 0;
    for (i = 0; i < a.len || i < b.len || cy > 0; i++)
    {
        if (i < a.len)
            cy += a.s[i];
        if (i < b.len)
            cy += b.s[i];
        res.s[i] = cy%base; cy /= base;
    }
    res.len = i;
    return res;
}
bint operator - (const bint &a, const bint &b)
{
    bint res; int i, cy = 0;
    for (res.len = a.len, i = 0; i < res.len; i++)
    {
        res.s[i] = a.s[i]-cy;
        if (i < b.len)
            res.s[i] -= b.s[i];
        if (res.s[i] < 0)
            cy = 1, res.s[i] += base;
        else
            cy = 0;
    }
    while (res.len > 0 && res.s[res.len-1] == 0)
        res.len--;
    return res;
}
bint operator * (const bint &a, const bint &b)
{
    bint res; res.len = 0;
    if (0 == b.len)
    {
        res.s[0] = 0;
        return res;
    }
    int i, j, cy;
    for (i = 0; i < a.len; i++)
    {
        for (j=cy=0; j < b.len || cy > 0; j++, cy/= base)
        {
            if (j < b.len)
                cy += a.s[i]*b.s[j];
            if (i+j < res.len)
                cy += res.s[i+j];
            if (i+j >= res.len)
                res.s[res.len++] = cy%base;
            else
                res.s[i+j] = cy%base;
        }
    }
    return res;
}
bint operator / (const bint &a, const bint &b)
{ // ! b != 0
    bint tmp, mod, res;
    int i, lf, rg, mid;
    mod.s[0] = mod.len = 0;
    for (i = a.len-1; i >= 0; i--)
    {
        mod = mod*base+a.s[i];
        for (lf = 0, rg = base-1; lf < rg; )
        {
            mid = (lf+rg+1)/2;
            if (b*mid <= mod)
                lf = mid;
            else
                rg = mid-1;
        }
        res.s[i] = lf;
        mod = mod-b*lf;
    }
    res.len = a.len;
    while (res.len > 0 && res.s[res.len-1] == 0)
        res.len--;
    return res; // return mod 就是%运算
}
int digits(bint &a) // 返回位数
{
    if (a.len == 0) return 0;
    int l = (a.len-1)*4;
    for (int t = a.s[a.len-1]; t; ++l, t/=10);
    return l;
}
bool read(bint &b, char buf[]) // 读取失败返回0
{
    if (1 != scanf("%s", buf)) return 0;
    int w, u, len = strlen(buf);
    memset(&b, 0, sizeof(bint));
    if ('0' == buf[0] && 0 == buf[1]) return 1;
    for (w = 1, u = 0; len; )
    {
        u += (buf[--len]-'0')*w;
        if (w*10 == base)
        {
            b.s[b.len++] = u;
            u = 0;
            w = 1;
        }
        else
            w *= 10;
    }
    if (w != 1)
        b.s[b.len++] = u;
    return 1;
}
void write(const bint &v)
{
    int i;
    printf("%d", v.len == 0 ? 0 : v.s[v.len-1]);
    for (i = v.len-2; i >= 0; i--)
        printf("%04d", v.s[i]); // ! 4 == width
    printf("\n");
}
int main()
{
    int a, b; scanf("%d%d", &a, &b);
```

```
    bint A(a), B(b);
    if (B < A)
    {
        write(A+B);
        write(A-B);
        write(A*B);
        write(A/B);
    }
    return 0;
}
```

bign-lrj

```
const int maxn = 200;
struct bign{
    int len, s[maxn];

    bign() {
        memset(s, 0, sizeof(s));
        len = 1;
    }

    bign(int num) {
        *this = num;
    }

    bign(const char* num) {
        *this = num;
    }

    bign operator = (int num) {
        char s[maxn];
        sprintf(s, "%d", num);
        *this = s;
        return *this;
    }

    bign operator = (const char* num) {
        len = strlen(num);
        for(int i = 0; i < len; i++) s[i] = num[len-i-1]
- '0';
        return *this;
    }

    string str() const {
        string res = "";
        for(int i = 0; i < len; i++) res = (char)(s[i] +
'0') + res;
        if(res == "") res = "0";
        return res;
    }

    bign operator + (const bign& b) const{
        bign c;
        c.len = 0;
        for(int i = 0, g = 0; g || i < max(len, b.len);
i++) {
            int x = g;
            if(i < len) x += s[i];
            if(i < b.len) x += b.s[i];
            c.s[c.len++] = x % 10;
            g = x / 10;
        }
        return c;
    }

    void clean() {
        while(len > 1 && !s[len-1]) len--;
    }

    bign operator * (const bign& b) {
        bign c; c.len = len + b.len;
        for(int i = 0; i < len; i++)
            for(int j = 0; j < b.len; j++)
                c.s[i+j] += s[i] * b.s[j];
        for(int i = 0; i < c.len-1; i++){
            c.s[i+1] += c.s[i] / 10;
            c.s[i] %= 10;
        }
        c.clean();
        return c;
    }

    bign operator - (const bign& b) {
        bign c; c.len = 0;
        for(int i = 0, g = 0; i < len; i++) {
            int x = s[i] - g;
            if(i < b.len) x -= b.s[i];
            if(x >= 0) g = 0;
            else {
                g = 1;
                x += 10;
            }
            c.s[c.len++] = x;
        }
        c.clean();
        return c;
    }

    bool operator < (const bign& b) const{
        if(len != b.len) return len < b.len;
        for(int i = len-1; i >= 0; i--)
            if(s[i] != b.s[i]) return s[i] < b.s[i];
        return false;
    }

    bool operator > (const bign& b) const{
        return b < *this;
    }

    bool operator <= (const bign& b) {
        return !(b > *this);
    }

    bool operator == (const bign& b) {
        return !(b < *this) && !(*this < b);
    }

    bign operator += (const bign& b) {
        *this = *this + b;
        return *this;
    }
};

istream& operator >> (istream &in, bign& x) {
    string s;
    in >> s;
    x = s.c_str();
    return in;
}

ostream& operator << (ostream &out, const bign& x) {
    out << x.str();
    return out;
}

int main() {
    bign a;
    cin >> a;
    a += "12345678912345678900000000";
    cout << a*2 << endl;
    return 0;
}
```

动态规划

MaxSum-最大连续和（hdu 1003）

```
const int MAXN = 100000+5, INF = 0x3f3f3f3f;
int T, n, a, sum, min_, max_, s_, s, t, cas;
int main()
{
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d", &n);
        sum = 0; min_ = 0; max_ = -INF, s_ = 1;
        for (int i = 1; i <= n; i++)
        {
            scanf("%d", &a);
            sum += a;
            if (sum-min_ > max_)
            {
                max_ = sum-min_;
                s = s_; t = i;
            }
            if (sum < min_)
            {
                min_ = sum;
                s_ = i+1;
            }
        }
        printf("Case %d:\n", ++cas);
        printf("%d %d %d\n", max_, s, t);
        if (T) printf("\n");
    }
    return 0;
}
```

SG 函数-博弈（poj 2311）

```
const int MAX = 200+5;
int W, H, sg[MAX][MAX];
int g(int w, int h)
{
    if (sg[w][h] != -1)
        return sg[w][h];
    if (2 <= w && w <= 3 && 2 <= h && h <= 3)
        return sg[w][h] = sg[h][w] = 0;
    bool vis[MAX];
    memset(vis, 0, sizeof(vis));
    for (int i = 2; i <= w/2; i++)
    {
        int x = g(i, h)^g(w-i, h);
        vis[x] = 1;
    }
    for (int i = 2; i <= h/2; i++)
    {
        int x = g(w, i)^g(w, h-i);
        vis[x] = 1;
    }
    for (int i = 0; ; i++)
        if (!vis[i])
            return sg[w][h] = sg[h][w] = i;
}
int main()
{
    memset(sg, -1, sizeof(sg));
    while (scanf("%d%d", &W, &H) != EOF)
    {
        if (g(W, H))
            printf("WIN\n");
        else
            printf("LOSE\n");
    }
    return 0;
}
```

LCS 最长公共子序列（uva 10405）

```
const int MAXL = 1000+5;
char str1[MAXL], str2[MAXL];
int dp[MAXL][MAXL];
inline int max(int x, int y) {return x>y?x:y;}
int main()
{
    while (fgets(str1, MAXL, stdin), fgets(str2, MAXL,
stdin))
    {
        memset(dp, 0, sizeof(dp));
        int len1 = strlen(str1)-1, len2 = strlen(str2)-1;
        for (int i = len1-1; i >= 0; i--)
            for (int j = len2-1; j >= 0; j--)
            {
                if (str1[i] == str2[j])
                    dp[i][j] = dp[i+1][j+1]+1;
                else
                    dp[i][j] = max(dp[i+1][j],
dp[i][j+1]);
            }
        printf("%d\n", dp[0][0]);
    }
    return 0;
}
```

TSP 旅行商问题-floyd+hamilton 回路（poj 3311）

```
//单独hamilton_path()用来求hamilton回路
//若求hamilton路，只需抽象出一个辅助源点
const int MAXN = 13;
const int INF = 0x3f3f3f3f;
int n, w[MAXN][MAXN];
int d[1<<11][MAXN];
void floyd()
{
    for (int k = 0; k < n; k++)
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                w[i][j] = min(w[i][j], w[i][k]+w[k][j]);
}
void hamilton_path()
{
    int m = (1<<n);
    for (int u = 0; u < m; u++)
        for(int i = 0; i < n; i++)
            d[u][i] = INF;
    for (int i = 0; i < n; i++)
        d[(1<<i)][i] = w[0][i];
    for (int u = 0; u < m; u++)
        for (int i = 0; i < n; i++) if ((1<<i)&u)
            for (int j = 0; j < n; j++) if ((1<<j)&u)
                if (i != j)
                {
                    int v = u^(1<<j);
                    d[u][j] = min(d[u][j],
d[v][i]+w[i][j]);
                }
}
int main()
{
    while (scanf("%d", &n) && n)
    {
        n++;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                scanf("%d", &w[i][j]);
        floyd(); hamilton_path();
        printf("%d\n", d[(1<<n)-1][0]);
    }
    return 0;
}
```

数据结构

线段树+离散化+扫描线（hysbz 1382）

```
const int MAXN = 10000+5;
int n;
int x_1, y_1, x_2, y_2, y[MAXN<<1];
map<int, int> My;
struct Line
{
    int x, ya, yb, InOut;
    Line(){}
    Line(int X, int Ya, int Yb, int inout)
    {
        x = X; ya = Ya; yb = Yb; InOut = inout;
    }
} line[MAXN<<1];
bool cmp(const Line a, const Line b)
{
    return a.x < b.x;
}
struct Node
{
    int cov, sum;
} T[MAXN<<3];
void Change(int idx, int L, int R, int l, int r, int inout)
{
    int left = idx<<1, right = (idx<<1)^1;
    if (L == l && R == r)
        T[idx].cov += inout;
    else
    {
        int mid = (L+R)>>1;
        if (r <= mid)
            Change(left, L, mid, l, r, inout);
        else if (mid <= l)
            Change(right, mid, R, l, r, inout);
        else
        {
            Change(left, L, mid, l, mid, inout);
            Change(right, mid, R, mid, r, inout);
        }
    }
    if (T[idx].cov)
        T[idx].sum = y[R]-y[L];
    else if (R-L == 1)
        T[idx].sum = 0;
    else
        T[idx].sum = T[left].sum+T[right].sum;
}
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d%d%d%d", &x_1, &y_1, &x_2, &y_2);
        line[i<<1] = Line(x_1, y_1, y_2, 1);
        line[(i<<1)^1] = Line(x_2, y_1, y_2, -1);
        y[i<<1] = y_1;
        y[(i<<1)^1] = y_2;
    }
    sort(line, line+(n<<1), cmp);
    sort(y, y+(n<<1));
    for (int i = 0; i < (n<<1); i++)
        My[y[i]] = i;
    long long ans = 0;
    for (int i = 0; i < (n<<1)-1; i++)
    {
        Change(1, 0, (n<<1)-1, My[line[i].ya],
My[line[i].yb], line[i].InOut);
        ans += (long
long)T[1].sum*(line[i+1].x-line[i].x);
    }
    printf("%lld\n", ans);
    return 0;
}
```

线段树+懒标记（poj 2777）

```
const int MAXN = 100000+5;
int N, T, O;
int Tr[MAXN<<2];
bool mark[MAXN<<2];
void Init(int idx, int L, int R)
{
    Tr[idx] = (1<<0);
    mark[idx] = 0;
    int left = (idx<<1), right = (idx<<1)^1;
    if (L < R)
    {
        int mid = ((L+R)>>1);
        Init(left, L, mid);
        Init(right, mid+1, R);
    }
}
void UpdateSon(int idx)
{
    int left = (idx<<1), right = (idx<<1)^1;
    Tr[left] = Tr[idx];
    mark[left] = 1;
    Tr[right] = Tr[idx];
    mark[right] = 1;
    mark[idx] = 0;
}
void Update(int idx, int L, int R, int l, int r, int c)
{
    if (L < R && mark[idx]) UpdateSon(idx);
    int left = (idx<<1), right = (idx<<1)^1;
    if (L == l && R == r)
    {
        Tr[idx] = (1<<(c-1));
        mark[idx] = 1;
    }
    else
    {
        int mid = ((L+R)>>1);
        if (r <= mid)
            Update(left, L, mid, l, r, c);
        else if (mid < l)
            Update(right, mid+1, R, l, r, c);
        else
        {
            Update(left, L, mid, l, mid, c);
            Update(right, mid+1, R, mid+1, r, c);
        }
        Tr[idx] = Tr[left]|Tr[right];
    }
}
int Query(int idx, int L, int R, int l, int r)
{
    if (L < R && mark[idx]) UpdateSon(idx);
    int left = (idx<<1), right = (idx<<1)^1;
    if (L == l && R == r)
        return Tr[idx];
    int mid = ((L+R)>>1);
    if (r <= mid)
        return Query(left, L, mid, l, r);
    else if (mid < l)
        return Query(right, mid+1, R, l, r);
    else
        return Query(left, L, mid, l, mid)|Query(right,
mid+1, R, mid+1, r);
}
int main()
{
    scanf("%d%d%d", &N, &T, &O);
    Init(1, 1, N);
    while (O--)
    {
        char op[5];
        scanf("%s", op);
```

```
        if (op[0] == 'C')
        {
            int A, B, C;
            scanf("%d%d%d", &A, &B, &C);
            if (A > B) swap(A, B);
            Update(1, 1, N, A, B, C);
        }
        else if (op[0] == 'P')
        {
            int A, B;
            scanf("%d%d", &A, &B);
            if (A > B) swap(A, B);
            int m = Query(1, 1, N, A, B);
            int cnt = 0;
            for (int i = 0; i < T; i++)
                if ((1<<i)&m) cnt++;
            printf("%d\n", cnt);
        }
    }
    return 0;
}
```

RMQ-ST

```
const int MAXN = 50000+5, MAXM = 16;
int N, Q;
int a[MAXN], st[MAXN][MAXM];
int pow2[MAXM];
inline int Most(const int &a, const int &b)
{
    return a > b ? a : b;
}
void InitRMQ(const int &n)
{
    pow2[0] = 1;
    for (int i = 1; i <= MAXM; i++)
        pow2[i] = pow2[i-1]<<1; //预处理2的i次方，最大次幂要大于MAXN
    for (int i = 1; i <= n; i++)
        stmax[i][0] = a[i];
    int k = int(log(double(n))/log(2.0))+1;
    for (int j = 1; j < k; j++)
        for (int i = 1; i <= n; i++)
        {
            if (i+pow2[j-1]-1 <= n)
                stmax[i][j] = Most(stmax[i][j-1],
stmax[i+pow2[j-1]][j-1]);
            else
                break; // st[i][j] = st[i][j-1];
        }
}
int Query(int x, int y) // x, y均为下标:1...n
{
    int k = int(log(double(y-x+1))/log(2.0));
    return Most(stmax[x][k], stmax[y-pow2[k]+1][k]);
}
int main()
{
    scanf("%d%d", &N, &Q);
    for (int i = 1; i <= N; i++)
        scanf("%d", &a[i]);
    InitRMQ(N);
    while (Q--)
    {
        int A, B;
        scanf("%d%d", &A, &B);
        int ans = Query(A, B);
    }
    return 0;
}
```

LCA-Tarjan（hdu 2586）

```
const int MAXN = 40000+5, MAXM = 200+5;
int T, a[MAXM], b[MAXM], lca[MAXM];
int n, m, e, qe;
int v[2*MAXN], next[2*MAXN], head[MAXN], w[2*MAXN];
int qv[2*MAXM], qnext[2*MAXM], qhead[MAXN], ord[2*MAXM];
int fa[MAXN];
bool vis[MAXN], rvs[2*MAXN];
void addedge(int x, int y, int z)
{
    v[e] = y; w[e] = z;
    next[e] = head[x]; head[x] = e;
    e++;
}
void addquery(int x, int y, int z)
{
    qv[qe] = y; ord[qe] = z;
    qnext[qe] = qhead[x]; qhead[x] = qe;
    qe++;
}
int find(int x)
{
    return fa[x] == x ? x : fa[x] = find(fa[x]);
}
void tarjan(int u)
{
    fa[u] = u;
    for (int i = head[u]; i != -1; i = next[i])
        if (!rvs[i])
        {
            rvs[i^1] = 1;
            tarjan(v[i]);
            fa[v[i]] = u;
        }
    vis[u] = 1;
    for (int i = qhead[u]; i != -1; i = qnext[i])
        if (vis[qv[i]])
            lca[ord[i]] = find(qv[i]);
}
int main()
{
    scanf("%d", &T);
    while (T--)
    {
        memset(vis, 0, sizeof(vis));
        memset(head, -1, sizeof(head));
        memset(rvs, 0, sizeof(rvs));
        memset(qhead, -1, sizeof(qhead));
        e = 0; qe = 0;
        scanf("%d%d", &n, &m);
        for (int i = 1; i < n; i++)
        {
            int x, y, z;
            scanf("%d%d%d", &x, &y, &z);
            addedge(x, y, z);
            addedge(y, x, z);
        }
        for (int i = 1; i <= m; i++)
        {
            scanf("%d%d", &a[i], &b[i]);
            addquery(a[i], b[i], i);
            addquery(b[i], a[i], i);
        }
        tarjan(1);
        for (int i = 1; i <= m; i++)
        {
            int r = lca[i], x = a[i], y = b[i], ans = 0;
            while (x != r)
            {
                for (int j = head[x]; j != -1; j = next[j])
                    if (rvs[j])
                    {
                        ans += w[j]; x = v[j];
                        break;
                    }
            }
            while (y != r)
            {
                for (int j = head[y]; j != -1; j = next[j])
                    if (rvs[j])
```

```
                    {
                        ans += w[j]; y = v[j];
                        break;
                    }
                }
            printf("%d\n", ans);
        }
    }
    return 0;
}
```

KMP（poj 3461）

```
const int MAXL = 1000000+5;
int T;
char w[MAXL], t[MAXL];
char *str, *pat;
int fail[MAXL];
void get_fail()
{
    int len2 = strlen(pat);
    fail[0] = -1;
    for (int i = 1, j = -1; i < len2; i++)
    {
        while (j != -1 && pat[j+1] != pat[i])
            j = fail[j];
        if (pat[j+1] == pat[i])
            j++;
        fail[i] = j;
    }
}
int kmp()
{
    int len1 = strlen(str), len2 = strlen(pat);
    int p = 0, q = 0;
    int cnt = 0;
    while (p < len1)
    {
        if (str[p] == pat[q])
            p++, q++;
        else if (q == 0)
            p++; //pat[0]匹配失败，从str下个字符开始
        else
            q = fail[q-1]+1; //pat[p]匹配失败，右移pat串
        if (q == len2)
            cnt++; //记录子串匹配次数
    }
    return cnt;
}
int main()
{
    scanf("%d", &T);
    while (T--)
    {
        scanf("%s%s", w, t);
        pat = w;
        get_fail();
        str = t;
        printf("%d\n", kmp());
    }
    return 0;
}
```

后缀数组（poj 2774-最长公共子串）

```
//sa[]为排好序的后缀数组
//rank[i]为suffix(i)在sa[]中的位置
//sa[rank[i]]=i
//height[i]=最长公共前缀LCP(i-1,i)
const int MAXN = 2*(100000+5);
string s, a, b;
//char s[MAXN]; // MAXN > 256
int len, sa[MAXN], height[MAXN], rank[MAXN], tmp[MAXN],
top[MAXN];
void makesa()
{ // O(N * log N)
    int lena = len < 256 ? 256 : len;
    memset(top, 0, lena*sizeof(int));
    for (int i = 0; i < len ; i++)
        top[rank[i] = s[i]&0xff]++;
    for (int i = 1; i < lena; i++)
        top[i] += top[i-1];
    for (int i = 0; i < len ; i++)
        sa[--top[rank[i]]] = i;
    for (int j, k = 1; k < len; k <<= 1)
    {
        for (int i = 0; i < len; i++)
        {
            j = sa[i]-k;
            if (j < 0)
                j += len;
            tmp[top[rank[j]]++] = j;
        }
        sa[tmp[top[0] = 0]] = j = 0;
        for (int i = 1; i < len; i++)
        {
            if (rank[tmp[i]] != rank[tmp[i-1]] ||
rank[tmp[i]+k] != rank[tmp[i-1]+k])
                top[++j] = i;
            sa[tmp[i]] = j;
        }
        memcpy(rank, sa , len*sizeof(int));
        memcpy(sa , tmp, len*sizeof(int));
        if (j >= len-1)
            break;
    }
}
void lcp()
{ // O(4 * N)
    for (int i, k, j = rank[height[i = k = 0] = 0]; i <
len-1; i++, k++)
        while (k >= 0 && s[i] != s[sa[j-1]+k])
            height[j] = k--, j = rank[sa[j]+1];
}
int main()
{
    cin>>a>>b;
    s = a+"$"+b;
    len = s.length()+1; //!!!
    makesa();
    lcp();
    int ans = 0, mid = a.length();
    for (int i = 1; i < len; i++)
        if ((sa[i-1] < mid && sa[i] > mid) || (sa[i-1] >
mid && sa[i] < mid))
            ans = max(ans, height[i]);
    cout<<ans<<endl;
    return 0;
}
```