省选基础算法

雷宇辰

2017 年 2 月 13 日

目录

1 day1 图论

1.1 有向图强连通分量的 Tarjan 算法

定义 在**有向图** G 中，如果两个顶点 u,v 间存在一条路径 u 到 v 的路径且也存在一条 v 到 u 的路径，则称这两个顶点 u,v 是**强连通的 (strongly connected)**。如果有向图 G 的每两个顶点都强连通，称 G 是一个**强连通图**。有向非强连通图的极大强连通子图，称为**强连通分量 (strongly connected components)**。若将有向图中的强连通分量都缩为一个点，则原图会形成一个 DAG（有向无环图）。

极大强连通子图 G 是一个极大强连通子图当且仅当 G 是一个强连通子图且不存在另一个强连通子图 G’ 使得 G 是 G’ 的真子集。

Tarjan 算法 定义 $dfn(u)$ 为结点 u 搜索的次序编号，给出函数 $low(u)$ 使得

$low(u) = min$

{

$dfn(u)$,

$low(v)$, (u,v) 为树枝边，u 为 v 的父结点

$dfn(v)$ (u,v) 为后向边或指向栈中结点的横叉边

}

当结点 u 的搜索过程结束后，若 $dfn(u) = low(u)$，则以 u 为根的搜索子树上所有还在栈中的结点是一个强连通分量。

代码

tarjan - SCC

```
void tarjan(int u)
{
    dfn[u] = low[u] = ++idx;
    st[top++] = u;
    for (Edge cur : G[u])
        if (!dfn[cur.to])
            tarjan(cur.to),
            low[u] = min(low[u], low[cur.to]);
        else if (!scc[cur.to])
            low[u] = min(low[u], dfn[cur.to]);
    if (dfn[u] == low[u] && ++cnt)
        do scc[st[--top]] = cnt;
        while (st[top] != u);
}
```

练习题

POJ2186/BZOJ1051 - Popular Cows 双倍的快乐

Popular Cows

```
#include <cstdio>
inline int min(int a, int b) { return a < b ? a : b; }
int head[10010], next[50010], to[50010], ecnt;
int dfn[10010], low[10010], stk[10010], scc[10010], top, idx, scccnt;
bool instk[10010];
```

```
int deg[10010];
inline void addEdge(int f, int t)
{
    ecnt++;
    next[ecnt] = head[f];
    head[f] = ecnt;
    to[ecnt] = t;
}
void tarjan(int x)
{
    dfn[x] = low[x] = ++idx;
    instk[stk[top++] = x] = true;
    for (int cur = head[x]; cur; cur = next[cur])
        if (!dfn[to[cur]])
            tarjan(to[cur]), low[x] = min(low[x], low[to[cur]]);
        else if (instk[to[cur]])
            low[x] = min(low[x], dfn[to[cur]]);
    if (dfn[x] == low[x])
    {
        scccnt++;
        do
        {
            top——;
            scc[stk[top]] = scccnt;
            instk[stk[top]] = false;
        } while (stk[top] != x);
    }
}
int main()
{
    int n, m;
    scanf(”%d%d”, &n, &m);
    for (int i = 0, x, y; i < m; i++)
    {
        scanf(”%d%d”, &x, &y);
        addEdge(x, y);
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i])
            tarjan(i);
    for (int i = 1; i <= n; i++)
        for (int cur = head[i]; cur; cur = next[cur])
            if (scc[i] != scc[to[cur]])
                deg[scc[i]]++;
    int zcnt = 0, id = 0;
    for (int i = 1; i <= scccnt; i++)
        if (deg[i] == 0)
            zcnt++, id = i;
    if (zcnt != 1)
        putchar(’0’);
    else
    {
        int ans = 0;
        for (int i = 1; i <= n; i++)
            if (scc[i] == id)
                ans++;
        printf(”%d”, ans);
    }
    return 0;
}
```

POJ3180 - The Cow Prom The N (2 <= N <= 10,000) cows are so excited.

The Cow Prom

```
#include <cstdio>
inline int min(int a, int b) { return a < b ? a : b; }
const int maxn = 100010;
int head[maxn], next[maxn << 1], to[maxn << 1], ecnt, n, m;
int dfn[maxn], scc[maxn], cnt[maxn], scccnt, stk[maxn], low[maxn], idx, top;
inline void addEdge(int f, int t)
{
    ecnt++;
    next[ecnt] = head[f];
    head[f] = ecnt;
    to[ecnt] = t;
}
void tarjan(int x)
{
    dfn[x] = low[x] = ++idx;
    stk[top++] = x;
    for (int i = head[x]; i; i = next[i])
        if (!dfn[to[i]])
            tarjan(to[i]), low[x] = min(low[x], low[to[i]]);
        else if (!scc[to[i]])
            low[x] = min(low[x], dfn[to[i]]);
    if (dfn[x] == low[x])
    {
        scccnt++;
        do
            scc[stk[--top]] = scccnt;
        while (stk[top] != x);
    }
}
int main()
{
    scanf(”%d%d”, &n, &m);
    for (int i = 0, x, y; i < m; i++)
    {
        scanf(”%d%d”, &x, &y);
        addEdge(x, y);
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i]) tarjan(i);
    int ans = 0;
    for (int i = 1; i <= n; i++) cnt[scc[i]]++;
    for (int i = 1; i <= scccnt; i++)
        if (cnt[i] > 1) ans++;
    printf(”%d”, ans);
    return 0;
}
```

POJ1236 - Network of Schools 强连通分量缩点求出度为 0 的和入度为 0 的分量个数

Network of Schools

```
#include <cstdio>
inline int min(int a, int b) { return a < b ? a : b; }
const int maxn = 110, maxm = 10100;
int head[maxn], next[maxm], to[maxm], ecnt, f[maxn], g[maxn];
inline void addEdge(int f, int t)
{
    ecnt++;
```

```
    next[ecnt] = head[f];
    head[f] = ecnt;
    to[ecnt] = t;
}
int dfn[maxn], low[maxn], stk[maxn], scc[maxn], scccnt, top, idx;
void tarjan(int x)
{
    dfn[x] = low[x] = ++idx;
    stk[top++] = x;
    for (int i = head[x]; i; i = next[i])
        if (!dfn[to[i]])
            tarjan(to[i]), low[x] = min(low[x], low[to[i]]);
        else if (!scc[to[i]])
            low[x] = min(low[x], dfn[to[i]]);
    if (dfn[x] == low[x])
    {
        scccnt++;
        do
            scc[stk[--top]] = scccnt;
        while (stk[top] != x);
    }
}
int main()
{
    int n;
    scanf("%d", &n);
    for (int i = 1, x; i <= n; i++)
        for (scanf("%d", &x); x; scanf("%d", &x))
            addEdge(i, x);
    for (int i = 1; i <= n; i++)
        if (!dfn[i]) tarjan(i);
    for (int i = 1; i <= n; i++)
        for (int j = head[i]; j; j = next[j])
            if (scc[i] != scc[to[j]])
                f[scc[i]]++, g[scc[to[j]]]++;
    int ans1 = 0, ans2 = 0;
    if (scccnt == 1)
        printf("1\n0");
    else
    {
        for (int i = 1; i <= scccnt; i++)
            ans1 += f[i] == 0, ans2 += g[i] == 0;
        printf("%d\n%d", ans2, ans1 > ans2 ? ans1 : ans2);
    }
    return 0;
}
```

1.2 图的割点、桥与双连通分量

定义

点连通度与边连通度 在一个**无向连通图**中，如果有一个顶点集合 V，删除顶点集合 V，以及与 V 中顶点相连（至少有一端在 V 中）的所有边后，原图**不连通**，就称这个点集 V 为**割点集合**。

一个图的**点连通度**的定义为：最小割点集合中的顶点数。

类似的，如果有一个边集合，删除这个边集合以后，原图不连通，就称这个点集为**割边集合**。

双连通图、割点与桥 如果一个无向连通图的**点连通度大于** 1，则称该图是**点双连通的 (point biconnected)**，简称双连通或重连通。一个图有**割点**，当且仅当这个图的点连通度为 1，则割点集合的唯一元素被称为**割点 (cut point)**，又叫关节点 (articulation point)。一个图可能有多个割点。
如果一个无向连通图的**边连通度大于** 1，则称该图是**边双连通的 (edge biconnected)**，简称双连通或重连通。一个图有**桥**，当且仅当这个图的边连通度为 1，则割边集合的唯一元素被称为**桥 (bridge)**，又叫关节边 (articulation edge)。一个图可能有多个桥。
可以看出，点双连通与边双连通都可以简称为双连通，它们之间是有着某种联系的，下文中提到的双连通，均既可指点双连通，又可指边双连通。（但这并不意味着它们等价）
双连通分量（分支）：在图 G 的所有子图 G' 中，如果 G' 是双连通的，则称 G' 为双连通子图。如果一个双连通子图 G' 它不是任何一个双连通子图的真子集，则 G' 为极大双连通子图。双连通分量 (biconnected component)，或重连通分量，就是图的极大双连通子图。特殊的，点双连通分量又叫做块。

Tarjan 算法 给出函数 $low(u)$ 使得
$low(u) = min$
{
$dfn(u)$,
$low(v)$, (u,v) 为树枝边 (父子边)
$dfn(v)$ (u,v) 为后向边 (返祖边) 等价于 $dfn(v) < dfn(u)$ 且 v 不为 u 的父亲结点
}

代码

tarjan - BCC

```
void tarjan(int u, int p)
{
    dfn[u] = low[u] = ++idx;
    for (int e = head[u]; e; e = next[e])
        if (!dfn[to[e]])
            tarjan(to[e], u), low[u] = min(low[u], low[to[e]]);
        else if (to[e] != p)
            low[u] = min(low[u], dfn[to[e]]);
}
```

练习题

POJ3177 - Redundant Paths 将一张有桥图通过加边变成边双连通图，至少要加 $\frac{leaf+1}{2}$ 条边。

Redundant Paths

```
#include <cstdio>
inline int min(int a, int b) { return a < b ? a : b; }
int head[5010], to[20010], next[20010], ecnt, map[5010][5010];
int dfn[5010], low[5010], idx, cnt[5010];
void addEdge(int f, int t)
{
    ecnt++;
    next[ecnt] = head[f];
    head[f] = ecnt;
```

```
    to[ecnt] = t;
}
void tarjan(int u, int p)
{
    dfn[u] = low[u] = ++idx;
    for (int e = head[u]; e; e = next[e])
        if (!dfn[to[e]])
            tarjan(to[e], u), low[u] = min(low[u], low[to[e]]);
        else if (to[e] != p)
            low[u] = min(low[u], dfn[to[e]]);
}
int main()
{
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 1, x, y; i <= m; i++)
    {
        scanf("%d%d", &x, &y);
        if (!map[x][y])
        {
            addEdge(x, y);
            addEdge(y, x);
            map[x][y] = map[y][x] = true;
        }
    }
    tarjan(1, 0);
    for (int i = 1; i <= n; i++)
        for (int e = head[i]; e; e = next[e])
            if (low[to[e]] != low[i])
                cnt[low[i]]++;
    int ans = 0;
    for (int i = 1; i <= n; i++)
        ans += cnt[i] == 1;
    printf("%d", (ans + 1) >> 1);
    return 0;
}
```

POJ1523 - SPF 求割点与删除这个点之后有多少个连通分量

Redundant Paths

```
#include <cstdio>
#include <cctype>
#include <cstring>
#define clz(X) memset(X, 0, sizeof(X))
inline int max(int a, int b) { return a > b ? a : b; }
inline int min(int a, int b) { return a < b ? a : b; }
inline void read(int &x)
{
    int ch = x = 0;
    while (!isdigit(ch = getchar()));
    for (; isdigit(ch); ch = getchar())
        x = x * 10 + ch - '0';
}
int map[1010][1010], range;
int dfn[1010], low[1010], idx;
int son, subnet[1010];
void tarjan(int u)
{
    dfn[u] = low[u] = ++idx;
    for (int v = 1; v <= range; v++)
        if (map[u][v])
```

```
            if (!dfn[v])
            {
                tarjan(v);
                low[u] = min(low[u], low[v]);
                if (low[v] >= dfn[u])
                    (u == 1 ? son : subnet[u])++;
            }
            else
                low[u] = min(low[u], dfn[v]);
}
int main()
{
    int x, y, T = 0;
    while (read(x), x)
    {
        clz(map), clz(dfn), clz(low), clz(subnet), son = idx = 0;
        read(y);
        map[x][y] = map[y][x] = 1;
        range = max(x, y);
        while (read(x), x)
        {
            read(y);
            map[x][y] = map[y][x] = 1;
            range = max(range, max(x, y));
        }
        printf("Network #%d\n", ++T);
        tarjan(1);
        bool flag = false;
        if (son > 1) subnet[1] = son - 1;
        for (int i = 1; i <= range; i++)
            if (subnet[i])
                printf("  SPF node %d leaves %d subnets\n", i, subnet[i] + 1),
                flag = true;
        if (!flag)
            puts("  No SPF nodes");
        putchar('\n');
    }
    return 0;
}
```

1.3 2-SAT

定义 给定一个布尔方程，判断是否存在一组布尔变量的取值方案，使得整个方程值为真的问题，被称为布尔方程的可满足性问题 (SAT)。SAT 问题是 NP 完全的，但对于一些特殊形式的 SAT 问题我们可以有效求解。

我们将下面这种布尔方程称为合取范式：

$$(a \vee b \vee c \vee \cdots) \wedge (d \vee e \vee f \vee \cdots) \wedge \cdots$$

其中 $a, b, c, \cdots$ 称为文字，它是一个布尔变量或其否定。像 $(a \vee b \vee c \vee \cdots)$ 这样用 $\vee$ 连接的部分称为子句。如果合取范式的每个子句中的文字个数都不超过两个，那么对应的 SAT 问题又称为 **2-SAT** 问题。

解法 对于给定的 **2-SAT** 问题，首先利用 $\Rightarrow$ 将每个子句 $(a \vee b)$ 改写成等价形式 $(\neg a \Rightarrow b \wedge a \Rightarrow \neg b)$. 这样原布尔公式就变成了把 $a \Rightarrow b$ 形式的布尔公式用 $\wedge$ 连接起来的形式。

对每个布尔变量 x 构造两个顶点分别代表 x 与 $\neg x$。以 $\Rightarrow$ 关系为边建立有向图。若在此图中 a 点能到达 b 点，就表示 a 为真时 b 也一定为真。因此该图中同一个强连通分量中所含的所有变量的布尔值

均相同。

若存在某个变量 x，代表 x 与 $\neg x$ 的两个顶点在同一个强连通分量中，则原布尔表达式的值无法为真。反之若不存在这样的变量，那么我们先将原图中所有的强连通分量缩为一个点，构出一个新图，新图显然是一个拓扑图，我们求出它的一个拓扑序。那么对于每个变量 x，

$$x\text{所在的强连通分量（新图中的点）的拓扑序在}\neg x\text{所在的强连通分量之后} \Leftrightarrow x\text{为真}$$

就是一组合适布尔变量赋值。注意到 Tarjan 算法所求的强连通分量就是按拓扑序的逆序得出的，因此不需要真的缩点建新图求拓扑序，直接利用强连通分量的编号来当做顺序即可。

练习题

POJ3648 - Wedding Additionally, there are several pairs of people conducting adulterous relationships (both different-sex and same-sex relationships are possible)

adulterous relationships

```
#include <cstdio>
#include <cstring>
inline int min(int a, int b) { return a < b ? a : b; }
const int maxn = 2010, maxm = 500010;
int head[maxn], next[maxm], to[maxm], ecnt;
inline void addEdge(int f, int t)
{
    next[ecnt] = head[f];
    head[f] = ecnt;
    to[ecnt] = t;
    ecnt++;
}
int dfn[maxn], low[maxn], stk[maxn], scc[maxn], top, idx, scccnt;
void tarjan(int x)
{
    dfn[x] = low[x] = ++idx;
    stk[top++] = x;
    for (int i = head[x]; ~i; i = next[i])
        if (!dfn[to[i]])
            tarjan(to[i]), low[x] = min(low[x], low[to[i]]);
        else if (!scc[to[i]])
            low[x] = min(low[x], dfn[to[i]]);
    if (dfn[x] == low[x])
    {
        scccnt++;
        do
            scc[stk[--top]] = scccnt;
        while (stk[top] != x);
    }
}
int main()
{
    int m, n;
    while (scanf("%d%d", &n, &m) && (m + n))
    {
        memset(head, -1, sizeof(head));
        memset(dfn, 0, sizeof(dfn));
        memset(low, 0, sizeof(low));
        memset(scc, 0, sizeof(scc));
        idx = top = ecnt = scccnt = 0;
        int a1, a2;
        char c1, c2;
```

```
        for (int i = 0; i < m; i++)
        {
            scanf("%d%c %d%c", &a1, &c1, &a2, &c2);
            a1 = a1 << 1 | (c1 == 'h'), a2 = a2 << 1 | (c2 == 'h');
            addEdge(a1, a2 ^ 1), addEdge(a2, a1 ^ 1);
        }
        addEdge(0, 1);
        for (int i = 0; i < (n << 1); i++)
            if (!dfn[i]) tarjan(i);
        bool flag = true;
        for (int i = 0; i < n && flag; i++)
            if (scc[i << 1] == scc[i << 1 | 1])
                flag = false;
        if (!flag)
            puts("bad luck");
        else if (n < 1)
            putchar('\n');
        else
            for (int i = 1; i < n; i++)
                printf("%d%c%c", i, (scc[i << 1] > scc[i << 1 | 1]) ? 'w' : 'h', " \n"[i == n - 1]);
    }
    return 0;
}
```

POJ3678 - Katu Puzzle 我什么时候做过这个题？

Katu Puzzle

```
#include <cstdio>
#include <cstring>
inline int min(int a, int b) { return a < b ? a : b; }
const int maxn = 10010, maxm = 4000010;
int head[maxn], next[maxm], to[maxm], ecnt;
inline void addEdge(int f, int t)
{
    next[ecnt] = head[f];
    head[f] = ecnt;
    to[ecnt] = t;
    ecnt++;
}
int dfn[maxn], low[maxn], stk[maxn], scc[maxn], top, idx, scccnt;
void tarjan(int x)
{
    dfn[x] = low[x] = ++idx;
    stk[top++] = x;
    for (int i = head[x]; ~i; i = next[i])
        if (!dfn[to[i]])
            tarjan(to[i]), low[x] = min(low[x], low[to[i]]);
        else if (!scc[to[i]])
            low[x] = min(low[x], dfn[to[i]]);
    if (dfn[x] == low[x])
    {
        scccnt++;
        do
            scc[stk[--top]] = scccnt;
        while (stk[top] != x);
    }
}
int main()
{
    int n, m;
    while (~scanf("%d%d", &n, &m))
```

```
    {
        memset(dfn, 0, sizeof(dfn));
        memset(low, 0, sizeof(low));
        memset(scc, 0, sizeof(scc));
        memset(head, -1, sizeof(head));
        ecnt = top = idx = scccnt = 0;
        for (int i = 0, u, v, w; i < m; ++i)
        {
            char op[5];
            scanf("%d%d%d%s", &u, &v, &w, op);
            if (op[0] == 'A')
                if (w)
                {
                    addEdge(u << 1, v << 1 | 1), addEdge(v << 1, u << 1 | 1);
                    addEdge(u << 1, v << 1), addEdge(v << 1 | 1, u << 1 | 1);
                    addEdge(u << 1 | 1, v << 1 | 1), addEdge(v << 1, u << 1);
                }
                else
                {
                    addEdge(u << 1 | 1, v << 1), addEdge(v << 1 | 1, u << 1);
                }
            if (op[0] == 'O')
                if (w)
                {
                    addEdge(u << 1, v << 1 | 1), addEdge(v << 1, u << 1 | 1);
                }
                else
                {
                    addEdge(u << 1, v << 1), addEdge(v << 1 | 1, u << 1 | 1);
                    addEdge(u << 1 | 1, v << 1 | 1), addEdge(v << 1, u << 1);
                    addEdge(u << 1 | 1, v << 1), addEdge(v << 1 | 1, u << 1);
                }
            if (op[0] == 'X')
                if (w)
                {
                    addEdge(u << 1, v << 1 | 1), addEdge(v << 1, u << 1 | 1);
                    addEdge(u << 1 | 1, v << 1), addEdge(v << 1 | 1, u << 1);
                }
                else
                {
                    addEdge(u << 1, v << 1), addEdge(v << 1 | 1, u << 1 | 1);
                    addEdge(u << 1 | 1, v << 1 | 1), addEdge(v << 1, u << 1);
                }
        }
        for (int i = 0; i < (n << 1); i++)
            if (!dfn[i]) tarjan(i);
        bool flag = true;
        for (int i = 0; i < n && flag; i++)
            if (scc[i << 1] == scc[i << 1 | 1])
                flag = false;
        puts(flag ? "YES" : "NO");
    }
    return 0;
}
```

POJ2749 - Building roads 杀光奶牛问题就会得到解决

Building roads

```
#include <cstdio>
#include <cstring>
inline int abs(int x) { return x >= 0 ? x : -x; }
```

```
inline int min(int a, int b) { return a < b ? a : b; }
const int inf = 0x3f3f3f3f, maxn = 10010, maxm = 1200010;
int head[maxn], next[maxm], to[maxm], ecnt, n, A, B;
int dfn[maxn], low[maxn], stk[maxn], scc[maxn], top, idx, scccnt;
int sx1, sy1, sx2, sy2, sLen, X[maxn], Y[maxn], hate[maxn][2], like[maxn][2],
d[maxn];
inline void addEdge(int f, int t)
{
    next[ecnt] = head[f];
    head[f] = ecnt;
    to[ecnt] = t;
    ecnt++;
}
void tarjan(int x)
{
    dfn[x] = low[x] = ++idx;
    stk[top++] = x;
    for (int i = head[x]; ~i; i = next[i])
        if (!dfn[to[i]])
            tarjan(to[i]), low[x] = min(low[x], low[to[i]]);
        else if (!scc[to[i]])
            low[x] = min(low[x], dfn[to[i]]);
    if (dfn[x] == low[x])
    {
        scccnt++;
        do
            scc[stk[--top]] = scccnt;
        while (stk[top] != x);
    }
}
bool check(int x)
{
    memset(dfn, 0, sizeof(dfn));
    memset(low, 0, sizeof(low));
    memset(scc, 0, sizeof(scc));
    memset(head, -1, sizeof(head));
    ecnt = top = idx = scccnt = 0;
    for (int i = 1; i <= n; i++)
        for (int j = i + 1; j <= n; j++)
        {
            int l1 = d[i << 1], l2 = d[i << 1 | 1];
            int r1 = d[j << 1], r2 = d[j << 1 | 1];
            if (l1 + r1 > x)
                addEdge(i << 1, j << 1 | 1), addEdge(j << 1, i << 1 | 1);
            if (l1 + r2 + sLen > x)
                addEdge(i << 1, j << 1), addEdge(j << 1 | 1, i << 1 | 1);
            if (l2 + r1 + sLen > x)
                addEdge(i << 1 | 1, j << 1 | 1), addEdge(j << 1, i << 1);
            if (l2 + r2 > x)
                addEdge(i << 1 | 1, j << 1), addEdge(j << 1 | 1, i << 1);
        }
    for (int i = 1, a, b; i <= A; i++)
    {
        a = hate[i][0], b = hate[i][1];
        addEdge(a << 1, b << 1 | 1);
        addEdge(a << 1 | 1, b << 1);
        addEdge(b << 1, a << 1 | 1);
        addEdge(b << 1 | 1, a << 1);
    }
    for (int i = 1, a, b; i <= B; i++)
    {
        a = like[i][0], b = like[i][1];
```

```
        addEdge(a << 1, b << 1);
        addEdge(a << 1 | 1, b << 1 | 1);
        addEdge(b << 1, a << 1);
        addEdge(b << 1 | 1, a << 1 | 1);
    }
    for (int i = 1; i <= (n << 1); i++)
        if (!dfn[i])
            tarjan(i);
    for (int i = 1; i <= n; i++)
        if (scc[i << 1] == scc[i << 1 | 1])
            return false;
    return true;
}
int main()
{
    scanf("%d%d%d%d%d%d%d", &n, &A, &B, &sx1, &sy1, &sx2, &sy2);
    sLen = abs(sx1 - sx2) + abs(sy1 - sy2);
    for (int i = 1; i <= n; i++)
        scanf("%d%d", X + i, Y + i);
    for (int i = 1; i <= n; i++)
        d[i << 1] = abs(X[i] - sx1) + abs(Y[i] - sy1),
        d[i << 1 | 1] = abs(X[i] - sx2) + abs(Y[i] - sy2);
    for (int i = 1; i <= A; i++)
        scanf("%d%d", &hate[i][0], &hate[i][1]);
    for (int i = 1; i <= B; i++)
        scanf("%d%d", &like[i][0], &like[i][1]);
    int l = 0, r = 8000000, m, ans = -1;
    while (l <= r)
        check(m = (l + r) >> 1) ? r = (ans = m) - 1 : l = m + 1;
    printf("%d\n", ans);
    return 0;
}
```