Bài toán 2-SAT
GrintonDH

I. Mở đầu

SAT (Boolean satisfiability problem) là dạng bài toán được ứng dụng nhiều trong lập trình thi đấu. Trong khi SAT là bài toán đầu tiên được chứng minh là bài toán NP-đầy đủ (theo định luật Cook-Levin) thì 2-SAT có thể giải được theo một thuật toán hiệu quả có độ phức tạp $O(n + m)$ với n là số biến và m là số phương trình.

II. SAT

1. Định nghĩa

SAT (Boolean satisfiability problem - Bài toán thỏa mãn biểu thức Boolean) là bài toán xác định có tồn tại các giá trị sao cho thỏa mãn một biểu thức Boolean cho trước. Nói cách khác, bài toán yêu cầu tìm cách gán giá trị cho các biến thành TRUE / FALSE sao cho biểu thức nhận được có giá trị là TRUE.

SAT là bài toán đầu tiên đã được chứng minh là NP - đầy đủ. Điều này có nghĩa là tất cả các bài toán có lớp độ phức tạp là NP, có độ khó tương đương với độ khó giải bài toán SAT. Không có thuật toán nào được biết đến để có thể giải bài toán SAT.

2. Biểu thức SAT

Một biểu thức mệnh đề toán học, hay biểu thức boolean, được xây dựng bởi các biến, phép AND ($\wedge$ hội), phép OR ($\vee$ tuyển), NOT ($\neg$ phủ định). Một biểu thức được gọi là *thỏa mãn (satisfiable)* nếu có thể có giá trị TRUE nếu gán lần lượt các biến với các giá trị logic (TRUE / FALSE). Bài toán SAT chính là kiểm tra xem có tồn tại cách gán như thế không.

Một biểu thức ở dạng chuẩn hội (conjunctive normal form - CNF) nếu nó là hội của các mệnh đề (hoặc chỉ một mệnh đề riêng lẻ). Ví dụ, biểu thức $(x1 \vee \neg x2) \wedge (\neg x1 \vee x2 \vee x3) \wedge \neg x1$ được gọi là biểu thức ở dạng chuẩn hội.

III. 2-SAT

Xét biểu thức $f = (x1 \vee y1) \wedge (x2 \vee y2) \wedge \ldots \wedge (xn \vee yn)$. Yêu cầu kiểm tra f có thỏa mãn hay không. Đây là dạng bài toán 2-SAT. Không như SAT là bài toán NP-đầy đủ, 2-SAT có thể giải quyết được trong thời gian đa thức sử dụng bài toán Tìm thành phần liên thông mạnh.

1. Nhắc lại Thành phần liên thông mạnh

Một đồ thị có hướng là liên thông mạnh nếu tồn tại đường đi giữa hai đỉnh bất kỳ của đồ thị. Bài toán tìm các thành phần liên thông mạnh có thể giải bằng

thuật toán Kosaraju hoặc thuật toán Tarjan. Ở đây sẽ sử dụng thuật toán Tarjan để giải quyết bài toán 2-SAT.

2. Giải quyết bài toán 2-SAT

a. Tạo đồ thị

Nhận thấy $a \vee b$ tương đương với $\neg a \Rightarrow b \wedge \neg b \Rightarrow a$. Điều này có nghĩa là nếu một biến là false, biến còn lại phải có giá trị bằng true.

Chúng ta cần tạo một đồ thị có hướng, với mỗi đỉnh x sẽ có hai đỉnh tương đương trong đồ thị là v_x và $v_{\neg x}$ - biểu diễn x và $\neg x$. Các cạnh sẽ được tạo tương ứng.

Ví dụ, xét biểu thức: $(a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee \neg c)$, các cạnh sau sẽ được tạo:

$\neg a \Rightarrow \neg b$	$b \Rightarrow a$	$a \Rightarrow b$	$\neg b \Rightarrow \neg a$
$a \Rightarrow \neg b$	$b \Rightarrow \neg a$	$\neg a \Rightarrow \neg c$	$c \Rightarrow a$

Chú ý rằng, nếu có cạnh nối từ a đến b thì cũng sẽ có cạnh nối từ $\neg a$ đến $\neg b$.

b. Giải bài toán 2-SAT

Ta sẽ sử dụng thuật toán Tarjan để tách đồ thị thành các thành phần liên thông mạnh.

Nếu x có thể đi đến $\neg x$ và ngược lại, tức x và $\neg x$ nằm trên cùng một thành phần liên thông mạnh, việc gán true hoặc false cho x sẽ gây ra mâu thuẫn. Khi đó biểu thức 2-SAT tương ứng sẽ không có nghiệm.

Sắp xếp lại các thành phần liên thông theo trình tự topo (có nghĩa, scc[v] <= scc[u] nếu có đường đi từ v đến u và scc[i] là chỉ số của thành phần liên thông mạnh tương ứng chứa i). Do đó, nếu scc[x] < scc[$\neg x$] ta gán x giá trị false, và gán giá trị true nếu scc[x] > scc[$\neg x$].

Code thuật toán 2-SAT:

```
struct Twosat {
        vector<vector<int>> adj;
        vector<int> num, low, scc;
        int cnt, k;
        stack<int> st;

        Twosat(int sz) {
```

```
        adj.resize(2 * sz);
        num.resize(2 * sz), low.resize(2 * sz);
        scc.resize(2 * sz);
        cnt = k = 0;
    }

    void add(int x, bool sx, int y, bool sy) {
        x = x << 1 ^ sx;
        y = y << 1 ^ sy;
        adj[x ^ 1].pb(y);
        adj[y ^ 1].pb(x);
        return ;
    }

    void dfs(int u) {
        num[u] = low[u] = ++cnt;
        st.push(u);
        repa(v, adj[u])
            if(num[v])
                low[u] = min(low[u], num[v]);
            else {
                dfs(v);
                low[u] = min(low[u], low[v]);
            }

        if(num[u] == low[u]) {
            ++k;
            int x = 0;
            do {
                x = st.top();
                scc[x] = k;
                st.pop();
                low[x] = num[x] = mod;
            } while(x != u);
        }
        return ;
    }

    vector<int> solve() {
        rep(i, 0, int(adj.size()) - 1)
            if(!num[i]) dfs(i);
        int n = int(adj.size()) / 2;
        vector<int> ans(n);
        rep(i, 0, n - 1)
```

```
                        if(scc[i << 1 ^ false] == scc[i << 1 ^ true])
                                return vector<int>(0);
                        else if(scc[i << 1 ^ false] > scc[i << 1 ^ true])
                                ans[i] = 1;
                return ans;
        }
};
```

c. Xây dựng các biểu thức 2-SAT

Xét một số trường hợp đặc biệt như sau:

Mệnh đề	Biểu thức 2-SAT
a kéo theo b	$\neg a \ OR \ \neg b = 1$
a = b (a và b có cùng giá trị)	$(a \ OR \ \neg b)$ AND $(\neg a \ OR \ b) = 1$
a != b (a và b luôn khác giá trị)	$(a \ OR \ b)$ AND $(\neg a \ OR \ \neg b) = 1$
a = true (a luôn đúng)	$a \ OR \ a = 1$
a = false (a luôn sai)	$\neg a \ OR \ \neg a = 1$
Ít nhất một trong hai biến *a* hoặc *b* nhận giá trị true	$a \ OR \ b = 1$
Ít nhất một trong hai biến *a* hoặc *b* nhận giá trị false	$\neg a \ OR \ \neg b = 1$

Cách xây dựng biểu thức

Bước 1: Xác định các biến của bài toán.

Bước 2: Xác định các điều kiện ràng buộc giữa các biến.

Bước 3: Xây dựng bảng giá trị theo các điều kiện ràng buộc. Từ đó tìm cách đưa về biểu thức (hoặc hệ các biểu thức) có dạng $a \ OR \ b = 1$ hoặc dựa vào các trường hợp đặc biệt trên.

IV. Các bài toán

Bài 1: https://cses.fi/problemset/task/1684 Giant Pizza

Tóm tắt đề bài

Có *n* người muốn ăn một chiếc Pizza khổng lồ. Có *m* loại topping khác nhau cho chiếc Pizza này. Mỗi người có hai yêu cầu muốn / không muốn có loại topping thứ *x* được sử dụng. Bạn cần phải chọn ra một số loại topping sao cho thỏa mãn ít nhất một trong hai yêu cầu của tất cả mọi người.

Input:

Dòng đầu tiên nhập hai số nguyên *n, m*.

n dòng tiếp theo mỗi dòng chứa hai yêu cầu, mỗi yêu cầu có dạng + *x* (muốn có loại topping thứ *x*) hoặc - *x* (không muốn có loại topping thứ *x*).

Output:

In ra một dòng chứa m kí hiệu. Kí hiệu thứ i là + nếu mua loại i, là - nếu không mua loại thứ i. Bạn có thể in ra lời giải bất kỳ. Nếu không tồn tại, in ra IMPOSSIBLE.

Cách tiếp cận

Xét yêu cầu của mỗi người là a và b. Do ít nhất một trong hai yêu cầu phải được thực hiện nên nếu gán a = true nếu thực hiện yêu cầu a, a = false nếu không thực hiện yêu cầu a, ta được bảng:

(a, b)	TRUE	FALSE
TRUE	(1, 1)	(1, 0)
FALSE	(0, 1)	Không thỏa mãn

Từ đó ta có thể xây dựng một biểu thức cho yêu cầu của mỗi người:

$$a \vee b = 1.$$

Do mỗi người trong n người đều phải thỏa mãn ít nhất một yêu cầu nên ta được một biểu thức 2-SAT. Sử dụng thuật toán Tarjan, ta có thể giải quyết bài toán.

ĐPT: $O(n + m)$.

Code

```
/** Author: GrintonDH **/
/** Problem: CSES_1684 **/

#include <bits/stdc++.h>

#define task                "CSES_1684"
#define rep(i, a, b)        for(int (i) = (a); (i) <= (b); ++(i))
#define repd(i, a, b)       for(int (i) = (a); (i) >= (b); --(i))
#define repa(a, u)          for(auto (a) : (u))

#define ft                  first
#define sd                  second
#define pb                  push_back
#define mp                  make_pair

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
```

```
const int maxn = 1e5 + 5;
const long long mod = 1e9 + 7;

struct Twosat {
    vector<vector<int>> adj;
    vector<int> num, low, scc;
    int cnt, k;
    stack<int> st;

    Twosat(int sz) {
        adj.resize(2 * sz);
        num.resize(2 * sz), low.resize(2 * sz);
        scc.resize(2 * sz);
        cnt = k = 0;
    }

    void add(int x, bool sx, int y, bool sy) {
        x = x << 1 ^ sx, y = y << 1 ^ sy;
        adj[x ^ 1].pb(y);
        adj[y ^ 1].pb(x);
        return ;
    }

    void dfs(int u) {
        num[u] = low[u] = ++cnt;
        st.push(u);
        repa(v, adj[u])
            if(num[v])
                low[u] = min(low[u], num[v]);
            else {
                dfs(v);
                low[u] = min(low[u], low[v]);
            }

        if(num[u] == low[u]) {
            ++k;
            int x = 0;
            do {
                x = st.top();
                scc[x] = k;
                st.pop();
                low[x] = num[x] = mod;
            } while(x != u);
        }
        return ;
```

```
    }

    vector<int> solve() {
        rep(i, 0, int(adj.size()) - 1)
            if(!num[i]) dfs(i);
        int n = int(adj.size()) / 2;
        vector<int> ans(n);
        rep(i, 0, n - 1)
            if(scc[i << 1 ^ false] == scc[i << 1 ^ true])
                return vector<int>(0);
            else if(scc[i << 1 ^ false] > scc[i << 1 ^ true])
                ans[i] = 1;
        return ans;
    }
};

main()
{
    ios_base::sync_with_stdio(false),cin.tie(0),cout.tie(0);
    int n, m;
    cin >> n >> m;
    Twosat G = Twosat(m);

    rep(i, 1, n) {
        int a, b, x, y;
        char c, d;
        cin >> c >> a >> d >> b;
        a--, b--;
        if(c == '+') x = 1; else x = 0;
        if(d == '+') y = 1; else y = 0;

        G.add(b, y, a, x);
    }

    vector<int> ans = G.solve();

    if(ans.size() == 0)
        cout << "IMPOSSIBLE\n";
    else {
        rep(i, 0, m - 1)
            if(ans[i] == 0)
                cout << "- ";
            else cout << "+ ";
    }
}
```

Bài 2: https://codeforces.com/problemset/problem/776/D The Door Problem

Tóm tắt đề bài

Có *n* căn phòng và *m* công tắc. Một công tắc có thể quản lý một vài căn phòng, nhưng một căn phòng chỉ có thể bị điều khiển bởi đúng duy nhất hai công tắc. Ban đầu, bạn biết trạng thái của mỗi phòng là mở hay đóng và biết rằng mỗi khi nhấn vào công tắc, trạng thái của tất cả các phòng mà nó quản lý sẽ bị thay đổi trạng thái. Hãy kiểm tra xem có cách nhấn công tắc sao cho tồn tại thời điểm mà tất cả *n* căn phòng đều mở.

Input

Dòng đầu tiên chứa hai số *n, m*.

Dòng thứ hai chứa *n* số nguyên - 0 nếu phòng đang đóng và 1 nếu phòng đang mở.

m dòng tiếp theo có dạng: *x* - số căn phòng mà công tắc tương ứng quản lý và *x* số nguyên phía sau là chỉ số của các căn phòng bị công tắc tương ứng quản lý.

Output

In ra "YES" nếu tồn tại thời điểm như đề bài, "NO" nếu không tồn tại.

Cách tiếp cận

Nhận xét:

+) Một công tắc không bao giờ bật hai lần.

+) Nếu cửa đang đóng, ta phải bật một trong hai công tắc lên.

+) Nếu cửa đang mở, ta không được bật hai công tắc tương ứng, hoặc bật cả hai công tắc đó lên.

Từ nhận xét, gọi hai công tắc tương ứng với căn phòng thứ *i* là *a* và *b*. Khi đó:

+) Nếu $r_i = 1$ thì $a \mathrel{!}= b$

+) Nếu $r_i = 0$ thì $a == b$

Code

```
/** Author: GrintonDH **/

#include <bits/stdc++.h>

#define task                    "776D"
#define rep(i, a, b)        for(int (i) = (a); (i) <= (b); ++(i))
```

```
#define repd(i, a, b)       for(int (i) = (a); (i) >= (b); --(i))
#define repa(u, v)              for(auto u : v)

#define bit(a, i)           (((a) >> (i)) & 1)
#define mask(i)                 (1LL << (i))
#define turn(a, i)              ((a) ^ mask(i))

#define pb                      push_back
#define ft                      first
#define sd                      second
#define mp                      make_pair

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef priority_queue<pii, vector<pii>, greater<pii>> minqueue;
typedef priority_queue<pii> maxqueue;

const int maxn = 1e5 + 5;
const long long mod = 1e9 + 7;

struct twosat {
    vector<vector<int>> adj;
    vector<int> scc, num, low;
    stack<int> st;
    int cnt, k;

    twosat(int sz) {
        adj.resize(2 * sz);
        scc.resize(2 * sz), num.resize(2 * sz), low.resize(2 * sz);
        cnt = k = 0;
    }

    void add(int x, bool sx, int y, bool sy) {
        x = x << 1 ^ sx;
        y = y << 1 ^ sy;
        adj[x ^ 1].pb(y);
        adj[y ^ 1].pb(x);
    }

    void dfs(int u) {
        num[u] = low[u] = ++cnt;
        st.push(u);
        repa(v, adj[u]) {
            if(num[v])
```

```
                low[u] = min(low[u], num[v]);
            else {
                dfs(v);
                low[u] = min(low[u], low[v]);
            }
        }

        if(num[u] == low[u]) {
            ++k;
            int x = 0;
            do {
                x = st.top();
                st.pop();
                scc[x] = k;
                num[x] = low[x] = mod;
            } while(x != u);
        }
    }

    vector<int> solve() {
        rep(i, 0, int(adj.size()) - 1)
            if(!num[i]) dfs(i);

        int n = int(adj.size()) / 2;
        vector<int> ans(n);

        rep(i, 0, n - 1)
            if(scc[i << 1 ^ false] == scc[i << 1 ^ true])
                return vector<int>(0);
            else
                ans[i] = scc[i << 1 ^ false] > scc[i << 1 ^ true];
            return ans;
    }
};

int n, m, c[maxn], a[maxn], b[maxn];

main() {
    ios_base::sync_with_stdio(false),cin.tie(0),cout.tie(0);

    cin >> n >> m;

    twosat G(m);

    rep(i, 1, n)
        cin >> c[i];
```

```
    rep(i, 1, m) {
        int x; cin >> x;
        rep(j, 1, x) {
            int u; cin >> u;
            if(a[u] > 0) b[u] = i;
            else a[u] = i;
        }
    }

    rep(i, 1, n) {
         a[i]--, b[i]--;
        if(b[i] == -1) {
            if(c[i] == 1)
                G.add(a[i], 0, a[i], 0);
            else
                G.add(a[i], 1, a[i], 1);
        }
        else {
            if(c[i] == 1) {
                G.add(a[i], 0, b[i], 1);
                G.add(a[i], 1, b[i], 0);
            }
            else {
                G.add(a[i], 0, b[i], 0);
                G.add(a[i], 1, b[i], 1);
            }
        }
    }

    vector<int> ans = G.solve();

    if(ans.size() == 0)
        cout << "NO\n";
    else
        cout << "YES\n";
}
```

Bài 3: (ELECT - VNSPOJ) Thống nhất đất nước

Tóm tắt đề bài

Nước *X* có *N* đảng khác nhau. Mỗi đảng có hai ứng viên. Tuy nhiên có *M* cặp ứng viên vì lí do khác nhau nên căm thù nhau. Bạn phải chọn ra *N* ứng viên, sao cho:

+) Mỗi đảng chỉ được chọn duy nhất một ứng viên.

+) Không có hai ứng viên ghét nhau cùng được chọn

Hãy in ra một cách chọn bất kỳ,

Input

Dòng đầu tiên chứa hai số *N, M*. Ứng viên thứ nhất của đảng *i* có số hiệu là $2i$, ứng viên thứ hai có số hiệu là $2i + 1$.

M dòng tiếp theo có dạng *u v* cho biết hai người có số hiệu này ghét nhau.

Output

Dòng đầu tiên in ra 0 nếu không tồn tại cách xếp, nếu tồn tại in ra 1.

Nếu tồn tại cách xếp, in ra những người được chọn.

Cách tiếp cận

Xét hai điều kiện:

+) Mỗi đảng chỉ được chọn duy nhất một ứng viên: $2i$!= $2i + 1$.

+) Không có hai ứng viên ghét nhau cùng được chọn: Ít nhất một trong hai biến *u* hoặc *v* nhận giá trị false.

Code

```
/** Author: GrintonDH **/

#include <bits/stdc++.h>

#define task                    "ELECT_SPOJ"
#define rep(i, a, b)            for(int (i) = (a); (i) <= (b); ++(i))
#define repd(i, a, b)           for(int (i) = (a); (i) >= (b); --(i))
#define repa(a, u)              for(auto (a) : (u))

#define ft                      first
#define sd                      second
#define pb                      push_back
#define mp                      make_pair

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef priority_queue<pii, vector<pii>, greater<pii>> minqueue;
typedef priority_queue<pii> maxqueue;

const int maxn = 1e5 + 5;
const long long mod = 1e9 + 7;

struct TWO_SAT {
```

```
vector<vector<int>> adj;
vector<int> num, low, scc;
int cnt, k;
stack<int> s;

TWO_SAT(int sz) {
	adj.resize(sz * 2);
	num.resize(sz * 2), low.resize(sz * 2), scc.resize(sz * 2);
	cnt = 0, k = 0;
}

void add(int x, int statX, int y, int statY) {
	int nodeX = x << 1 ^ statX;
	int nodeY = y << 1 ^ statY;
	adj[nodeX ^ 1].pb(nodeY);
	adj[nodeY ^ 1].pb(nodeX);
}

void dfs(int u) {
	num[u] = low[u] = ++cnt;
	s.push(u);

	repa(v, adj[u]) {
		if(num[v] != 0) low[u] = min(low[u], num[v]);
		else {
			dfs(v);
			low[u] = min(low[u], low[v]);
		}
	}

	if(low[u] == num[u]) {
		++k;
		int x = 0;
		do {
			x = s.top();
			s.pop();
			scc[x] = k;
			num[x] = low[x] = mod;
		} while(x != u);
	}
}

vector<int> solve() {
	rep(i, 0, int(adj.size()) - 1)
		if(num[i] == 0)
			dfs(i);
```

```
		int n = int(adj.size()) / 2;
		vector<int> ans(n);
		rep(i, 0, n - 1) {
			if(scc[i << 1 ^ false] == scc[i << 1 ^ true]) return vector<int>(0);
			ans[i] = (scc[i << 1 ^ false] > scc[i << 1 ^ true]);
		}
		return ans;
	}
} ;
int n, m;
main() {
	ios_base::sync_with_stdio(false),cin.tie(0),cout.tie(0);

	cin >> n >> m;
	TWO_SAT G(2 * n);
	rep(i, 1, 2 * n) { /// x != y => x or y = 1 and not(x) or not(y) = 1
		G.add(i - 1, 1, i, 1);
		G.add(i - 1, 0, i, 0);
		i++;
	}

	rep(i, 1, m) {
		int a, b;
		cin >> a >> b;
		a--, b--;
		G.add(a, 0, b, 0);
	}

	vector<int> ans = G.solve(), id;

	if(ans.size() == 0) {
		cout << 0;
		return 0;
	}
	else {
		rep(i, 0, 2 * n - 1)
			if(ans[i] == 1)
				id.pb(i + 1);
		if(id.size() != n)
			cout << 0 << "\n";
		else {
			cout << 1 << "\n";
			repa(i, id)
				cout << i << " ";
```

```
            }
        }
}
```

Các bài toán khác:

1. **11930 - Rectangles - UVa Online Judge**
2. **10319 - Manhattan - UVa Online Judge**
3. **11294 - Wedding - UVa Online Judge**
4. **TORNJEVI - SPOJ**

Gợi ý TORNJEVI: Ta cần tạo các ràng buộc về:
+) Chiều của súng (Chọn một trong hai: Left hoặc Right, Up hoặc Down)
+) Giữa các súng với nhau (Các súng không được bắn xuyên qua nhau)
+) Giữa mục tiêu với các súng (Chú ý mỗi mục tiêu chỉ có thể bị ngắm bởi tối đa hai súng).

5. **2-SAT problems on Codeforces Problemset**
6. **DATE - Sắp lịch hẹn hò**

Nguồn

1. **https://cp-algorithms.com/graph/2SAT.html - 2-SAT - Competitive Programming Algorithms**
2. **https://codeforces.com/blog/entry/16205 - 2SAT Tutorial - Codeforces.com (Handle: Swift)**
3. **https://en.wikipedia.org/wiki/Boolean_satisfiability_problem - Boolean satisfiability problem - Wikipedia**
4. **https://en.wikipedia.org/wiki/2-satisfiability 2-satisfiability - Wikipedia**
5. **https://vnoi.info**
6. **https://onlinejudge.org**
7. **https://spoj.com**
8. **https://codeforces.com**
9. **GSPVH CUTEPHOMAIQUE**