

## 2-SAT

SAT 是适定性 (Satisfiability) 问题的简称。一般形式为  $k$ -适定性问题, 简称  $k$ -SAT。而当  $k > 2$  时该问题为 NP 完全的。所以我们只研究  $k = 2$  的情况。



### 定义

2-SAT, 简单的说就是给出  $n$  个布尔方程, 每个方程和两个变量相关, 如  $a \vee b$ , 表示变量  $a, b$  至少满足一个。然后判断是否存在可行方案, 显然可能有多种选择方案, 一般题中只要求出一种即可。另外,  $\neg a$  表示  $a$  取反。

### 解决思路

#### 洛谷 P4782 「模板」2-SAT

有  $n$  个布尔变量  $x_1 \sim x_n$ , 另有  $m$  个需要满足的条件, 每个条件的形式都是「 $x_i$  为 true / false 或  $x_j$  为 true / false」。比如「 $x_1$  为真或  $x_3$  为假」、「 $x_7$  为假或  $x_2$  为假」。

2-SAT 问题的目标是给每个变量赋值使得所有条件得到满足。

使用布尔方程表示上述问题。设  $a$  表示  $x_a$  为真 ( $\neg a$  就表示  $x_a$  为假)。如果有个人提出的要求分别是  $a$  和  $b$ , 即  $(a \vee b)$  (变量  $a, b$  至少满足一个)。对这些变量关系建有向图, 则把  $a$  成立或不成立用图中的点表示,  $\neg a \rightarrow b$  和  $\neg b \rightarrow a$ , 表示  $a$  **不成立** 则  $b$  **一定成立**; 同理,  $b$  **不成立** 则  $a$  **一定成立**。建图之后, 我们就可以使用缩点算法来求解 2-SAT 问题了。

原式	建图
$\neg a \vee b$	$a \rightarrow b$ 和 $\neg b \rightarrow \neg a$
$a \vee b$	$\neg a \rightarrow b$ 和 $\neg b \rightarrow a$
$\neg a \vee \neg b$	$a \rightarrow \neg b$ 和 $b \rightarrow \neg a$

许多 2-SAT 问题都需要找出如  $a$  **不成立**, 则  $b$  **成立** 的关系。

### 求解

思考如果两点在同一强连通分量里有什么含义。根据前文边的逻辑意义可知：若两点在同一强连通分量内，则这两点代表的条件 **要么都满足，要么都不满足**。

建图后我们使用 **Tarjan 算法找 SCC**，判断对于任意布尔变量  $a$ ，表示  $a$  成立的点和表示  $a$  不成立的点是否在同一个 SCC 中（同一条件不可能既满足又不满足，或既不满足又并非不满足），若有则输出无解，否则有解。

输出方案时可以通过变量在图中的拓扑序确定该变量的取值。如果变量  $x$  的拓扑序在  $\neg x$  之后，那么取  $x$  值为真。应用到 Tarjan 算法的缩点，即  $x$  所在 SCC 编号在  $\neg x$  之前时，取  $x$  为真。因为 Tarjan 算法求强连通分量时使用了栈，如果跑完 Tarjan 缩点之后呈现出的拓扑序更大，在 Tarjan 会更晚被遍历到，就会更早地被弹出栈而缩点，分量编号会更小，所以 Tarjan 求得的 SCC 编号相当于 **反拓扑序**。

算法会把整张图遍历一遍，由于这张图  $n$  和  $m$  同阶，计算答案时复杂度为  $O(n)$ ，因此总复杂度为  $O(n)$ 。

代码实现

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <stack>
4  using namespace std;
5  const int N = 2e6 + 2;
6  int n, m, dfn[N], low[N], t, tot, head[N], a[N];
7  bool vis[N];
8  stack<int> s;
9
10 struct node {
11     int to, Next;
12 } e[N];
13
14 void adde(int u, int v) {
15     e[++tot].to = v;
16     e[tot].Next = head[u];
17     head[u] = tot;
18 }
19
20 void Tarjan(int u) {
21     dfn[u] = low[u] = ++t;
22     s.push(u);
23     vis[u] = 1;
24     for (int i = head[u]; i; i = e[i].Next) {
25         int v = e[i].to;
26         if (!dfn[v]) {
27             Tarjan(v);
28             low[u] = min(low[u], low[v]);
29         } else if (vis[v])
30             low[u] = min(low[u], dfn[v]);
31     }
32     if (dfn[u] == low[u]) {
33         int cur;
34         ++tot;
35         do {
36             cur = s.top();
37             s.pop();
38             vis[cur] = 0;
39             a[cur] = tot;
40         } while (cur != u);
41     }
42 }
43
44 int main() {
45     scanf("%d%d", &n, &m);
46     for (int i = 1, I, J, A, B; i <= m; i++) {
47         scanf("%d%d%d%d", &I, &A, &J, &B);
48         adde(A ? I + n : I, B ? J : J + n);
49         adde(B ? J + n : J, A ? I : I + n);

```

```

50     }
51     tot = 0;
52     for (int i = 1; i <= (n << 1); i++)
53         if (!dfn[i]) Tarjan(i);
54     for (int i = 1; i <= n; i++) {
55         if (a[i] == a[i + n]) {
56             printf("IMPOSSIBLE");
57             return 0;
58         }
59     }
60     puts("POSSIBLE");
61     for (int i = 1; i <= n; i++)
62         printf("%c%c", a[i] < a[i + n] ? '1' : '0', " \n"[i == n]);
63     return 0;
64 }

```

## 例题

### 例题 1

#### HDU3062 Party

有  $n$  对夫妻被邀请参加一个聚会，因为场地的问题，每对夫妻中只有一人可以列席。在  $2n$  个人中，某些人之间有着很大的矛盾（当然夫妻之间是没有矛盾的），有矛盾的两个人是不会同时出现在聚会上的。有没有可能会有  $n$  个人同时列席？

按照上面的分析，如果  $a_1$  中的丈夫和  $a_2$  中的妻子不合，我们就把  $a_1$  中的丈夫和  $a_2$  中的丈夫连边，把  $a_2$  中的妻子和  $a_1$  中的妻子连边，然后缩点染色判断即可。

## 参考代码

```
1  #include <algorithm>
2  #include <cstring>
3  #include <iostream>
4  constexpr int MAXN = 2018;
5  constexpr int MAXM = 4000400;
6  using namespace std;
7  int Index, instack[MAXN], DFN[MAXN], LOW[MAXN];
8  int tot, color[MAXN];
9  int numedge, head[MAXN];
10
11 struct Edge {
12     int nxt, to;
13 } edge[MAXM];
14
15 int sta[MAXN], top;
16 int n, m;
17
18 void add(int x, int y) {
19     edge[++numedge].to = y;
20     edge[numedge].nxt = head[x];
21     head[x] = numedge;
22 }
23
24 void tarjan(int x) { // 缩点看不懂请移步强连通分量上面有一个链接可以
25     点。
26     sta[++top] = x;
27     instack[x] = 1;
28     DFN[x] = LOW[x] = ++Index;
29     for (int i = head[x]; i; i = edge[i].nxt) {
30         int v = edge[i].to;
31         if (!DFN[v]) {
32             tarjan(v);
33             LOW[x] = min(LOW[x], LOW[v]);
34         } else if (instack[v])
35             LOW[x] = min(LOW[x], DFN[v]);
36     }
37     if (DFN[x] == LOW[x]) {
38         tot++;
39         do {
40             color[sta[top]] = tot; // 染色
41             instack[sta[top]] = 0;
42         } while (sta[top--] != x);
43     }
44 }
45
46 bool solve() {
47     for (int i = 0; i < 2 * n; i++)
48         if (!DFN[i]) tarjan(i);
49     for (int i = 0; i < 2 * n; i += 2)
```

```

50     if (color[i] == color[i + 1]) return false;
51     return true;
52 }
53
54 void init() {
55     top = 0;
56     tot = 0;
57     Index = 0;
58     numedge = 0;
59     memset(sta, 0, sizeof(sta));
60     memset(DFN, 0, sizeof(DFN));
61     memset(instack, 0, sizeof(instack));
62     memset(Low, 0, sizeof(Low));
63     memset(color, 0, sizeof(color));
64     memset(head, 0, sizeof(head));
65 }
66
67 int main() {
68     cin.tie(nullptr)->sync_with_stdio(false);
69     while (cin >> n >> m) {
70         init();
71         for (int i = 1; i <= m; i++) {
72             int a1, a2, c1, c2;
73             cin >> a1 >> a2 >> c1 >> c2;
74             add(2 * a1 + c1, 2 * a2 + 1 - c2);
75             // 对于第 i 对夫妇，我们用 2i+1 表示丈夫，2i 表示妻子。
76             add(2 * a2 + c2, 2 * a1 + 1 - c1);
77         }
78         if (solve())
79             cout << "YES\n";
80         else
81             cout << "NO\n";
82     }
83     return 0;
84 }

```

## 例题 2



### 2018-2019 ACM-ICPC Asia Seoul Regional K TV Show Game



有  $k$  盏灯，每盏灯是红色或者蓝色，但是初始的时候不知道灯的颜色。有  $n$  个人，每个人选择三盏灯并猜灯的颜色。一个人猜对两盏灯或以上的颜色就可以获得奖品。判断是否存在一个灯的着色方案使得每个人都能领奖，若有则输出一种灯的着色方案。

根据 伍昱 - 《由对称性解 2-sat 问题》，我们可以得出：如果要输出 2-SAT 问题的一个可行解，只需要在 tarjan 缩点后所得的 DAG 上自底向上地进行选择和删除。

具体实现的时候，可以通过构造 DAG 的反图后在反图上进行拓扑排序实现；也可以根据 tarjan 缩点后，所属连通块编号越小，节点越靠近叶子节点这一性质，优先对所属连通块编号小的节点进行选择。

下面给出第二种实现方法的代码。



## 参考代码

```
1  #include <algorithm>
2  #include <iostream>
3  using namespace std;
4  constexpr int MAXN = 1e4 + 5;
5  constexpr int MAXK = 5005;
6
7  int n, k;
8  int id[MAXN][5];
9  char s[MAXN][5][5], ans[MAXK];
10 bool vis[MAXN];
11
12 struct Edge {
13     int v, nxt;
14 } e[MAXN * 100];
15
16 int head[MAXN], tot = 1;
17
18 void addedge(int u, int v) {
19     e[tot].v = v;
20     e[tot].nxt = head[u];
21     head[u] = tot++;
22 }
23
24 int dfn[MAXN], low[MAXN], color[MAXN], stk[MAXN], ins[MAXN], top,
25 dfs_clock, c;
26
27 void tarjan(int x) { // tarjan算法求强联通
28     stk[++top] = x;
29     ins[x] = 1;
30     dfn[x] = low[x] = ++dfs_clock;
31     for (int i = head[x]; i; i = e[i].nxt) {
32         int v = e[i].v;
33         if (!dfn[v]) {
34             tarjan(v);
35             low[x] = min(low[x], low[v]);
36         } else if (ins[v])
37             low[x] = min(low[x], dfn[v]);
38     }
39     if (dfn[x] == low[x]) {
40         c++;
41         do {
42             color[stk[top]] = c;
43             ins[stk[top]] = 0;
44         } while (stk[top--] != x);
45     }
46 }
47
48 int main() {
49     cin.tie(nullptr)->sync_with_stdio(false);
```



```

50     cin >> k >> n;
51     for (int i = 1; i <= n; i++) {
52         for (int j = 1; j <= 3; j++) cin >> id[i][j] >> s[i][j];
53
54         for (int j = 1; j <= 3; j++) {
55             for (int k = 1; k <= 3; k++) {
56                 if (j == k) continue;
57                 int u = 2 * id[i][j] - (s[i][j][0] == 'B');
58                 int v = 2 * id[i][k] - (s[i][k][0] == 'R');
59                 addedge(u, v);
60             }
61         }
62     }
63
64     for (int i = 1; i <= 2 * k; i++)
65         if (!dfn[i]) tarjan(i);
66
67     for (int i = 1; i <= 2 * k; i += 2)
68         if (color[i] == color[i + 1]) {
69             cout << "-1\n";
70             return 0;
71         }
72
73     for (int i = 1; i <= 2 * k; i += 2) {
74         int f1 = color[i], f2 = color[i + 1];
75         if (vis[f1]) {
76             ans[(i + 1) >> 1] = 'R';
77             continue;
78         }
79         if (vis[f2]) {
80             ans[(i + 1) >> 1] = 'B';
81             continue;
82         }
83         if (f1 < f2) {
84             vis[f1] = true;
85             ans[(i + 1) >> 1] = 'R';
86         } else {
87             vis[f2] = true;
88             ans[(i + 1) >> 1] = 'B';
89         }
90     }
91     ans[k + 1] = 0;
92     cout << (ans + 1) << '\n';
93     return 0;
}


```


## 习题

- 洛谷 P5782 和平委员会

- [POJ3683 Priest John's Busiest Day](#)

 本页面最近更新：2025/8/28 21:35:18, [更新历史](#)

 发现错误？想一起完善？ [在 GitHub 上编辑此页！](#)

 本页面贡献者： [AndrewWayne](#), [Ir1d](#), [Backlight](#), [Tiphereth-A](#), [chu-yuehan](#), [Early0v0](#), [Enter-tainer](#), [frank-xjh](#), [H-J-Granger](#), [akakw1](#), [algosheep](#), [Anguei](#), [c-forrest](#), [felixesintot](#), [guodong2005](#), [HeRaNO](#), [jpy-cpp](#), [kenlig](#), [Konano](#), [ksyx](#), [ouuan](#), [sshwy](#)

© 本页面的全部内容 [在 CC BY-SA 4.0 和 SATA 协议之条款下](#) 提供，附加条款亦可能应用