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 个集合，每个集合有两个元素，已知若干个 $< a, b >$，表示 a 与 b 矛盾（其中 a 与 b 属于不同的集合）。然后从每个集合选择一个元素，判断能否一共选 n 个两两不矛盾的元素。显然可能有多种选择方案，一般题中只需要求出一种即可。

现实意义

比如邀请人来吃喜酒，夫妻二人必须去一个，然而某些人之间有矛盾（比如 A 先生与 B 女士有矛盾，C 女士不想和 D 先生在一起），那么我们要确定能否避免来人之间没有矛盾，有时需要方案。这是一类生活中常见的问题。

使用布尔方程表示上述问题。设 a 表示 A 先生去参加，那么 B 女士就不能参加（$\neg a$）；b 表示 C 女士参加，那么 $\neg b$ 也一定成立（D 先生不参加）。总结一下，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，则有：$\neg a \Rightarrow b \wedge \neg b \Rightarrow a$（$a$ 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。

常用解决方法

Tarjan SCC 缩点 [../scc/]

算法考究在建图这点，我们举个例子来讲：

假设有 $a1, a2$ 和 $b1, b2$ 两对，已知 $a1$ 和 $b2$ 间有矛盾，于是为了方案自洽，由于两者中必须选一个，所以我们就要拉两条条有向边 $(a1, b1)$ 和 $(b2, a2)$ 表示选了 $a1$ 则必须选 $b1$ ，选了 $b2$ 则必须选 $a2$ 才能够自洽。

然后通过这样子建边我们跑一遍 Tarjan SCC 判断是否有一个集合中的两个元素在同一个 SCC 中，若有则输出不可能，否则输出方案。构造方案只需要把几个不矛盾的 SCC 拼起来就好了。

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 $\neg x$ 的拓扑序在 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，所以 Tarjan 求得的 SCC 编号相当于反拓扑序。

显然地，时间复杂度为 $O(n + m)$ 。

暴搜

就是沿着图上一条路径，如果一个点被选择了，那么这条路径以后的点都将被选择，那么，出现不可行的情况就是，存在一个集合中两者都被选择了。

那么，我们只需要枚举一下就可以了，数据不大，答案总是可以出来的。

爆搜模板

```
// 来源：刘汝佳白书第 323 页
struct Twosat {
  int n;
  vector<int> g[maxn * 2];
  bool mark[maxn * 2];
  int s[maxn * 2], c;
  bool dfs(int x) {
    if (mark[x ^ 1]) return false;
    if (mark[x]) return true;
    mark[x] = true;
    s[c++] = x;
    for (int i = 0; i < (int)g[x].size(); i++)
      if (!dfs(g[x][i])) return false;
    return true;
  }
  void init(int n) {
    this->n = n;
    for (int i = 0; i < n * 2; i++) g[i].clear();
```

```
    memset(mark, 0, sizeof(mark));
  }
  void add_clause(int x, int y) {  // 这个函数随题意变化
    g[x].push_back(y ^ 1);         // 选了 x 就必须选 y^1
    g[y].push_back(x ^ 1);
  }
  bool solve() {
    for (int i = 0; i < n * 2; i += 2)
      if (!mark[i] && !mark[i + 1]) {
        c = 0;
        if (!dfs(i)) {
          while (c > 0) mark[s[--c]] = false;
          if (!dfs(i + 1)) return false;
        }
      }
    return true;
  }
};
```

例题

HDU3062 Party [http://acm.hdu.edu.cn/showproblem.php?pid=3062]

> 题面：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 1 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 2 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？

这是一道多校题，裸的 2-SAT 判断是否有方案，按照我们上面的分析，如果 $a1$ 中的丈夫和 $a2$ 中的妻子不合，我们就把 $a1$ 中的丈夫和 $a2$ 中的丈夫连边，把 $a2$ 中的妻子和 $a1$ 中的妻子连边，然后缩点染色判断即可。

参考代码

```
// 作者：小黑 AWM
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
#define maxn 2018
```

```
#define maxm 4000400
using namespace std;
int Index, instack[maxn], DFN[maxn], LOW[maxn];
int tot, color[maxn];
int numedge, head[maxn];
struct Edge {
  int nxt, to;
} edge[maxm];
int sta[maxn], top;
int n, m;
void add(int x, int y) {
  edge[++numedge].to = y;
  edge[numedge].nxt = head[x];
  head[x] = numedge;
}
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点。
  sta[++top] = x;
  instack[x] = 1;
  DFN[x] = LOW[x] = ++Index;
  for (int i = head[x]; i; i = edge[i].nxt) {
    int v = edge[i].to;
    if (!DFN[v]) {
      tarjan(v);
      LOW[x] = min(LOW[x], LOW[v]);
    } else if (instack[v])
      LOW[x] = min(LOW[x], DFN[v]);
  }
  if (DFN[x] == LOW[x]) {
    tot++;
    do {
      color[sta[top]] = tot;  // 染色
      instack[sta[top]] = 0;
    } while (sta[top--] != x);
  }
}
bool solve() {
  for (int i = 0; i < 2 * n; i++)
    if (!DFN[i]) tarjan(i);
  for (int i = 0; i < 2 * n; i += 2)
    if (color[i] == color[i + 1]) return 0;
  return 1;
}
void init() {
  top = 0;
  tot = 0;
  Index = 0;
  numedge = 0;
  memset(sta, 0, sizeof(sta));
```

```
  memset(DFN, 0, sizeof(DFN));
  memset(instack, 0, sizeof(instack));
  memset(LOW, 0, sizeof(LOW));
  memset(color, 0, sizeof(color));
  memset(head, 0, sizeof(head));
}
int main() {
  while (~scanf("%d%d", &n, &m)) {
    init();
    for (int i = 1; i <= m; i++) {
      int a1, a2, c1, c2;
      scanf("%d%d%d%d", &a1, &a2, &c1, &c2);  // 自己做的时候别用 cin 会被卡
      add(2 * a1 + c1, 2 * a2 + 1 - c2);
      // 对于第 i 对夫妇，我们用 2i+1 表示丈夫，2i 表示妻子。
      add(2 * a2 + c2, 2 * a1 + 1 - c1);
    }
    if (solve())
      printf("YES\n");
    else
      printf("NO\n");
  }
  return 0;
}
```

2018-2019 ACM-ICPC Asia Seoul Regional K TV Show Game [http://codeforces.com/gym/101987]

> 题面：有 $k(k>3)$ 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 3 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。

这道题在判断是否有方案的基础上，在有方案时还要输出一个可行解。

根据 伍昱 -《由对称性解 2-sat 问题》[https://wenku.baidu.com/view/31fd7200bed5b9f3f90f1ce2.html]，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

```cpp
// Author: Backl1ght
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e4 + 5;
const int maxk = 5005;

int n, k;
int id[maxn][5];
char s[maxn][5][5], ans[maxk];
bool vis[maxn];

struct Edge {
  int v, nxt;
} e[maxn * 100];
int head[maxn], tot = 1;
void addedge(int u, int v) {
  e[tot].v = v;
  e[tot].nxt = head[u];
  head[u] = tot++;
}

int dfn[maxn], low[maxn], color[maxn], stk[maxn],
ins[maxn], top, dfs_clock, c;
void tarjan(int x) {
  stk[++top] = x;
  ins[x] = 1;
  dfn[x] = low[x] = ++dfs_clock;
  for (int i = head[x]; i; i = e[i].nxt) {
    int v = e[i].v;
    if (!dfn[v]) {
      tarjan(v);
      low[x] = min(low[x], low[v]);
    } else if (ins[v])
      low[x] = min(low[x], dfn[v]);
  }
  if (dfn[x] == low[x]) {
    c++;
    do {
      color[stk[top]] = c;
      ins[stk[top]] = 0;
    } while (stk[top--] != x);
```

```
  }
}

int main() {
  scanf("%d %d", &k, &n);
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= 3; j++) scanf("%d%s", &id[i][j],
s[i][j]);

    for (int j = 1; j <= 3; j++) {
      for (int k = 1; k <= 3; k++) {
        if (j == k) continue;
        int u = 2 * id[i][j] - (s[i][j][0] == 'B');
        int v = 2 * id[i][k] - (s[i][k][0] == 'R');
        addedge(u, v);
      }
    }
  }

  for (int i = 1; i <= 2 * k; i++)
    if (!dfn[i]) tarjan(i);

  for (int i = 1; i <= 2 * k; i += 2)
    if (color[i] == color[i + 1]) {
      puts("-1");
      return 0;
    }

  for (int i = 1; i <= 2 * k; i += 2) {
    int f1 = color[i], f2 = color[i + 1];
    if (vis[f1]) {
      ans[(i + 1) >> 1] = 'R';
      continue;
    }
    if (vis[f2]) {
      ans[(i + 1) >> 1] = 'B';
      continue;
    }
    if (f1 < f2) {
      vis[f1] = 1;
      ans[(i + 1) >> 1] = 'R';
    } else {
      vis[f2] = 1;
      ans[(i + 1) >> 1] = 'B';
    }
  }
  ans[k + 1] = 0;
  printf("%s\n", ans + 1);
```

```
    return 0;
}
```

练习题

HDU1814 和平委员会 [http://acm.hdu.edu.cn/showproblem.php?pid=1814]

POJ3683 牧师忙碌日 [http://poj.org/problem?id=3683]

评论