目录

2-SAT

对于每个析取范式$a \bigvee b$ ，连两条边$(\neg a, b)$和$(\neg b, a)$，然后求强连通分量，进行缩点

如果存在两个对立的变量在一个强连通分量里面，则无解，否则必有解

缩点之后的DAG里面，对于两个对立变量a和b，若a的拓扑序在后边，则a为真

```
bool twoSAT(int n) {
    for(int i = 1; i <= 2*n; i++) if(!dfn[i]) tarjan(i);
    for(int i = 1; i <= n; i++) if(sccno[i] == sccno[i+n]) {
        printf("NO\n"); return ;
    };
    printf("YES\n");
    for(int i = 1; i <= n; i++) {
        if(sccno[i] < sccno[i+n]) printf("true\n");
        else printf("false\n");
    }
}
```

强连通分量

```
void init() {
    tot = dfs_clock = scc_cnt = 0;
    memset(head, 0, sizeof(head)); memset(dfn, 0, sizeof(dfn));
    memset(sccno, 0, sizeof(sccno));
}

void addEdge(int u, int v) {
    to[++tot] = v, nxt[tot] = head[u], head[u] = tot;
}

void tarjan(int u) {
    dfn[u] = low[u] = ++dfs_clock;
    S.push(u);
    for(int i = head[u]; i; i = nxt[i]) {
        int v = to[i];
        if(!dfn[v]) tarjan(v), low[u] = min(low[u], low[v]);
        else if(!sccno[v]) low[u] = min(low[u], dfn[v]);
    }
    if(low[u] == dfn[u]) {
        scc_cnt++;
        for(; ; ) {
            int x = S.top(); S.pop();
            sccno[x] = scc_cnt;
            if(x == u) break ;
        }
    }
}
```

最大流

```
const int N = 4100, M = 200010;
int tot = 1, src, sink;
int to[M], _next[M], cap[M], head[N], pre[N], vis[N], cur[N], used[N];

void addEdge(int u, int v, int c) {
    to[++tot] = v, _next[tot] = head[u], cap[tot] = c, head[u] = tot;
}

void init() {
    tot = 1; memset(head, 0, sizeof(head));
}

bool BFS() {
    memset(vis, false, sizeof(vis));
    queue<int>q; q.push(src);
    vis[src] = true;
    while(!q.empty()) {
        int u = q.front(); q.pop();
        for(int i = head[u]; i; i = _next[i]) {
            int v = to[i];
            if(!cap[i] || vis[v]) continue ;
            vis[v] = true, pre[v] = pre[u]+1;
            q.push(v);
        }
    }
    return vis[sink];
}

int DFS(int u, int c) {
    if(u==sink || c==0) return c;
    int flow = 0, f;
    for(int &i = cur[u]; i; i = _next[i]) {
        int v = to[i];
        if(pre[v]==pre[u]+1 && (f=DFS(v, min(c, cap[i])))>0) {
            flow += f, c -= f, cap[i] -= f, cap[i^1] += f;
        }
    }
    return flow;
}

int maxFlow() {
    int flow = 0;
    while(BFS()) {
        memcpy(cur, head, sizeof(cur));
        flow += DFS(src, INT_INF);
    }
    return flow;
}
```