Stratosphere's XCPC Templates

南京大学

平流层 Stratosphere

October 10, 2024

Contents

0 Header 与约定

1 图论

1.1 欧拉回路

```cpp
namespace Euler {
    bool directed;
    vector<pii> G[maxn];
    vector<int> ans;
    int vis[maxm];
    int dfs(int x) {
        vector<int> t;
        while (G[x].size()) {
            auto [to, id] = G[x].back();
            G[x].pop_back();
            if (!vis[abs(id)]) {
                vis[abs(id)] = 1, t.push_back(dfs(to)), ans.push_back(id);
            }
        }
        for (int i = 1; i < t.size(); i++) {
            if (t[i] != x) ans.clear();
        }
        return t.size() ? t[0] : x;
    }
    int n, m;
    pii e[maxm];
    int deg[maxn], vv[maxn];
    void clr() {
        for (int i = 1; i <= n; i++) G[i].clear(), deg[i] = vv[i] = 0;
        for (int i = 1; i <= m; i++) vis[i] = 0;
        ans.clear();
        n = m = 0;
    }
    void addedge(int x, int y) {
        chkmax(n, x), chkmax(n, y);
        e[++m] = {x, y};
        if (directed) {
            G[x].push_back({y, m});
            ++deg[x], --deg[y], vv[x] = vv[y] = 1;
        } else {
            G[x].push_back({y, m});
            G[y].push_back({x, -m});
            ++deg[x], ++deg[y], vv[x] = vv[y] = 1;
        }
    }
    using vi = vector<int>;
    pair<vi, vi> work() {
        if (!m) return clr(), pair<vi, vi>{{1}, {}};
        int S = 1;
        for (int i = 1; i <= n; i++)
            if (vv[i]) S = i;
        for (int i = 1; i <= n; i++)
            if (deg[i] > 0 && deg[i] % 2 == 1) S = i;
        dfs(S);
        if ((int)ans.size() != m) return clr(), pair<vi, vi>();
        reverse(ans.begin(), ans.end());
        vi ver, edge = ans;
        if (directed) {
            ver = {e[ans[0]].fir};
            for (auto t : ans) ver.push_back(e[t].sec);
        } else {
            ver = {ans[0] > 0 ? e[ans[0]].fir : e[-ans[0]].sec};
            for (auto t : ans) ver.push_back(t > 0 ? e[t].sec : e[-t].fir);
        }
        clr();
        return {ver, edge};
    }
}  // namespace Euler
```

1.2 Tarjan-SCC

```
void tarjan(int u) {
|   dfn[u] = low[u] = ++tim;
|   in[u] = 1;
|   st[++top] = u;
|   for (int v : G[u]) {
|   |   if (!dfn[v])
|   |   |   tarjan(v), ckmin(low[u], low[v]);
|   |   else if (in[v])
|   |   |   ckmin(low[u], dfn[v]);
|   }
|   if (dfn[u] == low[u]) {
|   |   ++totc;
|   |   int x;
|   |   do { x = st[top--], in[x] = 0, bel[x] = totc; } while (x != u);
|   }
}
```

1.3 点双

```
int T;  // assign = n
void tarjan(int u, int fa) {
|   dfn[u] = low[u] = ++tim;
|   stk[++top] = u;
|   for (int v : G[u]) {
|   |   if (v == fa) continue;
|   |   if (!dfn[v])
|   |   |   dfs(v, u), ckmin(low[u], low[v]);
|   |   else
|   |   |   ckmin(low[u], dfn[v]);
|   }
|   if (fa && low[u] >= dfn[fa]) {
|   |   int y;
|   |   ++T;
|   |   do {
|   |   |   y = stk[top--];
|   |   |   G2[T].push_back(y), G2[y].push_back(T);
|   |   } while (y != u);
|   |   G2[T].push_back(fa), G2[fa].push_back(T);
|   }
}
```

1.4 边双

```
void tarjan(int u, int f) {
|   dfn[u] = low[u] = ++tim;
|   for (int v : G[u]) {
|   |   int v = e[i].to;
|   |   if (v == f) continue;
|   |   if (!dfn[v]) {
|   |   |   tarjan(v, u);
|   |   |   ckmin(low[u], low[v]);
|   |   |   if (low[v] > dfn[u]) vis[i] = vis[i ^ 1] = 1;  // cut edge
|   |   } else
|   |   |   ckmin(low[u], dfn[v]);
|   }
}
```

1.5 2-SAT

构造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。
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。

```cpp
for (int i = 1; i <= n; i++)
    if (bel[i << 1] == bel[i << 1 | 1]) return puts("IMPOSSIBLE"), 0;
puts("POSSIBLE");
for (int i = 1; i <= n; i++) printf("%d ", bel[i << 1] > bel[i << 1 | 1]);
```

1.6 最大流 (Dinic)

1.7 最小费用最大流

1.8 三四元环计数

```cpp
static int id[maxn], rnk[maxn];
for (int i = 1; i <= n; i++) id[i] = i;
sort(id + 1, id + n + 1, [](int x, int y) {
    return pii{deg[x], x} < pii{deg[y], y};
});
for (int i = 1; i <= n; i++) rnk[id[i]] = i;
for (int i = 1; i <= n; i++)
    for (int v : G[i])
        if (rnk[v] > rnk[i]) G2[i].push_back(v);
int ans3 = 0;  // 3-cycle
for (int i = 1; i <= n; i++) {
    static int vis[maxn];
    for (int v : G2[i]) vis[v] = 1;
    for (int v1 : G2[i])
        for (int v2 : G2[v1])
            if (vis[v2]) ++ans3;  // (i,v1,v2)
    for (int v : G2[i]) vis[v] = 0;
}
ll ans4 = 0;  // 4-cycle
for (int i = 1; i <= n; i++) {
    static int vis[maxn];
    for (int v1 : G[i])
        for (int v2 : G2[v1])
            if (rnk[v2] > rnk[i]) ans4 += vis[v2], vis[v2]++;
    for (int v1 : G[i])
        for (int v2 : G2[v1]) vis[v2] = 0;
}
```

1.9 支配树

DAG 支配树

```cpp
namespace DomTree_DAG {
    int idom[maxn];
    vector<int> G[maxn], ANS[maxn];  // ANS: final tree
    int deg[maxn];
    int fa[maxn][25], dep[maxn];
    int lca(int x, int y) {
        if (dep[x] < dep[y]) swap(x, y);
        for (int i = 20; i >= 0; i--)
            if (fa[x][i] && dep[fa[x][i]] >= dep[y]) x = fa[x][i];
        if (x == y) return x;
        for (int i = 20; i >= 0; i--)
            if (fa[x][i] != fa[y][i]) x = fa[x][i], y = fa[y][i];
        return fa[x][0];
    }
    void work() {
        queue<int> q;
        q.push(1);
        while (!q.empty()) {
            int x = q.front();
            q.pop();
            ANS[idom[x]].push_back(x);
            fa[x][0] = idom[x];
            dep[x] = dep[idom[x]] + 1;
            for (int i = 1; i <= 20; i++) fa[x][i] = fa[fa[x][i - 1]][i - 1];
            for (int v : G[x]) {
                --deg[v];
                if (!deg[v]) q.push(v);
                if (!idom[v])
                    idom[v] = x;
                else
                    idom[v] = lca(idom[v], x);
            }
        }
    }
}  // namespace DomTree_DAG
namespace DomTree {
    vector<int> V[sz], rV[sz];
    int dfn[sz], id[sz], anc[sz], cnt;
    void dfs(int x) {
        id[dfn[x] = ++cnt] = x;
        for (int v : V[x])
            if (!dfn[v]) {
                BuildTree::V[x].push_back(v);
                BuildTree::deg[v]++;
                anc[v] = x;
                dfs(v);
            }
    }
    int fa[sz], mn[sz];
    int find(int x) {
        if (x == fa[x]) return x;
        int tmp = fa[x];
        fa[x] = find(fa[x]);
        chkmin(mn[x], mn[tmp]);
        return fa[x];
    }
    int semi[sz];
    void work() {
        dfs(1);
        rep(i, 1, n) fa[i] = i, mn[i] = 1e9, semi[i] = i;
        drep(w, n, 2) {
            int x = id[w];
            int cur = 1e9;
            if (w > cnt) continue;
            for (int v : rV[x]) {
                if (!dfn[v]) continue;
                if (dfn[v] < dfn[x])
                    chkmin(cur, dfn[v]);
                else
                    find(v), chkmin(cur, mn[v]);
            }
            semi[x] = id[cur];
            mn[x] = cur;
            fa[x] = anc[x];
            BuildTree::V[semi[x]].push_back(x);
            BuildTree::deg[x]++;
        }
    }
    void link(int x, int y) {
        V[x].push_back(y), rV[y].push_back(x);
    }
}  // namespace DomTree
```

2 数论

Delete This

3 数学

Delete This

4　字符串

Delete This

5 数据结构

Delete This

6 计算几何

Delete This

7 三维计算几何

Delete This

8 杂项

Delete This