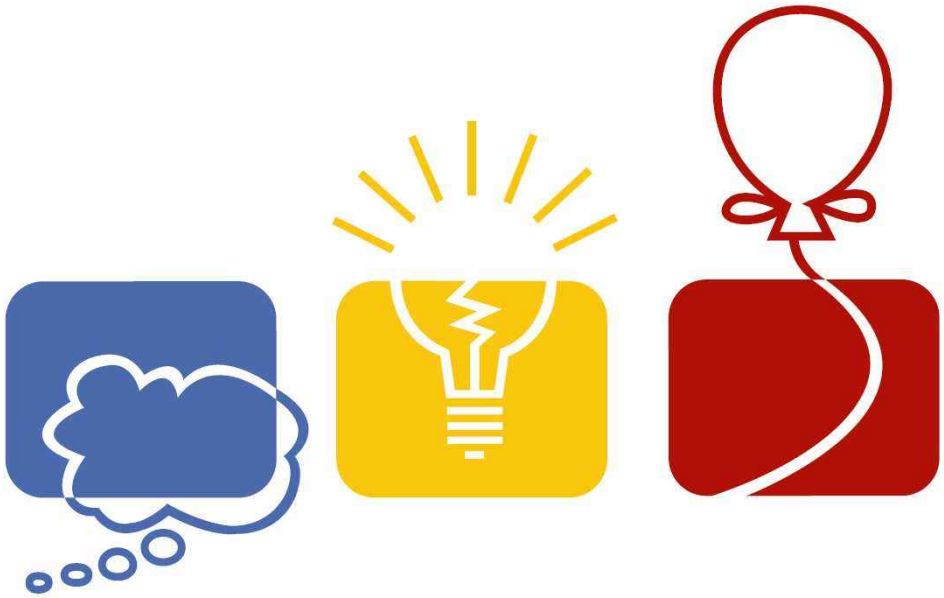

Stratosphere's XCPC Templates

南京大学

平流层 Stratosphere

October 2, 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

```
// etot should be initialized to 1 !!!
void tarjan(int u, int f) {
   dfn[u] = low[u] = ++tim;
   for (int i = head[u]; i; i = e[i].nxt) {
      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。

```
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 三四元环计数

```
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;
}
```

1.7 支配树

DAG 支配树

```
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();
```

```
21          ANS[idom[x]].push_back(x);
22          fa[x][0] = idom[x];
23          dep[x] = dep[idom[x]] + 1;
24          for (int i = 1; i <= 20; i++) fa[x][i] = fa[fa[x][i - 1]][i - 1];
25          for (int v : G[x]) {
26             --deg[v];
27             if (!deg[v]) q.push(v);
28             if (!idom[v])
29                idom[v] = x;
30             else
31                idom[v] = lca(idom[v], x);
32          }
33       }
34    }
35 } // namespace DomTree_DAG
```

2 数论

3 数学

4 字符串

5 数据结构

6 计算几何

7 三维计算几何

8 杂项