Standard Code Library

Andy Liao

November 11, 2021

Contents

1 写在前面

1.1 序言

那一年的 Final，是 SoftBear 谢幕之舞，最终排名 21。当时我看见三位学长坐在工位前挑灯夜战，这个画面我永生难忘。
那一刻我在想，如果我能成为主力队员，我一定要赢下所有。如今金牌就在眼前，我必须考虑这会不会是我此生仅有的机会。
我相信 FZU 能有过去的强校地位，SoftBear 功不可没。重铸 FZU 荣光，吾辈义不容辞。

Andy Liao, in honor with 48 Group

1.2 代码规范

- 采用 Google 代码规范（的来自 **MirAc1e** 的魔改版）。
- **不要觉得你比开发编译器的那帮老家伙更聪明**。
- 不要再使用已经被废弃的 `register` 和 `inline` 关键字。不要自我感觉良好地做一些只存在于想象中的优化。
- 预处理器指令应当处于文件的开头，且 `#include` 指令应在最上方，其次是 `#define` 指令。这之后应当有一个空行。大多数情况下需要包含的头文件只有 `<bits/stdc++.h>` 。极少数情况下允许使用 `<bits/extc++.h>` 。
- 一般情况下整数的最大值应由连续的几个 `0x3f` 组成。
- 一般情况下数组的大小应在预处理器指令中定义，且由一串相同的阿拉伯数字组成。
- **不要缩行**。
- 尽可能避免使用逗号表达式和奇怪的逻辑运算操作（反例可以在 **MirAc1e** 高中时的代码中找到）。
- 代码中应当包含足够多却又不多余的空格以保证可读性（可参照模板中的代码）。
- 在程序正常结束时 `main` 函数应当显式地返回 `0` 。
- 程序应当输出一系列完整的行。也就是说，输出应当以一个空行结尾。
- 文件末尾**不必**留有一个空行。

2 图论

2.1 图的存储与遍历

2.1.1 邻接矩阵

输入一张有向带权图，按编号从小到大输出由某点出发的所有边。

- 存储稠密图。
- 常与 **Floyd** 算法一同使用。
- 记得初始化。

```cpp
#include <bits/stdc++.h>
#define MAXN 1111

using namespace std;
int n, m, g[MAXN][MAXN];

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            g[i][j] = -1;
        }
    }
    for (int i = m, u, v, w; i; --i) {
        scanf("%d%d%d", &u, &v, &w);
        g[u][v] = w;
    }
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            if (~g[i][j]) {
                printf("%d %d %d\n", i, j, g[i][j]);
            }
        }
    }
    return 0;
}
```

2.1.2 邻接表

输入一张有向带权图，按编号从小到大输出由某点出发的所有边。

- 存储稀疏图。
- 在大多数情况下被使用。
- 记得初始化。
- 使用了 **vector** 代替 **MirAc1e** 使用了很久的基于指针的结构体。
- 使用了 **pair** 并将边权作为第一关键字以便于可能出现的排序。
- 可以使用任意能够遍历 **vector** 的方法来访问由某个点出发的所有边。

```cpp
#include <bits/stdc++.h>
#define MAXN 1111

using namespace std;
int n, m;
vector<pair<int, int> > g[MAXN];

int main() {
    scanf("%d%d", &n, &m);
    for (int i = m, u, v, w; i; --i) {
        scanf("%d%d%d", &u, &v, &w);
        g[u].push_back(make_pair(w, v));
    }
    for (int i = 1; i <= n; ++i) {
        for (auto &e : g[i]) {
            printf("%d %d %d\n", i, e.second, e.first);
        }
    }
    return 0;
}
```

2.2 最短路

2.2.1 Dijkstra

输入一张图，输出给定的两点间最短路径的长度。

- 模板题参考了 YZOJ P1127 。
- 使用邻接表存储图。
- 虽然模板题只要求输出到给定点的最短路径的长度，但是实际上已经算出了**由一点出发到所有点的最短路径长度**。
- 使用了 pbds 的配对堆。注意头文件和命名空间的变化。
- 注意距离数组的初始化。

```cpp
#include <bits/extc++.h>
#define MAXN 111111
#define INF 0x3f

using namespace std;
int n, m, s, t, dis[MAXN];
vector<pair<int, int> > g[MAXN];
__gnu_pbds::priority_queue<pair<int, int>, greater<pair<int, int> >,
↪  __gnu_pbds::pairing_heap_tag> q;
```

```cpp

int main() {
    memset(dis, INF, sizeof(dis));
    scanf("%d%d%d%d", &n, &m, &s, &t);
    for (int i = 1, u, v, w; i <= m; ++i) {
        scanf("%d%d%d", &u, &v, &w);
        g[u].push_back(make_pair(w, v));
    }
    dis[s] = 0;
    q.push(make_pair(0, s));
    while (!q.empty()) {
        int u = q.top().second;
        q.pop();
        for (auto &e : g[u]) {
            int v = e.second, w = e.first;
            if (dis[v] > dis[u] + w) {
                q.push(make_pair(dis[v] = dis[u] + w, v));
            }
        }
    }
    printf("%d\n", dis[t]);
    return 0;
}
```

2.2.2 SPFA

输入一张图，输出给定的两点间最短路径的长度。

- 模板题参考了 YZOJ P1127 。
- 使用邻接表存储图。
- 虽然模板题只要求输出到给定点的最短路径的长度，但是实际上已经算出了**由一点出发到所有点的最短路径长度**。
- 没事干就别用这玩意儿了。会被卡的。
- 听说真正的猛士会使用被称为 SLF 和 LLL 的东西来赌一把，试验一下人类能不能造出把自己卡到指数级复杂度的数据。
- 注意距离数组的初始化。
- 如果要求网格图或边权全部相同的图的最短路，写个 **BFS** 就好了。
- 这玩意儿能用来判断图中是否存在负环：记录下每个结点的入队次数，如果一个结点的入队次数已经超过该图的点数，这张图就存在负环，不存在最短路。
- 差分约束问题常用这个算法来解决：对于每个约束条件 $x_i - x_j \leq c_k$，从结点 j 向结点 i 连一条长度为 c_k 的有向边，则 $x_i = d_i$ 为该差分约束系统的一组解。

```cpp
#include <bits/extc++.h>
#define MAXN 111111
#define INF 0x3f

using namespace std;
int n, m, s, t, dis[MAXN];
bool vis[MAXN];
vector<pair<int, int> > g[MAXN];
queue<int> q;

int main() {
    memset(dis, INF, sizeof(dis));
    scanf("%d%d%d%d", &n, &m, &s, &t);
    for (int i = 1, u, v, w; i <= m; ++i) {
        scanf("%d%d%d", &u, &v, &w);
        g[u].push_back(make_pair(w, v));
    }
    dis[s] = 0;
    q.push(s);
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        vis[u] = 0;
        for (auto &e : g[u]) {
            int v = e.second, w = e.first;
            if (dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                if (!vis[v]) {
                    vis[v] = 1;
                    q.push(v);
                }
            }
        }
    }
    printf("%d\n", dis[t]);
    return 0;
}
```

2.2.3 Floyd

输入一张图，输出所有两点间最短路径的长度。

- 模板题参考了 YZOJ P1318 （福建省夏）。
- 使用邻接矩阵存储图。
- **注意三重循环的顺序**。
- 注意距离数组的初始化。

```cpp
#include <bits/stdc++.h>
#define MAXN 1111

using namespace std;
int n, m, g[MAXN][MAXN];

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            g[i][j] = -1;
        }
        g[i][i] = 0;
    }
    for (int i = m, u, v, w; i; --i) {
        scanf("%d%d%d", &u, &v, &w);
        g[u][v] = w;
        g[v][u] = w;
    }
    for (int k = 1; k <= n; ++k) {
        for (int i = 1; i <= n; ++i) {
            for (int j = 1; j <= n; ++j) {
                g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
            }
        }
    }
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            printf("%d ", g[i][j]);
        }
        putchar('\n');
    }
    return 0;
}
```

2.3 最小生成树

2.3.1 Kruskal

输入一张图，输出它的最小生成树的边权和。

- 模板题参考了 YZOJ P1126 。
- 常用于**稀疏图**。

```cpp
#include <bits/stdc++.h>
#define MAXN 111111
```

```

using namespace std;
int n, m, fa[MAXN];
vector<pair<int, pair<int, int> > > g;

int find(int x) {
        return fa[x] != x ? fa[x] = find(fa[x]) : x;
}

int main() {
        for (scanf("%d%d", &n, &m); m; --m) {
        int u, v, w;
                scanf("%d%d%d", &u, &v, &w);
        g.push_back(make_pair(w, make_pair(u, v)));
    }
        for (int i = 0; i < n; ++i) {
                fa[i] = i;
        }
        sort(g.begin(), g.end());
    int ans = 0;
        for (auto &e : g) {
        int p = find(e.second.first), q = find(e.second.second);
        if (p != q) {
            fa[q] = p;
            ans += e.first;
        }
    }
        printf("%d\n", ans);
        return 0;
}
```

2.3.2 Prim

输入一张图，输出它的最小生成树的边权和。

- 模板题参考了 YZOJ P1126 。
- 常用于**密集图**。

```
#include <bits/stdc++.h>
#define MAXN 1111
#define INF 0x3f3f3f3f

using namespace std;
int n, m, dis[MAXN], g[MAXN][MAXN];
bool vis[MAXN];

```

```
int main() {
    memset(g, INF, sizeof(g));
    memset(dis, INF, sizeof(dis));
    for (scanf("%d%d", &n, &m); m; --m) {
        int u, v, w;
        scanf("%d%d%d", &u, &v, &w);
        g[u][v] = g[v][u] = min(g[u][v], w);
    }
    int ans = 0;
    dis[0] = 0;
    for (int i = 0; i < n; ++i) {
        int d_min = INF, id_min = 0;
        for (int j = 0; j < n; ++j) {
            if (!vis[j] && d_min > dis[j]) {
                d_min = dis[j];
                id_min = j;
            }
        }
        ans += dis[id_min];
        vis[id_min] = 1;
        for (int j = 0; j < n; ++j) {
            if (!vis[j] && dis[j] > g[id_min][j]) {
                dis[j] = g[id_min][j];
            }
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

2.4 最近公共祖先

2.4.1 倍增

输入一棵树，询问某几对结点的最近公共祖先。

- 模板题参考了 YZOJ P1345 。
- 我是真的不知道，这玩意儿不好写的同时运行速度又慢，为什么还有这么多人在推荐把它作为求 LCA 的首选算法?

```
#include <bits/stdc++.h>
#define MAXN 111111
#define LOGN 33

using namespace std;
int n, q, fa[MAXN][LOGN], dep[MAXN];
```

```cpp
vector<int> g[MAXN];

void dfs(int u, int f) {
    dep[u] = dep[f] + 1;
    for (auto &v : g[u]) {
        dfs(v, u);
    }
}

int lca(int a, int b) {
    if (dep[a] > dep[b]) {
        swap(a, b);
    }
    int delta = dep[b] - dep[a];
    for (int j = 0; (1 << j) <= delta; ++j) {
        if (delta & (1 << j)) {
            b = fa[b][j];
        }
    }
    if (a == b) {
        return a;
    }
    int j;
    for (j = 1; (1 << j) <= n; ++j);
    for (--j; ~j; --j) {
        if (fa[a][j] != fa[b][j]) {
            a = fa[a][j];
            b = fa[b][j];
        }
    }
    a = fa[a][0];
    return a;
}

int main() {
    scanf("%d%d", &n, &q);
    for (int i = 1, x, y; i < n; ++i) {
        scanf("%d%d", &x, &y);
        fa[y][0] = x;
        g[x].push_back(y);
    }
    dfs(1, 0);
    for (int j = 1; (1 << j) <= n; ++j) {
        for (int i = 1; i <= n; ++i) {
            fa[i][j] = fa[fa[i][j - 1]][j - 1];
        }
    }
    while (q--) {
```

```
        int a, b;
        scanf("%d%d", &a, &b);
        printf("%d\n", lca(a, b));
    }
    return 0;
}
```

2.4.2 RMQ

输入一棵树，询问某几对结点的最近公共祖先。

- 模板题参考了 YZOJ P1345 。
- 在同样不好写的情况下还难以理解。看看就好。拓展一下思维吧。

```
#include <bits/stdc++.h>
#define MAXN 111111
#define LOGN 33

using namespace std;

int n, q, clk, dfn[MAXN], a[MAXN << 1], dep[MAXN << 1], rmq[MAXN << 1][LOGN];
vector<int> g[MAXN];

void dfs(int u, int d) {
    ++clk;
    dfn[u] = clk;
    a[clk] = u;
    dep[clk] = d;
    for (auto &v : g[u]) {
        dfs(v, d + 1);
        ++clk;
        a[clk] = u;
        dep[clk] = d;
    }
}

int query(int l, int r) {
    int k = 0;
    while ((1 << (k + 1)) <= r - l + 1) {
        ++k;
    }
    return dep[rmq[l][k]] <= dep[rmq[r - (1 << k) + 1][k]] ? rmq[l][k] : rmq[r - (1 << k) + 1][k];
}

```

```cpp
int main() {
    scanf("%d%d", &n, &q);
    for (int i = 1, x, y; i < n; ++i) {
        scanf("%d%d", &x, &y);
        g[x].push_back(y);
    }
    dfs(1, 1);
    for (int i = 1; i < clk; ++i) {
        rmq[i][0] = i;
    }
    for (int j = 1; (1 << j) < clk; ++j) {
        for (int i = 1; i + (1 << j) - 1 < clk; ++i) {
            rmq[i][j] = dep[rmq[i][j - 1]] <= dep[rmq[i + (1 << (j - 1))][j - 1]] ? rmq[i][j - 1] : rmq[i + (1 << (j - 1))][j - 1];
        }
    }
    while (q--) {
        int x, y;
        scanf("%d%d", &x, &y);
        printf("%d\n", a[query(min(x, y), max(x, y))]);
    }
    return 0;
}
```

2.4.3 树链剖分

输入一棵树，询问某几对结点的最近公共祖先。

- 模板题参考了 YZOJ P1345 。
- 这个算法思路如此清晰，运行速度也最快，因此**请把这个算法作为你求 LCA 的首选算法！**

```cpp
#include <bits/stdc++.h>
#define MAXN 111111

using namespace std;
int n, q, dep[MAXN], siz[MAXN], fa[MAXN], son[MAXN], top[MAXN];
vector<int> g[MAXN];

void dfs1(int u) {
    int f = fa[u], *s = &son[u];
    dep[u] = dep[f] + 1;
    siz[u] = 1;
    for (auto &v : g[u]) {
        if (v != f && !fa[v]) {
            fa[v] = u;
            dfs1(v);
```

```
            siz[u] += siz[v];
            if (siz[*s] < siz[v]) {
                *s = v;
            }
        }
    }
}

void dfs2(int u) {
    int f = fa[u];
    top[u] = u == son[f] ? top[f] : u;
    for (auto &v : g[u]) {
        if (fa[v] == u) {
            dfs2(v);
        }
    }
}

int query(int x, int y) {
    int a, b;
    while ((a = top[x]) != (b = top[y])) {
        dep[a] > dep[b] ? x = fa[a] : y = fa[b];
    }
    return dep[x] < dep[y] ? x : y;
}

int main() {
    scanf("%d%d", &n, &q);
    for (int i = 1, u, v; i < n; ++i) {
        scanf("%d%d", &u, &v);
        g[u].push_back(v);
    }
    dfs1(1);
    dfs2(1);
    while (q--) {
        int x, y;
        scanf("%d%d", &x, &y);
        printf("%d\n", query(x, y));
    }
    return 0;
}
```

2.4.4 Tarjan

输入一棵树，询问某几对结点的最近公共祖先。

- 模板题参考了 YZOJ P1345 。
- 仅适用于**询问可以离线**的情况。

- 线性复杂度。是这几种解法中最优秀的。

```cpp
#include <bits/stdc++.h>
#define MAXN 111111
#define MAXQ 111111

using namespace std;
int n, q, fa[MAXN], ans[MAXN];
bool vis[MAXN];
vector<int> g[MAXN];
vector<pair<int, int> > query[MAXQ];

int find(int o) {
    return o == fa[o] ? o : fa[o] = find(fa[o]);
}

void tarjan(int u, int f) {
    for (auto &v : g[u]) {
        if (v == f) {
            continue;
        }
        tarjan(v, u);
        fa[v] = u;
    }
    for (auto &q : query[u]) {
        int v = q.second, i = q.first;
        if (vis[v]) {
            ans[i] = find(v);
        }
    }
    vis[u] = 1;
}

int main() {
    scanf("%d%d", &n, &q);
    fa[n] = n;
    for (int i = 1, x, y; i < n; ++i) {
        fa[i] = i;
        scanf("%d%d", &x, &y);
        g[x].push_back(y);
    }
    for (int i = 1, a, b; i <= q; ++i) {
        scanf("%d%d", &a, &b);
        query[a].push_back(make_pair(i, b));
        query[b].push_back(make_pair(i, a));
    }
    tarjan(1, 0);
```

```
    for (int i = 1; i <= q; ++i) {
        printf("%d\n", ans[i]);
    }
    return 0;
}
```

2.5 欧拉回路

输入一张图，输出这张图上任意一个有效的欧拉回路。

- 模板题参考了 YZOJ P1037 。
- 模板只给出了有向图的情况。其他情况大同小异。
- 模板使用深度优先搜索后回溯的经典解法。还存在一种被称为 **Fleury** 的算法可以解决这个问题。
- 模板使用了当前弧优化。
- 使用前根据题意可能需要判断是否存在欧拉回路。
- 有向图欧拉回路存在的条件：图联通，且所有结点满足入度等于出度。
- 无向图欧拉回路存在的条件：图联通，且所有结点满足度数为偶数。
- 这个问题有一个变种：求欧拉道路。解法应该完全一致。
- 有向图欧拉回路存在的条件：图联通，且有且仅有一个结点的入度比出度大 1，有且仅有一个结点的出度比入度大 1，其他结点的入度等于出度
- 无向图欧拉回路存在的条件：图联通，且有且仅有两个结点的度数为奇数。

```
#include <bits/stdc++.h>
#define MAXN 1111

using namespace std;
int n, m, ans[MAXN];
queue<int> g[MAXN];

void dfs(int u) {
    while (!g[u].empty()) {
        int v = g[u].front();
        g[u].pop();
        dfs(v);
    }
    ans[++*ans] = u;
}

```

```
int main() {
    for (scanf("%d%d", &n, &m); m; --m) {
        int u, v;
        scanf("%d%d", &u, &v);
        g[u].push(v);
    }
    dfs(1);
    for (int i = *ans; i; --i) {
        printf("%d ", ans[i]);
    }
    putchar('\n');
    return 0;
}
```

2.6 拓扑排序

输入一张图，输出任意一种合法的拓扑排序序列。

- 模板题参考了 YZOJ P1593（福建省夏）。
- 对模板题做了修改，这样它就是完完全全的模板题了。
- 合法的方案可能有多种。模板程序会输出确定但无法预测的一种。
- 如果算法结束时存在结点未被得到拓扑排序，则这张图**存在环**。

```
#include <bits/stdc++.h>
#define MAXN 1111

using namespace std;
int n, m, ans[MAXN];
queue<int> g[MAXN];

void dfs(int u) {
    while (!g[u].empty()) {
        int v = g[u].front();
        g[u].pop();
        dfs(v);
    }
    ans[++*ans] = u;
}

int main() {
    for (scanf("%d%d", &n, &m); m; --m) {
        int u, v;
        scanf("%d%d", &u, &v);
        g[u].push(v);
```

```
    }
    dfs(1);
    for (int i = *ans; i; --i) {
        printf("%d ", ans[i]);
    }
    putchar('\n');
    return 0;
}
```

2.7 连通性

2.7.1 割点和桥

输入一张无向图，求这张图的割点和桥。

- 模板题来自 YZOJ P1292 。
- **若有多个连通分量，需要对每个连通分量都调用一次算法。**

```
#include <bits/stdc++.h>
#define MAXN 155

using namespace std;
int n, m, fa[MAXN];
vector<int> g[MAXN];
bool bridge[MAXN], cut[MAXN];
int dfs_clock, dfn[MAXN], low[MAXN];

void tarjan(int u, int fa) {
    dfn[u] = low[u] = ++dfs_clock;
    int siz = 0;
    for (auto &v : g[u]) {
        if (dfn[v]) {
            tarjan(v, u);
            ++siz;
            low[u] = min(low[u], low[v]);
            if (dfn[u] <= low[v] && fa) {
                cut[u] = 1;
            }
            if (dfn[u] < low[v]) {
                bridge[v] = 1;
            }
        }
        else if (v != fa) {
            low[u] = min(low[u], dfn[v]);
        }
    }
    if (!fa && siz > 1) {
```

```
            cut[u] = 1;
        }
    }

    int main() {
        for (cin >> n >> m; m; --m) {
            int x, y;
            cin >> x >> y;
            g[x].push_back(y);
            g[y].push_back(x);
        }
        for (int i = 1; i <= n; ++i) {
            if (!dfn[i]) {
                tarjan(i, 0);
            }
        }
        return 0;
    }
```

2.7.2 边双连通分量

输入一张无向图，求这张图的边双连通分量。

- 没有模板题。
- 不经过桥就能互相到达的两个点在同一个**边双连通分量**内。
- **若有多个连通分量，需要对每个连通分量都调用一次算法**。

```
#include <bits/stdc++.h>
#define MAXN 155

using namespace std;
int n, m;
bool vis[MAXN];
vector<int> g[MAXN];
int bcc, bel[MAXN];
int dfs_clock, top, dfn[MAXN], low[MAXN], stk[MAXN];

void tarjan(int u, int fa) {
    vis[u] = 1;
    stk[++top] = u;
    dfn[u] = low[u] = ++dfs_clock;
    for (auto &v : g[u]) {
        if (!vis[v]) {
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
```

```
        }
        else if (v != fa) {
            low[u] = min(low[u], dfn[v]);
        }
    }
    if (low[u] == dfn[u]) {
        ++bcc;
        while (1) {
            int x = stk[top--];
            bel[x] = bcc;
            if (u == x) {
                break;
            }
        }
    }
}

int main() {
    for (cin >> n >> m; m; --m) {
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    for (int i = 1; i <= n; ++i) {
        if (!dfn[i]) {
            tarjan(i, 0);
        }
    }
    return 0;
}
```

2.7.3 点双连通分量

输入一张无向图，求这张图的点双连通分量。

- 没有模板题。
- 不经过割点就能互相到达的两个点在同一个**点双连通分量**内。
- **若有多个连通分量，需要对每个连通分量都调用一次算法**。

```
#include <bits/stdc++.h>
#define MAXN 155

using namespace std;
int n, m;
```

```cpp
vector<int> g[MAXN];
bool cut[MAXN];
int bcc, bel[MAXN];
int dfs_clock, top, dfn[MAXN], low[MAXN], stk[MAXN];

void tarjan(int u, int fa) {
    stk[++top] = u;
    dfn[u] = low[u] = ++dfs_clock;
    int siz = 0;
    for (auto &v : g[u]) {
        if (!dfn[v]) {
            ++siz;
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] >= dfn[u]) {
                cut[u] = 1;
                bel[u] = ++bcc;
                while (1) {
                    int x = stk[top--];
                    bel[x] = bcc;
                    if (v == x) {
                        break;
                    }
                }
            }
        }
        else if (v != fa) {
            low[u] = min(low[u], dfn[v]);
        }
    }
    if (!fa && siz == 1) {
        cut[u] = 0;
    }
}

int main() {
    for (cin >> n >> m; m; --m) {
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    for (int i = 1; i <= n; ++i) {
        if (!dfn[i]) {
            tarjan(i, 0);
        }
    }
    return 0;
```

```
}
```

2.7.4 强连通分量

输入一张有向图，求这张图的强连通分量。

- 没有模板题。
- **若有多个连通分量，需要对每个连通分量都调用一次算法。**

```cpp
#include <bits/stdc++.h>
#define MAXN 155

using namespace std;
int n, m;
vector<int> g[MAXN];
bool cut[MAXN];
int scc, bel[MAXN];
int dfs_clock, top, dfn[MAXN], low[MAXN], stk[MAXN];

void tarjan(int u) {
    stk[++top] = u;
    dfn[u] = low[u] = ++dfs_clock;
    for (auto &v : g[u]) {
        if (!dfn[v]) {
            tarjan(v);
            low[u] = min(low[u], low[v]);
        }
        else if (!bel[v]) {
            low[u] = min(low[u], dfn[v]);
        }
    }
    if (low[u] == dfn[u]) {
        ++scc;
        while (1) {
            int x = stk[top--];
            bel[x] = scc;
            if (u == x) {
                break;
            }
        }
    }
}

int main() {
    for (cin >> n >> m; m; --m) {
        int x, y;
```

```
        cin >> x >> y;
        g[x].push_back(y);
    }
    for (int i = 1; i <= n; ++i) {
        if (!dfn[i]) {
            tarjan(i);
        }
    }
    return 0;
}
```

2.8 树链剖分

2.8.1 软件包管理器

来自 NOI 2015 的一道题。

- ~~这个算法也经常被视为数据结构的一部分。~~
- 常用于处理节点到根路径上的问题。
- 因重链上的点深度优先序号连续而具有一些很好的性质。

```
#include <bits/stdc++.h>
#define MAXN 111111
#define MAXL 11

using namespace std;
int n, q, dfs_clock, pts, root;
int fa[MAXN], dep[MAXN], siz[MAXN], son[MAXN];
int top[MAXN], dfn[MAXN], id[MAXN];
int l[MAXN << 2], r[MAXN << 2], lson[MAXN << 2], rson[MAXN << 2];
long long val[MAXN << 2], tag[MAXN << 2];
char cmd[MAXL];
vector<int> g[MAXN];

void dfs1(int u, int f, int d) {
    fa[u] = f;
    dep[u] = d;
    siz[u] = 1;
    for (auto &v : g[u]) {
        if (v == f) {
            continue;
        }
        dfs1(v, u, d + 1);
        siz[u] += siz[v];
        if (siz[son[u]] < siz[v] || !son[u]) {
            son[u] = v;
```

```
        }
    }
}

void dfs2(int u, int f, int t) {
    top[u] = t;
    dfn[u] = ++dfs_clock;
    id[dfs_clock] = u;
    if (!son[u]) {
        return;
    }
    dfs2(son[u], u, t);
    for (auto &v : g[u]) {
        if (v == f || v == son[u]) {
            continue;
        }
        dfs2(v, u, v);
    }
}

void pushup(int o) {
    int ls = lson[o], rs = rson[o];
    val[o] = val[ls] + val[rs];
    l[o] = l[ls];
    r[o] = r[rs];
}

void build(int o, int le, int ri) {
    if (le == ri) {
        lson[o] = rson[o] = tag[o] = -1;
        l[o] = r[o] = le;
        return;
    }
    int mid = (le + ri) >> 1;
    lson[o] = pts++;
    rson[o] = pts++;
    build(lson[o], le, mid);
    build(rson[o], mid + 1, ri);
    pushup(o);
}

void pushdown(int o) {
    int ls = lson[o], rs = rson[o];
    val[ls] = tag[o] * (r[ls] - l[ls] + 1);
    val[rs] = tag[o] * (r[rs] - l[rs] + 1);
    tag[ls] = tag[o];
    tag[rs] = tag[o];
    tag[o] = -1;
```

```
}

void update(int o, int le, int ri, long long delta) {
    if (le <= l[o] && r[o] <= ri) {
        val[o] = delta * (r[o] - l[o] + 1);
        tag[o] = delta;
        return;
    }
    if (~tag[o]) {
        pushdown(o);
    }
    int mid = (l[o] + r[o]) >> 1;
    if (le <= mid) {
        update(lson[o], le, ri, delta);
    }
    if (mid < ri) {
        update(rson[o], le, ri, delta);
    }
    pushup(o);
}

long long query(int o, int le, int ri) {
    if (le <= l[o] && r[o] <= ri) {
        return val[o];
    }
    if (~tag[o]) {
        pushdown(o);
    }
    int mid = (l[o] + r[o]) >> 1;
    long long res = 0;
    if (le <= mid) {
        res += query(lson[o], le, ri);
    }
    if (mid < ri) {
        res += query(rson[o], le, ri);
    }
    return res;
}

long long ask(int o) {
    int u = top[o];
    long long res = 0;
    while (u) {
        res += dfn[o] - dfn[u] - query(root, dfn[u], dfn[o]) + 1;
        update(root, dfn[u], dfn[o], 1);
        o = fa[u];
        u = top[o];
    }
```

```
    res += dfn[o] - dfn[0] - query(root, dfn[0], dfn[o]) + 1;
    update(root, dfn[0], dfn[o], 1);
    return res;
}

int main() {
    scanf("%d", &n);
    for (int i = 1, x; i < n; ++i) {
        scanf("%d", &x);
        g[i].push_back(x);
        g[x].push_back(i);
    }
    dfs1(0, -1, 1);
    dfs2(0, -1, 0);
    root = pts++;
    build(root, 1, n);
    for (scanf("%d", &q); q; --q) {
        int x;
        scanf("%s%d", cmd, &x);
        if (*cmd == 'i') {
            printf("%lld\n", ask(x));
        }
        else {
            printf("%lld\n", query(root, dfn[x], dfn[x] + siz[x] - 1));
            update(root, dfn[x], dfn[x] + siz[x] - 1, 0);
        }
    }
    return 0;
}
```

2.9 二分图匹配

2.9.1 Hungary

输入一张二分图，输出这张二分图的最大匹配数。

- 模板题参考了 Luogu P3386 。

```
#include <bits/stdc++.h>
#define MAXN 555
#define MAXM 555

using namespace std;
int n, m, c, p[MAXN + MAXM];
bool vis[MAXN + MAXM];
vector<int> g[MAXN];

```

```cpp
bool hungary(int u) {
        for (auto &v : g[u]) {
                if (vis[v])
                        continue;
                vis[v] = 1;
                if (!p[v] || hungary(p[v])) {
                        p[v] = u;
                        return 1;
                }
        }
        return 0;
}

int main() {
    for (scanf("%d%d%d", &n, &m, &c); c; --c) {
        int u, v;
        scanf("%d%d", &u, &v);
        g[u].push_back(v + n);
    }
    int ans = 0;
    for (int i = 1; i <= n; ++i) {
        memset(vis, 0, sizeof(vis));
        hungary(i) && ++ans;
    }
    printf("%d\n", ans);
    return 0;
}
```

2.10 点分治

2.10.1 聪聪可可

来自集训队互测的一道题。

- 常用于处理关于点对的问题。

```cpp
#include <bits/stdc++.h>
#define MAXN 22222
#define MAXD 11

using namespace std;
int n, siz_tot, siz_root, root, ans;
int siz_son[MAXN], siz[MAXN], dep[MAXN], cnt[MAXD];
bool vis[MAXN];
vector<pair<int, int> > g[MAXN];

void getroot(int u, int f) {
```

```
    siz[u] = 1;
    siz_son[u] = 0;
    for (auto &e : g[u]) {
        int v = e.second;
        if (vis[v] || v == f) {
            continue;
        }
        getroot(v, u);
        siz[u] += siz[v];
        siz_son[u] = max(siz_son[u], siz[v]);
    }
    siz_son[u] = max(siz_son[u], siz_tot - siz[u]);
    if (siz_son[root] > siz_son[u]) {
        root = u;
    }
}

void getdep(int u, int f) {
    ++cnt[dep[u]];
    for (auto &e : g[u]) {
        int v = e.second, w = e.first;
        if (vis[v] || v == f) {
            continue;
        }
        dep[v] = (dep[u] + w) % 3;
        getdep(v, u);
    }
}

int solve(int u, int w) {
    cnt[0] = cnt[1] = cnt[2] = 0;
    dep[u] = w;
    getdep(u, 0);
    return cnt[0] * cnt[0] + 2 * cnt[1] * cnt[2];
}

void dq(int u) {
    ans += solve(u, 0);
    vis[u] = 1;
    for (auto &e : g[u]) {
        int v = e.second, w = e.first;
        if (vis[v]) {
            continue;
        }
        ans -= solve(v, w);
        root = 0;
        siz_tot = siz[v];
        getroot(v, 0);
```

```
        dq(root);
    }
}

int main() {
    scanf("%d", &n);
    for (int i = 1, x, y, w; i < n; ++i) {
        scanf("%d%d%d", &x, &y, &w);
        w %= 3;
        g[x].push_back(make_pair(w, y));
        g[y].push_back(make_pair(w, x));
    }
    siz_son[0] = siz_tot = siz_root = n;
    getroot(1, 0);
    dq(root);
    int g = __gcd(ans, n * n);
    printf("%d/%d\n", ans / g, n * n / g);
    return 0;
}
```

3 网络流

3.1 最大流

3.1.1 Dinic

输入一个流网络，输出该网络的最大流。

- 模板题来自 Luogu P3376。
- 模板修改自 Oi Wiki。

```
#include <bits/stdc++.h>
#define MAXN 250
#define INF 0x3f3f3f3f

using namespace std;

struct Dinic {
    struct Edge {
        int from, to;
        long long cap, flow;
        Edge(int u, int v, long long c, long long f) : from(u), to(v),
 ↪  cap(c), flow(f) {}
    };

    int n, m, s, t;
```

```
    bool vis[MAXN];
    vector<Edge> edges;
    vector<int> g[MAXN];
    int d[MAXN], cur[MAXN];

    void init(int n) {
        for (int i = 0; i < n; ++i) {
            g[i].clear();
        }
        edges.clear();
    }

    void insert(int from, int to, long long cap) {
        edges.emplace_back(from, to, cap, 0);
        edges.emplace_back(to, from, 0, 0);
        m = edges.size();
        g[from].push_back(m - 2);
        g[to].push_back(m - 1);
    }

    bool bfs() {
        memset(vis, 0, sizeof(vis));
        queue<int> q;
        d[s] = 0;
        q.push(s);
        vis[s] = 1;
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (auto &x : g[u]) {
                Edge& e = edges[x];
                if (!vis[e.to] && e.cap > e.flow) {
                    vis[e.to] = 1;
                    d[e.to] = d[u] + 1;
                    q.push(e.to);
                }
            }
        }
        return vis[t];
    }

    int dfs(int u, long long c) {
        if (u == t || !c) {
            return c;
        }
        long long flow = 0, f;
        for (int& i = cur[u]; i < (int)g[u].size(); ++i) {
            Edge& e = edges[g[u][i]];
```

```
            if (d[u] + 1 == d[e.to] && (f = dfs(e.to, min(c, e.cap - e.flow))) > 0) {
                e.flow += f;
                edges[g[u][i] ^ 1].flow -= f;
                flow += f;
                c -= f;
                if (!c) {
                    break;
                }
            }
        }
        return flow;
    }

    long long dinic(int s, int t) {
        this -> s = s;
        this -> t = t;
        long long flow = 0;
        while (bfs()) {
            memset(cur, 0, sizeof(cur));
            flow += dfs(s, INF);
        }
        return flow;
    }
} andy;

int main() {
    int m;
    scanf("%d%d%d%d", &andy.n, &m, &andy.s, &andy.t);
    for (int i = 1, u, v, c; i <= m; ++i) {
        scanf("%d%d%d", &u, &v, &c);
        andy.insert(u, v, c);
    }
    printf("%lld\n", andy.dinic(andy.s, andy.t));
    return 0;
}
```

4 字符串

4.1 哈希

给出一系列字符串，求其中有多少个不同的字符串。

- 模板题来自Luogu P3370。
- 如果只需实现模板题要求的功能，可以直接使用 STL 中的 map 和 unordered_map（甚至是 set 和 unordered_set）。
- 为保险起见，可以使用两组或两组以上的基数和模数进行多次哈希。

- 可以使用自然溢出或 STL 等替代手写哈希表，但有被毒瘤出题人卡的可能...
- 如果使用热门模数无法通过本应通过的题，原因可能同上。

```cpp
#include <bits/stdc++.h>
#define MAXN 11111

using namespace std;
const int base = 131, mod = (int)1e9 + 727;
int n, ans, val[MAXN];
string s;

int main() {
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> s;
        long long cur = 0;
        for (auto &c : s) {
            cur = (cur * base + c) % mod;
        }
        val[i] = cur;
    }
    sort(val + 1, val + n + 1);
    for (int i = 1; i <= n; ++i) {
        if (val[i - 1] != val[i]) {
            ++ans;
        }
    }
    cout << ans << endl;
    return 0;
}
```

4.2 KMP

给出两个字符串，输出第二个字符串在第一个字符串中出现的位置。

定义一个字符串 s 的 border 为 s 的一个非 s 本身的子串 t，满足 t 既是 s 的前缀，又是 s 的后缀。输出第二个字符串长度为 i 的字串的最长 border 长度。

- 模板题来自Luogu P3375。

```cpp
#include <bits/stdc++.h>
#define MAXN 1111111

using namespace std;
int n, m, fail[MAXN];
```

```
string a, b;

void getfail() {
    int i = 0, j = -1;
    fail[0] = -1;
    while (i < m) {
        if (j == -1 || b[i] == b[j]) {
            fail[++i] = ++j;
        }
        else {
            j = fail[j];
        }
    }
}

void kmp() {
    int i = 0, j = 0;
    while (i < n) {
        if (j == -1 || a[i] == b[j]) {
            ++i;
            ++j;
        }
        else {
            j = fail[j];
        }
        if (j == m) {
            cout << i - m + 1 << endl;
            j = fail[j];
        }
    }
}

int main() {
    cin >> a >> b;
    n = a.length();
    m = b.length();
    getfail();
    kmp();
    for (int i = 1; i <= m; ++i) {
        cout << fail[i] << " \n"[i == m];
    }
    return 0;
}
```

5 数学

5.1 特征多项式

5.1.1 Determinant

输入一个矩阵和不同的特征值，输出特征多项式的值。

- 来自 2020-2021 Summer Petrozavodsk Camp, Day 6: Korean Contest 的 K 题。
- 感谢 yh 学长提供的模板。
- 包含了一个取模版的高斯消元。

```cpp
#include <bits/stdc++.h>
#define MAXN 555
#define MOD 998244353

using namespace std;
int n, q;
long long a[MAXN][MAXN], b[MAXN][MAXN];

long long xpow(long long a, long long b) {
    long long res = 1;
    while (b) {
        if (b & 1) {
            res = res * a % MOD;
        }
        a = a * a % MOD;
        b >>= 1;
    }
    return res;
}

long long inv(long long o) {
    return xpow(o, MOD - 2);
}

void gauss() {
    for (int i = 1; i <= n; ++i) {
        if (!a[i + 1][i]) {
            bool flag = 0;
            for (int j = i + 2; j <= n; ++j) {
                if (a[j][i]) {
                    for (int k = i; k <= n; ++k) {
                        swap(a[i + 1][k], a[j][k]);
                    }
                    for (int k = 1; k <= n; ++k) {
                        swap(a[k][i + 1], a[k][j]);
```

```
                    }
                    flag = 0;
                    break;
                }
            }
            if (flag) {
                continue;
            }
        }
        for (int j = i + 2; j <= n; ++j) {
            long long cur = MOD - a[j][i] * inv(a[i + 1][i]) % MOD;
            for (int k = i; k <= n; ++k) {
                a[j][k] = (a[j][k] + cur * a[i + 1][k] % MOD) % MOD;
            }
            for (int k = 1; k <= n; ++k) {
                a[k][i + 1] = (a[k][i + 1] + (MOD - cur) * a[k][j] % MOD)
↪  % MOD;
            }
        }
    }
}

void charpoly() {
    b[0][0] = 1;
    for (int i = 1; i <= n; ++i) {
        for (int j = 0; j < i; ++j) {
            b[i][j] = (b[i][j] + (b[i - 1][j] * a[i][i]) % MOD) % MOD;
            b[i][j + 1] = (b[i][j + 1] + (MOD - b[i - 1][j])) % MOD;
        }
        int cur = 1;
        for (int j = i - 1; j >= 1; --j) {
            cur = cur * a[j + 1][j] % MOD * (MOD - 1) % MOD;
            int res = cur * a[j][i] % MOD;
            for (int k = 0; k < j; ++k) {
                b[i][k] = (b[i][k] + (res * b[j - 1][k]) % MOD) % MOD;
            }
        }
    }
}

int main() {
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            scanf("%lld", &a[i][j]);
        }
    }
    gauss();
```

```
    charpoly();
    while (q--) {
        long long x, y = 1, ans = 0;
        scanf("%lld", &x);
        for (int i = 0; i <= n; ++i) {
            ans = (ans + (b[n][i] * y % MOD)) % MOD;
            y = y * x % MOD;
        }
        printf("%lld%c", ans, " \n"[q == 0]);
    }
    return 0;
}
```

6 二分

6.1 符合条件的最大值

小于等于答案的值都合法。

```
bool check(int o) {
    ...
}

int bs() {
    int l, r, mid, ans;
    while (l <= r) {
        if (check(mid = (l + r) >> 1)) {
            l = mid + 1;
            ans = max(ans, mid);
        }
        else {
            r = mid - 1;
        }
    }
    return ans;
}
```

6.2 符合条件的最小值

大于等于答案的值都合法。

```
const int INF = ...;

bool check(int o) {
    ...
}
```

```
6
7  int bs() {
8      int l, r, mid, ans = INF;
9      while (l <= r) {
10         if (check(mid = (l + r) >> 1)) {
11             r = mid - 1;
12             ans = min(ans, mid);
13         }
14         else {
15             l = mid + 1;
16         }
17     }
18     return ans;
19 }
```

Good Luck & Have Fun