

widsnoy's template

| | |
|--------------------------|---|
| 1. 数论 | 3 |
| 1.1. 取模还原分数 | 3 |
| 1.2. 解不定方程 | 3 |
| 1.3. 中国剩余定理 | 3 |
| 1.4. 卢卡斯定理 | 3 |
| 1.5. exBSGS | 3 |
| 1.6. 二次剩余 | 3 |
| 1.7. Miller-Rabin | 3 |
| 1.8. Pollard-rho | 3 |
| 1.9. 莫比乌斯反演 (两种形式) | 3 |
| 1.10. 除法分块 (上下取整) | 3 |
| 1.11. Min25 筛 | 3 |
| 1.12. 区间筛 | 3 |
| 1.13. 数论卷积 | 3 |
| 2. 动态规划 | 3 |
| 2.1. 缺 1 背包 | 3 |
| 3. 图论 | 3 |
| 3.1. 找环 | 4 |
| 3.2. SPFA 乱搞 | 4 |
| 3.3. 差分约束 | 5 |
| 3.4. 竞赛图 | 5 |
| 3.5. 有向图强连通分量 | 5 |
| 3.5.1. Tarjan | 5 |
| 3.5.2. Kosaraju | 5 |
| 3.6. 连通分量 | 5 |
| 3.6.1. 割点 | 5 |
| 3.6.2. 桥 | 5 |
| 3.6.3. 点双 | 5 |

| | |
|-------------------------|---|
| 3.6.4. 边双 | 5 |
| 3.7. 二分图匹配 | 5 |
| 3.7.1. 匈牙利算法 | 6 |
| 3.7.2. KM | 6 |
| 3.8. 网络流 | 6 |
| 3.8.1. 网络最大流 | 6 |
| 3.8.2. 最小费用最大流 | 6 |
| 3.8.2.1. spfa | 6 |
| 3.8.2.2. zkw | 6 |
| 3.8.3. 上下界网络流 | 6 |
| 3.9. 2-SAT | 6 |
| 3.9.1. 搜索 (最小字典序) | 6 |
| 3.9.2. tarjan | 6 |
| 3.10. 生成树 | 6 |
| 3.10.1. Prime | 6 |
| 3.10.2. Kruskal | 6 |
| 3.10.3. 次小生成树 | 6 |
| 3.10.4. 生成树计数 | 6 |
| 3.11. 三元环 | 6 |
| 3.12. 四元环 | 6 |
| 3.13. 欧拉路 | 6 |
| 3.14. 曼哈顿路 | 6 |
| 3.15. 建图优化 | 6 |
| 3.15.1. 前后缀优化 | 6 |
| 3.15.2. 线段树优化 | 6 |
| 4. 树论 | 6 |
| 4.1. prufer | 6 |
| 4.2. 圆方树 | 6 |
| 4.2.1. 广义 | 6 |

| | | | |
|------------------------|---|--------------------------------|---|
| 4.2.2. 仙人掌 | 6 | 8.1.1. 李超树 (最大, 次大, 第三大) | 7 |
| 4.3. 最近公共祖先 | 6 | 8.1.2. 合并分裂 | 7 |
| 4.4. 树分治 | 6 | 8.1.3. 线段树二分 | 7 |
| 4.4.1. 点分治 | 6 | 8.1.4. 免队线段树 | 7 |
| 4.4.2. 点分树 | 6 | 8.2. 平衡树 | 7 |
| 4.5. 链分治 | 6 | 8.2.1. 文艺平衡树 | 7 |
| 4.5.1. 重链分治 | 6 | 8.3. 历史版本信息线段树 | 7 |
| 4.5.2. 长链分治 | 6 | 8.4. 树状数组二分 | 7 |
| 4.6. dsu on tree | 6 | 8.5. 二维树状数组 | 7 |
| 5. 数学 | 7 | 8.6. ODT | 7 |
| 5.1. 组合恒等式 | 7 | 8.7. KDT | 7 |
| 5.2. min-max 容斥 | 7 | 8.8. 手写堆 | 7 |
| 5.3. 序列容斥 | 7 | 9. 字符串 | 7 |
| 5.4. 二项式反演 | 7 | 9.1. KMP | 7 |
| 5.5. 斯特林数 | 7 | 9.2. exKMP | 8 |
| 5.6. 高维前缀和 | 7 | 9.3. SA | 8 |
| 6. 线性代数 | 7 | 9.4. AC 自动机 | 8 |
| 6.1. 线性基 | 7 | 9.5. 马拉车 | 8 |
| 6.2. 行列式 | 7 | 10. 杂项 | 8 |
| 6.3. 高斯消元 | 7 | 10.1. gcd, xor, or 分块 | 8 |
| 7. 多项式 | 7 | 10.2. 超级钢琴 | 8 |
| 7.1. 快速数论变换 | 7 | 10.3. 平方计数 | 8 |
| 7.2. 快速傅里叶变换 | 7 | 10.4. FFT 字符串匹配 | 8 |
| 7.3. 任意模数 NTT | 7 | 10.5. 循环矩阵乘法 | 8 |
| 7.4. 自然数幂和 | 7 | 10.6. 线性逆元 | 8 |
| 7.5. 快速沃尔什变换 | 7 | 10.7. 快速幂 | 8 |
| 7.6. 子集卷积 | 7 | 10.8. 数学题基本预处理 | 8 |
| 8. 数据结构 | 7 | 10.9. fastio | 8 |
| 8.1. 线段树 | 7 | 10.10. 高精度 | 8 |

| | |
|-----------------------|---|
| 11. 配置相关 | 8 |
| 11.1. 对拍 | 8 |
| 11.2. vscode 配置 | 8 |

1. 数论

1.1. 取模还原分数

1.2. 解不定方程

1.3. 中国剩余定理

1.4. 卢卡斯定理

1.5. exBSGS

1.6. 二次剩余

1.7. Miller-Rabin

1.8. Pollard-rho

1.9. 莫比乌斯反演 (两种形式)

1.10. 除法分块 (上下取整)

1.11. Min25 筛

1.12. 区间筛

1.13. 数论卷积

2. 动态规划

2.1. 背包

3. 图论

3.1. 找环

```
1 const int N = 5e5 + 5;
2 int n, m, col[N], pre[N], pre_edg[N];
3 vector<pii> G[N];
4 vector<vector<int>> resp, rese;
5 //point
6 void get_cyc(int u, int v) {
7     if (!resp.empty()) return;
8     vector<int> cyc;
9     cyc.push_back(v);
10    while (true) {
11        v = pre[v];
12        if (v == 0) break;
13        cyc.push_back(v);
14        if (v == u) break;
15    }
16    reverse(cyc.begin(), cyc.end());
17    resp.push_back(cyc);
18 }
19 // edge
20 void get_cyc(int u, int v, int id) {
21     if (!rese.empty()) return;
22     vector<int> cyc;
23     cyc.push_back(id);
24     while (true) {
25         if (pre[v] == 0) break;
26         cyc.push_back(pre_edg[v]);
27         v = pre[v];
28         if (v == u) break;
29     }
30     reverse(cyc.begin(), cyc.end());
31     rese.push_back(cyc);
32 }
33 void dfs(int u, int edg) {
```

```
34     col[u] = 1;
35     for (auto [v, id] : G[u]) if (id != edg) {
36         if (col[v] == 1) {
37             get_cyc(v, u);
38             get_cyc(v, u, id);
39         } else if (col[v] == 0) {
40             pre[v] = u;
41             pre_edg[v] = id;
42             dfs(v, id);
43         }
44     }
45     col[u] = 2;
46 }
47 void MAIN() {
48     cin >> n >> m;
49     for (int i = 1; i <= m; i++) {
50         int u, v; cin >> u >> v;
51         // G[u].push_back({v, i});
52         // G[v].push_back({u, i});
53     }
54     for (int i = 1; i <= n; i++) if (!col[i])
55         dfs(i, -1);
56 }
```

3.2. SPFA 乱搞

```
1 mt19937_64
2   rng(chrono::steady_clock::now().time_since_epoch().count());
3 const int mod = 998244353;
4 const int N = 5e5 + 5;
5 const ll inf = 1e17;
6 int n, m, s, t, q[N], ql, qr;
```

```
7 int vis[N], fr[N];
8 ll dis[N];
9 vector<pii> G[N];
10 void MAIN() {
11     cin >> n >> m >> s >> t;
12     for (int i = 1; i <= m; i++) {
13         int u, v, w;
14         cin >> u >> v >> w;
15         G[u].push_back({v, w});
16     }
17     for (int i = 0; i <= n; i++) dis[i] = inf;
18     dis[s] = 0; q[qr] = s; vis[s] = 1;
19     while (ql <= qr) {
20         if (rng() % (qr - ql + 1) == 0) sort(q
+ ql, q + qr + 1, [](int x, int y) {
21             return dis[x] < dis[y];
22         });
23         int u = q[ql++];
24         vis[u] = 0;
25         for (auto [v, w] : G[u]) {
26             if (dis[u] + w < dis[v]) {
27                 dis[v] = dis[u] + w;
28                 fr[v] = u;
29                 if (!vis[v]) {
30                     if (ql > 0) q[--ql] = v;
31                     else q[++qr] = v;
32                     vis[v] = 1;
33                 }
34             }
35         }
36     }
37     if (dis[t] == inf) {
38         cout << "-1\n";
39     }
    return;
```

```
40     }
41     cout << dis[t] << ' ';
42     vector<pii> stk;
43     while (t != s) {
44         stk.push_back({fr[t], t});
45         t = fr[t];
46     }
47     reverse(stk.begin(), stk.end());
48     cout << stk.size() << '\n';
49     for (auto [u, v] : stk) cout << u << ' ' <<
v << '\n';
50 }
```

3.3. 差分约束

3.4. 竞赛图

3.5. 有向图强连通分量

3.5.1. Tarjan

3.5.2. Kosaraju

3.6. 连通分量

3.6.1. 割点

3.6.2. 桥

3.6.3. 点双

3.6.4. 边双

3.7. 二分图匹配

3.7.1. 匈牙利算法

3.7.2. KM

3.8. 网络流

3.8.1. 网络最大流

3.8.2. 最小费用最大流

3.8.2.1. spfa

3.8.2.2. zkw

3.8.3. 上下界网络流

3.9. 2-SAT

3.9.1. 搜索 (最小字典序)

3.9.2. tarjan

3.10. 生成树

3.10.1. Prime

3.10.2. Kruskal

3.10.3. 次小生成树

3.10.4. 生成树计数

3.11. 三元环

3.12. 四元环

3.13. 欧拉路

3.14. 曼哈顿路

3.15. 建图优化

3.15.1. 前后缀优化

3.15.2. 线段树优化

4. 树论

4.1. prufer

4.2. 圆方树

4.2.1. 广义

4.2.2. 仙人掌

4.3. 最近公共祖先

4.4. 树分治

4.4.1. 点分治

4.4.2. 点分树

4.5. 链分治

4.5.1. 重链分治

4.5.2. 长链分治

4.6. dsu on tree

5. 数学

5.1. 组合恒等式

5.2. min-max 容斥

5.3. 序列容斥

5.4. 二项式反演

5.5. 斯特林数

5.6. 高维前缀和

6. 线性代数

6.1. 线性基

6.2. 行列式

6.3. 高斯消元

7. 多项式

7.1. 快速数论变换

7.2. 快速傅里叶变换

7.3. 任意模数 NTT

7.4. 自然数幂和

7.5. 快速沃尔什变换

7.6. 子集卷积

8. 数据结构

8.1. 线段树

8.1.1. 李超树 (最大, 次大, 第三大)

8.1.2. 合并分裂

8.1.3. 线段树二分

8.1.4. 兔队线段树

8.2. 平衡树

8.2.1. 文艺平衡树

8.3. 历史版本信息线段树

8.4. 树状数组二分

8.5. 二维树状数组

8.6. ODT

8.7. KDT

8.8. 手写堆

9. 字符串

9.1. KMP

9.2. exKMP

9.3. SA

9.4. AC 自动机

9.5. 马拉车

10. 杂项

10.1. gcd, xor, or 分块

10.2. 超级钢琴

10.3. 平方计数

10.4. FFT 字符串匹配

10.5. 循环矩阵乘法

10.6. 线性逆元

10.7. 快快速幂

10.8. 数学题基本预处理

10.9. fastio

10.10. 高精度

11. 配置相关

11.1. 对拍

11.2. vscode 配置