

Standard Code Library

Andy Liao

November 11, 2021

Contents

1 写在前面	3
1.1 序言	3
1.2 代码规范	3
2 图论	3
2.1 图的存储与遍历	3
2.1.1 邻接矩阵	3
2.1.2 邻接表	4
2.2 最短路	5
2.2.1 Dijkstra	5
2.2.2 SPFA	6
2.2.3 Floyd	7
2.3 最小生成树	8
2.3.1 Kruskal	8
2.3.2 Prim	9
2.4 最近公共祖先	10
2.4.1 倍增	10
2.4.2 RMQ	12
2.4.3 树链剖分	13
2.4.4 Tarjan	14
2.5 欧拉回路	16
2.6 拓扑排序	17
2.7 连通性	18
2.7.1 割点和桥	18
2.7.2 边双连通分量	19
2.7.3 点双连通分量	20
2.7.4 强连通分量	22
2.8 树链剖分	23
2.8.1 软件包管理器	23
2.9 二分图匹配	26
2.9.1 Hungary	26
2.10 点分治	27
2.10.1 聪聪可可	27
3 网络流	29
3.1 最大流	29
3.1.1 Dinic	29

4 字符串	31
4.1 哈希	31
4.2 KMP	32
5 数学	34
5.1 特征多项式	34
5.1.1 Determinant	34
6 二分	36
6.1 符合条件的最大值	36
6.2 符合条件的最小值	36

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 **MirAcle** 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 **MirAcle** 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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 1111
3
4  using namespace std;
5  int n, m, g[MAXN][MAXN];
6
7  int main() {
8      scanf("%d%d", &n, &m);
9      for (int i = 1; i <= n; ++i) {
10         for (int j = 1; j <= n; ++j) {
11             g[i][j] = -1;
12         }
13     }
14     for (int i = m; i > 0; --i) {
15         scanf("%d%d%d", &u, &v, &w);
16         g[u][v] = w;
17     }
18     for (int i = 1; i <= n; ++i) {
19         for (int j = 1; j <= n; ++j) {
20             if (~g[i][j]) {
21                 printf("%d %d %d\n", i, j, g[i][j]);
22             }
23         }
24     }
25     return 0;
26 }
```

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 **MirAcle** 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。

```

1  #include <bits/stdc++.h>
2  #define MAXN 1111
3
4  using namespace std;
5  int n, m;
6  vector<pair<int, int> > g[MAXN];
7
8  int main() {
9      scanf("%d%d", &n, &m);
10     for (int i = m, u, v, w; i; --i) {
11         scanf("%d%d%d", &u, &v, &w);
12         g[u].push_back(make_pair(w, v));
13     }
14     for (int i = 1; i <= n; ++i) {
15         for (auto &e : g[i]) {
16             printf("%d %d %d\n", i, e.second, e.first);
17         }
18     }
19     return 0;
20 }

```

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。

```

1  #include <bits/extc++.h>
2  #define MAXN 111111
3  #define INF 0x3f
4
5  using namespace std;
6  int n, m, s, t, dis[MAXN];
7  vector<pair<int, int> > g[MAXN];
8  __gnu_pbds::priority_queue<pair<int, int>, greater<pair<int, int> >,
  ↪ __gnu_pbds::pairing_heap_tag> q;

```

```
9
10 int main() {
11     memset(dis, INF, sizeof(dis));
12     scanf("%d%d%d%d", &n, &m, &s, &t);
13     for (int i = 1, u, v, w; i <= m; ++i) {
14         scanf("%d%d%d", &u, &v, &w);
15         g[u].push_back(make_pair(w, v));
16     }
17     dis[s] = 0;
18     q.push(make_pair(0, s));
19     while (!q.empty()) {
20         int u = q.top().second;
21         q.pop();
22         for (auto &e : g[u]) {
23             int v = e.second, w = e.first;
24             if (dis[v] > dis[u] + w) {
25                 q.push(make_pair(dis[v] = dis[u] + w, v));
26             }
27         }
28     }
29     printf("%d\n", dis[t]);
30     return 0;
31 }
```

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 111111
3  #define INF 0x3f
4
5  using namespace std;
6  int n, m, s, t, dis[MAXN];
7  bool vis[MAXN];
8  vector<pair<int, int> > g[MAXN];
9  queue<int> q;
10
11 int main() {
12     memset(dis, INF, sizeof(dis));
13     scanf("%d%d%d%d", &n, &m, &s, &t);
14     for (int i = 1, u, v, w; i <= m; ++i) {
15         scanf("%d%d%d", &u, &v, &w);
16         g[u].push_back(make_pair(w, v));
17     }
18     dis[s] = 0;
19     q.push(s);
20     while (!q.empty()) {
21         int u = q.front();
22         q.pop();
23         vis[u] = 0;
24         for (auto &e : g[u]) {
25             int v = e.second, w = e.first;
26             if (dis[v] > dis[u] + w) {
27                 dis[v] = dis[u] + w;
28                 if (!vis[v]) {
29                     vis[v] = 1;
30                     q.push(v);
31                 }
32             }
33         }
34     }
35     printf("%d\n", dis[t]);
36     return 0;
37 }
```

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 1111
3
4  using namespace std;
5  int n, m, g[MAXN][MAXN];
6
7  int main() {
8      scanf("%d%d", &n, &m);
9      for (int i = 1; i <= n; ++i) {
10         for (int j = 1; j <= n; ++j) {
11             g[i][j] = -1;
12         }
13         g[i][i] = 0;
14     }
15     for (int i = m; i >= 1; --i) {
16         scanf("%d%d%d", &u, &v, &w);
17         g[u][v] = w;
18         g[v][u] = w;
19     }
20     for (int k = 1; k <= n; ++k) {
21         for (int i = 1; i <= n; ++i) {
22             for (int j = 1; j <= n; ++j) {
23                 g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
24             }
25         }
26     }
27     for (int i = 1; i <= n; ++i) {
28         for (int j = 1; j <= n; ++j) {
29             printf("%d ", g[i][j]);
30         }
31         putchar('\n');
32     }
33     return 0;
34 }
```

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。

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



```
3
4 using namespace std;
5 int n, m, fa[MAXN];
6 vector<pair<int, pair<int, int> > > g;
7
8 int find(int x) {
9     return fa[x] != x ? fa[x] = find(fa[x]) : x;
10 }
11
12 int main() {
13     for (scanf("%d%d", &n, &m); m; --m) {
14         int u, v, w;
15         scanf("%d%d%d", &u, &v, &w);
16         g.push_back(make_pair(w, make_pair(u, v)));
17     }
18     for (int i = 0; i < n; ++i) {
19         fa[i] = i;
20     }
21     sort(g.begin(), g.end());
22     int ans = 0;
23     for (auto &e : g) {
24         int p = find(e.second.first), q = find(e.second.second);
25         if (p != q) {
26             fa[q] = p;
27             ans += e.first;
28         }
29     }
30     printf("%d\n", ans);
31     return 0;
32 }
```

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**。

```
1 #include <bits/stdc++.h>
2 #define MAXN 1111
3 #define INF 0x3f3f3f3f
4
5 using namespace std;
6 int n, m, dis[MAXN], g[MAXN][MAXN];
7 bool vis[MAXN];
8
```

```

9  int main() {
10     memset(g, INF, sizeof(g));
11     memset(dis, INF, sizeof(dis));
12     for (scanf("%d%d", &n, &m); m; --m) {
13         int u, v, w;
14         scanf("%d%d%d", &u, &v, &w);
15         g[u][v] = g[v][u] = min(g[u][v], w);
16     }
17     int ans = 0;
18     dis[0] = 0;
19     for (int i = 0; i < n; ++i) {
20         int d_min = INF, id_min = 0;
21         for (int j = 0; j < n; ++j) {
22             if (!vis[j] && d_min > dis[j]) {
23                 d_min = dis[j];
24                 id_min = j;
25             }
26         }
27         ans += dis[id_min];
28         vis[id_min] = 1;
29         for (int j = 0; j < n; ++j) {
30             if (!vis[j] && dis[j] > g[id_min][j]) {
31                 dis[j] = g[id_min][j];
32             }
33         }
34     }
35     printf("%d\n", ans);
36     return 0;
37 }

```

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？

```

1  #include <bits/stdc++.h>
2  #define MAXN 111111
3  #define LOGN 33
4
5  using namespace std;
6  int n, q, fa[MAXN][LOGN], dep[MAXN];

```

```
7 vector<int> g[MAXN];
8
9 void dfs(int u, int f) {
10     dep[u] = dep[f] + 1;
11     for (auto &v : g[u]) {
12         dfs(v, u);
13     }
14 }
15
16 int lca(int a, int b) {
17     if (dep[a] > dep[b]) {
18         swap(a, b);
19     }
20     int delta = dep[b] - dep[a];
21     for (int j = 0; (1 << j) <= delta; ++j) {
22         if (delta & (1 << j)) {
23             b = fa[b][j];
24         }
25     }
26     if (a == b) {
27         return a;
28     }
29     int j;
30     for (j = 1; (1 << j) <= n; ++j);
31     for (--j; ~j; --j) {
32         if (fa[a][j] != fa[b][j]) {
33             a = fa[a][j];
34             b = fa[b][j];
35         }
36     }
37     a = fa[a][0];
38     return a;
39 }
40
41 int main() {
42     scanf("%d%d", &n, &q);
43     for (int i = 1, x, y; i < n; ++i) {
44         scanf("%d%d", &x, &y);
45         fa[y][0] = x;
46         g[x].push_back(y);
47     }
48     dfs(1, 0);
49     for (int j = 1; (1 << j) <= n; ++j) {
50         for (int i = 1; i <= n; ++i) {
51             fa[i][j] = fa[fa[i][j - 1]][j - 1];
52         }
53     }
54     while (q--) {
```

```

55     int a, b;
56     scanf("%d%d", &a, &b);
57     printf("%d\n", lca(a, b));
58 }
59 return 0;
60 }

```

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。

```

1  #include <bits/stdc++.h>
2  #define MAXN 111111
3  #define LOGN 33
4
5  using namespace std;
6
7  int n, q, clk, dfn[MAXN], a[MAXN << 1], dep[MAXN << 1], rmq[MAXN <<
  ↪ 1][LOGN];
8  vector<int> g[MAXN];
9
10 void dfs(int u, int d) {
11     ++clk;
12     dfn[u] = clk;
13     a[clk] = u;
14     dep[clk] = d;
15     for (auto &v : g[u]) {
16         dfs(v, d + 1);
17         ++clk;
18         a[clk] = u;
19         dep[clk] = d;
20     }
21 }
22
23 int query(int l, int r) {
24     int k = 0;
25     while ((1 << (k + 1)) <= r - l + 1) {
26         ++k;
27     }
28     return dep[rmq[l][k]] <= dep[rmq[r - (1 << k) + 1][k]] ? rmq[l][k] :
  ↪ rmq[r - (1 << k) + 1][k];
29 }
30

```

```

31 int main() {
32     scanf("%d%d", &n, &q);
33     for (int i = 1, x, y; i < n; ++i) {
34         scanf("%d%d", &x, &y);
35         g[x].push_back(y);
36     }
37     dfs(1, 1);
38     for (int i = 1; i < clk; ++i) {
39         rmq[i][0] = i;
40     }
41     for (int j = 1; (1 << j) < clk; ++j) {
42         for (int i = 1; i + (1 << j) - 1 < clk; ++i) {
43             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];
44         }
45     }
46     while (q--) {
47         int x, y;
48         scanf("%d%d", &x, &y);
49         printf("%d\n", a[query(min(x, y), max(x, y))]);
50     }
51     return 0;
52 }

```

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！

```

1  #include <bits/stdc++.h>
2  #define MAXN 111111
3
4  using namespace std;
5  int n, q, dep[MAXN], siz[MAXN], fa[MAXN], son[MAXN], top[MAXN];
6  vector<int> g[MAXN];
7
8  void dfs1(int u) {
9      int f = fa[u], *s = &son[u];
10     dep[u] = dep[f] + 1;
11     siz[u] = 1;
12     for (auto &v : g[u]) {
13         if (v != f && !fa[v]) {
14             fa[v] = u;
15             dfs1(v);

```

```

16         siz[u] += siz[v];
17         if (siz[*s] < siz[v]) {
18             *s = v;
19         }
20     }
21 }
22 }
23
24 void dfs2(int u) {
25     int f = fa[u];
26     top[u] = u == son[f] ? top[f] : u;
27     for (auto &v : g[u]) {
28         if (fa[v] == u) {
29             dfs2(v);
30         }
31     }
32 }
33
34 int query(int x, int y) {
35     int a, b;
36     while ((a = top[x]) != (b = top[y])) {
37         dep[a] > dep[b] ? x = fa[a] : y = fa[b];
38     }
39     return dep[x] < dep[y] ? x : y;
40 }
41
42 int main() {
43     scanf("%d%d", &n, &q);
44     for (int i = 1, u, v; i < n; ++i) {
45         scanf("%d%d", &u, &v);
46         g[u].push_back(v);
47     }
48     dfs1(1);
49     dfs2(1);
50     while (q--) {
51         int x, y;
52         scanf("%d%d", &x, &y);
53         printf("%d\n", query(x, y));
54     }
55     return 0;
56 }

```

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 111111
3  #define MAXQ 111111
4
5  using namespace std;
6  int n, q, fa[MAXN], ans[MAXN];
7  bool vis[MAXN];
8  vector<int> g[MAXN];
9  vector<pair<int, int> > query[MAXNQ];
10
11 int find(int o) {
12     return o == fa[o] ? o : fa[o] = find(fa[o]);
13 }
14
15 void tarjan(int u, int f) {
16     for (auto &v : g[u]) {
17         if (v == f) {
18             continue;
19         }
20         tarjan(v, u);
21         fa[v] = u;
22     }
23     for (auto &q : query[u]) {
24         int v = q.second, i = q.first;
25         if (vis[v]) {
26             ans[i] = find(v);
27         }
28     }
29     vis[u] = 1;
30 }
31
32 int main() {
33     scanf("%d%d", &n, &q);
34     fa[n] = n;
35     for (int i = 1, x, y; i < n; ++i) {
36         fa[i] = i;
37         scanf("%d%d", &x, &y);
38         g[x].push_back(y);
39     }
40     for (int i = 1, a, b; i <= q; ++i) {
41         scanf("%d%d", &a, &b);
42         query[a].push_back(make_pair(i, b));
43         query[b].push_back(make_pair(i, a));
44     }
45     tarjan(1, 0);
```

```
46     for (int i = 1; i <= q; ++i) {
47         printf("%d\n", ans[i]);
48     }
49     return 0;
50 }
```

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 1111
3
4  using namespace std;
5  int n, m, ans[MAXN];
6  queue<int> g[MAXN];
7
8  void dfs(int u) {
9      while (!g[u].empty()) {
10         int v = g[u].front();
11         g[u].pop();
12         dfs(v);
13     }
14     ans[++ans] = u;
15 }
16
```



```
17 int main() {
18     for (scanf("%d%d", &n, &m); m; --m) {
19         int u, v;
20         scanf("%d%d", &u, &v);
21         g[u].push(v);
22     }
23     dfs(1);
24     for (int i = *ans; i; --i) {
25         printf("%d ", ans[i]);
26     }
27     putchar('\n');
28     return 0;
29 }
```

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 1111
3
4  using namespace std;
5  int n, m, ans[MAXN];
6  queue<int> g[MAXN];
7
8  void dfs(int u) {
9      while (!g[u].empty()) {
10         int v = g[u].front();
11         g[u].pop();
12         dfs(v);
13     }
14     ans[++*ans] = u;
15 }
16
17 int main() {
18     for (scanf("%d%d", &n, &m); m; --m) {
19         int u, v;
20         scanf("%d%d", &u, &v);
21         g[u].push(v);
```

```
22     }
23     dfs(1);
24     for (int i = *ans; i; --i) {
25         printf("%d ", ans[i]);
26     }
27     putchar('\n');
28     return 0;
29 }
```

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 155
3
4  using namespace std;
5  int n, m, fa[MAXN];
6  vector<int> g[MAXN];
7  bool bridge[MAXN], cut[MAXN];
8  int dfs_clock, dfn[MAXN], low[MAXN];
9
10 void tarjan(int u, int fa) {
11     dfn[u] = low[u] = ++dfs_clock;
12     int siz = 0;
13     for (auto &v : g[u]) {
14         if (dfn[v]) {
15             tarjan(v, u);
16             ++siz;
17             low[u] = min(low[u], low[v]);
18             if (dfn[u] <= low[v] && fa) {
19                 cut[u] = 1;
20             }
21             if (dfn[u] < low[v]) {
22                 bridge[v] = 1;
23             }
24         }
25         else if (v != fa) {
26             low[u] = min(low[u], dfn[v]);
27         }
28     }
29     if (!fa && siz > 1) {
```

```

30         cut[u] = 1;
31     }
32 }
33
34 int main() {
35     for (cin >> n >> m; m; --m) {
36         int x, y;
37         cin >> x >> y;
38         g[x].push_back(y);
39         g[y].push_back(x);
40     }
41     for (int i = 1; i <= n; ++i) {
42         if (!dfn[i]) {
43             tarjan(i, 0);
44         }
45     }
46     return 0;
47 }

```

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。

```

1  #include <bits/stdc++.h>
2  #define MAXN 155
3
4  using namespace std;
5  int n, m;
6  bool vis[MAXN];
7  vector<int> g[MAXN];
8  int bcc, bel[MAXN];
9  int dfs_clock, top, dfn[MAXN], low[MAXN], stk[MAXN];
10
11 void tarjan(int u, int fa) {
12     vis[u] = 1;
13     stk[++top] = u;
14     dfn[u] = low[u] = ++dfs_clock;
15     for (auto &v : g[u]) {
16         if (!vis[v]) {
17             tarjan(v, u);
18             low[u] = min(low[u], low[v]);

```

```
19     }
20     else if (v != fa) {
21         low[u] = min(low[u], dfn[v]);
22     }
23 }
24 if (low[u] == dfn[u]) {
25     ++bcc;
26     while (1) {
27         int x = stk[top--];
28         bel[x] = bcc;
29         if (u == x) {
30             break;
31         }
32     }
33 }
34 }
35
36 int main() {
37     for (cin >> n >> m; m; --m) {
38         int x, y;
39         cin >> x >> y;
40         g[x].push_back(y);
41         g[y].push_back(x);
42     }
43     for (int i = 1; i <= n; ++i) {
44         if (!dfn[i]) {
45             tarjan(i, 0);
46         }
47     }
48     return 0;
49 }
```

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 155
3
4  using namespace std;
5  int n, m;
```

```
6 vector<int> g[MAXN];
7 bool cut[MAXN];
8 int bcc, bel[MAXN];
9 int dfs_clock, top, dfn[MAXN], low[MAXN], stk[MAXN];
10
11 void tarjan(int u, int fa) {
12     stk[++top] = u;
13     dfn[u] = low[u] = ++dfs_clock;
14     int siz = 0;
15     for (auto &v : g[u]) {
16         if (!dfn[v]) {
17             ++siz;
18             tarjan(v, u);
19             low[u] = min(low[u], low[v]);
20             if (low[v] >= dfn[u]) {
21                 cut[u] = 1;
22                 bel[u] = ++bcc;
23                 while (1) {
24                     int x = stk[top--];
25                     bel[x] = bcc;
26                     if (v == x) {
27                         break;
28                     }
29                 }
30             }
31         }
32         else if (v != fa) {
33             low[u] = min(low[u], dfn[v]);
34         }
35     }
36     if (!fa && siz == 1) {
37         cut[u] = 0;
38     }
39 }
40
41 int main() {
42     for (cin >> n >> m; m; --m) {
43         int x, y;
44         cin >> x >> y;
45         g[x].push_back(y);
46         g[y].push_back(x);
47     }
48     for (int i = 1; i <= n; ++i) {
49         if (!dfn[i]) {
50             tarjan(i, 0);
51         }
52     }
53     return 0;
```

54 }

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 155
3
4  using namespace std;
5  int n, m;
6  vector<int> g[MAXN];
7  bool cut[MAXN];
8  int scc, bel[MAXN];
9  int dfs_clock, top, dfn[MAXN], low[MAXN], stk[MAXN];
10
11 void tarjan(int u) {
12     stk[++top] = u;
13     dfn[u] = low[u] = ++dfs_clock;
14     for (auto &v : g[u]) {
15         if (!dfn[v]) {
16             tarjan(v);
17             low[u] = min(low[u], low[v]);
18         }
19         else if (!bel[v]) {
20             low[u] = min(low[u], dfn[v]);
21         }
22     }
23     if (low[u] == dfn[u]) {
24         ++scc;
25         while (1) {
26             int x = stk[top--];
27             bel[x] = scc;
28             if (u == x) {
29                 break;
30             }
31         }
32     }
33 }
34
35 int main() {
36     for (cin >> n >> m; m; --m) {
37         int x, y;
```

```

38         cin >> x >> y;
39         g[x].push_back(y);
40     }
41     for (int i = 1; i <= n; ++i) {
42         if (!dfn[i]) {
43             tarjan(i);
44         }
45     }
46     return 0;
47 }

```

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。

```

1  #include <bits/stdc++.h>
2  #define MAXN 111111
3  #define MAXL 11
4
5  using namespace std;
6  int n, q, dfs_clock, pts, root;
7  int fa[MAXN], dep[MAXN], siz[MAXN], son[MAXN];
8  int top[MAXN], dfn[MAXN], id[MAXN];
9  int l[MAXN << 2], r[MAXN << 2], lson[MAXN << 2], rson[MAXN << 2];
10 long long val[MAXN << 2], tag[MAXN << 2];
11 char cmd[MAXL];
12 vector<int> g[MAXN];
13
14 void dfs1(int u, int f, int d) {
15     fa[u] = f;
16     dep[u] = d;
17     siz[u] = 1;
18     for (auto &v : g[u]) {
19         if (v == f) {
20             continue;
21         }
22         dfs1(v, u, d + 1);
23         siz[u] += siz[v];
24         if (siz[son[u]] < siz[v] || !son[u]) {
25             son[u] = v;

```

```
26     }
27 }
28 }
29
30 void dfs2(int u, int f, int t) {
31     top[u] = t;
32     dfn[u] = ++dfs_clock;
33     id[dfs_clock] = u;
34     if (!son[u]) {
35         return;
36     }
37     dfs2(son[u], u, t);
38     for (auto &v : g[u]) {
39         if (v == f || v == son[u]) {
40             continue;
41         }
42         dfs2(v, u, v);
43     }
44 }
45
46 void pushup(int o) {
47     int ls = lson[o], rs = rson[o];
48     val[o] = val[ls] + val[rs];
49     l[o] = l[ls];
50     r[o] = r[rs];
51 }
52
53 void build(int o, int le, int ri) {
54     if (le == ri) {
55         lson[o] = rson[o] = tag[o] = -1;
56         l[o] = r[o] = le;
57         return;
58     }
59     int mid = (le + ri) >> 1;
60     lson[o] = pts++;
61     rson[o] = pts++;
62     build(lson[o], le, mid);
63     build(rson[o], mid + 1, ri);
64     pushup(o);
65 }
66
67 void pushdown(int o) {
68     int ls = lson[o], rs = rson[o];
69     val[ls] = tag[o] * (r[ls] - l[ls] + 1);
70     val[rs] = tag[o] * (r[rs] - l[rs] + 1);
71     tag[ls] = tag[o];
72     tag[rs] = tag[o];
73     tag[o] = -1;
```



```
74 }
75
76 void update(int o, int le, int ri, long long delta) {
77     if (le <= l[o] && r[o] <= ri) {
78         val[o] = delta * (r[o] - l[o] + 1);
79         tag[o] = delta;
80         return;
81     }
82     if (~tag[o]) {
83         pushdown(o);
84     }
85     int mid = (l[o] + r[o]) >> 1;
86     if (le <= mid) {
87         update(lson[o], le, ri, delta);
88     }
89     if (mid < ri) {
90         update(rson[o], le, ri, delta);
91     }
92     pushup(o);
93 }
94
95 long long query(int o, int le, int ri) {
96     if (le <= l[o] && r[o] <= ri) {
97         return val[o];
98     }
99     if (~tag[o]) {
100         pushdown(o);
101     }
102     int mid = (l[o] + r[o]) >> 1;
103     long long res = 0;
104     if (le <= mid) {
105         res += query(lson[o], le, ri);
106     }
107     if (mid < ri) {
108         res += query(rson[o], le, ri);
109     }
110     return res;
111 }
112
113 long long ask(int o) {
114     int u = top[o];
115     long long res = 0;
116     while (u) {
117         res += dfn[o] - dfn[u] - query(root, dfn[u], dfn[o]) + 1;
118         update(root, dfn[u], dfn[o], 1);
119         o = fa[u];
120         u = top[o];
121     }
```

```

122     res += dfn[o] - dfn[0] - query(root, dfn[0], dfn[o]) + 1;
123     update(root, dfn[0], dfn[o], 1);
124     return res;
125 }
126
127 int main() {
128     scanf("%d", &n);
129     for (int i = 1, x; i < n; ++i) {
130         scanf("%d", &x);
131         g[i].push_back(x);
132         g[x].push_back(i);
133     }
134     dfs1(0, -1, 1);
135     dfs2(0, -1, 0);
136     root = pts++;
137     build(root, 1, n);
138     for (scanf("%d", &q); q; --q) {
139         int x;
140         scanf("%s%d", cmd, &x);
141         if (*cmd == 'i') {
142             printf("%lld\n", ask(x));
143         }
144         else {
145             printf("%lld\n", query(root, dfn[x], dfn[x] + siz[x] - 1));
146             update(root, dfn[x], dfn[x] + siz[x] - 1, 0);
147         }
148     }
149     return 0;
150 }

```

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](#)。

```

1  #include <bits/stdc++.h>
2  #define MAXN 555
3  #define MAXM 555
4
5  using namespace std;
6  int n, m, c, p[MAXN + MAXM];
7  bool vis[MAXN + MAXM];
8  vector<int> g[MAXN];
9

```

```

10 bool hungary(int u) {
11     for (auto &v : g[u]) {
12         if (vis[v])
13             continue;
14         vis[v] = 1;
15         if (!p[v] || hungary(p[v])) {
16             p[v] = u;
17             return 1;
18         }
19     }
20     return 0;
21 }
22
23 int main() {
24     for (scanf("%d%d%d", &n, &m, &c); c; --c) {
25         int u, v;
26         scanf("%d%d", &u, &v);
27         g[u].push_back(v + n);
28     }
29     int ans = 0;
30     for (int i = 1; i <= n; ++i) {
31         memset(vis, 0, sizeof(vis));
32         hungary(i) && ++ans;
33     }
34     printf("%d\n", ans);
35     return 0;
36 }

```

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。

```

1  #include <bits/stdc++.h>
2  #define MAXN 22222
3  #define MAXD 11
4
5  using namespace std;
6  int n, siz_tot, siz_root, root, ans;
7  int siz_son[MAXN], siz[MAXN], dep[MAXN], cnt[MAXD];
8  bool vis[MAXN];
9  vector<pair<int, int> > g[MAXN];
10
11 void getroot(int u, int f) {

```

```
12     siz[u] = 1;
13     siz_son[u] = 0;
14     for (auto &e : g[u]) {
15         int v = e.second;
16         if (vis[v] || v == f) {
17             continue;
18         }
19         getroot(v, u);
20         siz[u] += siz[v];
21         siz_son[u] = max(siz_son[u], siz[v]);
22     }
23     siz_son[u] = max(siz_son[u], siz_tot - siz[u]);
24     if (siz_son[root] > siz_son[u]) {
25         root = u;
26     }
27 }
28
29 void getdep(int u, int f) {
30     ++cnt[dep[u]];
31     for (auto &e : g[u]) {
32         int v = e.second, w = e.first;
33         if (vis[v] || v == f) {
34             continue;
35         }
36         dep[v] = (dep[u] + w) % 3;
37         getdep(v, u);
38     }
39 }
40
41 int solve(int u, int w) {
42     cnt[0] = cnt[1] = cnt[2] = 0;
43     dep[u] = w;
44     getdep(u, 0);
45     return cnt[0] * cnt[0] + 2 * cnt[1] * cnt[2];
46 }
47
48 void dq(int u) {
49     ans += solve(u, 0);
50     vis[u] = 1;
51     for (auto &e : g[u]) {
52         int v = e.second, w = e.first;
53         if (vis[v]) {
54             continue;
55         }
56         ans -= solve(v, w);
57         root = 0;
58         siz_tot = siz[v];
59         getroot(v, 0);
```

```

60         dq(root);
61     }
62 }
63
64 int main() {
65     scanf("%d", &n);
66     for (int i = 1, x, y, w; i < n; ++i) {
67         scanf("%d%d%d", &x, &y, &w);
68         w %= 3;
69         g[x].push_back(make_pair(w, y));
70         g[y].push_back(make_pair(w, x));
71     }
72     siz_son[0] = siz_tot = siz_root = n;
73     getroot(1, 0);
74     dq(root);
75     int g = __gcd(ans, n * n);
76     printf("%d/%d\n", ans / g, n * n / g);
77     return 0;
78 }

```

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 [Oj Wiki](#)。

```

1  #include <bits/stdc++.h>
2  #define MAXN 250
3  #define INF 0x3f3f3f3f
4
5  using namespace std;
6
7  struct Dinic {
8      struct Edge {
9          int from, to;
10         long long cap, flow;
11         Edge(int u, int v, long long c, long long f) : from(u), to(v),
12         ↪ cap(c), flow(f) {}
13     };
14     int n, m, s, t;

```

```
15     bool vis[MAXN];
16     vector<Edge> edges;
17     vector<int> g[MAXN];
18     int d[MAXN], cur[MAXN];
19
20     void init(int n) {
21         for (int i = 0; i < n; ++i) {
22             g[i].clear();
23         }
24         edges.clear();
25     }
26
27     void insert(int from, int to, long long cap) {
28         edges.emplace_back(from, to, cap, 0);
29         edges.emplace_back(to, from, 0, 0);
30         m = edges.size();
31         g[from].push_back(m - 2);
32         g[to].push_back(m - 1);
33     }
34
35     bool bfs() {
36         memset(vis, 0, sizeof(vis));
37         queue<int> q;
38         d[s] = 0;
39         q.push(s);
40         vis[s] = 1;
41         while (!q.empty()) {
42             int u = q.front();
43             q.pop();
44             for (auto &x : g[u]) {
45                 Edge& e = edges[x];
46                 if (!vis[e.to] && e.cap > e.flow) {
47                     vis[e.to] = 1;
48                     d[e.to] = d[u] + 1;
49                     q.push(e.to);
50                 }
51             }
52         }
53         return vis[t];
54     }
55
56     int dfs(int u, long long c) {
57         if (u == t || !c) {
58             return c;
59         }
60         long long flow = 0, f;
61         for (int& i = cur[u]; i < (int)g[u].size(); ++i) {
62             Edge& e = edges[g[u][i]];
```

```

63         if (d[u] + 1 == d[e.to] && (f = dfs(e.to, min(c, e.cap -
↪ e.flow))) > 0) {
64             e.flow += f;
65             edges[g[u][i] ^ 1].flow -= f;
66             flow += f;
67             c -= f;
68             if (!c) {
69                 break;
70             }
71         }
72     }
73     return flow;
74 }
75
76 long long dinic(int s, int t) {
77     this->s = s;
78     this->t = t;
79     long long flow = 0;
80     while (bfs()) {
81         memset(cur, 0, sizeof(cur));
82         flow += dfs(s, INF);
83     }
84     return flow;
85 }
86 } andy;
87
88 int main() {
89     int m;
90     scanf("%d%d%d%d", &andy.n, &m, &andy.s, &andy.t);
91     for (int i = 1, u, v, c; i <= m; ++i) {
92         scanf("%d%d%d", &u, &v, &c);
93         andy.insert(u, v, c);
94     }
95     printf("%lld\n", andy.dinic(andy.s, andy.t));
96     return 0;
97 }

```

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。

```

1  #include <bits/stdc++.h>
2  #define MAXN 11111
3
4  using namespace std;
5  const int base = 131, mod = (int)1e9 + 727;
6  int n, ans, val[MAXN];
7  string s;
8
9  int main() {
10     cin >> n;
11     for (int i = 1; i <= n; ++i) {
12         cin >> s;
13         long long cur = 0;
14         for (auto &c : s) {
15             cur = (cur * base + c) % mod;
16         }
17         val[i] = cur;
18     }
19     sort(val + 1, val + n + 1);
20     for (int i = 1; i <= n; ++i) {
21         if (val[i - 1] != val[i]) {
22             ++ans;
23         }
24     }
25     cout << ans << endl;
26     return 0;
27 }

```

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](#)。

```

1  #include <bits/stdc++.h>
2  #define MAXN 1111111
3
4  using namespace std;
5  int n, m, fail[MAXN];

```



```
6 string a, b;
7
8 void getfail() {
9     int i = 0, j = -1;
10    fail[0] = -1;
11    while (i < m) {
12        if (j == -1 || b[i] == b[j]) {
13            fail[++i] = ++j;
14        }
15        else {
16            j = fail[j];
17        }
18    }
19 }
20
21 void kmp() {
22     int i = 0, j = 0;
23     while (i < n) {
24         if (j == -1 || a[i] == b[j]) {
25             ++i;
26             ++j;
27         }
28         else {
29             j = fail[j];
30         }
31         if (j == m) {
32             cout << i - m + 1 << endl;
33             j = fail[j];
34         }
35     }
36 }
37
38 int main() {
39     cin >> a >> b;
40     n = a.length();
41     m = b.length();
42     getfail();
43     kmp();
44     for (int i = 1; i <= m; ++i) {
45         cout << fail[i] << " \n"[i == m];
46     }
47     return 0;
48 }
```

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。

```
1  #include <bits/stdc++.h>
2  #define MAXN 555
3  #define MOD 998244353
4
5  using namespace std;
6  int n, q;
7  long long a[MAXN][MAXN], b[MAXN][MAXN];
8
9  long long xpow(long long a, long long b) {
10     long long res = 1;
11     while (b) {
12         if (b & 1) {
13             res = res * a % MOD;
14         }
15         a = a * a % MOD;
16         b >>= 1;
17     }
18     return res;
19 }
20
21 long long inv(long long o) {
22     return xpow(o, MOD - 2);
23 }
24
25 void gauss() {
26     for (int i = 1; i <= n; ++i) {
27         if (!a[i + 1][i]) {
28             bool flag = 0;
29             for (int j = i + 2; j <= n; ++j) {
30                 if (a[j][i]) {
31                     for (int k = i; k <= n; ++k) {
32                         swap(a[i + 1][k], a[j][k]);
33                     }
34                     for (int k = 1; k <= n; ++k) {
35                         swap(a[k][i + 1], a[k][j]);
```

```

36         }
37         flag = 0;
38         break;
39     }
40 }
41 if (flag) {
42     continue;
43 }
44 }
45 for (int j = i + 2; j <= n; ++j) {
46     long long cur = MOD - a[j][i] * inv(a[i + 1][i]) % MOD;
47     for (int k = i; k <= n; ++k) {
48         a[j][k] = (a[j][k] + cur * a[i + 1][k] % MOD) % MOD;
49     }
50     for (int k = 1; k <= n; ++k) {
51         a[k][i + 1] = (a[k][i + 1] + (MOD - cur) * a[k][j] % MOD)
↵ % MOD;
52     }
53 }
54 }
55 }
56
57 void charpoly() {
58     b[0][0] = 1;
59     for (int i = 1; i <= n; ++i) {
60         for (int j = 0; j < i; ++j) {
61             b[i][j] = (b[i][j] + (b[i - 1][j] * a[i][i]) % MOD) % MOD;
62             b[i][j + 1] = (b[i][j + 1] + (MOD - b[i - 1][j])) % MOD;
63         }
64         int cur = 1;
65         for (int j = i - 1; j >= 1; --j) {
66             cur = cur * a[j + 1][j] % MOD * (MOD - 1) % MOD;
67             int res = cur * a[j][i] % MOD;
68             for (int k = 0; k < j; ++k) {
69                 b[i][k] = (b[i][k] + (res * b[j - 1][k]) % MOD) % MOD;
70             }
71         }
72     }
73 }
74
75 int main() {
76     scanf("%d%d", &n, &q);
77     for (int i = 1; i <= n; ++i) {
78         for (int j = 1; j <= n; ++j) {
79             scanf("%lld", &a[i][j]);
80         }
81     }
82     gauss();

```

```
83     charpoly();
84     while (q--) {
85         long long x, y = 1, ans = 0;
86         scanf("%lld", &x);
87         for (int i = 0; i <= n; ++i) {
88             ans = (ans + (b[n][i] * y % MOD)) % MOD;
89             y = y * x % MOD;
90         }
91         printf("%lld%c", ans, " \n"[q == 0]);
92     }
93     return 0;
94 }
```

6 二分

6.1 符合条件的最大值

小于等于答案的值都合法。

```
1  bool check(int o) {
2      ...
3  }
4
5  int bs() {
6      int l, r, mid, ans;
7      while (l <= r) {
8          if (check(mid = (l + r) >> 1)) {
9              l = mid + 1;
10             ans = max(ans, mid);
11         }
12         else {
13             r = mid - 1;
14         }
15     }
16     return ans;
17 }
```

6.2 符合条件的最小值

大于等于答案的值都合法。

```
1  const int INF = ...;
2
3  bool check(int o) {
4      ...
5  }
```

```
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