矩阵树

矩阵树

无向图

矩阵形式：

$A(i,i)=\deg(i)$，$A(i,j)=-\operatorname{cnt}(i,j)$（i 到 j 的边权/重边数量）

有向图

矩阵形式：

- 内向树：$A_{out}(i,i)=\deg_{out}(i)$，$A(i,j)=-\operatorname{cnt}(i,j)$（i 到 j 的边权/重边数量）
- 外向树：$A_{in}(i,i)=\deg_{in}(i)$，$A(i,j)=-\operatorname{cnt}(i,j)$（i 到 j 的边权/重边数量）

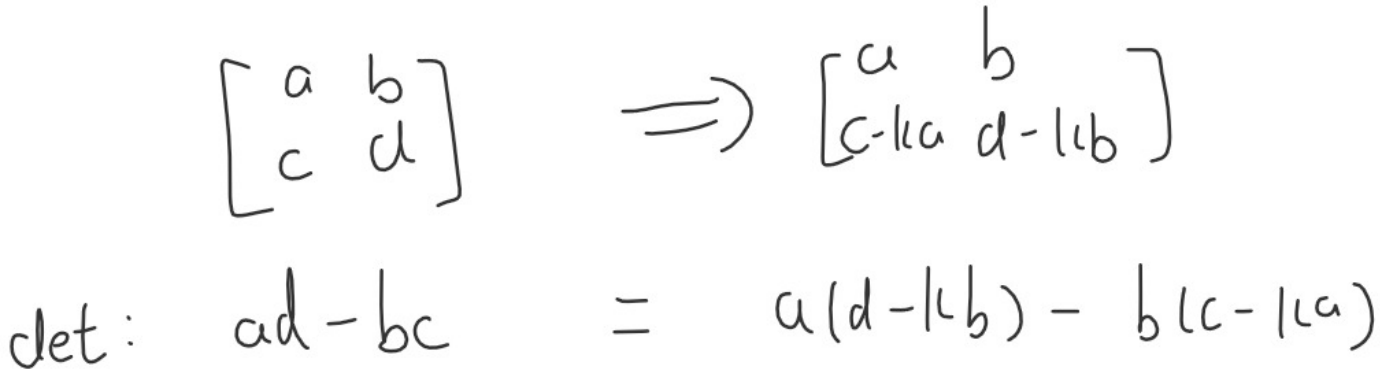
以 i 为根的生成树数量为 A 删去第 i 行以及第 i 列，之后求行列式。

行列式

定义式：

$$\det(A)=\sum_{\text{排列}P}(-1)^{\text{P 中逆序对数}}\prod_{i=1}^{n}a_{i,P(i)}$$

- 交换两行，结果取反
- 某行乘 k，结果乘 k
- 某行减去另一行乘一个系数，结果不变，直观理解：



trick

- dp 提前算贡献
- 分阶段 dp（寿司晚宴&皮配）
- 时光倒流（常用于从环、序列上删数，且过程与所删数当前的前驱、后继相关的时候，或者贪心的候选集合递减（蔬菜））

- 消消乐 dp
- 子树合并 dp 复杂度
- dp 变为 dag 路径计数（然后可以搞一些操作，比如在反图上跑）
- n 选 k 使得和最大，n 很大，k 很小，考虑排除没有可能的方案（CF1572D）
- 模拟最大流转模拟最小割
- 找不变量（交换差分，逆序对个数奇偶性....）
- prim 最小生成树
- 按余数分类（定长区间异或 1）
- 切糕模型
- 取关键点（字符串循环节/答案对应的长度不超过 B，并支持 n^2 计算，则可取若干关键区间计算 $[1,2B],[B,3B],[2B,4B]\cdots$）
- 二进制分组斜率优化（当斜率不递增时，可以分别开若干个大小为 $2^0,2^1,2^2\cdots$ 的单调栈，加的时候先在最小的里面加，一旦大小撑爆了就跟下一个单调栈合并）
- i 行 j 列转化为 i 行向 j 列连边。
- 在生成树上构造。
- 点减边容斥。求连通块个数转化为求包含每个点连通块个数-包含每个边连通块个数
- 线段树 tag 不会合并？考虑变成矩阵乘法。
- 把统计某个东西转化为统计图上的边
- 棋盘异或行列之和的奇偶数
- 问题转化成实时维护一个集合，并判断集合是否与某个目标集合相等，然后就可以哈希。
- 不仅有二进制分组，还可以随机分组，比如随机分k组就可以解决k个不同点
- 一个01序列，可以对其进行若干次奇怪操作（包含异或运算），最终要变成另一个序列，考虑先算出每个位置的操作奇偶性，然后再考虑构造1操作顺序
- 利用 Lucas 定理，$\binom{x}{y}\pmod 2=[y\subset x]$
- 如下的序列递推 $a_i=\max(b_i,a_i-1+c_i)$ 是可以维护的，只需要维护这段区间中顶过上界的答案和没有顶过上界的答案。
  - https://codeforces.com/gym/104076/problem/B
  - 某年 pkusc
- 比如说，求满足某个序关系的最小的点对，可以考虑是否有少数支配对，用类似单调栈的思想解决，例子：
  - https://codeforces.com/gym/104076/problem/L（在树上，经典套路！）
- 求不同颜色序列数的一些技巧：
  - https://www.luogu.com.cn/problem/AT_agc002_f

我的常犯错误

做题策略（惨痛教）

1. 开始把题全看一遍（15 min 左右）
2. 开始想题前手玩样例（5 min 以内可以玩的话）
3. 除非分数有保障，想题超过 30 min 考虑换换，不要有心理负担，暴力也能搞好多分
4. 思路较复杂，且不好调的题，重新回忆下算法，不一定是写错了，可能是假了。不要一心认为某nb做法是对的，其他做法是假的。
5.

STL

- `multimap` 的 `erase(x)` 会把所有值等于 x 的删掉

细节

- 建图把无向图建成有向图
- 组合数计算函数忘记特判 `x>y`，`x,y<0`，开 O2 后 UB 爆零了。。
- 预处理阶乘/扫描线分开存储修改查询时，数组忘 **开两倍**

数据结构

- 线段树着急写错左右儿子
- 值域线段树 `pushdown` 忘记把空节点新建出来

fhqTreap 专栏

- `pushdown` 时要 **判空节点**
- 按 `rank` 分裂时应该判断 `sz[ls]+1>=rank`
- `push_up` 时 `sz[p]=sz[ls]+sz[rs]+1` 忘记写 `+1`
- 把 `split` 复制粘贴改成 `splitrk` 忘记改递归的函数名
- `split(rt,b,c)` 之后应该 `split(b,a,b)`，写成 `split(rt,a,b)`

算法

- 求树的直径时忘记把父亲的跟儿子取 max
- 线段树合并忘记 **判断叶子节点**
- 点分治忘记考虑 **分治中心的贡献**
- 扫描线先改后查
- **莫队排序写挂**
- AC 自动机，在把 fail 树上子树加起来的时候，必须**显式建图**，倒着遍历所有节点，并加到自己的 father 上不对
- dinic 中 bfs 的 queue 忘记清零
- 写线性基形式的高斯消元时，消元这一步：

```
inline void eli(double *x,double *y,int ind){
    if(is0(x[ind]))return;
    double rate=x[ind]/y[ind];
    for(int i=0;i<=ind;i++)//第四行
        x[i]-=y[i]*rate;
}
```

第 4 行**不可以**循环到 n，否则会**被卡精度**。

其他

- 函数内开大数组 RE 了
- 对拍时要把 `std` 中的小数据暴力注释掉
- 编译不会开 O2 的命令 `g++ -O2 -Wall *.cpp -o *`

Tarjan 专栏

- 有向图建成无向图
- 缩完点后建图把原来的点和缩成的点搞混。。。
- `instack` 数组应该在弹栈时置零

判断条件

- 强连通：`dfn[p]==low[p]`，弹到 `p`。
- 点双：`low[nx]==dfn[p]`，则如果 `p` 不是搜索树的根，或者 `p` 在搜索树上有多个儿子，`p` 为割点，弹栈直到 `nx`（`nx` 要保留，因为割点可能在多个点双中）
- 边双：`dfn[p]==low[p]`，弹到 `p`，当前边为桥。

总是忘记的

- sa 中求 h 数组时：`h[rk[i]]>=h[rk[i-1]]-1`
- 2-SAT 中最终应该选择编号小的强连通分量

模板

hash.py

```
import hashlib
import sys
import re
fname = sys.argv[1]
f = open(fname, encoding="utf8")
fout = open(fname+".hash", "w", encoding="utf8")
for i in f.readlines():
    no_space = "".join(i.split())
```

```
    no_comment = no_space.split("//")[0]
    fout.write(f"{hashlib.sha256(no_comment.encode('utf8')).hexdigest()[:4]}
| {i}")
f.close()
fout.close()
```

rho

```
a2f9 | #include <bits/stdc++.h>
790b | using namespace std;
8f5d | typedef long long ll;
c83f | typedef long double ld;
d9e4 | inline ll qmul(ll x, ll y, ll mod) {
942d |     ll res = x * y - mod * ll((ld)x / mod * y);
00f3 |     if (res < 0) return res + mod;
571b |     if (res < mod) return res;
f26b |     return res - mod;
d10b | }
a02e | inline ll qpow(ll x, ll y, ll mod) {
ebfd |     ll res = 1;
4aed |     while (y) {
3685 |         if (y & 1) res = qmul(res, x, mod);
93b5 |         x = qmul(x, x, mod), y >>= 1;
d10b |     }
f4f0 |     return res;
d10b | }
70a7 | inline bool ispri(ll x) {
6645 |     if (x < 3) return x == 2;
a8d0 |     ll y = x - 1, h = 0;
0745 |     while (!(y & 1)) y >>= 1, h++;
4ea8 |     for (ll i = 0; i < 8; i++) {
f9d5 |         ll v = qpow(rand() % (x - 2) + 2, y, x), th = h;
6673 |         if (v == 1) continue;
2c61 |         while (th--) {
264a |             if (v == x - 1) break;
c0ae |             v = qmul(v, v, x);
d10b |         }
ad3d |         if (th == -1) return 0;
d10b |     }
31a0 |     return 1;
d10b | }
e3b0 |
6a7a | inline ll gcd(ll x, ll y) { return y ? gcd(y, x % y) : x; }
6c64 | inline ll f(ll x, ll c, ll mod) {
```

```
907f |     ll res = qmul(x, x, mod) + c;
8daf |     return res < mod ? res : res - mod;
d10b | }
e49c | inline ll rho(ll x) {
2940 |     ll c = rand() % (x - 1), s = 0;
3eb2 |     for (ll rg = 1;; rg <<= 1) {
df0b |         ll t = s, v = 1;
0b97 |         for (ll j = 1; j <= rg; j++) {
3e1a |             s = f(s, c, x);
e364 |             v = qmul(v, abs(s - t), x);
efe3 |             if (j % 127 == 0) {
ccb1 |                 ll g = gcd(v, x);
c0a0 |                 if (g > 1) return g;
d10b |             }
d10b |         }
cdfc |         ll g = gcd(s, x);
c0a0 |         if (g > 1) return g;
d10b |     }
d10b | }
7154 | inline void fact(ll x) {
206b |     if (x == 1) return;
ebd4 |     if (ispri(x)) {
1179 |         printf("%lld ", x);
8bea |         return;
d10b |     }
f030 |     ll p;
ac0b |     do
7154 |         p = rho(x);
3327 |     while (p == x);
0841 |     fact(x / p), fact(p);
d10b | }
80cc | int main(int cnt, char **va) {
e1a4 |     srand(time(0));
9e5b |     for (int i = 1; i < cnt; i++) {
824e |         ll x;
9a36 |         sscanf(va[i], "%lld", &x);
0376 |         printf("%lld:", x);
105a |         fact(x);
d728 |         putchar('\n');
d10b |     }
7145 |     return 0;
d10b | }
```

kd

```
e3b0 | // P4148 简单题
a2f9 | #include <bits/stdc++.h>
fe2f | #define sq(x) (x) * (x)
790b | using namespace std;
2ae2 | const double rate = 0.75;
12cc | const int MXN = 2e5 + 5;
2ffd | struct p {
6764 |     int x, y, v;
22e2 | } c[MXN];
80d4 | typedef int arrn[MXN];
6d44 | int rt, nodec, flatc;
993c | int n, ql, qr, qd, qu;
f8f8 | arrn ls, rs, L, R, D, U, dim, sz, sum, t;
8673 | inline void upd(int x, int y) {
1c83 |     sz[x] += sz[y], sum[x] += sum[y];
949b |     L[x] = min(L[x], L[y]);
d959 |     R[x] = max(R[x], R[y]);
6e26 |     D[x] = min(D[x], D[y]);
5624 |     U[x] = max(U[x], U[y]);
d10b | }
cbc7 | inline void pushu(int p) {
2460 |     sum[p] = c[p].v, sz[p] = 1;
e206 |     L[p] = R[p] = c[p].x;
bf0e |     D[p] = U[p] = c[p].y;
26b7 |     if (ls[p]) upd(p, ls[p]);
de48 |     if (rs[p]) upd(p, rs[p]);
d10b | }
e3b0 |
eb36 | inline bool cmpx(int x, int y) { return c[x].x < c[y].x; }
9ba8 | inline bool cmpy(int x, int y) { return c[x].y < c[y].y; }
4fe5 | inline int build(int l, int r) {
5895 |     if (l > r) return 0;
1326 |     double avx = 0, avy = 0, vax = 0, vay = 0;
064c |     for (int i = l; i <= r; i++) avx += c[t[i]].x, avy += c[t[i]].y;
47ad |     avx /= (r - l + 1), avy /= (r - l + 1);
1954 |     for (int i = l; i <= r; i++) vax += sq(avx - c[t[i]].x), vay +=
sq(avy - c[t[i]].y);
e3b0 |
46a6 |     int mid = (l + r) >> 1;
aaa5 |     if (vax > vay)
b301 |         dim[t[mid]] = 1, nth_element(t + l, t + mid, t + r + 1,
```

```
cmpx);
7dd5 |     else
cf75 |         dim[t[mid]] = 0, nth_element(t + l, t + mid, t + r + 1,
cmpy);
dcaf |     ls[t[mid]] = build(l, mid - 1), rs[t[mid]] = build(mid + 1, r);
07d2 |     return pushu(t[mid]), t[mid];
d10b | }
5c22 | inline void flat(int p) {
43d6 |     if (!p) return;
9dea |     flat(ls[p]);
aa5e |     t[++flatc] = p;
539e |     flat(rs[p]);
d10b | }
5f6f | inline void rebuild(int &p) {
ec7e |     flatc = 0;
7b87 |     flat(p);
4869 |     p = build(1, flatc);
d10b | }
07ad | inline bool balance(int p) { return rate * sz[p] >= max(sz[ls[p]],
sz[rs[p]]); }
5450 | inline void ins(int &p, int x) {
1ef4 |     if (!p) {
7af4 |         pushu(p = x);
8bea |         return;
d10b |     }
3470 |     if (dim[p]) {
ae20 |         if (c[x].x <= c[p].x)
1b8a |             ins(ls[p], x);
7dd5 |         else
ea82 |             ins(rs[p], x);
282f |     } else {
99cb |         if (c[x].y <= c[p].y)
1b8a |             ins(ls[p], x);
7dd5 |         else
ea82 |             ins(rs[p], x);
d10b |     }
4497 |     pushu(p);
493d |     if (!balance(p)) rebuild(p);
d10b | }
e64d | inline int que(int p) {
8296 |     if (!p || ql > R[p] || qr < L[p] || qd > U[p] || qu < D[p])
return 0;
c9a7 |     if (ql <= L[p] && qr >= R[p] && qd <= D[p] && qu >= U[p]) return
```

```
sum[p];
7c82 |     int res = que(ls[p]) + que(rs[p]);
22d5 |     if (ql <= c[p].x && qr >= c[p].x && qd <= c[p].y && qu >= c[p].y)
res += c[p].v;
f4f0 |     return res;
d10b | }
e3b0 |
565c | int main() {
e570 |     scanf("%*d");
d89a |     int last = 0, op;
cd1f |     while (scanf("%d", &op) != EOF) {
4f83 |         if (op == 1) {
f00b |             ++nodec;
5b58 |             scanf("%d%d%d", &c[nodec].x, &c[nodec].y, &c[nodec].v);
769f |             c[nodec].x ^= last, c[nodec].y ^= last, c[nodec].v ^=
last;
087b |             ins(rt, nodec);
86e4 |         } else if (op == 2) {
fb7e |             scanf("%d%d%d%d", &ql, &qd, &qr, &qu);
3f8b |             ql ^= last, qr ^= last, qu ^= last, qd ^= last;
e467 |             printf("%d\n", last = que(rt));
7a6d |         } else
42af |             break;
d10b |     }
7145 |     return 0;
d10b | }
```

dinic

```
a2f9 | #include <bits/stdc++.h>
e3b0 |
790b | using namespace std;
e3b0 |
e3b0 | // {{{ flow
e3b0 | // 原始版费用流
6798 | template <const int MXN, typename T = int>
f3fa | struct raw_flow {
bb6d |     const T INF = numeric_limits<T>::max();
e67e |     struct edge {
8664 |         int v, o;
9095 |         T c, w;
e9e5 |         edge(int _v, T _c, T _w, int _o) : v(_v), o(_o), c(_c), w(_w)
{}
df39 |     };
```

```
9054 |     vector<edge> g[MXN];
c98c |     queue<int> q;
98f2 |     int s, t, cure[MXN];
c1dd |     bool vis[MXN];
962d |     T dis[MXN];
2034 |     void addedge(int u, int v, T c, T w) {
8c8d |         g[u].push_back(edge(v, c, w, g[v].size()));
ecbb |         g[v].push_back(edge(u, 0, -w, g[u].size() - 1));
d10b |     }
fa43 |     void adduedge(int u, int v, T c) {
c26d |         g[u].push_back(edge(v, c, 1, g[v].size()));
9aab |         g[v].push_back(edge(u, 0, 1, g[u].size() - 1));
d10b |     }
f933 |     bool spfa() {
0050 |         for (int i = 0; i < MXN; i++) dis[i] = INF, cure[i] = 0;
94f2 |         dis[s] = 0;
dc94 |         q.push(s);
e1b4 |         while (!q.empty()) {
2abc |             int p = q.front();
d22c |             q.pop();
b5f3 |             vis[p] = 0;
99eb |             for (edge &nx : g[p])
2b43 |                 if (nx.c && dis[nx.v] > dis[p] + nx.w) {
a54f |                     dis[nx.v] = dis[p] + nx.w;
c0e0 |                     if (!vis[nx.v]) {
a7bf |                         vis[nx.v] = 1;
538e |                         q.push(nx.v);
d10b |                     }
d10b |                 }
d10b |         }
3e48 |         return dis[t] != INF;
d10b |     }
2403 |     T dinic(int p, T fi) {
b642 |         if (p == t) return fi;
e49f |         T fo = 0;
82e7 |         vis[p] = 1;
c2d6 |         for (int &i = cure[p]; i < (int)g[p].size(); i++) {
ba9a |             edge &nx = g[p][i];
a132 |             if (dis[nx.v] == dis[p] + nx.w && !vis[nx.v] && nx.c) {
eb12 |                 T delt = dinic(nx.v, min(fi - fo, nx.c));
7238 |                 if (delt) {
f447 |                     nx.c -= delt;
2350 |                     g[nx.v][nx.o].c += delt;
```

```
991d |                     fo += delt;
435a |                     if (fi == fo) return vis[p] = 0, fo;
7a6d |                 } else
b122 |                     dis[nx.v] = -1;
d10b |             }
d10b |         }
42e1 |         return vis[p] = 0, fo;
d10b |     }
a3a8 |     pair<T, T> run(int _s, int _t) {
a55e |         s = _s, t = _t;
9a6a |         pair<T, T> res = {0, 0};
9976 |         while (spfa()) {
85de |             T delt = dinic(s, INF);
3b0b |             res.first += delt, res.second += delt * dis[t];
d10b |         }
f4f0 |         return res;
d10b |     }
df39 | };
e3b0 | // 封装的上下界网络流
6798 | template <const int MXN, typename T = int>
0749 | struct lim_flow {
bb6d |     const T INF = numeric_limits<T>::max();
f04e |     raw_flow<MXN, T> f;
0fa7 |     T deg[MXN];
038f |     pair<T, T> res;
e3b0 |     // 加边函数 起点 终点 流量下界 流量上界 [是否有负环=false]
9676 |     void addedge(int u, int v, T l, T r, T w, bool cycle = 0) {
adf5 |         if (cycle && w < 0) {
aece |             w = -w;
abe1 |             swap(v, u), swap(l, r);
7de1 |             l = -l, r = -r;
d10b |         }
16f2 |         deg[u] -= l, deg[v] += l;
bb12 |         res.second += l * w;
fd82 |         f.addedge(u, v, r - l, w);
d10b |     }
e3b0 |     // 加单位边的函数（只求最大流，不求费用的时候用这个加边，跑的比较快）
5d87 |     void adduedge(int u, int v, T l, T r) {
16f2 |         deg[u] -= l, deg[v] += l;
bfc9 |         f.adduedge(u, v, r - l);
d10b |     }
e3b0 |     // 超级源点 超级汇点 源点 汇点 [选项=1]
e3b0 |     // 选项:
```

```
e3b0 |     // 0->最小费用可行流
e3b0 |     // 1->最小费用最大流
e3b0 |     // 2->最小费用最小流
e3b0 |     // 返回值 {流量，费用} 如果没有可行流返回 {-1,-1}
fa21 |     pair<T, T> run(int super_s, int super_t, int s, int t, int opt =
1) {
57a0 |         T all = 0;
cd9d |         for (int i = 0; i < MXN; i++) {
a95e |             if (deg[i] > 0)
06da |                 f.addedge(super_s, i, deg[i], 0), all += deg[i];
4ee7 |             else if (deg[i] < 0)
d229 |                 f.addedge(i, super_t, -deg[i], 0);
d10b |         }
b5fc |         f.addedge(t, s, INF, 0);
c60e |         pair<T, T> tres = f.run(super_s, super_t);
3e6d |         if (tres.first != all) return {-1, -1};
ad9b |         res.second += tres.second;
6759 |         res.first += f.g[s].back().c;
f867 |         f.g[s].back().c = 0;
d30e |         f.g[t].back().c = 0;
1f71 |         if (opt == 1) {
0911 |             tres = f.run(s, t);
18a1 |             res.first += tres.first, res.second += tres.second;
9551 |         } else if (opt == 2) {
1696 |             tres = f.run(t, s);
cb3e |             res.first -= tres.first, res.second += tres.second;
d10b |         }
f4f0 |         return res;
d10b |     }
df39 | };
e3b0 | // }}}
e3b0 |
565c | int main() {
7145 |     return 0;
d10b | }
```

sa

```
a2f9 | #include <bits/stdc++.h>
790b | using namespace std;
4aeb | const int MXN = 2e5 + 5, LG = 31 - __builtin_clz(MXN);
4c1c | int t, n, tot;
2d9a | char str[MXN];
e3b0 |
```

```
87fe | namespace SA {
80d4 | typedef int arrn[MXN];
614b | arrn sa, rk, tmp, ork, cnt;
69ad | int h[LG + 1][MXN];
275e | inline bool cmp(int x, int y, int w) { return ork[x] == ork[y] &&
ork[x + w] == ork[y + w]; }
2c45 | template <typename T>
bad5 | inline void init(int n, int m, T *arr) {
d837 |     for (int i = 1; i <= m; i++) cnt[i] = 0;
d916 |     for (int i = 1; i <= n; i++) cnt[rk[i] = arr[i]]++;
85ac |     for (int i = 1; i <= m; i++) cnt[i] += cnt[i - 1];
305b |     for (int i = n; i; i--) sa[cnt[rk[i]]--] = i;
0922 |     for (int w = 1; w <= n; w <<= 1) {
a7ee |         int ind = 0;
c7a8 |         for (int i = n - w + 1; i <= n; i++) tmp[++ind] = i;
fd5c |         for (int i = 1; i <= n; i++)
c2c2 |             if (sa[i] > w) tmp[++ind] = sa[i] - w;
d837 |         for (int i = 1; i <= m; i++) cnt[i] = 0;
1df0 |         for (int i = 1; i <= n; i++) cnt[rk[i]]++;
85ac |         for (int i = 1; i <= m; i++) cnt[i] += cnt[i - 1];
5061 |         for (int i = n; i; i--) sa[cnt[rk[tmp[i]]]--] = tmp[i],
ork[i] = rk[i];
7b65 |         m = 0;
5aea |         for (int i = 1; i <= n; i++) rk[sa[i]] = cmp(sa[i], sa[i -
1], w) ? m : ++m;
e453 |         if (m == n) break;
d10b |     }
e3b0 |
d835 |     arr[n + 1] = -1;
0bb4 |     for (int i = 1, lcp = 0; i <= n; i++) {
4869 |         lcp -= !!lcp;
95ab |         while (arr[i + lcp] == arr[sa[rk[i] - 1] + lcp]) ++lcp;
c363 |         h[0][rk[i]] = lcp;
d10b |     }
aad6 |     for (int i = 1; i <= LG; i++)
7cac |         for (int w = 1 << (i - 1), j = n - (1 << i) + 1; j > 0; j--)
h[i][j] = min(h[i - 1][j], h[i - 1][j + w]);
d10b | }
4a44 | inline int lcp(int x, int y) {
5069 |     x = rk[x], y = rk[y];
f144 |     assert(x != y);
0c62 |     if (x > y) swap(x, y);
557a |     ++x;
```

```
1062 |     int lg = 31 - __builtin_clz(y - x + 1);
7eb9 |     return min(h[lg][x], h[lg][y - (1 << lg) + 1]);
d10b | }
d10b | } // namespace SA
e3b0 |
565c | int main() {
2bf3 |     scanf("%d", &t);
73d1 |     while (t--) {
33c4 |         scanf("%d", &n);
8a9b |         tot = n * 2 + 2;
5c1f |         for (int i = 1; i <= n; i++) {
31e3 |             char x;
ac0b |             do
82c1 |                 x = getchar();
18ee |             while (x != 'a' && x != 'b');
adf6 |             str[i] = str[tot - i] = x;
d10b |         }
bdde |         str[n + 1] = '#';
8ec9 |         str[tot] = 0;
abe0 |         SA::init(tot - 1, 130, str);
83bd |         int ans = 0;
fd5c |         for (int i = 1; i <= n; i++)
bac4 |             for (int j = i << 1; j <= n; j += i)
82a7 |                 ans = max(ans, (SA::lcp(tot - j, tot - j + i) +
SA::lcp(j, j - i) + i - 1) / i);
d054 |         printf("%d\n", ans);
d10b |     }
7145 |     return 0;
d10b | }
```

圆方树

```
a2f9 | #include <bits/stdc++.h>
790b | using namespace std;
8f5d | typedef long long ll;
4f5e | const ll MXN = 3e5 + 5;
2451 | ll n, m;
07fb | vector<ll> g[MXN], t[MXN];
c5bc | void ae(vector<ll> *_g, ll u, ll v) {
2a3c |     _g[u].push_back(v);
6205 |     _g[v].push_back(u);
d10b | }
ffa6 | ll dfn[MXN], low[MXN], dfnc, sqrc;
9f8f | stack<ll> stk;
```

```
ff72 | void tj(ll p) {
b613 |     dfn[p] = low[p] = ++dfnc;
99c2 |     stk.push(p);
5cea |     for (ll nx : g[p]) {
fc99 |         if (!dfn[nx]) {
9d42 |             tj(nx);
5889 |             low[p] = min(low[nx], low[p]);
351b |             if (low[nx] == dfn[p]) {
824e |                 ll x;
42f1 |                 ++sqrc;
f774 |                 do {
1ff9 |                     x = stk.top();
383f |                     stk.pop();
8c49 |                     ae(t, x, sqrc);
bc7f |                 } while (x != nx);
6468 |                 ae(t, p, sqrc);
d10b |             }
7a6d |         } else
4cff |             low[p] = min(low[p], dfn[nx]);
d10b |     }
d10b | }
c1dd | bool vis[MXN];
eaca | ll sz[MXN], ans;
3163 | void dfssz(ll p, ll fa) {
eedd |     sz[p] = p <= n;
2d9b |     for (ll nx : t[p])
cccf |         if (nx != fa) {
0337 |             dfssz(nx, p);
37cc |             sz[p] += sz[nx];
d10b |         }
d10b | }
a8b3 | ll sqr(ll x) { return x * (x - 1); }
fca1 | void cal(ll p, ll fa, ll tot) {
82e7 |     vis[p] = 1;
92f4 |     ll curw = (p <= n ? -1 : t[p].size()), cnt = sqr(tot) - sqr(tot -
sz[p]);
2d9b |     for (ll nx : t[p])
cccf |         if (nx != fa) {
82b3 |             cnt -= sqr(sz[nx]);
759e |             cal(nx, p, tot);
d10b |         }
2369 |     ans += cnt * curw;
d10b | }
```

```
565c | int main() {
2f87 |     scanf("%lld%lld", &n, &m);
c8c0 |     sqrc = n;
76d2 |     while (m--) {
f6e3 |         ll u, v;
d759 |         scanf("%lld%lld", &u, &v);
e6c6 |         ae(g, u, v);
d10b |     }
029e |     for (ll i = 1; i <= n; i++)
8944 |         if (!dfn[i]) tj(i);
ee2c |     for (ll i = 1; i <= sqrc; i++)
3f61 |         if (!vis[i]) {
18f9 |             dfssz(i, 0);
1d95 |             cal(i, 0, sz[i]);
d10b |         }
57e2 |     printf("%lld", ans);
e3b0 |
7145 |     return 0;
d10b | }
```

正常 tarjan

```
e3b0 | // P3387 【模板】缩点
a2f9 | #include <bits/stdc++.h>
790b | using namespace std;
0c05 | const int MXN = 1e4 + 5;
5a2b | vector<int> e[MXN], ne[MXN];
1328 | int n, m, arr[MXN], dp[MXN];
6809 | int col[MXN], tot[MXN], colc;
4b91 | int dfn[MXN], low[MXN], dfsc;
ac7d | bool instk[MXN];
1b0a | stack<int> st;
3377 | inline void tj(int p) {
1ac8 |     dfn[p] = low[p] = ++dfsc;
80cd |     st.push(p), instk[p] = 1;
5ea7 |     for (int nx : e[p]) {
9c9c |         if (!dfn[nx])
fb1e |             tj(nx), low[p] = min(low[p], low[nx]);
11a6 |         else if (instk[nx])
4cff |             low[p] = min(low[p], dfn[nx]);
d10b |     }
e3b0 |
a001 |     if (dfn[p] == low[p]) {
6b91 |         int x;
```

```
a707 |         ++colc;
f774 |         do {
54c8 |             x = st.top();
3438 |             st.pop();
e635 |             col[x] = colc, instk[x] = 0;
b0f8 |             tot[colc] += arr[x];
cf44 |         } while (x != p);
d10b |     }
d10b | }
4f94 | inline int dfs(int p) {
b492 |     if (~dp[p]) return dp[p];
65a9 |     dp[p] = tot[p];
4f59 |     for (int nx : ne[p]) dp[p] = max(dp[p], dfs(nx) + tot[p]);
e767 |     return dp[p];
d10b | }
e3b0 |
565c | int main() {
81dd |     scanf("%d%d", &n, &m);
05f2 |     for (int i = 1; i <= n; i++) scanf("%d", arr + i);
a6a1 |     for (int i = 1, ts, tt; i <= m; i++) {
a58c |         scanf("%d%d", &ts, &tt);
9446 |         e[ts].push_back(tt);
d10b |     }
fd5c |     for (int i = 1; i <= n; i++)
8944 |         if (!dfn[i]) tj(i);
fd5c |     for (int i = 1; i <= n; i++)
fed9 |         for (int nx : e[i])
fbba |             if (col[nx] != col[i]) ne[col[nx]].push_back(col[i]);
55e8 |     memset(dp, -1, sizeof(dp));
83bd |     int ans = 0;
484b |     for (int i = 1; i <= colc; i++) ans = max(ans, dfs(i));
35cb |     printf("%d", ans);
7145 |     return 0;
d10b | }
```

树哈希

这类方法需要一个多重集的哈希函数。以某个结点为根的子树的哈希值，就是以它的所有儿子为根的子树的哈希值构成的多重集的哈希值，即：

$$h_x = f(\{h_i \mid i \in son(x)\})$$

其中 h_x 表示以 x 为根的子树的哈希值，f 是多重集的哈希函数。

以代码中使用的哈希函数为例：

$$f(S) = \left(c + \sum_{x \in S} g(x) \right) \bmod m$$

其中 c 为常数，一般使用 1 即可。m 为模数，一般使用 2^{32} 或 2^{64} 进行自然溢出，也可使用大素数。g 为整数到整数的映射，代码中使用 xor shift，也可以选用其他的函数，但是不建议使用多项式。为了预防出题人对着 xor hash 卡，还可以在映射前后异或一个随机常数。

这种哈希十分好写。如果需要换根，第二次 DP 时只需把子树哈希减掉即可。

```
ull shift(ull x) {
  x ^= mask;
  x ^= x << 13;
  x ^= x >> 7;
  x ^= x << 17;
  x ^= mask;
  return x;
}
```

李超树

```
a2f9 | #include <bits/stdc++.h>
990c | #define fi first
1eee | #define se second
a4fb | #define mp make_pair
790b | using namespace std;
8f5d | typedef long long ll;
c38c | typedef pair<ll, ll> pi;
00a9 | const ll INF = 1e18;
4f5e | const ll MXN = 3e5 + 5;
e3b0 |
890a | inline long double inter(const pi &x, const pi &y) {
50b0 |     return x.fi == y.fi ? (x.se > y.se ? -INF : INF) : (long double)
(y.se - x.se) / (x.fi - y.fi);
d10b | }
e3b0 |
db9e | ll n, s, sc[MXN], st[MXN], dp[MXN];
5633 | pi q[MXN];
25b9 | ll ql = 1, qr;
565c | int main() {
94a1 |     scanf("%lld%lld", &n, &s);
5c1f |     for (int i = 1; i <= n; i++) {
d822 |         scanf("%lld%lld", st + i, sc + i);
```

```
51a2 |         st[i] += st[i - 1], sc[i] += sc[i - 1];
d10b |     }
6566 |     q[++qr] = mp(0, s * sc[n]);
5c1f |     for (int i = 1; i <= n; i++) {
b8a0 |         ll l = ql, r = qr;
1cb8 |         while (l < r) {
bbce |             ll mid = (l + r) >> 1;
7fe1 |             if (inter(q[mid], q[mid + 1]) >= st[i])
5300 |                 r = mid;
7dd5 |             else
8091 |                 l = mid + 1;
d10b |         }
d70c |         dp[i] = q[l].fi * st[i] + q[l].se + st[i] * sc[i];
6ab6 |         pi curseg = mp(-sc[i], dp[i] + s * (sc[n] - sc[i]));
2645 |         while (ql < qr && inter(q[qr - 1], q[qr]) >= inter(q[qr],
curseg)) qr--;
a4bc |         q[++qr] = curseg;
d10b |     }
9089 |     printf("%lld", dp[n]);
e3b0 |
7145 |     return 0;
d10b | }
```