South China University of Technology

SCUT_gugugu

MODEL

Last build at October 25, 2020

Contents

1 Graph Theory

1.1 Shortest Path

1.2 Network Flow

修订稿 Created By ChrisJaunes 最大权闭合子图建模 eg: (+++) - (—)
定义：在一个有向图中，每个点都有一个点权。闭合子图：对于这个子图，它任意一个点的的后继必须在这个子图中。最大权闭合子图：在所有的闭合子图中，该图的点权和最大。
建模：
1. 建立超级源 S 和超级汇 T
2. 把所有点权为正的点与 S 连接一条有向边，方向是从 S 到 u，边权是点权；把所有点权为负的点与 T 连接一条有向边，方向是从 u 到 T，边权是点权的相反数；
3. 原图中所有有向边的边权是 INT_MAX；也可以根据情况按照割的思想处理。
答案：最大权闭合子图的权值和就是所有点的点权和-该生成图的最小割

独立集: 一个点集，点集中的各点没有关系。
最大独立集: 点的个数最多的独立集。
求解一般无向图的最大团、最大独立集是 NPC 问题。
二分图最大独立集 = 点的总数 - 最小点覆盖。
最大点权独立集的点权和 = 所有点权和-最小点权覆盖的点权
路径覆盖：给定有向图 G=(V,E)。设 P 是 G 的一个简单路 (顶点不相交) 的集合。如果 V 中每个定点恰好在 P 的一条路上，则称 P 是 G 的一个路径覆盖。
最小路径覆盖：G 的最小路径覆盖是 G 所含路径条数最少的路径覆盖
建模：拆点，对于原图的 (u,v), 连接 (u->v')
答案：原图的结点数-新图的最大匹配数

最小割的可行边和必须边 (所有割集的交集和并集)
注意：必须边 可行边.
1. 残余网络中有剩余流量的边一定不在最小割中
2. 残余网络中一条满流边的首尾还能相互到达，那么这条边不是可行边
3. 残余网络中一条满流的首尾分别与 S 和 T 在一个强连通分量中，那么这条边为必须边
具体实现, 需要先跑最大流, 然后 Tarjan 缩强连通分量, 条件是:
可行边: 两端不在一个强连通分量内。
必须边: 一端在 S 的分量内, 另一端在 T 的分量内。
必须边对于小规模也可以强行割去然后跑流判断流是否减少，暴力对费用流必须边同样适用

求割边最少的最小割: 把边的权值全部乘以一个较大的数 E 再加 1, E 要严格大于边的数量
最小割: ans/E;
最小割下的最小割边: ans % E
公平分配模型：
题意：n 个球队，已经有一些胜负场，现在还有一些场次，你去分配胜负，问每支球队有没有可能获胜
解：
1. 枚举每一个人，贪心计算比赛最大获胜次数 max_win

2. 将比赛当作节点，从源到比赛连比赛次数边
3. 分配比赛胜利者，从比赛到人连无穷边
4. 从人到汇连 max_win - 已经获胜的比赛次数的边
5. 检查是否能将比赛全部安排完

混合图重定向为欧拉回路
1. 将所有的无向边随便定向，计算得到每个点的出度与入度之差 deg
2. 建立源点汇点 S,T
3. 对于 deg>0 的点 v，连一条边 Su, 流为 deg/2;
4. 对于 deg<0 的点 u，连一条边 uT，流为-deg/2;
5. 对于有向边，忽略；对于无向边 (u,v)，连 (u,v,1)
5. 判断是否能跑满流

费用流：物品通过代价重复使用
1. 拆点，使用物品前和使用物品后
2. 对于节点 [使用物品前]：考虑物品来源: 源点购买能力、购买费用、节点 [使用物品后]INF、购买费用; 去向: 汇点使用量、0
3. 对于节点 [使用物品后]：考虑物品来源: 源点使用量、0，去向: 节点 [使用物品前]

费用流：物品通过代价延后销售
1. 拆点，生产物品和出售物品
2. 对于节点 [生产物品]：考虑物品来源: 源点生产能力、生产费用; 去向: 节点 [出售物品]INF、保存的代价
3. 对于节点 [出售物品]：考虑物品来源: 通过节点生产物品，去向: 汇点销售能力、销售费用

费用流：n 个点 m 条带权有向边的图，要给边染色，染色的边形成若干个回路且每个点都恰好属于其中 k 个回路。问最少要染多少边权和的路。
一个回路里面各个点的入度 = 出度 =1，各个点如果都恰好属于 k 个回路那么各个点的入度 = 出度 =k。
这样就考虑用最小费用最大流了：
1. 所有点 u 拆成两点 u 和 u'，分别代表出度和入度
2. 原点向 u 连容量 k 费用 0 的边，u' 向汇点连容量 k 费用 0 的边
3. 所有有向边 <u,v>，u 向 v' 连容量 1 费用边权的边
4. 这样跑最小费用最大流，如果最大流等于 n*k，也就是说各个点的出度 = 入度 =k，那么就有解，最小费用就是最小的解

费用流：费用不是线性关系
费用为 $a * x^2$：拆边，拆成 a * 1, a * 3, a * 5, ...
费用为之前的等待时间 + a：反向考虑，倒数第 i 个点对整体贡献为 a * i, 拆点，拆成 a, a*2, a*3, ...

费用流：平面上给定一些黑点白点，要求一个黑点连接一个白点，并且所有线段都不相交
合法情况的连线，总长度明显比不合法情况的小，题意可以转化为求让每条线段的长度和最小的方案
1、建立超级源超级汇
2、从超级源向左侧点连流量为 1，费用为 0 的边
3、从左侧点向右侧点连流量为 1，费用为两点欧几里得距离的边

4、从右侧点向超级汇连流量为 1，费用为 0 的边
5、跑最小费用最大流

区间 k 覆盖问题：给定 n 个带权开区间，区间最多可以选一次，要求保证实轴上的所有数被选择的次数要小于 k 次，问最大的权值。
1. 建立超级源超级汇
2. 源点向第一个点连一条容量为 k、费用为 0 的边
3. 相邻节点连上用容量为 k 费用为 0 的边
4. 最后一个点向超级汇连一条容量为 k、费用为 0 的边
5. 区间的两个端点之间连接一条容量为 1 的，费用为 w 的边

区间至少 ai 覆盖问题
1. 建立超级源超级汇
2. 源点向第一个点连一条容量为 INF、费用为 0 的边
3. 相邻节点连上用容量为 INF-ai 费用为 0 的边
4. 最后一个点向超级汇连一条容量为 INF、费用为 0 的边
5. 区间的两个端点之间连接一条容量为 INF 的，费用为 c 的边

分治优化建图
1. 费用为 |ai-aj| 可以利用差分性质通过累积权值之间的差值实现费用的绝对值
这样可以采用分治方式将边的数量从 n*n 将为 n*logn
2. 一个点向一个区间的每一个点连边，可以利用线段树转为每个点最多向 log 个虚点连边
线段树上的点向孩子节点连边 INF，叶子节点向孩子连边 INF

1.3 Tree Related

1.3.1 Kruskal 重构树

构造

除了连边，其他过程都和 Kruskal 最小生成树算法完全一样。
当我们需要在 u, v 之间连一条权值为 w 的边时，Kruskal 重构树算法是这样实现的：
新建节点 x，将 x 的点权设为 w。
设 u, v 所属集合分别为 S_u 和 S_v，那么连边 (x, S_u) 和 (x, S_v)，此处连有向边即可，注意没有边权！
将 u 和 v 所属集合都改为 x。
其余过程直接套用 Kruskal 即可！
重构树的根应该为 $2n-1$，也就是最后一个点。原因为：如果以 $1 \sim n$ 中的任何一个点作为根会破坏重构树的形态；连边时向下连边，只有以 $2n-1$ 为根时才能遍历整棵树。

性质

最后形成一棵有 $2n-1$ 个节点的树。
我们需要将 n 个原来的点最后放到同一个集合中，那么需要进行 $n-1$ 次合并。每次合并都会新建 1 个节点和 2 条无向边。那么一共会有 $2n-1$ 个点和 $2n-2$ 条边。

重构树中的叶子节点为原树中的节点，其余每个节点代表了一条边。
我们每次将 x 和下方的 u,v 连边，这个证明很显然吧！
原树 u 到 v 路径上的边权最大值为重构树上 u 和 v 的 LCA 的点权。
根据 Kruskal 的过程，我们把边按照权值从小到大排序，那么对于所有非叶子节点，它的点权（原图的边权）一定不大于父亲节点的点权，所以路径上的最大值即为 LCA 的点权。

Example: bzoj3551Peaks 加强版
有 N 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 M 条路径，每条路径有一个困难值，这个值越大表示越难走，现在有 Q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出-1。强制在线

```
// 3551Peaks加强版.cpp
#include <bits/stdc++.h>
using namespace std;
inline int read() {
    int x = 0, ch = getchar(); bool fg = 1;
    while(ch < '0' || ch > '9') { if(ch == '-') fg = 0; ch = getchar(); }
    while(ch >= '0' && ch <= '9') { x = x * 10 + ch - '0'; ch = getchar(); }
    return fg ? x : -x;
}
typedef pair<int, int> P;
const int MAXN = 2e5 + 5, S = 17;
const int MAXE = 5e5 + 5;
pair<int, P> e[MAXE];
vector<int> G[MAXN];
inline void addEdge(int u, int v) {
    G[u].push_back(v);
}
namespace uset{
    vector<int> fa(MAXN);
    void init(int n) {
        for(int i = 0; i <= n; i++) fa[i] = i;
    }
    int Find(int x) { return fa[x] == x ? x : fa[x] = Find(fa[x]); }
    void unite(int x, int y, int z) {
        fa[Find(x)] = z;
        fa[Find(y)] = z;
    }
}
int fa[MAXN][21], val[MAXN], h[MAXN];
namespace Tree {
    struct Node {
        int l, r, s;
    }t[MAXN * 40];
    int cnt;
    vector<int>rt(MAXN, 0);
    void update(int pre, int &x, int L, int R, int v) {
        t[x = ++cnt] = t[pre]; t[x].s++;
        if(L == R) return;
        int mid = (L + R) >> 1;
        if(v <= mid) update(t[pre].l, t[x].l, L, mid, v);
        else update(t[pre].r, t[x].r, mid + 1, R, v);
    }
    int query(int x, int y, int L, int R, int k) {
        if(L == R) return L;
        if(t[y].s - t[x].s < k) return -1;
        int mid = (L + R) >> 1;
        int sum = t[t[y].r].s - t[t[x].r].s;
```

```
        if(sum >= k) return query(t[x].r, t[y].r, mid + 1, R, k);
        return query(t[x].l, t[y].l, L, mid, k - sum);
    }
}
int num[MAXN], cnt_num;
int last[MAXN], dfs_val[MAXN];
void dfs(int x) {
    dfs_val[num[x] = ++cnt_num] = h[x];
    for(auto y : G[x]) {
        dfs(y);
    }
    last[x] = cnt_num;
}
int up(int x, int y) {
    for(int i = S; i >= 0; i--) {
        if(fa[x][i] && val[fa[x][i]] <= y) x = fa[x][i];
    }
    return x;
}
int main() {
    int n = read(), m = read(), Q = read();
    uset::init(n << 1);
    for(int i = 1; i <= n; i++) cin >> h[i];
    for(int i = 1, u, v, w; i <= m; i++) {
        u = read(), v = read(), w = read();
        e[i] = make_pair(w, P(u, v));
    }
    sort(e + 1, e + m + 1);
    for(int i = 1, x, y; i <= m; i++) {
        P cur = e[i].second;
        if((x = uset::Find(cur.first)) == (y = uset::Find(cur.second))) continue;
        val[++n] = e[i].first;
        fa[x][0] = fa[y][0] = n;
        uset::unite(x, y, n);
        addEdge(n, x); addEdge(n, y);
    }
    for(int k = 1; k <= S; k++) {
        for(int i = 1; i <= n; i++) {
            fa[i][k] = fa[fa[i][k - 1]][k - 1];
        }
    }
    dfs(n);
    for(int i = 1; i <= cnt_num; i++) Tree::update(Tree::rt[i - 1], Tree::rt[i], 0, 1e9
    + 1, dfs_val[i]);
    int a, b, c, ans = -1;
    while(Q--) {
        a = read(), b = read(), c = read();
        if(~ans) {
            a ^= ans; b ^= ans; c ^= ans;
        }
        int root = up(a, b);
        ans = Tree::query(Tree::rt[num[root] - 1], Tree::rt[last[root]], 0, 1e9 + 1, c);
        if(!ans || ans == 1e9 + 1) ans = -1;
        printf("%d\n", ans);
    }
    return 0;
}
```

1.4 LCA

尚未开始

1.5 Tarjan

尚未开始

1.6 Cactus

Tarjan 缩点后，每个 BCC 内的点为圆点，连到同一个方点。

Example:bzoj2125 最短路

仙人掌上 q 次询问两点最短路径

做法：加边权。如果一条边是圆圆边，那么就是原来的边权，如果是圆方边，那么边权等于环的根到那个圆点的最短路径长度。现在我们要询问两个点的最短路，我们查一下 lca，如果是圆点，那么这个最短路就是树上的长度。如果是方点，我们倍增跳到这个环上，在每个环上记一下前缀和，判一下走哪一侧就行了。

```cpp
// 2125最短路.cpp
#include <bits/stdc++.h>
using namespace std;
inline int read() {
    int x = 0, ch = getchar(); bool fg = 1;
    while(ch < '0' || ch > '9') { if(ch == '-') fg = 0; ch = getchar(); }
    while(ch >= '0' && ch <= '9') { x = x * 10 + ch - '0'; ch = getchar(); }
    return fg ? x : -x;
}
typedef pair<int, int> P;
const int MAXN = 2e4 + 5;
const int S = 15;
namespace Tree {
    struct Edge {
        int to, nxt, w;
    }e[MAXN << 1];
    int ecnt, head[MAXN];
    int rt, isrt[MAXN], fa[MAXN][S + 3];
    int sz[MAXN];
    inline void addEdge(int u, int v, int w) {
        e[++ecnt] = (Edge) {v, head[u], w}; head[u] = ecnt;
        fa[v][0] = u;
    }
}
int n, m, Q;
namespace BCC{
    struct Edge {
        int to, nxt, w;
    }e[MAXN << 1];
    int ecnt, head[MAXN];
    int dfs_clock, dfn[MAXN], low[MAXN];
    int is_vertex[MAXN], vbcc_cnt, vbccno[MAXN];
    vector<P> vbcc[MAXN];
    stack<P> vs;
    int tag[MAXN];
    inline void addEdge(int u, int v, int w) {
        e[++ecnt] = (Edge) {v, head[u], w}; head[u] = ecnt;
        e[++ecnt] = (Edge) {u, head[v], w}; head[v] = ecnt;
```

```
    }
    inline void init(int n) {
        ecnt = 1;
        dfs_clock = 0;
        vbcc_cnt = 0;
        for(int i = 0; i <= 2 * n; i++){
            head[i] = dfn[i] = low[i] = 0;
            vbccno[i] = 0;
            tag[i] = 0;
        }
        while(!vs.empty()) vs.pop();
    }
    //root's edge = -1;
    void tarjan(int u, int edge) {
        dfn[u] = low[u] = ++dfs_clock;
        vs.push(P(u, e[edge ^ 1].w));
        for(int i = head[u], v; i; i = e[i].nxt) {
            if(!dfn[v = e[i].to]) {
                tarjan(v, i ^ 1);
                low[u] = min(low[u], low[v]);
                if(low[v] >= dfn[u]) {
                    if(vs.top().first == v) {
                        Tree::addEdge(u, v, vs.top().second);
                        vs.pop();
                        continue;
                    }
                    vbcc[++vbcc_cnt].clear();
                    vbcc[vbcc_cnt].push_back(P(u, 0));
                    Tree::isrt[u] = 1;
                    int &sz = Tree::sz[n + vbcc_cnt];
                    tag[vs.top().first] = n + vbcc_cnt;
                    //Tree::addEdge(u, rt, 0);
                    for(P x;;) {
                        x = vs.top(); vs.pop();
                        sz += x.second;
                        //Tree::addEdge(rt, x.first, sz);
                        vbcc[vbcc_cnt].push_back(x);
                        vbccno[x.first] = vbcc_cnt;
                        if(x.first == v) break;
                    }
                }
            }
            else if(dfn[v] < dfn[u] && i != edge)
                low[u] = min(low[u], dfn[v]);
        }
        for(int i = head[u], v; i; i = e[i].nxt) {
            if(tag[v = e[i].to]) {
                int r = tag[v]; Tree::sz[r] += e[i].w;
                tag[v] = 0;
            }
        }
    }
    void findBCC(int n) {
        for(int i = 1; i <= n; i++)
            if(!dfn[i]) tarjan(i, -1);
    }
}
namespace Tree {
    int dis[MAXN], dep[MAXN], len[MAXN];
    inline void init(int n) {
        BCC::init(n);
```

```
        rt = n;
        ecnt = 1;
        for(int i = 0; i <= 2 * n; i++) {
            head[i] = 0;
            fa[i][0] = isrt[i] = dis[i] = dep[i]  = len[i] = 0;
        }
    }
    void dfs(int x) {
        for(int i = head[x], y; i; i = e[i].nxt) {
            if(!dep[y = e[i].to]) {
                dep[y] = dep[x] + 1;
                dis[y] = dis[x] + e[i].w;
                dfs(y);
            }
        }
    }
    void pre() {
        for(int k = 1; k <= BCC::vbcc_cnt; k++) {
            rt++;
            vector<P> &E = BCC::vbcc[k];
            addEdge(E[0].first, rt, 0);
            int cnt = 0;
            for(int i = E.size() - 1; i >= 1; i--) {
                cnt += E[i].second;
                len[E[i].first] = cnt;
                addEdge(rt, E[i].first, min(cnt, sz[rt] - cnt));
            }
        }
        for(int k = 1; k <= S; k++) {
            for(int i = 1; i <= rt; i++) {
                fa[i][k] = fa[fa[i][k - 1]][k - 1];
            }
        }
        dep[1] = 1;
        dfs(1);
    }
    int up(int x, int d) {
        for(int i = S; i >= 0; i--) {
            if(dep[fa[x][i]] >= d) x = fa[x][i];
        }
        return x;
    }
    int lca(int u, int v) {
        if(dep[u] > dep[v]) swap(u, v);
        v = up(v, dep[u]);
        if(u == v) return u;
        for(int i = S; i >= 0; i--) {
            if(fa[u][i] != fa[v][i]) {
                u = fa[u][i], v = fa[v][i];
            }
        }
        return fa[u][0];
    }
    int query(int u, int v) {
        int l = lca(u, v);
        if(l <= n) return dis[u] + dis[v] - 2 * dis[l];
        int x = up(u, dep[l] + 1), y = up(v, dep[l] + 1);
        int res = dis[u] - dis[x] + dis[v] - dis[y];
        int tmp = abs(len[x] - len[y]);
        return res + min(tmp, sz[l] - tmp);
    }
```

```
}

int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    using namespace Tree;
    cin >> n >> m >> Q;
    init(n);
    for(int i = 1, u, v, w; i <= m; i++) {
        cin >> u >> v >> w;
        BCC::addEdge(u, v, w);
    }
    BCC::findBCC(n);
    pre();
    int u, v;
    while(Q--) {
        cin >> u >> v;
        cout << query(u, v) << endl;
    }
    return 0;
}
```

Example:bzoj4316 小 C 的独立集

仙人掌上的最大独立集

做法：dp[x][0/1] 为 x 不选/选时这棵子树内的最大独立集。类似于树上 dp，考虑如果圆点的 dp 方程不变，方点的 dp 就应该是：dp[x][0] 表示这个点的父亲选，dp[x][1] 表示这个点的父亲可以选也可以不选。

```
// 4316小C的独立集.cpp
#include <bits/stdc++.h>
using namespace std;

inline int read() {
    int x = 0, ch = getchar(); bool fg = 1;
    while(ch < '0' || ch > '9') { if(ch == '-') fg = 0; ch = getchar(); }
    while(ch >= '0' && ch <= '9') { x = x * 10 + ch - '0'; ch = getchar(); }
    return fg ? x : -x;
}
const int MAXN = 1e6 + 5;
int n, m;
namespace Tree {
    vector<int> G[MAXN];
    int rt[MAXN];
    inline void addEdge(int u, int v) {
        G[u].push_back(v);
    }
}
namespace BCC{
    struct Edge {
        int to, nxt;
    }e[MAXN << 1];
    int ecnt, head[MAXN];
    int dfs_clock, dfn[MAXN], low[MAXN];
    int is_vertex[MAXN], vbcc_cnt, vbccno[MAXN];
    vector<int> vbcc[MAXN];
    stack<int> vS;
    inline void addEdge(int u, int v) {
        e[++ecnt] = (Edge) {v, head[u]}; head[u] = ecnt;
        e[++ecnt] = (Edge) {u, head[v]}; head[v] = ecnt;
    }
    inline void init(int n) {
        ecnt = 1;
        dfs_clock = 0;
```

```
        vbcc_cnt = 0;
        for(int i = 1; i <= n; i++){
            head[i] = dfn[i] = low[i] = 0;
            is_vertex[i] = 0;
            vbccno[i] = 0;
        }
        while(!vS.empty()) vS.pop();
    }
    //root's edge = -1;
    void tarjan(int u, int edge) {
        dfn[u] = low[u] = ++dfs_clock;
        int ch = 0;
        vS.push(u);
        for(int i = head[u], v; i; i = e[i].nxt) {
            if(!dfn[v = e[i].to]) {
                tarjan(v, i ^ 1);
                low[u] = min(low[u], low[v]);
                if(low[v] >= dfn[u]) {
                    ch++;
                    if(edge > 0 || ch > 1) is_vertex[u] = 1;
                    if(vS.top() == v) {
                        vS.pop();
                        Tree::addEdge(u, v);
                        continue;
                    }
                    vbcc[++vbcc_cnt].clear();
                    vbcc[vbcc_cnt].push_back(u);
                    int rt = n + vbcc_cnt;
                    Tree::rt[u] = 1;
                    Tree::addEdge(u, rt);
                    for(int x;;) {
                        x = vS.top(); vS.pop();
                        Tree::addEdge(rt, x);
                        vbcc[vbcc_cnt].push_back(x);
                        vbccno[x] = vbcc_cnt;
                        if(x == v) break;
                    }
                }
            }
            else if(dfn[v] < dfn[u] && i != edge)
                low[u] = min(low[u], dfn[v]);
        }
    }
    void findBCC(int n){
        for(int i = 1; i <= n; i++)
            if(!dfn[i]) tarjan(i, -1);
    }
}
namespace Tree {
    int vis[MAXN], dp[MAXN][2];
    int tmp[MAXN][2];
    inline void init(int n) {
        BCC::init(n);
        for(int i = 1; i <= n; i++) {
            G[i].clear();
            dp[i][0] = dp[i][1] = vis[i] = rt[i] = 0;
        }
    }
    void dfs(int x) {
        if(vis[x]) return;
        vis[x] = 1;
```

```cpp
        if(!rt[x] && x <= n) {
            dp[x][1] = 1; dp[x][0] = 0;
            for(auto y : G[x]) {
                dfs(y);
                dp[x][1] += dp[y][0];
                dp[x][0] += max(dp[y][0], dp[y][1]);
            }
        }
        else {
            vector<int> &E = G[x];
            for(auto y : E) dfs(y);
            if(x <= n) {
                dp[x][1] = 1; dp[x][0] = 0;
                for(auto y : E) {
                    dp[x][1] += dp[y][1];
                    dp[x][0] += max(dp[y][0], dp[y][1]);
                }
                return;
            }
            int u, v;
            //第一个点不加
            tmp[E[0]][0] = tmp[E[0]][1] = 0;
            for(int i = 1; i < E.size(); i++) {
                v = E[i], u = E[i - 1];
                tmp[v][1] = dp[v][1] + tmp[u][0];
                tmp[v][0] = dp[v][0] + max(tmp[u][0], tmp[u][1]);
            }
            u = E.front(), v  = E.back();
            dp[x][1] = tmp[v][0] + dp[u][0];
            //第一个点加
            tmp[E[0]][0] = dp[E[0]][0];
            tmp[E[0]][1] = dp[E[0]][1];
            for(int i = 1; i < E.size(); i++) {
                v = E[i], u = E[i - 1];
                tmp[v][1] = dp[v][1] + tmp[u][0];
                tmp[v][0] = dp[v][0] + max(tmp[u][0], tmp[u][1]);
            }
            dp[x][0] = max(tmp[v][0], tmp[v][1]);
        }
    }
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    using namespace Tree;
    cin >> n >> m;
    init(n);
    for(int i = 1, u, v; i <= m; i++) {
        cin >> u >> v;
        BCC::addEdge(u, v);
    }
    BCC::findBCC(n);
    int ans = 0;
    for(int i = 1; i <= n; i++) {
        if(!vis[i]) dfs(i);
        ans = max(ans, max(dp[i][0], dp[i][1]));
    }
    cout << ans << endl;
    return 0;
}
```

2 Data Structures

2.1 Basic Structures

尚未开始

2.2 Heap Structures

尚未开始

2.3 Sequence Structures

尚未开始

2.4 Persistent Data Structures

尚未开始

2.5 Tree Structures

尚未开始

3 String

3.1 Basics

尚未开始

3.2 String Matching

尚未开始

3.3 Suffix Related

尚未开始

3.4 Palindrome Related

周期：若 $0 < p \leq |S|, \forall 1 \leq i \leq |s| - p, s[i] = s[i+p]$ 就称 p 是 s 的周期
border：若 $0 \leq r < |S|, pre(s,r) = suf(s,r)$ 就称 pre(s,r) 是 s 的 border
周期和 border 的关系：t 是 s 的 border，当且仅当 $|s| - |t|$ 是 s 的周期
t 是回文串 s 的后缀，t 是 s 的 border 当且仅当 t 是回文串
t 是回文串 s 的 border($|s| \leq 2|t|$)，s 是回文串当且仅当 t 是回文串
t 是回文串 s 的 border，则 |s|-|t| 是 s 的周期，|s|-|t| 为 s 的最小周期，当且仅当 t 是 s 的最长回文真后缀
x 是一个回文串，y 是 x 的最长回文真后缀，z 是 y 的最长回文真后缀。令 u,v 分别为满足 x=uy,y=vz 的字符串，则有下面三条性质
1. $|u| \geq |v|$
2. 如果 $|u| > |v|$，那么 $|u| > |z|$
3. 如果 $|u| = |v|$，那么 $u = v$
推论：s 的所有回文后缀按照长度排序后，可以划分成 log|s| 段等差数列
遍历回文自动机的 fail 链，能得到当前串的所有 Border

1. 计算以该 i 结尾的回文子串个数. 解：输出 num[i];
2. 计算本质不同的 (奇/偶/总) 回文串个数. 解：奇回文 for(int i = 2; i < pcnt; i++) if(len[i]%2 == 1) ++ans;
3. 计算 (奇/偶/总) 回文串个数. 解：
奇回文 for(int i = 2; i < pcnt; i++) if(len[i]%2 == 1) ans += cnt[i];
动态情况下,总串累计 num[i] 4. 计算 $ww^{-1}ww^{-1}$ 的最长长度. 解: if(len[trans[last]]+len[trans[last]]==len[last] && len[last]%4==0) ans = max(ans, len[last]);
5. 计算 $aa^{-1}bb^{-1}$ 的最长长度. 解：正反各建立一颗回文树，以 i-1 结尾的最长回文串和以 i 开始的最长回文串最大值即为答案
6. 计算相交回文串的个数. 输入：字符串长度 n($1 <= n <= 2*10^6$)、字符串. 解: 总数 - 计算不相交的个数

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 2e6 + 7;
const LL MOD = 51123987;
namespace PAM {
    int scnt, S[MAXN];
```

```
    int pcnt, last, len[MAXN], fail[MAXN];
    struct node{int c, pos, nxt;} e[MAXN];
    int ecnt, head[MAXN];
    int num[MAXN];
    int newnode(int _len) {
        len[pcnt] = _len;
        num[pcnt] = head[pcnt] = 0;
        return pcnt++;
    }
    inline void init() {
        S[scnt = 0] = -1;
        pcnt = 0;newnode(0);newnode(-1);
        fail[0] = 1;last = 0;
        ecnt = 0;
        memset(head, 0, sizeof head);
    }
    int getfail(int x) {
        while(S[scnt - len[x] - 1] != S[scnt]) x = fail[x];
        return x;
    }
    int ch(int u, int c) {
        for(int i = head[u]; i; i = e[i].nxt) if(e[i].c == c) return e[i].pos;
        return 0;
    }
    void setCh(int u, int c, int pos) {
        e[++ecnt] = (node) {c, pos, head[u]}; head[u] = ecnt;
    }
    void extend(int c) {
        S[++scnt] = c;
        int cur = getfail(last);
        if(!(last = ch(cur, c))) {
            last = newnode(len[cur] + 2);
            fail[last] = ch(getfail(fail[cur]), c);
            setCh(cur, c, last);
            num[last] = num[fail[last]] + 1;
        }
    }
};
using namespace PAM;
LL sum[MAXN];
int main() {
    ios::sync_with_stdio(false); cin.tie(0);
    int n; cin >> n;
    string s; cin >> s;
    init();
    for(int i = 1; i <= n; i++) {
        extend(s[i-1] - 'a');
        sum[i] = sum[i-1] + num[last];
        if(sum[i] >= MOD) sum[i] -= MOD;
    }
    LL ans = sum[n] * (sum[n] - 1) / 2;
    init();
    for(int i = n; i; i--) {
        extend(s[i-1] - 'a');
        ans = (ans - sum[i-1] * num[last] % MOD + MOD) % MOD ;
    }
    cout << ans << endl;
    return 0;
}
```

7. 操作一: 串开头或末尾加一个字符, 操作二: 串开头或末尾加一个该串的逆串 从空串到指定串最少步骤

解：每个偶回文串 v 的情况为：min(dp[u] + 1, dp[trans[v]] + len[v]/2 - len[trans[v]] + 1); 答案为 min(ans, dp[v] + n - len[v])

8. 公共回文子串的个数, 重复计算. 输入两个串 s1，s2. 解：分别建立回文树，在两颗树上同时跑

```
int main() {
    ios::sync_with_stdio(false); cin.tie(0);
    string s1, s2; cin >> s1 >> s2;
    T1.init();for(auto it : s1) T1.extend(it-'A');T1.count();
    T2.init();for(auto it : s2) T2.extend(it-'A');T2.count();
    queue<pair<int,int> > q;
    q.push({0, 0}); q.push({1, 1});
    LL ans = 0;
    while(!q.empty()) {
        int cur1 = q.front().first, cur2 = q.front().second; q.pop();
        for(int i = 0; i < 26; i++) {
            int t1 = T1.ch[cur1][i], t2 = T2.ch[cur2][i];
            if(t1 && t2) {
                q.push({t1, t2});
                ans += (LL)T1.cnt[t1] * T2.cnt[t2];
            }
        }
    }
    cout << ans<< endl;
    return 0;
}
```

9. 给定一个串 S，把串分为偶数段，假设分为了 s1,s2,s3....sk，求满足 $s_1 = s_k, s_2 = s_{k-1}$...... 的方案数解：题目转化为 $S' = S_1 S_n S_2 S_{n2}$...，求把 S' 分为若干个长度为偶数的回文串的方案数辅助数组 g：g[v] 表示 v 所在的等差数列的 dp 值之和，且 v 是这个等差数列中长度最长的节点 $g[v] = \sum_{x,slink[x]=v} dp[i - len[x]]$ $g[v] = g[fail[x]] + dp$，多出来的这个位置为 $i - (len[slink] + diff[x])$ 然后利用 g[v] 更新 f[i]，正确性依赖于：如果 v 和 fail[v] 属于同一个等差数列，那么 fail[x] 上一次出现位置是 i-diff[v] 无论 f[i] 是否要求修改，都要更新对应位置的 g

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1e6 + 7;
int MOD = 1e9 + 7;
namespace PAM {
    int scnt, S[MAXN];
    int pcnt, last, len[MAXN], fail[MAXN], ch[MAXN][26];
    int diff[MAXN], slink[MAXN];
    int newnode(int _len) {
        len[pcnt] = _len; diff[pcnt] = slink[pcnt] = 0;
        for(int i = 0; i < 26; i++) ch[pcnt][i] = 0;
        return pcnt++;
    }
    inline void init() {
        S[scnt = 0] = -1;
        pcnt = 0;newnode(0);newnode(-1);
        fail[0] = 1;last = 0;
    }
    int getfail(int x) {
        while(S[scnt - len[x] - 1] != S[scnt]) x = fail[x];
        return x;
    }
    void extend(int c) {
        S[++scnt] = c;
        int cur = getfail(last);
```

```
        if(!ch[cur][c]) {
            int now = newnode(len[cur] + 2);
            fail[now] = ch[getfail(fail[cur])][c];
            ch[cur][c] = now;
            diff[now] = len[now] - len[fail[now]];
            slink[now] = (diff[now] == diff[fail[now]]) ? slink[fail[now]] : fail[now];
        }
        last = ch[cur][c];
    }
};
using namespace PAM;
int g[MAXN], f[MAXN];
void sol(int i, int cur) {
    for(; cur; cur = slink[cur]) {
        g[cur] = f[i - len[slink[cur]] - diff[cur]];
        if (slink[cur] != fail[cur]) g[cur] = (g[cur] + g[fail[cur]])%MOD;
        f[i] = (f[i] + g[cur]) % MOD;
    }
}
int main() {
    ios::sync_with_stdio(false); cin.tie(0);
    string s; cin >> s;
    int sn = s.size();
    init();f[0]=1;
    for(int i = 0; i < sn; i += 2) {
        extend(s[i >> 1] - 'a');sol(i+1, last); f[i+1] = 0;
        extend(s[sn-1 - (i>>1)] - 'a'); sol(i+2, last);
    }
    cout << f[sn] << endl;
    return 0;
}
```

10. 前后均支持插入，利用不基于势能分析的 PAM 模板 11. 在 Trie 上构建回文自动机

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1e5 + 7;
int MOD = 1e9 + 7;
namespace PAM {
    int scnt, S[MAXN<<1];
    int pcnt, last[MAXN];
    int len[MAXN<<1], fa[MAXN<<1], quick[MAXN<<1][26], ch[MAXN<<1][26];
    int newnode(int _len) {
        len[pcnt] = _len;
        for(int i = 0; i < 26; i++) ch[pcnt][i] = 0;
        return pcnt++;
    }
    inline void init() {
        pcnt = 0;newnode(0);newnode(-1);
        fa[0] = 1;
        for(int i = 0; i < 26; i++) quick[0][i] = quick[1][i] = 1;
        S[scnt = 0] = -1;
    }
    int extend(int p, int c) {
        if (S[scnt-len[p]-1] ^ c) p = quick[p][c];
        if (!ch[p][c]) {
            int np = newnode(len[p]+2), q = fa[p];
            if (S[scnt-len[q]-1] ^ c) q = quick[q][c];
            fa[np] = ch[q][c];
            memcpy(quick[np],quick[fa[np]], sizeof(quick[np]));
            quick[np][S[scnt-len[fa[np]]]] = fa[np];
            ch[p][c] = np;
```

```cpp
        }
        return ch[p][c];
    }
    int ilen[MAXN]; char s[MAXN];
    long long ans;
    vector<int> G[MAXN];
    void dfs(int u, int _fa) {
        S[++scnt] = s[u] - 'a';
        last[u] = extend(last[_fa], s[u]-'a');
        ilen[u] = max(len[last[u]], ilen[_fa]);
        ans += ilen[u];
        for(auto v : G[u]) if(v != _fa) dfs(v, u);
        --scnt;
    }
};
using namespace PAM;
int main() {
    int n; scanf("%d", &n);
    scanf("%s", s + 1);
    init();
    for(int i = 1; i < n; i++) {
        int x, y;
        cin >> x >> y;
        G[x].push_back(y);
        G[y].push_back(x);
    }
    dfs(1, 0);
    printf("%lld", ans);
    return 0;
}
```

3.5 Substring Automaton

求解子序列个数，不重复，包括空串

```cpp
int dfs(int x) {
    if(dp[x]) return dp[x];
    dp[x] = 1;
    for(int i = 0; i < 26; i++)
        if(ch[x][i] != -1) dp[x] += dfs(ch[x][i]);
    return dp[x];
}
```

求解回文公共子序列个数，不重复，包括空串

原串与反串都建一遍，这样有两个字符串,x+y<=n+1 这个序列才是合法的

x+y=n+1 以外的情况统计奇回文序列要 ++f[x][y]

特殊处理 x+y=n+1、x+y<n+1 时不 ++f[x][y]，可以统计偶回文序列

相减可以统计奇回文序列

```cpp
LL dfs(LL x,LL y) {
    if(f[x][y]) return f[x][y];
    for(LL i = 0; i < 26; i++)
    if(ch1[x][i] != -1 && ch2[y][i] != -1) {
        if(ch1[x][i]+ch2[y][i] > n+1) continue;
        if(ch1[x][i]+ch2[y][i] < n+1) f[x][y]++;
        f[x][y] = (f[x][y]+dfs(ch1[x][i],ch2[y][i]))%mod;
    }
    return ++f[x][y];
}
```

求解两个字符串的所有公共子序列个数，可重复

```
for(int i=1;i<=n;i++)
    for(int j=1;j<=m;j++) {
        f[i][j]=f[i-1][j]+f[i][j-1]-f[i-1][j-1];
        if(arr[i]==brr[j]) f[i][j]+=f[i-1][j-1]+1;
        if(f[i][j]<0) f[i][j]+=mod;
        if(f[i][j]>=mod) f[i][j]%=mod;
    }
```

求解两个字符串的所有公共子序列个数，去重复，包括了空串输入：a 串长度、b 串长度、a 串、b 串、(0：输出所有的公共子序列个数 |1：按照字典序输出所有不重复公共子序列)

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const LL MOD = 1e9;
const int MAXN = 3020;
const int MAXM = 3e6 + 7;
const int MAXS = 58;
struct Big {//压位高精度
    int len, a[20];
    Big() {
        len = 0;
        for (int i = 0; i < 20; i++) a[i] = 0;
    }
    void put() {
        printf("%d", a[len]);
        for (int i = len - 1; i >= 0; i--) printf("%09d", a[i]);
    }
    void operator=(int o) {
        if (o >= MOD) {
            a[0] = o % MOD;
            a[len = 1] = 0 / MOD;
        } else
            a[len = 0] = o;
    }
    void operator+=(const Big& o) {
        len = max(len, o.len);
        for (int i = 0; i <= len; i++) {
            a[i] += o.a[i];
            if (a[i] >= MOD) {
                a[i + 1] += a[i] / MOD;
                a[i] %= MOD;
            }
        }
        while (a[len + 1]) len++;
    }
} dp[MAXM];

char sta[MAXN];
int top = -1;

char a[MAXN], b[MAXN];
int an, bn, k;
LL ans;
int cha[MAXN][MAXS], chb[MAXN][MAXS];

int markn, mark[MAXN][MAXN];
void build(char* s, int sn, int (*ch)[58]) {
    for (int j = 0; j < 58; j++) ch[sn][j] = ch[sn + 1][j] = -1;
    for (int i = sn; i >= 1; i--) {
```

```
        for (int j = 0; j < 58; j++) ch[i - 1][j] = ch[i][j];
        ch[i - 1][s[i] - 'A'] = i;
    }
}
int dfs0(int x, int y) {
    if (mark[x][y])
        return mark[x][y];
    int p = mark[x][y] = ++markn;
    dp[p] = 1;
    for (int i = 0; i < 58; i++) {
        if (cha[x][i] != -1 && chb[y][i] != -1) {
            dp[p] += dp[dfs0(cha[x][i], chb[y][i])];
        }
    }
    return mark[x][y];
}
LL dfs1(int x, int y) {
    puts(sta);
    LL cnt = 1;
    for (int i = 0; i < 58; i++) {
        if (cha[x][i] != -1 && chb[y][i] != -1) {
            sta[++top] = i + 'A';
            cnt += dfs1(cha[x][i], chb[y][i]);
            sta[top--] = '\0';
        }
    }
    return cnt;
}
int main() {
    scanf("%d%d%s%s%d", &an, &bn, a + 1, b + 1, &k);
    build(a, an, cha);
    build(b, bn, chb);
    if (k == 1)
        dfs1(0, 0);
    dfs0(0, 0);
    dp[mark[0][0]].put();
    return 0;
}
```

4 Math

4.1 FFT

分治 FFT:

对集合 S，$val(s) = F_{sum(s)}$,where sum(s) is the sum of all the elements of the set.

Given a set S having N elements, please calculate the sum of the values of all subsets of S of size K.

$$F_n = \frac{1 - a^n}{1 - a}$$

生成函数:

$$(x + a^{s_1}) \cdot (x + a^{s_2}) \cdots (x + a^{s_n})$$

答案为 x^{n-k} 的系数

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 300005, mod = 1e5 + 3, P = mod - 1;
const double pi = acos(-1.);
struct comp {
    double x, y;
    comp operator + (const comp& a) const { return (comp) {x + a.x, y + a.y}; }
    comp operator - (const comp& a) const { return (comp) {x - a.x, y - a.y}; }
    comp operator * (const comp& a) const { return (comp) {x * a.x - y * a.y, x * a.y +
    y * a.x}; }
};
#define conj(a) ((comp){a.x, -a.y})
int rev[MAXN], T;
comp Sin[MAXN], tmp;
void fft(comp *a, int r) {
    for(int i = 1; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
    for(int i = 2, mid = 1, s = (T >> 1); i <= T; mid = i, i <<= 1, s >>= 1) {
        for(int j = 0; j < T; j += i) {
            for(int k = j, cur = 0; k < j + mid; k++, cur += s) {
                tmp = a[k + mid] * Sin[cur];
                a[k + mid] = a[k] - tmp;
                a[k] = a[k] + tmp;
            }
        }
    }
}
comp A[MAXN];
void init(int n) {
    for(T = 1; T <= n; T <<= 1);
    for(int i = 0; i < T; i++) {
        if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
        else rev[i] = rev[i >> 1] >> 1;
        A[i] = (comp) {0, 0};
    }
    for(int i = 0; i < (T >> 1); i++) {
        Sin[i] = (comp) {cos(2 * pi * i / T), sin(2 * pi * i / T)};
    }
}
void mtt(int *x, int *y) {
    for(int i = 0; i < T; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
    static comp a[MAXN], b[MAXN];
    static comp dfta[MAXN], dftb[MAXN], dftc[MAXN], dftd[MAXN];
```

```
    for(int i = 0; i < T; i++) {
        a[i] = {x[i] & 0x7fff, x[i] >> 15};
        b[i] = {y[i] & 0x7fff, y[i] >> 15};
    }
    fft(a, 1); fft(b, 1);
    for(int i = 0; i < T; i++) {
        int j = (T - i) & (T - 1);
        static comp da, db, dc, dd;
        da = (a[i] + conj(a[j])) * (comp){0.5, 0};
        db = (a[i] - conj(a[j])) * (comp){0, -0.5};
        dc = (b[i] + conj(b[j])) * (comp){0.5, 0};
        dd = (b[i] - conj(b[j])) * (comp){0, -0.5};
        dfta[i] = da * dc;
        dftb[i] = da * dd;
        dftc[i] = db * dc;
        dftd[i] = db * dd;
    }
    for(int i = 0; i < T; i++) {
        a[i] = dfta[i] + dftb[i] * (comp) {0, 1};
        b[i] = dftc[i] + dftd[i] * (comp) {0, 1};
    }
    for(int i = 0; i < (T >> 1); i++) Sin[i].y = -Sin[i].y;
    fft(a, -1); fft(b, -1);
    for(int i = 0; i < T; i++) {
        static int da, db, dc, dd;
        da = (LL)(a[i].x / T + 0.5) % mod;
        db = (LL)(a[i].y / T + 0.5) % mod;
        dc = (LL)(b[i].x / T + 0.5) % mod;
        dd = (LL)(b[i].y / T + 0.5) % mod;
        x[i] = ((da + ((LL)(db + dc) << 15) + ((LL)dd << 30)) % mod + mod) % mod;
    }
}
LL f[MAXN], g[MAXN];
int x[MAXN], y[MAXN];
void solve(int l, int r) {
    if(l == r) {
        f[0] = 1;
        f[1] = g[l];
        return;
    }
    int mid = (l + r) >> 1;
    solve(l, mid);
    LL tmp[mid - l + 5];
    for(int i = 0; i <= mid - l + 1; i++) tmp[i] = f[i];
    solve(mid + 1, r);
    init(r - l + 1);
    for(int i = 0; i < T; i++) x[i] = y[i] = 0;
    for(int i = 0; i <= mid - l + 1; i++) x[i] = tmp[i];
    for(int i = 0; i <= r - mid; i++) y[i] = f[i];
    mtt(x, y);
    for(int i = 0; i <= r - l + 1; i++) {
        f[i] = x[i];
    }
}
LL fac[mod + 5], inv[mod + 5];
void init_C() {
    fac[0] = 1;
    for(int i = 1; i < mod; i++) fac[i] = fac[i - 1] * i % mod;
    inv[0] = inv[1] = 1;
    for(int i = 2; i < mod; i++) inv[i] = (mod - mod / i) * inv[mod % i] % mod;
    for(int i = 1; i < mod; i++) inv[i] = inv[i] * inv[i - 1] % mod;
```

```
}
LL C(int a, int b) {
    if(b > a) return 0;
    if(a < mod) return fac[a] * inv[b] % mod * inv[a - b] % mod;
    return C(a / mod, b / mod) * C(a % mod, b % mod) % mod;
}
LL pw[MAXN];
LL qpow(LL x, LL y) {
    LL res = 1;
    while(y) {
        if(y & 1) res = res * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return res;
}
int main() {
    init_C();
    int n, a, Q;
    scanf("%d%d%d", &n, &a, &Q);
    pw[0] = 1;
    for(int i = 1; i < mod; i++) pw[i] = pw[i - 1] * a % mod;
    for(int i = 1, s; i <= n; i++) {
        scanf("%d", &s);
        g[i] = pw[s % P];
    }
    solve(1, n);
    int k;
    int INV = qpow(a - 1, mod - 2);
    while(Q--) {
        scanf("%d", &k);
        printf("%lld\n", (f[k] + mod - C(n, k)) % mod * INV % mod);
    }
    return 0;
}
```

4.2 FWT

题目大意:

给一个 $n*m$ 矩阵 $(a)_{ij}$, 行向量为 v_i, 求 $\oplus_{x=0}^{2^k}(count(x)*3^x mod(10^9+7))$ 其中 $count(x)$ 表示 $\frac{1}{2^m}\sum_i^n\prod_j^m(1-(-1)^{|a_{ij} and x|})$, $\oplus$ 表示异或,$|a|$ 表示 a 的二进制中 1 的个数。

解:

考虑化简以上 $count(x)$, 由于 $(-1)^{|a and x|+|b and x|}=(-1)^{|(a\oplus b) and x|}$ 则展开 $\prod$ 后得到:

$$\frac{1}{2^m}\sum_i^n\sum_{S\subseteq v_i}(-1)^{size(S)}*(-1)^{|(\oplus_{j\in S}a_{ij}) and x|}$$

其中 s 为枚举每个行向量的二进制集合。

这里可以观察到和离散沃尔什变换相似的地方，即:

$$
\begin{cases}
C_k = \sum_{i \oplus j = k} A_i * B_j \\
DWT(A)_i = \sum_{j}^{n} A_j * f_{i,j} \\
DWT(C)_i = DWT(A)_i * DWT(B)_i \\
f_{i,j} \cdot f_{i,k} = f_{i,j \oplus k} \\
f_{i,j} = [i\ and\ j == i] & (and) \\
f_{i,j} = [i\ and\ j == j] & (or) \\
f_{i,j} = (-1)^{|i\ and\ j|} & (xor)
\end{cases}
$$

注意到 xor 的形式与上述形式完全一致，所以令

$$cnt_x = \sum_{i}^{n} \sum_{S \in i} (-1)^{size(S)}$$

其中 $x = \oplus_{j \in S} a_{i,j}$，则对于 cnt 做沃尔什变换可得：

$$DWT(cnt)_x = \sum_{j}^{2^k} \sum_{i}^{n} \sum_{S \subseteq v_i} (-1)^{size(S)} * (-1)^{|j\ and\ x|}$$

由此可得：

$$count(x) = \frac{1}{2^m} * DWT(cnt)_x$$

预处理 cnt_x 复杂度 $O(n\ 2^m)$，FWT 复杂度 $O(k\ 2^k)$，总复杂度为 $O(n\ 2^m + k\ 2^k)$

```
#include <bits/stdc++.h>
using namespace std;
const int mod = 1e9 + 7;
void FWT(vector<int> &a,int n) {
    for(int i = 2;i <= n; i <<= 1) {
        for(int j = 0; j < n; j += i) {
            for(int d = 0, w = i >> 1; d < w; d++){
                int u = a[j + d], v = a[j + d + w];
                a[j + d] = u + v, a[j + d + w] = u - v;
            }
        }
    }
}
void dfs(vector<int> &cnt, const vector<int> &a, int i, int x, int p) {
    if(i < a.size()) {
        dfs(cnt, a, i + 1, x, p);
        dfs(cnt, a, i + 1, x ^ a[i], -p);
    }
    else {
        cnt[x] += p;
    }
}
int main() {
    int n, m, k;
    while(~scanf("%d%d%d", &n, &m, &k)) {
        auto cnt = vector<int>(1 << k, 0);
        auto a = vector<int>(m);
        for(int i = 0; i < n; i++) {
```

```
            for(int j = 0; j < m; j++) a[j] = read();
            dfs(cnt, a, 0, 0, 1);
        }
        FWT(cnt, 1 << k);
        int ans = 0, pw = 1 << m, w = 1;
        for(int x = 0; x < (1 << k); x++) {
            ans ^= (long long)w * (cnt[x] / pw) % mod;
            w = (long long)w * 3 % mod;
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

4.3 Functions

题目大意：求 $\Sigma_i^n \Sigma_j^m gcd(i,j) * 2 - 1$

解:
$\Sigma_i^n \Sigma_j^m gcd(i,j) * 2 - 1$
$= -n * m + 2 * \Sigma_i^n \Sigma_j^m gcd(i,j)$
$= -n * m + 2 * \Sigma_i^n \Sigma_j^m \Sigma_{d|gcd(i,j)} \phi(d)$
$= -n * m + 2 * \Sigma_d^{max(n,m)} \phi(d) * \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
枚举 d，复杂度 O(n)

```
#include <cstdio>
#include <algorithm>
#include <cmath>
using namespace std;
const int MAXN = 100003;
int phi[MAXN], prime[MAXN], cnt;
bool isp[MAXN];
void Euler(int n) {
    phi[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime[++cnt] = i;
            phi[i] = i - 1;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            else phi[i * prime[j]] = phi[i] * (prime[j] - 1);
        }
    }
}
long long ans;
int main() {
    int n, m, k;
    scanf("%d%d", &n, &m);
    k = max(n, m);
    Euler(k);
    ans -= (long long)n * m;
    for(int d = 1; d <= k; d++) {
```

```
        ans += 2LL * phi[d] * (n / d) * (m / d);
    }
    printf("%lld\n", ans);
    return 0;
}
```

题目大意：对于给定的整数 N,M 和 d，有多少正整数对 x,y，满足 x<=N，y<=M，并且 gcd(x,y)=d。

解：
等价于 x<=N/d，y<=M/d，互质的 x，y 的对数。
那么我们另 n=N/d, m = M/d
于是原题就变成了求 $\Sigma_i^n\Sigma_j^m e(gcd(i,j))$
其中 $e(x) = (x == 1) = \Sigma_{d|x}\mu(d)$
然后根据套路我们再来推一波式子
$\Sigma_i^n\Sigma_j^m e(gcd(i,j))$
$=\Sigma_i^n\Sigma_j^m\Sigma_{d|gcd(i,j)}\mu(d)$
$=\Sigma_d^{min(n,m)}\mu(d) * \lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$
根据上一题的做法，每次询问复杂度 O(n)，显然不能胜任。
但是我们观察式子，显然 $\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$ 的取值只有 $\sqrt{n}+\sqrt{m}$
个，那么我们可以预处理 μ 的前缀和，并分块完成。

```
#include <cstdio>
#include <algorithm>
using namespace std;
const int MAXN = 50003;
int mu[MAXN], prime[MAXN], cnt;
bool isp[MAXN];
void Euler(int n) {
    mu[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime[++cnt] = i;
            mu[i] = -1;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                mu[i * prime[j]] = 0;
                break;
            }
            mu[i * prime[j]] = -mu[i];
        }
    }
}
int cal(int n, int m) {
    if(n > m) swap(n, m);
    int res = 0, last;
    for(int i = 1; i <= n; i = last + 1) {
        last = min(n / (n / i), m / (m / i));
        res += (n / i) * (m / i) * (mu[last] - mu[i - 1]);
    }
    return res;
}
int main() {
    Euler(50000);
    for(int i = 2; i <= 50000; i++) mu[i] += mu[i - 1];
```

```
    int T, a, b, d;
    scanf("%d", &T);
    while(T--) {
        scanf("%d%d%d", &a, &b, &d); a /= d; b /= d;
        printf("%d\n", cal(a, b));
    }
    return 0;
}
```

题目大意：另 F(i) 表示 i 的约数和，q 次给定 n,m,a，求 $\Sigma_i^n \Sigma_{j and F(gcd(i,j))<=a}^m F(gcd(i,j)) mod 2^{31}$

解：
有个 a 的限制，式子并不能推了。所以先把它去掉。
令 $g(i) = \Sigma_x^n \Sigma_y^m e(gcd(x,y) == i)$
那么可以得到 $g(i) = \Sigma_{i|d} \mu(\frac{d}{i}) * \lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$
于是 $ans = \Sigma_i^{min(n,m)} F(i) * g(i)$
展开化简得到 $\Sigma_d^{min(n,m)} \lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor \Sigma_{i|d} F(i)\mu(\frac{d}{i})$
然后对 $\Sigma_{i|d} F(i)\mu(\frac{d}{i})$ 求个前缀和
枚举每个 i 更新 i 的倍数即可
现在考虑将 a 离线处理，询问按 a 排序
用树状数组维护前缀和即可。

```
#include <cstdio>
#include <algorithm>
using namespace std;
typedef pair<int, int> P;
const int MAXN = 100003;
int prime[MAXN], cnt, mu[MAXN], mnsum[MAXN], a[MAXN];
bool isp[MAXN];
P f[MAXN];
void Euler(int n) {
    mu[1] = 1;
    f[1] = P(1, 1);
    for(int i = 2; i <= n; i++) {
        f[i].second = i;
        if(!isp[i]) {
            prime[++cnt] = a[i] = i;
            mu[i] = -1;
            f[i].first = mnsum[i] = i + 1;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                mu[i * prime[j]] = 0;
                mnsum[i * prime[j]] = mnsum[i] + a[i] * prime[j];
                f[i * prime[j]].first = f[i].first / mnsum[i] * mnsum[i * prime[j]];
                a[i * prime[j]] = a[i] * prime[j];
                break;
            }
            mu[i * prime[j]] = -mu[i];
            f[i * prime[j]].first = f[i].first * (prime[j] + 1);
            a[i * prime[j]] = prime[j];
            mnsum[i * prime[j]] = prime[j] + 1;
        }
    }
}
```

```
int bit[MAXN];
void add(int x, int v) {
    for(; x <= 100000; x += x & -x) bit[x] += v;
}
int sum(int x) {
    int res = 0;
    for(; x; x -= x & -x) res += bit[x];
    return res;
}
struct Node {
    int n, m, a, id;
    bool operator < (const Node &x) const {
        return a < x.a;
    }
    inline void read(int i) {
        id = i; scanf("%d%d%d", &n, &m, &a);
    }
}q[MAXN];
int ans[MAXN];
int cal(int n, int m) {
    if(n > m) swap(n, m);
    int res = 0, last;
    for(int i = 1; i <= n; i = last + 1) {
        last = min(n / (n / i), m / (m / i));
        res += (n / i) * (m / i) * (sum(last) - sum(i - 1));
    }
    return res & 0x7fffffff;
}
int main() {
    Euler(100000);
    sort(f + 1, f + 100001);
    int T;
    scanf("%d", &T);
    for(int i = 1; i <= T; i++) q[i].read(i);
    sort(q + 1, q + T + 1);
    int cur = 1, N, M, A;
    for(int Q = 1; Q <= T; Q++) {
        N = q[Q].n, M = q[Q].m, A = q[Q].a;
        while(cur <= 100000 && f[cur].first <= A) {
            for(int i = f[cur].second; i <= 100000; i += f[cur].second)
                add(i, f[cur].first * mu[i / f[cur].second]);
            cur++;
        }
        ans[q[Q].id] = cal(N, M);
    }
    for(int i = 1; i <= T; i++) printf("%d\n", ans[i]);
    return 0;
}
```

题目大意：求 $\Sigma_i^n \Sigma_j^m lcm(i,j) mod 20101009$

解：
这里跟之前的区别在于，gcd(i,j) 在分母上，那么考虑反演
令 $f(x) = \frac{1}{x}$
$F(x) = f$ X μ 这里 X 是 Dirichlet 积
即 $F(x) = \Sigma_{d|x} f(d) * \mu(\frac{x}{d})$
那么由莫比乌斯反演我们有 $f = F X 1$
那么 $\Sigma_i^n \Sigma_j^m lcm(i,j)$

$=\Sigma_i^n\Sigma_j^m i*j*f(gcd(i,j))$

$=\Sigma_i^n\Sigma_j^m i*j*\Sigma_{d|gcd(i,j)F(d)}$

另 $sum(x,y)=\Sigma_i^x\Sigma_j^y i*j$

于是可以进一步化简 $ans=\Sigma_d^n F(d)*d*d*sum(\frac{n}{d},\frac{m}{d})$

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
typedef long long ll;
const int mod = 20101009, MAXN = 10000003;
int mu[MAXN], prime[MAXN], sum[MAXN];
int cnt;
bool isp[MAXN];
void getmu(int n) {
    mu[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            mu[i] = -1;
            prime[++cnt] = i;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                mu[i * prime[j]] = 0;
                break;
            }
            mu[i * prime[j]] = -mu[i];
        }
    }
}
ll n, m, ans;
ll query(ll x, ll y) { return (x * (x + 1) / 2 % mod) * (y * (y + 1) / 2 % mod) % mod; }
ll F(ll x, ll y) {
    ll res = 0, last;
    for(ll i = 1; i <= min(x, y); i = last + 1) {
        last = min(x / (x / i), y / (y / i));
        res = (res + (sum[last] - sum[i - 1]) * query(x / i, y / i) % mod) % mod;
    }
    return res;
}
int main() {
    cin>>n>>m;
    getmu(min(n, m));
    for(ll i = 1; i <= min(n, m); i++) sum[i] = (sum[i - 1] + (i * i * mu[i]) % mod) % mod;
    ll last;
    for(ll d = 1; d <= min(n, m); d = last + 1) {
        last = min(n / (n / d), m / (m / d));
        ans = (ans + (last - d + 1) * (d + last) / 2 % mod * F(n / d, m / d) % mod) % mod;
    }
    ans = (ans + mod) % mod;
    cout<<ans<<endl;
    return 0;
}
```

总结一下：当推式子陷入僵局的时候，尝试将一些东西处理出来，通过分块、反演等方法降低复杂度。对于式子本身的特性进行观察，有时也可以发现优化的地方。

在推式子过程中，将有关变量放在一块处理是化简的常用技巧。

4.4 Sieve

Example: 51nod1239 欧拉函数之和

S(n) = Phi(1) + Phi(2) + ... Phi(n)，给出 n，求 S(n)

```
//杜教筛
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 1e7 + 5, mod = 1000000007;
vector<int> prime;
int phi[MAXN], P[MAXN];
bool isp[MAXN];
unordered_map<LL, int> mp;
void Euler(int n) {
    phi[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime.push_back(i);
            phi[i] = i - 1;
        }
        for(auto x : prime) {
            if(i * x > n) break;
            isp[i * x] = 1;
            if(i % x == 0) {
                phi[i * x] = phi[i] * x;
                break;
            }
            phi[i * x] = phi[i] * (x - 1);
        }
    }
    for(int i = 1; i <= n; i++) P[i] = (P[i - 1] + phi[i]) % mod;
}
LL cal(LL n) {
    if(n < MAXN) return P[n];
    if(mp.count(n)) return mp[n];
    LL res = 0;
    for(LL i = 2, last; i <= n; i = last + 1) {
        last = n / (n / i);
        res += (last - i + 1) % mod * cal(n / i) % mod;
        res %= mod;
    }
    mp[n] = res = ((__int128)n * (n + 1) / 2 % mod + mod - res) % mod;
    return res;
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    Euler(MAXN - 1);
    LL n; cin >> n;
    cout << cal(n) << endl;
    return 0;
}
//min25
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 1e6 + 5, mod = 1e9 + 7;
```

```cpp
const int inv2 = (mod + 1) / 2, inv6 = (mod + 1) / 6;
int prime[MAXN], isp[MAXN], cnt;
LL g[3][MAXN << 1], h[3][MAXN << 1];
LL w[MAXN << 1];
int id1[MAXN], id2[MAXN];
inline int MOD(LL x) { return x >= mod ? x - mod : x; }
//inline int MOD(LL x) { return x % mod; }
inline int add(LL x, LL y) { return MOD(MOD(x) + MOD(y)); }
void Euler(int n) {
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime[++cnt] = i;
            h[0][cnt] = h[0][cnt - 1] + 1;
            h[1][cnt] = add(h[1][cnt - 1], i);
            h[2][cnt] = add(h[2][cnt - 1], (LL)i * i % mod);
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                break;
            }
        }
    }
}
LL n;
int sz, m;
inline int id(LL x) {
    return x <= sz ? id1[x] : id2[n / x];
}
//f(p ^ k)
inline int f(int p, LL pk) {
    return pk / p * (p - 1) % mod;
}
LL S(LL x, int y) {
    if(x <= 1 || prime[y] > x) return 0;
    //g(x) - h(j - 1)
    LL res = add(add(g[1][id(x)], mod - g[0][id(x)]), mod - add(h[1][y - 1], mod - h[0][y - 1]));
    for(int j = y, k = 1; j <= cnt && (LL)prime[j] * prime[j] <= x; j++, k = 1) {
        for(LL pk = prime[j]; pk * prime[j] <= x; pk *= prime[j], k++) {
            res = add(res, S(x / pk, j + 1) * f(prime[j], pk) % mod + f(prime[j], pk * prime[j]));
        }
    }
    return res;
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    cin >> n;
    sz = sqrt(n) + 1;
    Euler(sz);
    for(LL i = 1, last, t; i <= n; i = last + 1) {
        last = n / (n / i);
        w[++m] = n / i, t = n / i % mod;
        w[m] <= sz ? id1[w[m]] = m : id2[last] = m;
        g[0][m] = MOD(t + mod - 1);
        g[1][m] = add(t * (t + 1) % mod * inv2 % mod, mod - 1);
        g[2][m] = add((2 * t + 1) % mod * t * (t + 1) % mod * inv6 % mod, mod - 1);
    }
    for(int j = 1; j <= cnt; j++) {
        for(int i = 1; i <= m && (LL)prime[j] * prime[j] <= w[i]; i++) {
```

```
            g[0][i] = MOD(g[0][i] + mod - (g[0][id(w[i] / prime[j])] - h[0][j - 1]));
            g[1][i] = MOD(g[1][i] + mod - ((LL)prime[j] * MOD(g[1][id(w[i] / prime[j])] + mod - h[1][j - 1]) % mod));
            g[2][i] = MOD(g[2][i] + mod - ((LL)prime[j] * prime[j] % mod * MOD(g[2][id(w[i] / prime[j])] + mod - h[2][j - 1]) % mod));
        }
    }
    //S(n, 1) + F(1);
    LL ans = MOD(S(n, 1) + 1);
    cout << ans << endl;
    return 0;
}
```

Example: 51nod1224 莫比乌斯函数之和

S(a,b) = miu(a) + miu(a + 1) + ... miu(b)

```
//杜教筛
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 3e7 + 5;
vector<int> prime;
int mu[MAXN];
int M[MAXN];
bool isp[MAXN];
unordered_map<LL, LL> mp;
void Euler(int n) {
    mu[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime.push_back(i);
            mu[i] = -1;
        }
        for(auto x : prime) {
            if(i * x > n) break;
            isp[i * x] = 1;
            if(i % x == 0) {
                mu[i * x] = 0;
                break;
            }
            mu[i * x] = -mu[i];
        }
    }
    for(int i = 1; i <= n; i++) M[i] = M[i - 1] + mu[i];
}
LL cal(LL n) {
    if(n < MAXN) return M[n];
    if(mp.count(n)) return mp[n];
    LL res = 0;
    for(LL i = 2, last; i <= n; i = last + 1) {
        last = n / (n / i);
        res += (last - i + 1) * cal(n / i);
    }
    mp[n] = 1 - res;
    return 1 - res;
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    Euler(MAXN - 1);
    LL a, b;
```

```
    cin >> a >> b;
    cout << cal(b) - cal(a - 1) << endl;
    return 0;
}
//min25
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 1e6 + 5;
int prime[MAXN], isp[MAXN], cnt;
LL g[3][MAXN << 1], h[3][MAXN << 1];
LL w[MAXN << 1];
int id1[MAXN], id2[MAXN];
void Euler(int n) {
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime[++cnt] = i;
            h[0][cnt] = h[0][cnt - 1] + 1;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                break;
            }
        }
    }
}
LL a, b;
LL n;
int sz, m;
inline int id(LL x) {
    return x <= sz ? id1[x] : id2[n / x];
}
//f(p ^ k)
inline int f(int p, int k) {
    return k == 1 ? -1 : 0;
}
LL S(LL x, int y) {
    if(x <= 1 || prime[y] > x) return 0;
    //g(x) - h(j - 1)
    LL res = - g[0][id(x)] + h[0][y - 1];
    for(int j = y, k = 1; j <= cnt && (LL)prime[j] * prime[j] <= x; j++, k = 1) {
        for(LL pk = prime[j]; pk * prime[j] <= x; pk *= prime[j], k++) {
            res += S(x / pk, j + 1) * f(prime[j], k) + f(prime[j], k + 1);
        }
    }
    return res;
}
LL cal(LL x) {
    n = x;
    m = 0;
    sz = sqrt(n);
    for(LL i = 1, last, t; i <= n; i = last + 1) {
        last = n / (n / i);
        w[++m] = n / i, t = n / i;
        w[m] <= sz ? id1[w[m]] = m : id2[last] = m;
        g[0][m] = t - 1;
    }
    for(int j = 1; j <= cnt; j++) {
        for(int i = 1; i <= m && (LL)prime[j] * prime[j] <= w[i]; i++) {
            g[0][i] = g[0][i] - (g[0][id(w[i] / prime[j])] - h[0][j - 1]);
```

```
        }
    }
    return S(x, 1) + 1;
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    cin >> a >> b;
    Euler(sqrt(b));

    //S(n, 1) + F(1);
    cout << cal(b) - cal(a - 1) << endl;
    return 0;
}
```

5 Geometry

尚未开始

6 Game Theory

6.1 Bash's Game

Bash's Game 巴什博弈
有一堆个数为 n 的石子，游戏双方依次从中拿取，满足:
1. 每次至少取 1 个，最多取 m 个.
最后取光者得胜。
结论: $n = t(m+1) + r$, 必败态:$r = 0$;
巴什博弈变种:
取一个指定集合的石头个数
取到最后一个石子输, $n = t(m + 1) + r, r = 1$;

6.2 Wythoff' s Game

Wythoff' s Game（威佐夫博弈）
有两堆分别为 (an, bn) 的石子, 游戏双方依次从中拿取，满足:
1. 从任意一堆中取任意个 > 1。2. 从两堆中取同样多个。最后取完者胜.
结论: 对于任意的局势 $(a, b)(a < b)$, 必败点为 (b-a)*(sqrt(5)+1)/2=a.

6.3 Fibonacci' s Game / Zeckendorf's theory

Fibonacci' s Game（斐波那契博弈）
有一堆个数为 n 的石子，游戏双方轮流取石子，满足：
1. 先手不能在第一次把所有的石子取完；
2. 之后每次可以取的石子数介于 1 到对手刚取的石子数的 2 倍之间（包含 1 和对手刚取的石子数的 2 倍）。
结论: 必败点是斐波那契数

齐肯多夫定理: 任何正整数可以表示为若干个不连续的 Fibonacci 数之和

6.4 Nim' s Game / Anti-Nim's Game / K-Nim's Game / Anti-K-Nim's Game

Nim' s Game（尼姆博弈）
石子的个数可以等价成某个游戏的 SG 函数。

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 规定每次只能从一堆中取若干根，可将一堆全取走，但不可不取.
最后取完者为胜。
结论:
T 态: 所有火柴数异或和为 0
S 态: 所有火柴数异或和不为 0

必胜态:S

有 n 堆石子, 游戏双方依次从中拿取, 满足:
1. 规定每次只能从一堆中取若干根，可将一堆全取走，但不可不取.
最后取完者为败。
结论:
S0 态: 即仅有奇数个孤单堆
T0 态: 即仅有偶数个孤单堆
S1 态: 异或和大于 0, 且有 1 个充裕堆
T1 态: 不存在
S2 态: 异或和大于 0, 且有多个充裕堆
T2 态: 异或和等于 0, 且有多个充裕堆
必胜态:T0,S1,S2
必败态:S0,T2

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 规定每次只能至多 k 堆中取若干根，可将 k 堆全取走，但不可不取.
最后取完者为胜。
结论:
对于每一堆，把它石子的个数用二进制表示
必败态: 对所有的石子堆，如果在任何一个二进制位上 1 的个数总是 k+1 的整数倍

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 规定每次只能至多 k 堆中取若干根，可将 k 堆全取走，但不可不取
最后取完者为败。
结论:
1. 对于每一堆，把它石子的个数用二进制表示
2. 所有的堆（非零堆，下同）全是 1, 此时如果 1 堆个数模 k+1 的结果是 1 则必败，否则必胜 (我们可以通过拿走 0 到 k 个堆来随意调整当前状态模的结果，然后再将所有大于 1 的堆降到 1 就行了)
3. 有多于 k 个堆的个数大于 1。必胜

6.5 阶梯博弈

有 n 个阶梯呈升序排列，每个阶梯上有若干个石子，游戏双方轮流取石子, 满足:
1. 将一个阶梯上的石子移任意个（>0）到前一个台阶。
当没有可行操作时（所有石子都被移动到了地面，即第 0 号台阶）输。
结论:
奇数号台阶的 Nim 游戏
变种 1: 树上, 每个石子只能往父亲节点移动.
变种 2:
游戏双方在一个 1*N 的格子内挪动棋子，刚开始在若干个位置上有棋子, 每个位置至多一个棋子
每一个选手可以进行的操作时选择一个棋子并把它向左方移动，当然不能越过其它的棋子，也不能超出边界。
谁不能移动谁就输了。求谁会赢?

结论:
将棋子位置按升序排列, 然后从后往前两两绑成一对, 如果个数是奇数, 那么将第一个和边界外绑定.
一对棋子的前一个和前一对棋子的后一个之间有多少个空位置对最终的结果是没有影响的。
于是我们只需要考虑同一对的两个棋子之间有多少空位, 将同一对棋子间的空位视为石子, 做 nim 游戏
两对棋子间的空格数当奇数位石子, 其他当偶数位石子, 石子相右边移动
变种 3:
山上有 n 个人，每个人给出距离山顶的距离，给出其中一个人为 king，每次能挑选一个人向上移动，不能越过其他人，最后将 king 移动到山顶者获胜。问获胜者。
结论:
只要把 King 当作普通人一样处理即可。除了两种特殊情况：
1. 当 King 是第一个人时，Alice 直接胜
2. 当 King 是第二个人且一共有奇数个人时，第一堆的大小需要减 1。

6.6 Multi-Nim

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 任意一堆石子中拿任意多个石子 (不能不拿)
2. 把一堆数量不少于 2 石子分为两堆不为空的石子
最后取完者为胜。
结论:
操作一与普通的 Nim 游戏等价
操作二实际上是将一个游戏分解为两个游戏，根据 SG 定理，我们可以通过异或运算把两个游戏连接到一起，作为一个后继状态

$$SG(x) \equiv \begin{cases} x-1 & (xmod4=0) \\ x & (xmod4=1or2) \\ x+1 & (xmod4=3) \end{cases} \tag{1}$$

Multi-SG 游戏规定，在符合拓扑原则的前提下，一个单一游戏的后继可以为多个单一游戏。
Multi-SG 其他规则与 SG 游戏相同。
注意在这里要分清楚后继与多个单一游戏
对于一个状态来说，不同的划分方法会产生多个不同的后继，而在一个后继中可能含有多个独立的游戏
一个后继状态的 SG 值即为后继状态中独立游戏的异或和
该状态的 SG 值即为后继状态的 SG 值中未出现过的最小值

6.7 Every-SG

给定一张无向图，上面有一些棋子，两个顶尖聪明的人在做游戏，每人每次必须将可以移动的棋子进行移动，不能移动的人
因为两个人都顶尖聪明，因此当一个人知道某一个游戏一定会输的话，它一定会尽力缩短游戏的时间，当它知道某一个游戏一定会赢的话，一定会尽力延长游戏的时间
对于还没有结束的单一游戏，游戏者必须对该游戏进行一步决策；
其他规则与普通 SG 游戏相同

Every-SG 游戏与普通 SG 游戏最大的不同就是它多了一维时间
对于 SG 值为 0 的点，我们需要知道最少需要多少步才能走到结束
对于 SG 值不为 0 的点，我们需要知道最多需要多少步结束
这样我们用 step 变量来记录这个步数

$$step(x) \equiv \begin{cases} 0 & u \\ maxstep(v) & sg(u)! = 0vusg(v) = 0 \\ minstep(v) & sg(u) = 0vu \end{cases} \tag{2}$$

6.8 树的删边游戏

给出一个有 N 个点的树，有一个点作为树的根节点。游戏者轮流从树中删去边，删去一条边后，不与根节点相连的部分将被移走。谁无法移动谁输。
结论:
Colon Principle: 对于树上的某一个点，ta 的分支可以转化成以这个点为根的一根竹子，这个竹子的长度就是 ta 各个分支的边的数量的异或和
叶子节点的 SG 值为 0；中间节点的 SG 值为它的所有子节点的 SG 值加 1 后的异或和。

6.9 Chomp's theory?

取一个无关紧要的位置, 如果对方必胜, 则学习其策略, 我方必胜.

6.10 Other's theory?

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 规定每次能从任意多堆中取 1 根, 不可不取.
最后取完者为胜。
结论: 如果全是偶数, 先手必败, 否者先手必胜

一个无相联通图，有一个点作为图的根。
游戏者轮流从图中删去边，删去一条边后，不与根节点相连的部分将被移走。
谁无路可走谁输。
结论:
Fusion Principle: 环上的点可以融合，且不改变图的 SG 值, 我们可以把一个带有奇数边的环等价成只有一个端点的一条边而偶数边的环等价于一个点

6.11 SG Theory

```
memset(mex, 0, sizeof mex);
for (int i = 1; i < maxN;++i) {
    for (int j = 1; j <= n;++j) {
        if (a[j] <= i)
            mex[SG[i - a[j]]] = i;
            for (int k = 1; i - k - a[j] > 0;++k)
                mex[SG[k] ^ SG[i - k - a[j]]] = 1;
```

```
    }
    for (int j = 0;; ++j)
        if (mex[j]!=i){
            SG[i] = j;
            break;
        }
}
```

6.12 SJ Theory

反公平游戏 Anti-SG Game

DAG 上没有出度的点为胜利状态，其它定义与一般游戏相同。

现在的问题是解决多个反公平游戏的合并。

SJ 定理说明：先手必胜，当且仅当以下两个条件同时成立或同时不成立：

1. 合并的 SG 值为 0；
2. 所有游戏的 SG 值不超过 1。

6.13 Surreal Number Theory

7 DP

7.1 最大子矩阵

单调栈法:

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1007;
int a[MAXN][MAXN], b[MAXN][MAXN];
pair<int, int> mp[MAXN * MAXN];
pair<int, int> off[MAXN][MAXN];
int tail;
pair<int, int> st[MAXN];
int Up[MAXN][MAXN];
int ans = 0;
void maintain(pair<int, int> v) {
    for(;tail && st[tail].first >= v.first; tail--) {
        ans = max(ans, st[tail].first * (v.second - st[tail-1].second-1));
    }
}
void clear(int pos) {
    for(;tail; tail--)
        ans = max(ans, st[tail].first * (pos -st[tail-1].second-1));
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n, m;
    cin >> n >> m;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++) {
            cin >> a[i][j];
            mp[a[i][j]] = {i,j};
        }
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++) {
            cin >> b[i][j];
            off[i][j] = {i - mp[b[i][j]].first, j - mp[b[i][j]].second};

        }
    for(int i = 1; i <= n; i++) {
        st[0].second = 0;
        for(int j = 1; j <= m; j++) {
            Up[i][j] = i > 1 && off[i][j] == off[i-1][j] ? Up[i-1][j] + 1 : 1;
            if(j > 1 && off[i][j] == off[i][j-1])
                maintain({Up[i][j], j}); else {clear(j); st[0].second = j-1;}
            st[++tail] = {Up[i][j], j};
        }
        clear(m+1);
    }
    cout << ans << endl;
    return 0;
}
```

悬线法:

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1007;
int a[MAXN][MAXN], b[MAXN][MAXN];
pair<int, int> mp[MAXN * MAXN];
```

```cpp
pair<int, int> off[MAXN][MAXN];
int L[MAXN][MAXN], R[MAXN][MAXN], UP[MAXN][MAXN];
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n, m;
    cin >> n >> m;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++) {
            cin >> a[i][j];
            mp[a[i][j]] = {i,j};
        }
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++) {
            cin >> b[i][j];
            off[i][j] = {i - mp[b[i][j]].first, j - mp[b[i][j]].second};
            L[i][j] = R[i][j] = j;
            UP[i][j] = 1;
        }
    for(int i = 1; i <= n; i++)
        for(int j = 2; j <= m; j++) {
            if(off[i][j] == off[i][j-1]) L[i][j] = L[i][j-1];
        }
    for(int i = 1; i <= n; i++)
        for(int j = m - 1; j >= 1; j--) {
            if(off[i][j] == off[i][j+1]) R[i][j] = R[i][j+1];
        }
    int ans = 0;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++) {
            if(i > 1 && off[i][j] == off[i-1][j]) {
                UP[i][j] = UP[i-1][j] + 1;
                L[i][j] = max(L[i][j], L[i-1][j]);
                R[i][j] = min(R[i][j], R[i-1][j]);
            }
            ans = max(ans, (R[i][j] - L[i][j] + 1) * UP[i][j]);
        }
    cout << ans << endl;
    return 0;
}
```

7.2 斜率优化

7.3 四边形不等式优化

7.4 忘情水二分